BOUNDED-DEGREE FACTORS OF LACUNARY MULTIVARIATE POLYNOMIALS

BRUNO GRENET

Abstract. In this paper, we present a new method for computing bounded-degree factors of lacunary multivariate polynomials. In particular for polynomials over number fields, we give a new algorithm that takes as input a multivariate polynomial f in lacunary representation and a degree bound d and computes the irreducible factors of degree at most d of f in time polynomial in the lacunary size of f and in d. Our algorithm consists in a reduction of the problem to the univariate case on the one hand and to the irreducible factorization of multivariate low-degree polynomials on the other hand, which is valid for any field of zero characteristic.

The reduction algorithms we propose are elementary in that they only manipulate the exponent vectors of the input polynomial. The proof of correctness and the complexity bounds rely on the valuations of certain formal power series with rational exponents, called Puiseux series, and on considerations on the Newton polytope of the polynomial.

1. Introduction

In this paper, we give a new polynomial-time algorithm for computing bounded-degree factors of lacunary multivariate polynomials. The *lacunary representation* of a polynomial consists in the list of its nonzero monomials and its size is logarithmic in the degree of the polynomial. Our algorithm is a reduction of the problem to the univariate case on the one hand, and to the irreducible factorization of low-degree polynomials on the other hand.

The factorization of polynomials is a well-studied subject in symbolic computation. Although there exist effective fields in which testing irreducibility of polynomials is undecidable [10], the irreducible factorization of univariate or multivariate polynomials can be computed in time polynomial in the degree of the input polynomial for many base fields. Without claim of exhaustivity, one can cite the cases of polynomials over rational numbers [21, 14] or more generally algebraic number fields [22, 20, 23] or over finite fields [2]. In a somewhat different perspective, one can also compute the factorization in an extension of the base field, such as (approximate) factorization in the real or complex numbers [25, 17] or absolute

factorization, that is factorization over an algebraic closure of the base field [8].

The purpose of this paper is to study the possibility of algorithms of complexity polynomial in the size of the lacunary representation of the input polynomial, that is in particular logarithmic rather than polynomial in the degree. Note that in lacunary representation, even evaluating a polynomial over an input is intractable: For instance, the monomial X^d has lacunary size $O(\log d)$ while its evaluation on the input 2 is an integer of size d. More generally, testing the irreducibility of lacunary polynomials or computing the greatest common divisor of two lacunary polynomials are NP-hard problems [26, 18, 15]. This motivates to reduce the ambitions and to compute only a partial factorization of the input polynomial.

Cucker, Koiran and Smale gave an algorithm to compute the integer roots, that is linear factors, of univariate integer polynomials in time polynomial in the lacunary representation [9]. This result was extended by Lenstra who described an algorithm to compute the bounded-degree factors of polynomials over number fields [24]. His algorithm takes as input a description of the number field by means of an irreducible polynomial with integer coefficients in dense representation, the polynomial to factor in lacunary representation, and a bound on the degree of the factors it computes. The complexity is polynomial in the size of the input and in the degree bound (rather than in its bit-size). Then, Kaltofen and Koiran extended this result to the computation of linear factors of bivariate polynomials over the rational numbers [15], and then to the computation of bounded-degree factors of multivariate polynomials over number fields [16]. Seemingly independently of this latest result, Avendaño, Krick and Sombra gave an algorithm to compute the bounded-degree factors of bivariate polynomials over number fields [1]. They also explained how to compute the bounded-degree factors over an algebraic closure of the rational numbers, but the binomial ones. Note that the binomial factors include univariate linear factors the number of which cannot be polynomially bounded in the logarithm of the degree. These are the only known polynomial-time algorithms for (partial) factorization of polynomials given in lacunary representation. Nonetheless, algorithms have been given for the computation of the real roots of integer polynomials [27] and the computation of roots of polynomials over finite fields [4] that run in sublinear time in the degree. Note that in the last case, the authors also proved that the problem is NP-hard (under randomized reductions).

The algorithms we propose are an extension of the algorithms we described for computing linear factors of bivariate and multivariate polynomials [7, 6]. We identify two distinct kind of factors, namely the *unidimensional* and the *multidimensional* factors. Roughly

speaking, a polynomial is said unidimensional if it can be written as $f(\boldsymbol{X}^\delta)$ where f is a univariate polynomial and $\boldsymbol{X}^\delta$ a multivariate monomial. We describe an algorithm to reduce the computation of the bounded-degree unidimensional factors of a lacunary multivariate polynomial to the computation of the bounded-degree factors of some lacunary univariate polynomials. It is based on the fact that unidimensional factors of a multivariate polynomial are in bijection with the irreducible factors of some *univariate projections* of the polynomial. It is valid for any base field, in any characteristic. For the multidimensional factors, we give an algorithm to reduce their computation to the irreducible factorization of a low-degree polynomial. This algorithm is based on a so-called *Gap Theorem*, valid for any field of characteristic 0, that asserts that if a polynomial f can be written $f_1 + \boldsymbol{X}^\delta f_2$ where δ is large enough, each low-degree factor of a f is a common factor of f_1 and f_2. This technique of Gap Theorem was introduced by Cucker, Koiran and Smale and the subsequent works mentioned above were all based on a Gap Theorem. Both algorithms are elementary since only the exponent vectors of f and not its coefficients are manipulated.

The proof of our Gap Theorem is based on the notion of Puiseux expansion of a bivariate polynomial, and makes use of the Newton polygon of the polynomial. Given a polynomial $g \in \mathbb{K}[X, Y]$, one can describe its *roots* in some algebraic closure $\overline{\mathbb{K}}(X)$ of $\mathbb{K}(X)$ in terms of Puiseux series, that is formal power series whose exponents are rational numbers with a common denominator. For a Puiseux series ϕ cancelling a polynomial g and a lacunary polynomial f, we give a bound on the valuation of $f(X, \phi(X))$ in terms of the degree of g and the number of nonzero monomials of f. The proof of this valuation bound is based on the wronskian determinant of a family of linearly independent power series. This bound is used to deduce the Gap Theorem.

As a corollary, we obtain a new proof of the main result of Kaltofen and Koiran [16] stating that over algebraic number fields, one can compute the degree-d factors of a lacunary multivariate polynomial f in deterministic time $(\mathrm{size}(f) + d)^{O(n)}$ or probabilistic time $(\mathrm{size}(f) + d)^{O(1)}$, where $\mathrm{size}(f)$ is the lacunary size of f. Their algorithm uses a universal constant arising from number theory that is not explicitly given. In contrast, our algorithm is entirely explicit and can easily be implemented.

Since our Gap Theorem applies to any field of characteristic 0, we obtain partial results for other fields. In particular, for any field that admits a multivariate factorization algorithm running in time polynomial in the degree of the input polynomial, we obtain a algorithm to compute the bounded-degree multidimensional factors

of multivariate polynomials that runs in time polynomial in the lacunary size of the input polynomial and the degree bound. Such fields include the fields of real or complex numbers, the fields of p-adic numbers or the algebraic closure of the rational number. In this latest case, we obtain a new proof of Avendaño, Krick and Sombra's result [1] since the unidimensional irreducible polynomials over $\overline{\mathbb{Q}}$ are exactly the binomials. Indeed, we prove that the unidimensional factors of a polynomial are in bijection with the factors of some univariate projections of the polynomial, and irreducible univariate polynomials over $\overline{\mathbb{Q}}$ are linear polynomials.

Our results leave open some questions. A first one concerns the fields of positive characteristic. In this case, our Gap Theorem does not apply as such. Yet, in the specific case of multilinear factors and large characteristic, we proved that it is applicable yielding an algorithm for fields of large characteristic [7]. We suspect that a similar result holds here though we were not able to prove it yet. For fields of small characteristic, the same gap argument does not seem to apply but there may well exist a different approach that exploits the fact that the characteristic is small.

Another question concerns the factors computed. As sketched in Sec. 3, our technique for computing unidimensional factors can be used in other contexts such as the computation of lacunary factors. As mentioned above, we cannot hope for a polynomial time algorithm computing the irreducible factorization of a lacunary polynomial, whence the restriction on the degree of the factors. A generalization of this would be to impose a bound on the number of nonzero monomials of the factors instead of their degree. We do not know whether our techniques may be adapted to this case or whether new techniques could be used. Since this problem concerns lacunary polynomials, we should not exclude the possibility of a hardness result too.

A preliminary version of this article appeared in the proceedings of ISSAC 2014 [12]. The current paper is a complete rewriting of the preceding article. In particular, the algorithms for multivariate polynomials were only sketched. In this version, the algorithms have been simplified and are described in full details. Moreover, we tighten our complexity analyses.

Organization of the paper. In Sec. 2, we introduce the necessary notions used throughout the paper. Sec. 3 is devoted to the unidimensional factors, and Sec. 4 to the multidimensional factors.

Acknowledgements. I would like to thank again P. Koiran, N. Portier, Y. Strozecki, A. Bostan, P. Lairez and B. Salvy for discussions that helped me to prepare the preliminary version of this paper. I am also grateful to R. Lebreton for discussions on this new version.

2. Preliminaries

2.1. **Notations.** Let $\boldsymbol{X}$ denote the tuple of variables $(X_1, \ldots, X_n)$, and $\boldsymbol{\alpha}$ the tuple $(\alpha_1, \ldots, \alpha_n)$. The notation $\boldsymbol{X}^{\boldsymbol{\alpha}}$ denotes the monomial $\prod_{i=1}^{n} X_i^{\alpha_i}$. Addition and subtraction of tuples are defined componentwise: $\boldsymbol{\alpha} - \boldsymbol{\beta} = (\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n)$. For any scalar k, $k\boldsymbol{\alpha}$ denotes the vector $(k\alpha_1, \ldots, k\alpha_n)$.

We consider polynomials in a ring $\mathbb{K}[X_1, \ldots, X_n]$. The degree of a polynomial f with respect to the variable X_i is denoted $\deg_{X_i}(f)$ and its valuation with respect to X_i is denoted $\mathrm{val}_{X_i}(f)$. Let also $\mathrm{mdeg}(f)$ denotes the multidegree $(\deg_{X_1}(f), \ldots, \deg_{X_n}(f))$ and $\mathrm{mval}(f)$ the multivaluation $(\mathrm{val}_{X_1}(f), \ldots, \mathrm{val}_{X_n}(f))$. If f is a univariate polynomial, we shall use the usual notations $\deg(f)$ and $\mathrm{val}(f)$ for its degree and valuation. We say that a polynomial has multidegree *at most* $(d_1, \ldots, d_n)$ if $\deg_{X_i}(f) \leq d_i$ for all i.

The multiplicity of g as a factor of f, that is the maximum integer μ such that g^μ divides f, is denoted by $\mathrm{mult}_g(f)$.

A *partition* of a polynomial $f = \sum_{j=1}^{k} c_j \boldsymbol{X}^{\boldsymbol{\alpha}_j}$ is a set of polynomials $\{f_1, \ldots, f_s\}$ defined by a partition of $\{1, \ldots, k\}$ into disjoint subsets. The polynomials $f_1, \ldots, f_s$ are the *summand* of the partition. In particular, $f = f_1 + \cdots + f_s$ and two distinct summand do not share any common monomial. We shall mainly write partitions as sums of the form $f_1 + \cdots + f_s$ rather than as sets $\{f_1, \ldots, f_s\}$.

We use several notions of size for a polynomial f. Let $\mathrm{size}(f)$ denotes its *lacunary size*: if $f = \sum_{j=1}^{k} c_j \boldsymbol{X}^{\boldsymbol{\alpha}_j} \in \mathbb{K}[X_1, \ldots, X_n]$,

$$\mathrm{size}(f) = k(n \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq k}} (\log(\alpha_{i,j})) + \mathrm{size}(c_j))$$

where the size of c_j depends on the field $\mathbb{K}$. Actually since our algorithms only manipulate the exponent vectors, we shall express the complexities as functions of k, n and the total degree D of f. More precisely, we aim to describe algorithms of complexity polynomial in k, n and $\log(D)$. The integer k is called the *sparsity* of f.

We shall also use the notion of *convex size* which denotes the volume of its Newton polytope (*cf.* next section).

To express the complexity of some of our algorithms, we shall use the notation $\mathsf{M}(D)$ which denotes the complexity of standard mathematical operations, such as multiplication, on integers bounded by D, that is of size $\log(D)$. It satisfies

$$\mathsf{M}(D) = O(\log(D) \log\log(D) \log\log\log(D)) \text{ [28]},$$

or even $\mathsf{M}(D) = O(\log(D) 8^{\log^\star(D)})$ [11, 13] where $\log^\star$ denotes the iterated logarithm which increases slower than any iteration

$\log \circ \log \circ \cdots \circ \log$ of the logarithm. Note for instance that computing the greatest common divisor of two integers of size at most $\log(D)$ takes $O(\mathsf{M}(D)\log(D))$ bit-operations.

2.2. **Newton polytope.**

Definition 2.1. Let $f \in \mathbb{K}[X_1, \ldots, X_n]$. Its *support* is the set of vectors $(\alpha_1, \ldots, \alpha_n)$ such that the coefficient of the monomial $X_1^{\alpha_1} \cdots X_n^{\alpha_n}$ in f is nonzero. The *Newton polytope* of f, denoted by $\mathrm{Newt}(f)$, is the convex hull of its support.

A polytope in two dimensions is called a *polygon*, whence the appellation *Newton polygon* when f is bivariate.

Two convex polytopes can be added using the *Minkowski sum*, defined by $A + B = \{a + b : a \in A, b \in B\}$ for two polytopes A and B. If A and B are both convex, then so does $A + B$. Minkowski sum is related to factorization of polynomials by Ostrowski's Theorem.

Ostrowski's Theorem. *Let f, g, $h \in \mathbb{K}[X_1, \ldots, X_n]$ such that $f = gh$. Then* $\mathrm{Newt}(f) = \mathrm{Newt}(g) + \mathrm{Newt}(h)$.

If we consider the Newton polygon of a bivariate polynomial, its boundary is made of edges whose extremities are points of the support. In n variables, the boundary of a Newton polytope is made of faces which have dimension 1 to $(n-1)$. We shall only consider faces of dimension 1 that we still call edges. Again, the extremities of an edge are points in the support of the polynomial.

We define the *direction of an edge* of extremities $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as the unique vector $\boldsymbol{\delta} \in \mathbb{Z}^n$ collinear to $\boldsymbol{\alpha} - \boldsymbol{\beta} = (\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n)$ such that $\gcd(\delta_1, \ldots, \delta_n) = 1$ and whose first nonzero coordinate is positive. Then, there exists $\lambda \in \mathbb{Z}$ such that $\boldsymbol{\alpha} - \boldsymbol{\beta} = \lambda\boldsymbol{\delta}$.

Ostrowski's theorem shall mainly be used through a corollary concerning the edges of the Newton polytopes.

Corollary 2.2. *Let f, g, $h \in \mathbb{K}[X_1, \ldots, X_n]$ such that $f = gh$. Then each edge in* $\mathrm{Newt}(f)$ *is parallel to either an edge of* $\mathrm{Newt}(g)$ *or an edge of* $\mathrm{Newt}(h)$.

For some algorithms, we may want to have access to the Newton polytope of f. Actually, this is doable in polynomial time for a fixed number of variables only.

Proposition 2.3. *Let $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k \in \mathbb{Z}^n$, $\|\boldsymbol{\alpha}_1\|_\infty, \ldots, \|\boldsymbol{\alpha}_k\|_\infty \leq D$. Their convex hull can be deterministically computed in $O(k^{\lfloor n/2 \rfloor} n \mathsf{M}(D))$ bit-operations.*

2.3. **Puiseux series.**

Definition 2.4. Let $\overline{\mathbb{K}}$ be an algebraic closure of a field $\mathbb{K}$ of characteristic 0. Then the *field of Puiseux series over* $\overline{\mathbb{K}}$, denoted by $\overline{\mathbb{K}}\langle\langle X \rangle\rangle$, is

defined as the set of formal power series of the form

$$\phi(X) = \sum_{t \geq t_0} f_t X^{t/d}$$

where $t_0 \in \mathbb{Z}$, $d \in \mathbb{N}$, and $f_t \in \overline{\mathbb{K}}$ for all t.

If $f_{t_0} \neq 0$, the *valuation* of ϕ is $\mathrm{val}(\phi) = t_0/d$.

Addition and multiplication are defined as for standard formal power series. A theorem of Puiseux states that Puiseux series actually form a field, and that this field is algebraically closed. We can give a constructive version of this result using the Newton polygon of a bivariate polynomial $f \in \mathbb{K}[X, Y]$.

Since $\mathrm{Newt}(f)$ lives in this case in a plane, one can choose a system of coordinates to draw it and define the lower and upper parts of the Newton polygon. To each edge, we attach a normal vector pointing inside the Newton polygon. The *lower hull* is defined as the set of edges whose normal vectors have a positive second coordinate, the *upper hull* as the set of edges whose normal vectors have a negative second coordinate, and *vertical edges* are the (at most 2) edges whose normal vectors have their second coordinate equal to zero. The names are clear if we choose to represent the exponents of Y on the x-axis and the exponents of X on the y-axis.

Newton-Puiseux Theorem. *Let $g \in \mathbb{K}[X, Y]$. Then there exists $\phi \in \overline{\mathbb{K}}\langle\langle X \rangle\rangle$ of valuation v such that $g(X, \phi(X)) = 0$ if and only if the lower hull of* $\mathrm{Newt}(g)$ *contains an edge of direction (p, q) such that $v = -p/q$.*

One particular consequence of the Newton-Puiseux Theorem is that if g has bidegree (d_X, d_Y), any edge of its Newton polygon is contained in the square $[0, d_X] \times [0, d_Y]$, hence has direction (p, q) with $|p| \leq d_X$ and $|q| \leq d_Y$. One can rephrase it by saying that the valuation v of any root $\phi \in \overline{\mathbb{K}}\langle\langle X \rangle\rangle$ of g satisfy $1/d_Y \leq |v| \leq d_X$.

2.4. **The wronskian determinant.** Our main technical result uses the wronskian determinant of a family of Puiseux series.

Definition 2.5. Let $(\phi_1, \ldots, \phi_\ell)$ be a family of Puiseux series. Its *wronskian* is

$$\mathrm{wr}(\phi_1, \ldots, \phi_\ell) = \det \begin{pmatrix} \phi_1 & \cdots & \phi_\ell \\ \phi_1' & \cdots & \phi_\ell' \\ \vdots & & \vdots \\ \phi_1^{(\ell-1)} & \cdots & \phi_\ell^{(\ell-1)} \end{pmatrix}.$$

The main property we shall use is the relation of the wronskian to the linear independence. Bostan and Dumas give a proof of the following proposition in the case of formal power series [5] but the exact same proof applies to Puiseux series.

Proposition 2.6. *Let $\phi_1, \dots, \phi_\ell \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$ be Puiseux series. Then the family $(\phi_j)_{1\leq j\leq \ell}$ is linearly independent if and only if its wronskian does not vanish.*

We aim to give bounds on the valuation of the wronskian of certain particular families of Puiseux series in Sec. 4.1. We can first give a general lower bound.

Lemma 2.7. *Let $\phi_1, \dots, \phi_\ell \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$. Then*

$$\operatorname{val}(\operatorname{wr}(\phi_1, \dots, \phi_k)) \geq \sum_{j=1}^{\ell} \operatorname{val}(\phi_j) - \binom{\ell}{2}.$$

Proof. Let us consider the matrix W such that $\operatorname{wr}(\phi_1, \dots, \phi_\ell) = \det(W)$. The entry (i, j) of the W is $\phi_j^{(i-1)}$. Each term in $\det(W)$ is thus a product of the form

$$\phi_1^{(i_1)} \dots \phi_\ell^{(i_\ell)}$$

such that $\{i_1, \dots, i_\ell\} = \{0, \dots, \ell - 1\}$. This product has valuation at least

$$\sum_{j=1}^{\ell} \operatorname{val}(\phi_j) - i_j = \sum_{j=1}^{\ell} \operatorname{val}(\phi_j) - \sum_{j=0}^{\ell-1} j.$$

As a sum of such terms, the wronskian has valuation at least as large as these terms. □

2.5. **Uni- and multidimensional polynomials.** In our algorithms, we treat in very different way unidimensional and multidimensional polynomials. We first give a formal definition of these terms.

Definition 2.8. A polynomial $f \in \mathbb{K}[X_1, \dots, X_n]$ is *unidimensional* if the dimension of its Newton polytope is exactly 1, that is if f has at least two monomials and its Newton polygon is a line segment. The *direction* $\delta \in \mathbb{Z}^n$ of f is the direction of the unique edge of its Newton polytope.

A polynomial is *multidimensional* if its Newton polytope has dimension at least 2.

Note that monomials are neither unidimensional nor multidimensional. Since the computation of the monomial factors is obvious, we ignore them in the rest of the paper.

We remark that for bivariate polynomials, being unidimensional is the same as being weighted-homogeneous. This is not true anymore for polynomials in more variables. We now define several notion by analogy with homogeneous polynomials.

We begin with the decomposition of a polynomial as a sum of homogeneous, here unidimensional, components. For a polynomial $f \in \mathbb{K}[X_1, \dots, X_n]$ and $\delta \in \mathbb{Z}^n$, one can write $f = f_1 + \dots + f_s$ where

f_t is unidimensional of direction $\boldsymbol{\delta}$ for all t. If further no sum $f_{t_1} + f_{t_2}$ is unidimensional (that is the f_t's are maximal), the f_t's are the *unidimensional components of direction* $\boldsymbol{\delta}$ of f, in short the $\boldsymbol{\delta}$*-components* of f.

The notions analogous to homogenization and dehomogenization are called projection and lifting in our settings.

Lemma 2.9. *Let $f \in \mathbb{K}[X_1, \ldots, X_n]$ be a unidimensional polynomial of direction $\boldsymbol{\delta}$. There exists a unique univariate polynomial $f_\pi \in \mathbb{K}[Z]$ of valuation 0 such that $f(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}} f_\pi(\boldsymbol{X}^{\boldsymbol{\delta}})$ for some $\boldsymbol{\alpha} \in \mathbb{Z}^n$. Furthermore, $\boldsymbol{\alpha}$ is in this case nonnegative, that is $\boldsymbol{\alpha} \in \mathbb{N}^n$.*

Proof. Let $f(\boldsymbol{X}) = \sum_{j=1}^{\ell} c_j \boldsymbol{X}^{\boldsymbol{\alpha}_j}$. Since f is unidimensional of direction $\boldsymbol{\delta}$, there exists for all j an integer λ_j such that $\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_1 = \lambda_j \boldsymbol{\delta}$. In particular, $\lambda_1 = 0$. Let j_0 be the index of the smallest λ_j and $\lambda_j^\star = \lambda_j - \lambda_{j_0}$ for all j. Then, the integers $\lambda_1^\star, \ldots, \lambda_\ell^\star$ are nonnegative and satisfy $\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j_0} = \lambda_j^\star \boldsymbol{\delta}$ for all j. Moreover, it is clear that the index j_0 is the unique index with the property that $(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j_0})$ is a nonnegative multiple of $\boldsymbol{\delta}$.

Let us define $f_\pi = \sum_{j=1}^{\ell} c_j Z^{\lambda_j^\star}$. Then f_π belongs to $\mathbb{K}[Z]$ and has valuation 0. Moreover if we let $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{j_0}$, we have

$$\boldsymbol{X}^{\boldsymbol{\alpha}} f_\pi(\boldsymbol{X}^{\boldsymbol{\delta}}) = \sum_{j=1}^{\ell} c_j \boldsymbol{X}^{\boldsymbol{\alpha} + \lambda_j^\star \boldsymbol{\delta}} = \sum_{j=1}^{\ell} c_j \boldsymbol{X}^{\boldsymbol{\alpha}_j}$$

since $\boldsymbol{\alpha} + \lambda_j^\star \boldsymbol{\delta} = \boldsymbol{\alpha}_j$ by definition of $\lambda_j^\star$. This proves the existence of f_π.

To prove its unicity, let us consider $g = \sum_{p=1}^{\ell} b_p Z^{\gamma_p}$ with $0 = \gamma_1 < \cdots < \gamma_\ell$ such that $f(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}} g(\boldsymbol{X}^{\boldsymbol{\delta}})$ for some $\boldsymbol{\alpha} \in \mathbb{Z}^n$. Clearly, since f is a polynomial and g has valuation 0, $\boldsymbol{\alpha}$ belongs to $\mathbb{N}^n$ and is the exponent of some monomial of f. Now, for all j there exists a p such that the term $c_j \boldsymbol{X}^{\boldsymbol{\alpha}_j}$ of f is the image of the term $b_p Z^{\gamma_p}$ of g. In particular, $\boldsymbol{\alpha}_j - \boldsymbol{\alpha} = \gamma_p \delta$. Since $\gamma_p \geq 0$, the uniqueness of the index j_0 defined in the first paragraph shows that $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{j_0}$. Then, each γ_p is uniquely defined by the differences $(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j_0})$ and $g = f_\pi$. □

From this lemma, one can define *the* projection of a unidimensional polynomial.

Definition 2.10. Let $f \in \mathbb{K}[X_1, \ldots, X_n]$ a unidimensional polynomial of direction $\boldsymbol{\delta}$. Its *(univariate) projection* is the unique polynomial $f_\pi \in \mathbb{K}[Z]$ such that $f(X_1, \ldots, X_n) = \boldsymbol{X}^{\boldsymbol{\alpha}} f_\pi(\boldsymbol{X}^{\boldsymbol{\delta}})$ for some $\boldsymbol{\alpha} \in \mathbb{N}^n$

Let $g \in \mathbb{K}[Z]$ a univariate polynomial and $\boldsymbol{\delta} \in \mathbb{Z}^n$. Its *lifting in direction* $\boldsymbol{\delta}$ is the unidimensional polynomial $f \in \mathbb{K}[X_1, \ldots, X_n]$ defined by $f(X_1, \ldots, X_n) = \boldsymbol{X}^{\boldsymbol{\alpha}} g(\boldsymbol{X}^{\boldsymbol{\delta}})$ for some $\boldsymbol{\alpha} \in \mathbb{Z}^n$ such that the valuation of f is zero with respect to each of its variables.

Note that if $f \in \mathbb{K}[X_1, \dots, X_n]$ is a unidimensional polynomial of valuation zero with respect to each of its variables, the operations of projection and lifting are inverse of each other. That is, the lifting in direction $\boldsymbol{\delta}$ of f_π is f itself. In general, the lifting of f_π is the polynomial f° defined by $f^\circ(\boldsymbol{X}) = f(\boldsymbol{X})/\boldsymbol{X}^{\mathrm{mval}(f)}$. In other words, if two unidimensional polynomials f and g of direction $\boldsymbol{\delta}$ have the same projection, there exists $\boldsymbol{\alpha} \in \mathbb{Z}^n$ such that $f(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}} g(\boldsymbol{X})$.

We shall need a bound on the degree of the projection g_π of a unidimensional polynomial g. Let us assume that the valuation of g in each variable is null, for example that g is irreducible, and let $\boldsymbol{\delta}$ be the direction of g and $(d_1, \dots, d_n)$ its multidegree. There exists $\boldsymbol{\alpha}$ such that $g(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}} g_\pi(\boldsymbol{X}^{\boldsymbol{\delta}})$. Consider a variable X_i such that $\delta_i \neq 0$. If $\delta_i > 0$, then $d_i = \alpha_i + \delta_i \deg(g_\pi)$, and since g and g_π have valuation 0, $\alpha_i = 0$. Thus $\deg(g_\pi) = d_i/\delta_i$. If $\delta_i < 0$, we get $\alpha_i = d_i$ and $0 = \alpha_i + \delta_i \deg(g_\pi)$. Whence in both cases $\deg(g_\pi) = d_i/|\delta_i|$. In particular, let us assume that we only a bound on the multidegree of g, then $\deg(g_\pi) = \min_i(d_i/|\delta_i|)$ where the minimum is taken over the indices i such that $\delta_i \neq 0$.

3. Unidimensional factors

In this section, we show how to reduce the computation of the unidimensional factors of some polynomial $f \in \mathbb{K}[X_1, \dots, X_n]$ to the factorization of (several) univariate polynomials. Let us sketch the main ingredients of the algorithm. Suppose we want to compute the unidimensional factors of some $f \in \mathbb{K}[X_1, \dots, X_n]$. First, a set $\Delta \subset \mathbb{Z}^n$ of directions is computed such that Δ contains the direction of each unidimensional factor of f. For each $\boldsymbol{\delta} \in \Delta$, one can compute the unidimensional factors of direction $\boldsymbol{\delta}$ as follows. The polynomial f is written as a sum of $\boldsymbol{\delta}$-components, and these components are projected to $\mathbb{K}[Z]$. Using an univariate factorization algorithm, the common factors of these projections are computed, and then lifted in direction $\boldsymbol{\delta}$. Theorem 3.1 below proves that the set of polynomial computed in such a way is indeed the set of unidimensional factors of f of direction $\boldsymbol{\delta}$. And Lemma 3.10 proves that the cost of the algorithm, discarding the cost of the univariate factorization algorithm, is $O(\mathrm{size}(\mathcal{F})kn \log(D))$ where k is the sparsity of f, n its number of variables, D its degree, $\mathcal{F}$ the set of factors computed and $\mathrm{size}(\mathcal{F})$ the sum of the sizes of the elements of $\mathcal{F}$.

In the context of this paper, we are mostly interested in bounded-degree unidimensional factors of lacunary polynomials and we show that their computation reduces to the computation of the bounded-degree factors of lacunary univariate polynomials. However, the techniques we use are more general and could be used as well for

low-degree polynomials, and for the computation of all unidimensional factors or for the computation of the lacunary unidimensional factors for instance.

3.1. **Structural result.**

Theorem 3.1. *Let $f \in \mathbb{K}[X_1, \ldots, X_n]$ and $\boldsymbol{\delta} \in \mathbb{Z}^n$. Let $f_1, \ldots, f_s$ its $\boldsymbol{\delta}$-components and $(f_1)_\pi, \ldots, (f_s)_\pi \in \mathbb{K}[Z]$ their projections. Then for any unidimensional polynomial g of direction $\boldsymbol{\delta}$,*

$$\operatorname{mult}_g(f) = \min_{1 \le t \le s} \operatorname{mult}_{g_\pi}((f_t)_\pi)$$

where g_π is the projection of g.

This theorem is a direct consequence of the two following lemmas.

Lemma 3.2. *Let $f \in \mathbb{K}[X_1, \ldots, X_n]$, and $\boldsymbol{\delta} \in \mathbb{Z}^n$. The unidimensional factors of direction $\boldsymbol{\delta}$ of f are the common factors of its $\boldsymbol{\delta}$-components. More precisely,*

$$\operatorname{mult}_g(f) = \min_{1 \le t \le s} \operatorname{mult}_g(f_t)$$

where $f_1, \ldots, f_t$ are the $\boldsymbol{\delta}$-components of f.

Proof. The product of two unidimensional polynomial g and h of direction $\boldsymbol{\delta}$ is itself a unidimensional polynomial of direction $\boldsymbol{\delta}$. Indeed let $f = gh$ and consider two monomials $\boldsymbol{X}^{\boldsymbol{\alpha}}$ and $\boldsymbol{X}^{\boldsymbol{\beta}}$ of f. Each monomial is a product of a monomial of g and a monomial of h. Let us assume that $\boldsymbol{X}^{\boldsymbol{\alpha}} = \boldsymbol{X}^{\boldsymbol{\alpha}_g}\boldsymbol{X}^{\boldsymbol{\alpha}_h}$ and $\boldsymbol{X}^{\boldsymbol{\beta}} = \boldsymbol{X}^{\boldsymbol{\beta}_g}\boldsymbol{X}^{\boldsymbol{\beta}_h}$ where $\boldsymbol{X}^{\boldsymbol{\alpha}_g}$ and $\boldsymbol{X}^{\boldsymbol{\beta}_g}$ are monomials of g and $\boldsymbol{X}^{\boldsymbol{\alpha}_h}$ and $\boldsymbol{X}^{\boldsymbol{\beta}_h}$ are monomials of h. Then

$$\begin{aligned}\boldsymbol{\alpha} - \boldsymbol{\beta} &= (\boldsymbol{\alpha}_g + \boldsymbol{\alpha}_h) - (\boldsymbol{\beta}_g + \boldsymbol{\beta}_h) = (\boldsymbol{\alpha}_g - \boldsymbol{\beta}_g) + (\boldsymbol{\alpha}_h - \boldsymbol{\beta}_h) \\ &= \lambda_g \boldsymbol{\delta} + \lambda_h \boldsymbol{\delta} = (\lambda_g + \lambda_h)\boldsymbol{\delta}\end{aligned}$$

for some $\lambda_g, \lambda_h \in \mathbb{Z}$. This shows that f is unidimensional of direction $\boldsymbol{\delta}$.

Consider now a unidimensional factor g of direction $\boldsymbol{\delta}$ of some polynomial $f \in \mathbb{K}[X_1, \ldots, X_n]$, and let $h = f/g$. Let us write $h = h_1 + \cdots + h_s$ as a sum of $\boldsymbol{\delta}$-components. Then $gh = gh_1 + \cdots + gh_s$ and each gh_t is unidimensional of direction $\boldsymbol{\delta}$. This proves in particular that g divides each $\boldsymbol{\delta}$-component of f. To conclude, it remains to notice that the same argument works for g^μ where $\mu = \operatorname{mult}_g(f)$. □

Lemma 3.3. *Let f and $g \in \mathbb{K}[X_1, \ldots, X_n]$ be unidimensional polynomials of same direction and f_π and g_π their respective projections. Then*

$$\operatorname{mult}_g(f) = \operatorname{mult}_{g_\pi}(f_\pi).$$

Proof. We first prove that the projection commutes with multiplication: If $f = gh$ is a unidimensional polynomial, its projection is $g_\pi h_\pi$ where g_π is the projection of g and h_π the projection of h.

Let δ be the direction of f, g and h. By definition, there exist $\boldsymbol{\alpha}_g$ and $\boldsymbol{\alpha}_h \in \mathbb{Z}^n$ such that $g(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}_g} g_\pi(\boldsymbol{X}^\delta)$ and $h(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}_h} h_\pi(\boldsymbol{X}^\delta)$. Thus, $f(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}_g+\boldsymbol{\alpha}_h} g_\pi(\boldsymbol{X}^\delta) h_\pi(\boldsymbol{X}^\delta)$. We can define $f_\pi = g_\pi h_\pi$ and we obtain $f(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}_g+\boldsymbol{\alpha}_h} f_\pi(\boldsymbol{X}^d)$. Hence f_π is indeed the projection of f, by unicity of the projection.

Assume that g^μ divides f for some $\mu > 0$, that is there exists h such that $f = g^\mu h$. The projection of f is then $f_\pi = g_\pi^\mu h_\pi$, and $\mathrm{mult}_{g_\pi}(f_\pi) \geq \mu$. Conversely, let us assume that $f_\pi = g_\pi^\mu h_\pi$ for some g_π, $h_\pi \in \mathbb{K}[Z]$, and denote by g and h the respective liftings of g_π and h_π in direction δ. Let $f^\circ = g^\mu h$, so that $f_\pi^\circ = g_\pi^\mu h_\pi = f_\pi$. Then there exists $\boldsymbol{\alpha} \in \mathbb{Z}^n$ such that $f^\circ(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\alpha}} f(\boldsymbol{X})$. Since g and h are prime with $\boldsymbol{X}^{\boldsymbol{\alpha}}$, g^μ divides f and $\mathrm{mult}_g(f) \geq \mu$. This concludes the proof. □

3.2. **Computing the set of directions.** The goal of this section is to compute, given $f \in \mathbb{K}[X_1, \dots, X_n]$, the set $\Delta_0(f) \subset \mathbb{Z}^n$ of directions δ such that f has a unidimensional factor of direction δ. More precisely, we are going to compute an approximation of $\Delta_0(f)$, that is a set Δ that contains $\Delta_0(f)$. We give several algorithms with distinct and often incomparable complexities, that compute different approximations of $\Delta_0(f)$.

Let $f, g \in \mathbb{K}[X_1, \dots, X_n]$ such that g is unidimensional of direction δ and divides f. Then the Newton polytope of f has two parallel edges of direction δ. This follows from Corollary 2.2. More precisely, $\mathrm{Newt}(f)$ can in this case be partitioned into line segments of direction δ, none of which in reduced to a single point. This motivates the definition of three supersets of $\Delta_0(f)$.

Definition 3.4. Let $f \in \mathbb{K}[X_1, \dots, X_n]$. We define three sets $\Delta_1(f)$, $\Delta_2(f)$, $\Delta_3(f) \subset \mathbb{Z}^n$ as follows:

- $\delta \in \Delta_1(f)$ if the support of f can be partitioned into line segments of direction δ, none of which is reduced to a single point;
- $\delta \in \Delta_2(f)$ if $\mathrm{Newt}(f)$ has two parallel edges of direction δ;
- $\delta \in \Delta_3(f)$ if the support of f has two points $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ such that $\boldsymbol{\alpha} - \boldsymbol{\beta}$ has direction δ.

Lemma 3.5. *Let $f \in \mathbb{K}[X_1, \dots, X_n]$. of sparsity k. Then*

$$\Delta_0(f) \subseteq \Delta_1(f) \subseteq \Delta_2(f) \subseteq \Delta_3(f)$$

and $|\Delta_3(f)| \leq \binom{k}{2}$.

Proof. The first two inclusions were proved above, and follow from Corollary 2.2. The last inclusion simply comes from the fact that an edge of $\mathrm{Newt}(f)$ connects two points of the support of f. And the bound on the cardinality of $\Delta_3(f)$ follows from the same observation since there are at most $\binom{k}{2}$ pairs of points in the support of f. □

We now give algorithms to compute $\Delta_1(f)$, $\Delta_2(f)$ and $\Delta_3(f)$, beginning with the easiest to compute.

Lemma 3.6. *Given $f \in \mathbb{K}[X_1, \ldots, X_n]$ of sparsity k and total degree D, one can compute $\Delta_3(f)$ in $O(k^2 n \mathsf{M}(D) \log(D))$ bit-operations.*

Proof. The algorithm is straightforward: For each pair of exponent vectors $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ of f, one computes the direction of $(\boldsymbol{\alpha} - \boldsymbol{\beta})$. There are $\binom{k}{2}$ such pairs to consider. Computing the direction of a vector $(\boldsymbol{\alpha} - \boldsymbol{\beta})$ reduces to some arithmetic operations and gcd computations on $O(n)$ integers of size at most $\log(D)$.

It remains to detect collisions in the directions. One can to this end sort the directions using some total order, say lexicographic. This can be computed in $O(k^2 n \log(D))$ operations using Radix Sort, since there are $O(k^2)$ vectors of n integers of size at most $\log(D)$. The conclusion follows since $\log(D) \leq \mathsf{M}(D)$. □

We turn to the computation of $\Delta_2(f)$. This can be done without computing first $\Delta_3(f)$ in order to avoid considering all the $\binom{k}{2}$ pairs of points.

Lemma 3.7. *Given $f \in \mathbb{K}[X_1, \ldots, X_n]$ of sparsity k and total degree D, one can compute $\Delta_2(f)$ in $O(k^{\lfloor n/2 \rfloor} n \mathsf{M}(D) + k^2 n \mathsf{M}(D) \log(D))$ bit-operations.*

Proof. One can use Proposition 2.3 to compute the Newton polytope of f. The output of such an algorithm gives a list of facets, from which one can extract the edges. We simply have the return the set $\Delta_2(f)$ of directions $\boldsymbol{\delta}$ such that there are two distinct edges of direction $\boldsymbol{\delta}$ in $\mathrm{Newt}(f)$. The cost is $O(k^{\lfloor n/2 \rfloor} n \mathsf{M}(D))$ to compute the Newton polytope, and $O(k^2 n \mathsf{M}(D) \log(D))$ to compute the directions of the at most $\binom{k}{2}$ edges. □

Computing $\Delta_2(f)$ is thus expensive. Though, if $n = 3$ for instance, $\mathrm{Newt}(f)$ cannot have more than $O(k)$ edges and the cost become linear in k.

We now turn to the computation of $\Delta_1(f)$. We propose two approaches. In the first one, one computes $\Delta_3(f)$ (or $\Delta_2(f)$) and extract $\Delta_1(f)$ from it by removing the directions $\boldsymbol{\delta}$ such that the support of f cannot be partitioned into line segments of direction $\boldsymbol{\delta}$. The second one is direct.

Lemma 3.8. *Let $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k$ and $\boldsymbol{\delta} \in \mathbb{Z}^n$, $\|\boldsymbol{\alpha}_1\|_\infty, \ldots, \|\boldsymbol{\alpha}_k\|_\infty, \|\boldsymbol{\delta}\|_\infty \leq D$. One can compute a partition of $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k\}$ into line segments of direction $\boldsymbol{\delta}$ in time $O(nk\mathsf{M}(D))$.*

Proof. Let $\mathcal{H}$ denote the hyperplane whose normal vector is $\boldsymbol{\delta}$. Two points $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}_j$ belong to the same line segment of direction $\boldsymbol{\delta}$ if

and only if their projections onto $\mathcal{H}$ coincide. The projections can be computed using a dot product: The projection of $\boldsymbol{\alpha}$ onto $\mathcal{H}$ is given by

$$\boldsymbol{\alpha} - \frac{(\boldsymbol{\alpha} \cdot \boldsymbol{\delta})}{\|\boldsymbol{\delta}\|_2^2}\boldsymbol{\delta}$$

where $(\boldsymbol{\alpha} \cdot \boldsymbol{\delta}) = \sum_i \alpha_i \delta_i$ is the dot (or scalar) product of $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$. This projection can be computed in time $O(n\mathsf{M}(D))$ using arithmetic operations on the coordinates of the vectors. Therefore, one can compute the projection of each $\boldsymbol{\alpha}_j$ onto $\mathcal{H}$ in time $O(nk\mathsf{M}(D))$.

It remains to detect the collisions between these projections as in Lemma 3.6.

Altogether, a partition of $\{\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_k\}$ into line segments can be computed in $O(nk\mathsf{M}(D))$ bit-operations. □

Lemma 3.9. *Given $f \in \mathbb{K}[X_1, \dots, X_n]$ of sparsity k and total degree D, one can compute $\Delta_1(f)$ in $O(k^2n^2\mathsf{M}(D)\log(D))$ bit-operations.*

Proof. As mentioned before, the first strategy is to use Lemma 3.6 to compute $\Delta_3(f)$ and then for each $\boldsymbol{\delta} \in \Delta_3(f)$, to check whether the support of f can be partitioned into line segments of direction $\boldsymbol{\delta}$ using Lemma 3.8. This takes $O(k^2n\mathsf{M}(D)\log(D) + k^3n\mathsf{M}(D))$ bit-operations. Similarly, computing first $\Delta_2(f)$ using Lemma 3.7 and refining it to obtain $\Delta_1(f)$ takes $O(k^{\lfloor n/2 \rfloor}n\mathsf{M}(D) + k^2n\mathsf{M}(D)\log(D) + k^3n\mathsf{M}(D))$.

Let us now turn to the second approach. We consider projections of $\mathrm{Newt}(f)$ onto two-dimensional planes. More precisely, for two distinct variables X_i and X_j, let us consider f as an element of $\mathbb{L}_{ij}[X_i, X_j]$ where $\mathbb{L}_{ij} = \mathbb{K}[\boldsymbol{X} \setminus \{X_i, X_j\}]$ is the ring of polynomials in the other variables. If $g \in \mathbb{K}[X_1, \dots, X_n]$ is a unidimensional polynomial of direction $\boldsymbol{\delta}$ with $\delta_i \neq 0$, it is still unidimensional when seen as an element of $\mathbb{L}_{ij}[X_i, X_j]$ for all j, and its direction is (δ_i, δ_j). This means that if g divides f, the support of f viewed as an element of $\mathbb{L}_{ij}[X_i, X_j]$ can be partitioned into line segments of direction (δ_i, δ_j). Thus if f can be partitioned into line segments of direction $\boldsymbol{\delta}$, then for all i and j such that $(\delta_i, \delta_j) \neq (0, 0)$ the support of $f \in \mathbb{L}_{ij}[X_i, X_j]$ can be partitioned into line segments of direction (δ_i, δ_j). Let us define for all (i, j) the set $\Delta_1^{ij}(f) \subset \mathbb{Z}^2$ corresponding to $f \in \mathbb{L}_{ij}[X_i, X_j]$. The set $\Delta_1(f)$ can be computed as follows:

1: Compute $\Delta^{ij}(f)$ for $1 \leq i < j \leq n$;
2: $\Delta_1(f) \leftarrow \{\boldsymbol{\delta} \neq 0 : \forall i, j, (\delta_i, \delta_j) \in \Delta^{ij}(f) \cup \{(0, 0)\}\}$;
3: **return** $\Delta_1(f)$.

To analyze the complexity of this algorithm, first note that each $\Delta^{ij}(f)$ can be computed in time $O(k\mathsf{M}(D)\log(D))$. Then even though the size of each $\Delta^{ij}(f)$ can be linear in k, the size of $\Delta_1(f)$ is at most

quadratic since each pair of monomials of f defines at most one direction δ. Therefore, the total complexity of the algorithm is bounded by $O(k^2n^2\mathsf{M}(D)\log(D))$. □

3.3. **Computing unidimensional factors.** This section is devoted to an algorithm to compute the unidimensional factors of direction δ of a lacunary polynomial f, as soon as one has an algorithm for factoring lacunary univariate polynomials. One first computes the δ-components of f, then their projections, and then the set $\mathcal{F}_\pi$ of common factors of these projections, with multiplicities. The set $\mathcal{F}$ of factors of f is then obtained by lifting the elements of $\mathcal{F}_\pi$ in direction δ. Next lemma shows that the complexity of this strategy is roughly speaking the complexity of the underlying univariate factorization algorithm.

Lemma 3.10. *Let $f \in \mathbb{K}[X_1, \ldots, X_n]$ be a polynomial of total degree D and sparsity k, and $\delta \in \mathbb{Z}^n$ a direction with $\|\delta\|_\infty \leq D$.*

- *The δ-components of f can be computed in $O(kn\mathsf{M}(D))$ bit-operations;*
- *If f is unidimensional of direction δ, its projection can be computed in $O(kn\mathsf{M}(D))$ bit-operations;*
- *If $g \in \mathbb{K}[Z]$ has degree $\leq D$ and sparsity ℓ, its lifting in direction δ can be computed in $O(\ell n\mathsf{M}(D))$.*

Proof. The complexity of computing the δ-components is directly given by Lemma 3.8. Projection and lifting are computed using arithmetic functions on the components of the vectors, whence the same bound. □

Altogether, this proves that if one has an algorithm for computing factors of lacunary univariate polynomials, one has an algorithm for computing unidimensional factors of lacunary multivariate polynomials. Yet, such algorithms are usually not known. We give here a more formal description of the algorithm for computing the bounded-degree unidimensional factors of a lacunary polynomial with coefficients in a number field, since in this case Lenstra gives an algorithm to compute the degree-d factors of lacunary univariate polynomials [24].

Theorem 3.11. *Given an irreducible polynomial $\varphi \in \mathbb{Q}[\xi]$ in dense representation, a polynomial $f \in \mathbb{K}[X_1, \ldots, X_n]$, where $\mathbb{K} = \mathbb{Q}[\xi]/\langle\varphi\rangle$, given in lacunary representation, and a multidegree bound $(d_1, \ldots, d_n) \in \mathbb{N}^n$, one can compute the unidimensional factors of multidegree at most $(d_1, \ldots, d_n)$ of f in deterministic time $\mathsf{poly}(\mathsf{size}(f) + \max_i d_i)$.*

Proof. The algorithm is as follows.

1: Compute $\Delta_1(f)$; ▷ Lemma 3.9
2: $\mathcal{F} \leftarrow \emptyset$;

for all $\delta \in \Delta_1(f)$ **do**
  $\{f^1, \dots, f^s\} \leftarrow \delta$-components of f; ▷ Lemma 3.10
  $d \leftarrow \min_{1 \le i \le n}(d_i/|\delta_i|)$;
  for $t = 1$ to s **do**
    $f^t_\pi \leftarrow$ projection of f^t in direction δ;
    $\mathcal{F}^t_\pi \leftarrow$ degree-d factors of f^t_π; ▷ Lenstra's algorithm
    $\mathcal{F}^t \leftarrow$ the set of liftings in direction δ of elements of $\mathcal{F}^t_\pi$;
  end for
  $\mathcal{F} \leftarrow \mathcal{F} \cup \bigcap_t \mathcal{F}^t$;
end for
return $\mathcal{F}$.

The correctness and the complexity of this algorithm follow from the complexity and the correctness of Lenstra's algorithm, and from the lemmas of this section. □

4. Multidimensional factors

In this section, we focus on multidimensional factors. Their computation is based on a Gap Theorem. This Gap Theorem follows from a bound on the valuation of an expression $f(X, \phi(X))$ where f is a lacunary polynomial and $\phi \in \overline{\mathbb{K}}\langle\langle X \rangle\rangle$ cancels a low-degree polynomial. This bound is given in Sec. 4.1. The Gap Theorem is stated for bivariate polynomials in Sec. 4.2 and yields an algorithm for bivariate polynomials which consists in reducing the computation to several factorizations of low-degree polynomials. To obtain the result on multivariate polynomials, we actually simplify the bivariate algorithm in Sec. 4.3. The resulting algorithm has a higher complexity but extends in a straightforward way to any number of variables. Actually, it reads as follows: Given $f \in \mathbb{K}[X_1, \dots, X_n]$ and $d_1, \dots, d_n$, there exists a bound δ depending only on $k, d_1, \dots, d_n$ such that if we partition $f = f_1 + \cdots + f_s$ so that each f_t has degree at most δ in each variable, each multidimensional polynomial of multidegree at most $(d_1, \dots, d_n)$ satisfies $\mathrm{mult}_g(f) = \min_t(\mathrm{mult}_g(f_t))$. Therefore, as soon as we have a factorization algorithm for multivariate polynomials over a field $\mathbb{K}$ that runs in time polynomial in the degree of the input, this gives an algorithm to compute the bounded-degree multidimensional factors of multivariate polynomials over $\mathbb{K}$ in time polynomial in the lacunary size of the input and in the degree bounds.

4.1. Bound on the valuation.

The goal of this section is to bound the valuation of an expression $f(X, \phi(X))$ where f is a lacunary polynomial of sparsity at least 2 and ϕ a Puiseux series with a low-degree minimal polynomial.

To express the bound, let us define

$$\gamma_v(\ell) = 4d_X d_Y(\ell-1)^2 - \frac{1}{2}(\ell-1)((3\ell-4)d_X + v\ell)$$

where the dependency of $\gamma_v(\ell)$ on d_X and d_Y is not explicitly stated since these quantities shall not vary in the following. Note that if v denotes the valuation of a root of a polynomial of bidegree at most (d_X, d_Y), it is bounded in absolute value by d_X. Thus, $(3\ell-4)d_X + v\ell \geq (2\ell-4)d_X$. Furthermore, $\ell \geq 2$ shall denote the sparsity of f, whence $\gamma_v(\ell) \leq 4d_Xd_Y(\ell-1)^2$ for all v. We define $\gamma(\ell) = 4d_Xd_Y(\ell-1)^2$, so that $\gamma_v(\ell) \leq \gamma(\ell)$ for all $|v| \leq d_X$.

The bound reads as follows.

Theorem 4.1. *Let $\phi \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$ of valuation v and $g \in \mathbb{K}[X,Y]$ its minimal polynomial of bidegree (d_X, d_Y).*

Let $f = \sum_{j=1}^{\ell} c_j X^{\alpha_j} Y^{\beta_j}$ be a polynomial with exactly ℓ terms, and suppose that the family $(X^{\alpha_j}\phi(X)^{\beta_j})_{1\leq j\leq \ell}$ is linearly independent over $\mathbb{K}$.

Then

$$\operatorname{val}\big(f(X,\phi(X))\big) \leq \min_{1\leq j\leq \ell}(\alpha_j + v\beta_j) + \gamma_v(\ell).$$

The highest order term of the bound in the theorem is $4d_xd_Y\ell^2$. This is provably not tight since for $d_X = d_Y = 1$, one can prove that a bound $\ell^2 + O(\ell)$ holds [7].

We state and prove a series of lemmas to finally get a proof of the theorem. Note that some of these lemmas were proven in a slightly different form by Koiran, Portier and Tavenas [19]. Our formulations are more precise and distinguish in particular the degrees in X and Y.

Lemma 4.2. *Let $\phi \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$, $h \in \mathbb{K}[X,Y]$ and $H(X) = h(X,\phi(X))$. For all u and v, let $H_{X^uY^v}(X) = (\partial^{u+v}h/\partial X^u Y^v)(X,\phi(X))$. Then for all $k \geq 1$,*

$$H^{(k)} = \phi^{(k)}H_Y + \sum_{\substack{i_1+\cdots+i_p\leq k\\ 1\leq i_j<k}} c_{\boldsymbol{i}}\phi^{(i_1)}\cdots\phi^{(i_p)}H_{X^{k-i}Y^p}$$

where the sum ranges over all tuples $\boldsymbol{i} = (i_1,\ldots,i_p)$, $p \geq 0$, with $1 \leq i_j < k$ for all j and $i = i_1 + \ldots + i_p \leq k$.

Proof. For a tuple $\boldsymbol{i}$, let $T^k_{\boldsymbol{i}} = \phi^{(i_1)}\cdots\phi^{(i_p)}H_{X^{k-i}Y^p}$. We aim to prove by induction on k that $H^{(k)} = \phi^{(k)}H_Y + \sum_{\boldsymbol{i}} c_{\boldsymbol{i}}T^k_{\boldsymbol{i}}$. Note that the empty tuple corresponding to $p = 0$ is allowed, in which case we have the term $c_\emptyset H_{X^k}$.

For $k = 1$, the chain rule implies $H' = \phi' H_Y + H_X = 0$ and the result holds with $c_\emptyset = 1$.

To compute $H^{(k+1)}$, let us first consider

$$(\phi^{(k)}H_Y)' = \phi^{(k+1)}H_Y + \phi^{(k)}(\phi' H_{Y^2} + H_{XY}).$$

This gives us the term $\phi^{(k+1)}H_Y$ and two terms of the sum, namely $T^{k+1}_{(k,1)} = \phi^{(k)}\phi' H_{Y^2}$ and $T^{k+1}_{(k)} = \phi^{(k)}H_{XY}$. Now the product rule

applied to $T_{\boldsymbol{i}}^k$ shows that $(T_{\boldsymbol{i}}^k)'$ is a sum of terms, in which either $H_{X^iY^j}$ or some $\phi^{(i_j)}$ has been derived. In the first case, since $H'_{X^iY^j} = \phi' H_{X^iY^{j+1}} + H_{X^{i+1}Y^j}$ by the chain rule, we obtain the sum of two terms $T_{\boldsymbol{i}_1}^{k+1} + T_{\boldsymbol{i}}^{k+1}$ where $\boldsymbol{i}_1 = (i_1, \dots, i_p, 1)$. In the second case, we get the term $T_{\boldsymbol{i}'}^{k+1}$ where $\boldsymbol{i}'$ is obtained from $\boldsymbol{i}$ by replacing the j-th component i_j by $i_j + 1$. Altogether

$$(T_{\boldsymbol{i}}^k)' = \sum_{j=1}^{p} T_{(i_1,\dots,i_j+1,\dots,i_p)}^{k+1} + T_{(i_1,\dots,i_p,1)}^{k+1} + T_{(i_1,\dots,i_p)}^{k+1}.$$

This proves the lemma since $H^{(k+1)} = (\phi^{(k)} H_Y)' + \sum_{\boldsymbol{i}} c_{\boldsymbol{i}} (T_{\boldsymbol{i}}^k)'$. □

Lemma 4.3. *Let $\phi \in \overline{\mathbb{K}}\langle\langle X \rangle\rangle$ with minimal polynomial $g \in \mathbb{K}[X, Y]$ of bidegree (d_X, d_Y). Then*

$$\phi^{(k)}(X) = \frac{r_k}{g_Y^{2k-1}}(X, \phi(X)) \tag{1}$$

where $g_Y = \partial g / \partial Y$ and $r_k \in \mathbb{K}[X, Y]$ satisfies

$$\begin{cases} \deg_X(r_k) \leq (2k-1)d_X - k & \text{and} \\ \deg_Y(r_k) \leq (2k-1)d_Y - 2(k-1). \end{cases}$$

.

Proof. Let $G(X) = g(X, \phi(X))$ and for all nonnegative i and j, let $G_{X^iY^j}(X) = (\partial^{i+j} g / \partial X^i Y^j)(X, \phi(X))$. By definition $G(X) = 0$, hence $G^{(k)}(X) = 0$ for all $k \geq 0$. Using Lemma 4.2, we obtain $\phi^{(k)} = -\sum_{\boldsymbol{i}} c_{\boldsymbol{i}} T_{\boldsymbol{i}}^k / G_Y$ where $T_{\boldsymbol{i}}^k = \phi^{(i_1)} \cdots \phi^{(i_p)} G_{X^{k-i}Y^p}$. Let us prove the lemma by induction on k. For $k = 1$, $\phi' = -G_X / G_Y$ so the lemma holds. Let us assume that the lemma for all $j < k$, and consider a term $T_{\boldsymbol{i}}^k$. Let $R_k(X) = r_k(X, \phi(X))$ for all k. Then

$$\begin{aligned} T_{\boldsymbol{i}}^k &= \phi^{(i_1)} \cdots \phi^{(i_p)} G_{X^{k-i}Y^p} \\ &= \frac{R_{i_1}}{G_Y^{2i_1-1}} \cdots \frac{R_{i_p}}{G_Y^{2i_p-1}} G_{X^{k-i}Y^p} \qquad \text{(by induction)} \\ &= \frac{1}{G_Y^{2k-1}} R_{i_1} \cdots R_{i_p} G_{X^{k-i}Y^p} G_Y^{2(k-i)+p-2}. \end{aligned}$$

Let $r_{k,\boldsymbol{i}} = r_{i_1} \cdots r_{i_p} g_{X^{k-i}Y^p} g_Y^{2(k-i)+p-2}$ and $r_k = \sum_{\boldsymbol{i}} c_{\boldsymbol{i}} r_{k,\boldsymbol{i}}$. To conclude, it is enough to bound the degree of r_k.

By induction, we have $\deg_X(r_i) \leq (2i-1)d_X - i$ and $\deg_Y(r_i) \leq (2i-1)d_Y - 2(i-1)$. Whence

$$\begin{aligned}\deg_X(r_{k,\boldsymbol{i}}) &\leq \sum_{j=1}^{p}((2i_j-1)d_X - i_j) \\ &\qquad + (d_X - k + i) + (2(k-i) + p - 2)d_X \\ &\leq (2i-p)d_X - i + d_X - (k-i) + (2(k-i)+p-2)d_X \\ &\leq (2k-1)d_X - k\end{aligned}$$

and $\deg_Y(r_{k,\boldsymbol{i}}) \leq (2k-1)d_X - 2(k-1)$ with a similar computation. □

Lemma 4.4. *Let $\phi \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$ with minimal polynomial $g \in \mathbb{K}[X,Y]$ of bidegree (d_X, d_Y). Let $\psi(X) = X^\alpha \phi(X)^\beta$ for some integer $\alpha, \beta \geq k$. Then*

$$\psi^{(k)}(X) = X^{\alpha-k}\phi(X)^{\beta-k}\frac{s_k}{g_Y^{2k-1}}(X, \phi(X))$$

where $g_Y = \partial g/\partial Y$ and s_k satisfies

$$\begin{cases}\deg_X(s_k) \leq (2k-1)d_X & \text{and} \\ \deg_Y(s_k) \leq (2k-1)d_Y - (k-1).\end{cases}$$

Proof. By Lemma 4.2 with $h(X,Y) = X^\alpha Y^\beta$,

$$\psi^{(k)} = \phi^{(k)}X^\alpha\beta\phi^{\beta-1} + \sum_{\boldsymbol{i}}\tilde{c}_{\boldsymbol{i}}\phi^{(i_1)}\cdots\phi^{(i_p)}X^{\alpha-k+i}\phi^{\beta-p}$$

where $\tilde{c}_{\boldsymbol{i}} = (\alpha)_{k-i}(\beta)_p c_{\boldsymbol{i}}$ for all $\boldsymbol{i}$, and X denotes by a slight abuse of notation the identity polynomial. As previously, let $G_Y(X) = g_Y(X, \phi(X))$ and $R_k(X) = r_k(X, \phi(X))$ for all k. By Lemma 4.3,

$$\phi^{(k)}X^\alpha\beta\phi^{\beta-1} = \frac{X^{\alpha-k}\phi^{\beta-k}}{G_Y^{2k-1}}\left(\beta R_k X^k \phi^{k-1}\right)$$

and

$$\begin{aligned}&\phi^{(i_1)}\cdots\phi^{(i_p)}X^{\alpha-k+i}\phi^{\beta-p} \\ &= \frac{1}{G_Y^{2i-p}}R_{i_1}\cdots R_{i_p}X^{\alpha-k+i}\phi^{\beta-p} \\ &= \frac{X^{\alpha-k}\phi^{\beta-k}}{G_Y^{2k-1}}\left(R_{i_1}\cdots R_{i_p}X^i\phi^{k-p}G_Y^{2(k-i)+p-1}\right).\end{aligned}$$

The function

$$s_k = \beta X^k Y^{k-1}r_k + \sum_{\boldsymbol{i}}\tilde{c}_{\boldsymbol{i}}X^iY^{k-p}r_{i_1}\cdots r_{i_p}g_Y^{2(k-i)+p-1}$$

satisfies the lemma and it only remains to bound its degree.

To this end, a simple computation shows that

$$\begin{aligned}\deg_X(s_k) &\leq \max_i\big(i+(2i-p)d_X - i + (2(k-i)+p-1)d_X\big)\\ &\leq (2k-1)d_X\end{aligned}$$

and similarly $\deg_Y(s_k) \leq (2k-1)d_Y - (k-1)$. □

Lemma 4.5. *Let $\phi \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$ with minimal polynomial $g \in \mathbb{K}[X,Y]$ of bidegree (d_X, d_Y). For $1 \leq j \leq \ell$, let $\psi_j(X) = X^{\alpha_j}\phi(X)^{\beta_j}$ where α_j and β_j are nonnegative integers. If the family $(\psi_j)_j$ is linearly independent, then*

$$\operatorname{wr}(\psi_1,\dots,\psi_\ell) = X^{A-\binom{\ell}{2}}\phi(X)^{B-\binom{\ell}{2}}\frac{t_\ell(X,\phi(X))}{g_Y^{(\ell-1)^2}(X,\phi(X))} \tag{2}$$

where $A = \sum_j \alpha_j$, $B = \sum_j \beta_j$ and $t_\ell \in \mathbb{K}[X,Y]$ satisfies

$$\begin{cases}\deg_X(t_\ell) \leq (\ell-1)^2 d_X & \text{and}\\ \deg_Y(t_\ell) \leq (\ell-1)^2 d_Y - \binom{\ell-1}{2}.\end{cases}$$

Proof. Let us first assume that $\alpha_j, \beta_j \geq \ell$. The wronskian can be expressed using Leibniz formula for the determinant. If we write $W = (W_{jk})_{1\leq j,k\leq \ell}$ the wronskian matrix of the family $(\psi_j)_j$, each term in the wronskian is a product of the form $W_{1,k_1}\cdots W_{\ell,k_\ell}$ where $\{k_1,\dots,k_\ell\} = \{1,\dots,\ell\}$. Moreover $W_{jk} = \psi_j^{(k-1)}$ for all j and k.

By Lemma 4.4,

$$\psi_j^{(k-1)} = X^{\alpha_j-k+1}\phi(X)^{\beta_j-k+1}s_{k-1}(X,\phi(X))/g_Y^{2k-3}(X,\phi(X))$$

for $k \geq 2$. (Note the in the above expression, s_{k-1} actually depends on j.) Thus each term in the wronskian has the form

$$X^{\sum_j \alpha_j - \binom{\ell}{2}}\phi(X)^{\sum_j \beta_j - \binom{\ell}{2}}\frac{\tilde{t}_\ell}{g_Y^{(l-1)^2}}(X,\phi(X))$$

where $\tilde{t}_\ell = \prod_{k\geq 2} s_{k-1}$. Thus

$$\deg_X(\tilde{t}_\ell) \leq \sum_{k=1}^{\ell-1}(2k-1)d_X = (\ell-1)^2 d_X$$

and $\deg_Y(\tilde{t}_\ell) \leq (\ell-1)^2 d_Y - \binom{\ell-1}{2}$.

To conclude, t_ℓ is defined as the sum of the $\tilde{t}_\ell$'s and the degree bounds still hold.

To remove the assumption $\alpha_j, \beta_j \geq \ell$, we apply the above proof to the family $(\chi_j)_j$ where $\chi_j = X^\ell \phi^\ell \psi_j$ and we use the fact that $\operatorname{wr}(\chi_1,\dots,\chi_\ell) = X^{\ell^2}\phi(X)^{\ell^2}\operatorname{wr}(\psi_1,\dots,\psi_\ell)$. □

Lemma 4.6. *Let $\phi \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$ with minimal polynomial $g \in \mathbb{K}[X,Y]$ of bidegree (d_X, d_Y). Let $h(X,Y)$ be a polynomial of bidegree (δ_X, δ_Y). Then*

$$|\mathrm{val}(h(X,\phi(X)))| \leq d_X\delta_Y + \delta_X d_Y.$$

Proof. Let us consider the resultant

$$r(X,Y) = \mathrm{res}_Z(g(X,Z), Y - h(X,Z)).$$

Then $r(X,h(X,\phi(X))) = 0$ since both $g(X,Z)$ and $h(X,\phi(X)) - h(X,Z)$ vanish when $Z = \phi(X)$.

Let us now consider the degree of r in X. The coefficients of $g(X,Z)$ viewed as a polynomial in Z have degree at most d_X in X by definition. In the Sylvester matrix, δ_Y rows are made of the coefficients of g since $Y - h(X,Z)$ has degree δ_Y in Z. In the same way, the Sylvester matrix contains d_Y rows with the coefficients of $Y - h(X,Z)$, each of which has degree at most δ_X in X. Altogether, each term in the resultant has degree at most $d_X\delta_Y + \delta_X d_Y$ in X.

We have shown that $h(X,\phi)$ is a Puiseux series which cancels a polynomial r of degree at most $(d_X\delta_Y + \delta_X d_Y)$ in X. By Newton-Puiseux Theorem, this quantity also bounds the absolute value of its valuation. □

Proof of Theorem 4.1. Let W be the wronskian of the family of Puiseux series $(X^{\alpha_1}\phi^{\beta_1}, \ldots, X^{\alpha_\ell}\phi^{\beta_\ell})$ and $F(X) = f(X,\phi(X))$. Without loss of generality, let us assume that $\min_j(\alpha_j + v\beta_j)$ is attained for $j = 1$.

Since F is a linear combination of the family $(X^{\alpha_j}\phi(X)^{\beta_j})_j$, the wronskian W_F of the family $(F, X^{\alpha_2}\phi^{\beta_2}, \ldots, X^{\alpha_\ell}\phi^{\beta_\ell})$ satisfies $W_F = a_1 W$ and their valuations coincide. By Lemma 2.7,

$$\mathrm{val}(W_F) \geq \mathrm{val}(F) + \sum_{j>1}(\alpha_j + v\beta_j) - \binom{\ell}{2}.$$

On the other hand, since the family $(X^{\alpha_j}\phi^{\beta_j})_j$ is linearly independent, there exists a nonzero t_ℓ such that

$$W = X^{A-\binom{\ell}{2}}\phi^{B-\binom{\ell}{2}}\frac{t_\ell(X,\phi)}{g_Y^{(\ell-1)^2}(X,\phi)}$$

according to Lemma 4.5. Moreover $\mathrm{val}(t_\ell(X,\phi)) \leq 2d_Xd_Y(\ell-1)^2 - d_X\binom{\ell-1}{2}$ and $\mathrm{val}(g_Y(X,\phi)) \geq -2d_Xd_Y + d_X$ by Lemma 4.6. Therefore,

$$\mathrm{val}(W) \leq A - \binom{\ell}{2} + vB - v\binom{\ell}{2} + \frac{1}{2}d_X(\ell-1)(8d_Y(\ell-1) - 3\ell + 4).$$

Since $A = \sum_j \alpha_j$ and $B = \sum_j \beta_j$,

$$\mathrm{val}(F) \leq \alpha_1 + v\beta_1 - v\binom{\ell}{2} + \frac{1}{2}d_X(\ell-1)(8d_Y(\ell-1) - 3\ell + 4).$$

The theorem follows, since $\alpha_1 + v\beta_1 = \min_j(\alpha_j + v\beta_j)$. □

4.2. **The bivariate case.** In this section we state a Gap Theorem that derive from Theorem 4.1 for bivariate polynomials, and deduce an algorithm to compute the multidimensional factors of bounded bidegree of a lacunary polynomial.

In order to simplify the exposition, we shall use the bound $\gamma_v(\ell) \leq \gamma(\ell)$, valid as soon as $|v| \leq d_X$. Using $\gamma_v(\ell)$ instead of $\gamma(\ell)$ yields only slightly better results at the price of much more cumbersome proofs.

Theorem 4.7 (Gap Theorem). *Let $v \in \mathbb{Q}$, $d_X, d_Y \in \mathbb{N}$ and $f = f_1 + f_2 \in \mathbb{K}[X,Y]$ where*

$$f_1 = \sum_{j=1}^{\ell} c_j X^{\alpha_j} Y^{\beta_j} \text{ and } f_2 = \sum_{j=\ell+1}^{k} c_j X^{\alpha_j} Y^{\beta_j}$$

satisfy $\alpha_j + v\beta_j \leq \alpha_{j+1} + v\beta_{j+1}$ for $1 \leq j < k$. If ℓ is the smallest index such that

$$\alpha_{\ell+1} + v\beta_{\ell+1} > \alpha_1 + v\beta_1 + \gamma(\ell)$$

then for every irreducible polynomial of bidegree at most (d_X, d_Y) such that g has a root of valuation v in $\overline{\mathbb{K}}\langle\langle X\rangle\rangle$,

$$\operatorname{mult}_g(f) = \min(\operatorname{mult}_g(f_1), \operatorname{mult}_g(f_2)).$$

Proof. We first prove that under the assumptions of the theorem, g divides f if and only if it divides both f_1 and f_2. In a second part, we shall obtain the full statement on the multiplicities.

Consider a polynomial g as in the theorem, and $\phi \in \overline{\mathbb{K}}\langle\langle X\rangle\rangle$ such that $g(X, \phi(X)) = 0$. If g divides f, $f(X, \phi(X)) = 0$. Actually, the converse holds since g is irreducible. Let us assume that g does not divide f_1. Using the same remark, this means that $f_1(X, \phi(X)) \neq 0$. Let us consider the family $(X^{\alpha_j}\phi^{\beta_j})_{1\leq j\leq \ell}$. One can extract a basis $(X^{\alpha_{j_t}}\phi^{\beta_{j_t}})_{1\leq t\leq m}$ of this family and rewrite

$$f_1(X, \phi(X)) = \sum_{t=1}^{m} b_t X^{\alpha_{j_t}} \phi(X)^{\beta_{j_t}}$$

where $b_1, \dots, b_m$ are linear combinations of $c_1, \dots, c_\ell$. Without any loss of generality, less us assume that $b_t \neq 0$ for all t. Since a basis is a linearly independent family by definition, Theorem 4.1 applies and

$$\operatorname{val}(f_1(X, \phi(X))) \leq \min_t(\alpha_{j_t} + v\beta_{j_t}) + \gamma(m).$$

By minimality of ℓ, $\alpha_{j_t} + v\beta_{j_t} \leq \alpha_1 + v\beta_1 + \gamma(j_t - 1)$ for all t. Since $j_t + m - 1 \leq \ell$ and $\gamma(\ell_1) + \gamma(\ell_2) \leq \gamma(\ell_1 + \ell_2)$ for all ℓ_1 and ℓ_2,

$$\operatorname{val}(f_1(X, \phi(X))) \leq \alpha_1 + v\beta_1 + \gamma(\ell).$$

This means that by assumption $\operatorname{val}(f_2(X, \phi(X))) > \operatorname{val}(f_1(X, \phi(X)))$, whence $f_1(X, \phi(X)) + f_2(X, \phi(X)) \neq 0$. In other words, if g does not divide f_1, it does not divide f either.

To obtain the statement on the multiplicities, consider the p-th derivatives of f, f_1 and f_2. Then $\mathrm{mult}_g(f) > p$ if and only if g divides $f^{(p)}$. Let us assume without loss of generality that $\alpha_j, \beta_j > p$ for all j. (For one can multiply f by X^pY^p without changing its irreducible factors but the multiplicity of X and Y as factors of f.) Now $f^{(p)}$ has the same form as f, and can be written $f_1^{(p)} + f_2^{(p)}$. Furthermore, the condition on the exponents in the lemma is satisfied in $f^{(p)}$ if and only if it is satisfied in f since it is based on the difference of the exponents. This means that for all p, g divide $f^{(p)}$ if and only if it divides both $f_1^{(p)}$ and $f_2^{(p)}$. The conclusion follows. □

In order to avoid any misunderstanding, we formalize what it means for a polynomial f to have a gap relative to a valuation v.

Definition 4.8. Let $v \in \mathbb{Q}$, d_X, $d_Y \in \mathbb{N}$ and $f = \sum_{j=1}^{k} c_j X^{\alpha_j} Y^{\beta_j}$ such that $\alpha_j + v\beta_j \leq \alpha_{j+1} + v\beta_{j+1}$ for $1 \leq j < k$. We say that f *has no gap relative to* v if for $1 < \ell \leq k$,

$$\alpha_\ell + v\beta_\ell \leq \alpha_1 + v\beta_1 + \gamma(\ell - 1).$$

Otherwise, f *has a gap relative to* v.

The Gap Theorem can be used to partition an input polynomial f into a sum $f_1 + \cdots + f_s$ such that for all irreducible polynomial g of bidegree at most (d_X, d_Y) with a root of valuation v, $\mathrm{mult}_g(f) = \min_t(\mathrm{mult}_g(f_t))$. Graphically, this partition corresponds to a partition of the support of f into oblique stripes, each of which has width $\gamma(\ell_t)$ where ℓ_t is the number of points of the support of f it contains. This is the algorithm Partition (Algorithm 1).

Lemma 4.9. *If f has sparsity k and total degree D, and $d_X, d_Y \leq D$, the algorithm* Partition(f, d_X, d_Y, v, σ) *runs in time $O(k \log D)$ and outputs a partition $f_1 + \cdots + f_s$ of f such that* $\mathrm{mult}_g(f) = \min_{1 \leq t \leq s}(\mathrm{mult}_g(f_t))$ *for all polynomial g such that the lower hull of* $\mathrm{Newt}(g)$ *contains an edge of direction (p, q) with $v = -p/q$.*

Proof. The correctness of the algorithm is a direct consequence of the Gap Theorem: Indeed, g has a root of valuation $v = -p/q$ in this case. The complexity of the algorithm is bounded by $O(k \log D)$ since there are only comparisons of integers of size at most $\log(D)$. □

Alone, this partition bounds neither the degree in X nor the degree in Y of each f_t. Using the graphical interpretation again, the support of f is partitioned into stripes that have finite width but are infinite though. The idea is then to use another set of stripes, not parallel to the first ones, to partition again the support (or equivalently f). Since the stripes are not parallel, this will partition the support into parallelograms which are indeed finite.

Algorithm 1 PARTITION(f, d_X, d_Y, v, σ)

Input: $f = \sum_{j=1}^{k} c_j X^{\alpha_j} Y^{\beta_j}$, d_X, $d_Y \in \mathbb{N}$, $v \in \mathbb{Q}$ and σ a permutation such that $j \mapsto \alpha_{\sigma(j)} + v\beta_{\sigma(j)}$ is non-decreasing;
Output: A partition $f = f_1 + \cdots + f_s$.

```
t ← 1;
j_m ← 1;
for ℓ = 2 to k do
    if α_{σ(ℓ)} + vβ_{σ(ℓ)} > α_{σ(j_m)} + vβ_{σ(j_m)} + γ(ℓ − j_m) then
        f_t ← Σ_{j=j_m}^{ℓ−1} c_{σ(j)} X^{α_{σ(j)}} Y^{β_{σ(j)}};
        j_m ← ℓ + 1;
        t ← t + 1;
    end if
end for
f_t ← Σ_{j=j_m}^{k} c_{σ(j)} X^{α_{σ(j)}} Y^{β_{σ(j)}};
return {f_1, ..., f_t}.
```

There comes multidimensionality. If g is multidimensional, its Newton polygon has by definition two non-parallel edges. Let us assume that these two edges belong to the lower hull of the Newton polygon. Then g has a root of valuation v_1 and another one of valuation $v_2 \neq v_1$ where v_1 and v_2 are determined by the directions of the two non-parallel edges. One can partition f with respect to v_1 and then each summand in the partition can be again partitioned, this time with respect to v_2. This yields the algorithm BIPARTITION (Algorithm 2).

Lemma 4.10. *If f has sparsity k and total degree D, and $d_X, d_Y \leq D$, the algorithm* BIPARTITION(f, d_X, d_Y, v_1, v_2) *runs in time $O(k^2 \log D)$ and outputs a partition $f_1 + \cdots + f_s$ such that*

$$\operatorname{mult}_g(f) = \min_{1 \leq t \leq s} (\operatorname{mult}_g(f_t))$$

for all multidimensional polynomial g such that the lower hull of $\operatorname{Newt}(g)$ *contains two edges of directions (p_1, q_1) and (p_2, q_2) with $v_1 = -p_1/q_1$ and $v_2 = -p_2/q_2$.*

Furthermore, the convex size of each f_t is at most $O(d_X^2 d_Y^2 \ell_t^4)$ where ℓ_t is the sparsity of f_t.

Proof. Again, the correctness of this algorithm directly follows from the correctness of the algorithm of Lemma 4.9, that is ultimately from the Gap Theorem. To estimate its complexity, note first that the total number of monomials in $\mathcal{S}$ remains constant, equal to k, during the computation. At each iteration of the while loop, the procedure PARTITION is called on polynomials whose total number of

Algorithm 2 BIPARTITION(f, d_X, d_Y, v_1, v_2)

Input: $f = \sum_{j=1}^{k} c_j X^{\alpha_j} Y^{\beta_j}$, d_X, $d_Y \in \mathbb{N}$, v_1, $v_2 \in \mathbb{Q}$;
Output: A partition $f = f_1 + \cdots + f_s$.

for $i = 1, 2$ **do**
  $\sigma_i \leftarrow$ permutation such that $j \mapsto \alpha_{\sigma_i(j)} + v_i \beta_{\sigma_i(j)}$ is non-decreasing;
end for
$\mathcal{S} \leftarrow \{f\}$ and $s \leftarrow 0$;
while $|\mathcal{S}| > s$ **do**
  $s \leftarrow |\mathcal{S}|$;
  for all $h \in \mathcal{S}$ **do**
    $\mathcal{S}_h \leftarrow$ PARTITION$(h, d_X, d_Y, v_1, \sigma_1)$;
    for all $h' \in \mathcal{S}_h$ **do**
      $\mathcal{S}_{h'} \leftarrow$ PARTITION$(h', d_X, d_Y, v_2, \sigma_2)$;
    end for
    $\mathcal{S}_h \leftarrow \bigcup_{h' \in \mathcal{S}_h} \mathcal{S}_{h'}$;
  end for
  $\mathcal{S} \leftarrow \bigcup_{h \in \mathcal{S}} \mathcal{S}_h$;
end while
return $\mathcal{S}$.

monomials is $2k$. Thus the complexity of this algorithm is $O(k^2 \log D)$ since the sorting phase can be also performed within this complexity bound.

The bound on the convex size comes from the fact that at the end of the algorithm, there is no gap anymore in any f_t. Thus, the support of each f_t is contained in two stripes of widths bounded by $\gamma(\ell_t)$, that is in a parallelogram of area $\gamma(\ell_t)^2 = 16 d_X^2 d_Y^2 (\ell_t - 1)^4$. □

From the previous lemma, we obtain a reduction to low-degree factorization for bivariate polynomials. Note that we do not state any degree bound in the next theorem (such bounds are given in the next section) but rather a bound on the convex size of the output polynomials. To really have an algorithm to compute bounded-degree factors of bivariate polynomials, one can branch any bivariate factorization algorithm. In order to get the best complexity bounds, one can preprocess the output polynomials before their factorization with the techniques of Berthomieu and Lecerf [3]. This allows to compute the irreducible factorization of a polynomial in time polynomial in the convex size rather than the degree of the polynomial. This is particularly interesting in our settings when the support of the input polynomial is partitioned into very *flat* parallelograms, that is parallelograms of large dimensions but small area.

Theorem 4.11. *Let $f \in \mathbb{K}[X,Y]$ of sparsity k and total degree D and d_X, $d_Y \in \mathbb{N}$ be degree bounds. One can reduce the computation of the multidimensional bidegree-(d_X, d_Y) factors of f to the irreducible factorization of at most k polynomials of convex size $O(d_X^2 d_Y^2 k^4)$ in time $\mathsf{poly}(k, \log D, d_X, d_Y)$.*

Proof. Note first that the Newton polygon of a multidimensional factor g has two non-parallel edges. There are three cases to handle: Either the lower hull of $\mathrm{Newt}(g)$ has two non parallel-edges, or its upper hull has two parallel edges, or $\mathrm{Newt}(g)$ has two vertical edges. These three cases are treated separately.

Let us first sketch an algorithm which computes a set $\mathcal{S}$ of polynomials of convex size at most $O(d_X^2 d_Y^2 k^4)$ adapted to polynomials g such that the lower hull of $\mathrm{Newt}(g)$ has two non-parallel edges. For, let us suppose that we have a procedure $\textsc{Gcd}(\mathcal{S})$ that given a set of polynomials computes their greatest common divisor. The algorithm is as follows:

1: $\mathcal{S} \leftarrow \emptyset$;
2: $\Delta \leftarrow$ the directions of the edges in the lower hull of $\mathrm{Newt}(f)$;
3: $\Delta \leftarrow \Delta \cap \{(p,q) : p \le d_X, |q| \le d_Y\}$;
4: **for all** $(p_1,q_1),(p_2,q_2) \in \Delta$ **do**
5: $\quad \mathcal{S}_{p_1,q_1,p_2,q_2} = \textsc{Bipartition}(f, d_X, d_Y, -p_1/q_1, -p_2/q_2)$
6: $\quad h \leftarrow \textsc{Gcd}(\mathcal{S}_{p_1,q_1,p_2,q_2})$;
7: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup h$;
8: **end for**
9: **return** $\mathcal{S}$.

Let g be a factor of f of bidegree at most (d_X, d_Y), and suppose that the lower hull of its Newton polygon has two non-parallel edges of directions (p_1,q_1) and (p_2,q_2). Using Corollary 2.2, the lower hull of the Newton polygon of f also has two non-parallel edges of directions (p_1,q_1) and (p_2,q_2). That is, $(p_1,q_1),(p_2,q_2) \in \Delta$. Therefore the correctness of Bipartition implies that the factor g is a common factor of the polynomials of $\mathcal{S}_{p_1,q_1,p_2,q_2}$, and thus a factor of the corresponding h. Conversely, the correctness of Bipartition also implies that each computed factor is a factor of f. To obtain exactly the factors of bidegree at most (d_X, d_Y), it simply remains to remove the higher-degree factors from $\mathcal{F}$.

For the complexity of the algorithm, there are at most $O(k)$ edges in Δ, whence at most $O(k^2)$ iterations of the loop. The call to Bipartition takes polynomial time. Now, using the techniques of Berthomieu and Lecerf [3], one can compute the gcd of $\mathcal{S}$ in time polynomial in the convex size of the elements of $\mathcal{S}$. This convex size is bounded by $O(d_X^2 d_Y^2 k^4)$ according to Lemma 4.10.

It remains to prove that one can give similar algorithms for the factors of f that have two non-parallel edges in the upper hull of the Newton polygon, or two vertical edges. For the first remaining case, one can simply consider $f^X(X,Y) = Y^{\deg_X(f)} f(1/X, Y)$ and apply the previous algorithm to f^X. Since f has two non-parallel edges in its upper hull, one easily sees that f^X has two non-parallel edges in its lower hull. One can reconstruct the factor of f from the factors of f^X since $\mathrm{mult}_g(f) = \mathrm{mult}_{g^X}(f^X)$ for all g, and $(g^X)^X = g$ for an irreducible polynomial g (or actually for any g of valuation 0 in X).

The last case is a bit more different, though the algorithm is actually slightly simpler. One has to modify the algorithm Bipartition. Since we do not have two distinct valuations to call Partition, one has to replace the second call to Partition at line 10 of Bipartition. For, let us define $\bar{f}(X,Y) = f(Y,X)$ and $\bar{g}(X,Y) = g(Y,X)$ for any g. Clearly, $\mathrm{mult}_{\bar{g}}(\bar{f}) = \mathrm{mult}_g(f)$ and if g has two vertical edges, $\bar{g}$ has two horizontal edges. This means that one can replace the call $\textsc{Partition}(h', d_X, d_Y, v_2, \sigma_2)$ at line 10 by $\textsc{Partition}(\bar{h}', d_X, d_Y, 0, \sigma_0)$ where σ_0 is the permutation such that $\beta_{\sigma_0(j)} \leq \beta_{\sigma_0(j+1)}$ for all j. The correctness of this algorithm to find the factors of g having two vertical edges once again follows from the Gap Theorem. The complexity and the size bounds remain valid in this case. □

4.3. **The multivariate case.** In this section, we aim to give a generalization of the algorithm of the preceding section to multivariate polynomials. Actually, the algorithm is a simplification of the previous one that can be used for bivariate polynomials as well. Yet the price for the simplicity is an increase of complexity.

The first step is a new analysis of the algorithm Bipartition. We have given a bound on the convex size of the polynomials in the output. For our simplified algorithm, we need a bound on their degree. As mentioned earlier, the output of Bipartition is a partition of the support of the input polynomial into parallelograms. The area of these parallelograms is bounded by $\gamma(\ell)^2$ where ℓ is the number of points of the support of the input polynomial contained in the parallelogram. Our aim is to bound their coordinates.

Lemma 4.12. *Let (p_1, q_1) and (p_2, q_2) be the directions of two non-parallel edges in the lower hull of the Newton polygon of some polynomial of bidegree (d_X, d_Y), and let $v_1 = p_1/q_1$ and $v_2 = p_2/q_2$. Then*

$$\frac{1}{|v_1 - v_2|} \leq d_Y^2/4 \text{ and } \frac{|v_1| + |v_2|}{|v_1 - v_2|} \leq d_X d_Y.$$

Proof. Since (p_1, q_1) and (p_2, q_2) are the directions of two edges in the lower hull of some Newton polygon, there exist $\lambda, \mu \in \mathbb{N}$ such that $\lambda|q_1| + \mu|q_2| \leq d_Y$, whence $|q_1| + |q_2| \leq d_Y$. And for similar reasons, $|p_1|, |p_2| \leq d_X$.

Since $|q_1| + |q_2| \leq d_Y$, $|q_1 q_2| \leq |q_1|(d_Y - |q_1|) \leq (d_Y/2)^2$. Thus

$$\frac{1}{|v_1 - v_2|} = \frac{|q_1 q_2|}{|p_1 q_2 - p_2 q_1|} \leq |q_1 q_2| \leq \frac{d_Y^2}{4}.$$

Similarly, $|p_1 q_2| + |p_2 q_1| \leq d_X(|q_1| + (d_Y - |q_1|)) \leq d_X d_Y$ and

$$\frac{|v_1| + |v_2|}{|v_1 - v_2|} = \frac{|p_1 q_2| + |p_2 q_1|}{|p_1 q_2 - p_2 q_1|} \leq d_X d_Y.$$

□

Lemma 4.13. *Let d_X, $d_Y \in \mathbb{N}$, v_1, $v_2 \in \mathbb{Q}$ and $f \in \mathbb{K}[X,Y]$. Let $\mathcal{S} = \textsc{Bipartition}(f, d_X, d_Y, v_1, v_2)$. Then for all $h \in \mathcal{S}$ of sparsity ℓ,*

$$\begin{cases} \deg_X(h) - \mathrm{val}_X(h) \leq \frac{1}{2} d_Y^2 \gamma(\ell) & \text{and} \\ \deg_Y(h) - \deg_Y(h) \leq d_X d_Y \gamma(\ell). \end{cases}$$

Proof. If $h \in \mathcal{S}$, there exists no gap in h with respect to either v_1 or v_2. Let us write $h = \sum_{j=1}^{\ell} c_j X^{\alpha_j} Y^{\beta_j}$. Then for all j,

$$\begin{cases} \alpha_j + v_1 \beta_j \leq \min_{1 \leq j \leq \ell}(\alpha_j + v_1 \beta_j) + \gamma(\ell) & \text{and} \\ \alpha_j + v_2 \beta_j \leq \min_{1 \leq j \leq \ell}(\alpha_j + v_2 \beta_j) + \gamma(\ell). \end{cases}$$

In particular, for all p and q, and $i = 1, 2$,

$$(\alpha_p - \alpha_q) + v_i(\beta_p - \beta_q) \leq \gamma(\ell).$$

We aim to bound $|\alpha_p - \alpha_q|$ and $|\beta_p - \beta_q|$ for all p and q. Let us fix some p and some q, and let $\Delta_\alpha = \alpha_p - \alpha_q$ and $\Delta_\beta = \beta_p - \beta_q$. Since the above bound is valid if we exchange p and q, we have $|\Delta_\alpha + v_i \Delta_\beta| \leq \gamma(\ell)$ for $i = 1, 2$.

Now $\Delta_\alpha + v_1 \Delta_\beta - (\Delta_\alpha + v_2 \Delta_\beta) = (v_1 - v_2)\Delta_\beta \leq 2\gamma(\ell)$. Thus,

$$|\Delta_\beta| \leq \frac{2\gamma(\ell)}{|v_1 - v_2|} \leq \frac{1}{2} d_Y^2 \gamma(\ell)$$

according to Lemma 4.12.

To bound Δ_α, first note that $v_2 \Delta_\alpha + v_1 v_2 \Delta_\beta \leq |v_2| \gamma(\ell)$ and $v_1 \Delta_\alpha + v_1 v_2 \Delta_\beta \geq -|v_1| \gamma(\ell)$. Thus $(v_2 - v_1)\Delta_\alpha \leq (|v_1| + |v_2|)\gamma(\ell)$ and

$$|\Delta_\alpha| \leq \frac{|v_1| + |v_2|}{|v_1 - v_2|} \gamma(\ell) \leq d_X d_Y \gamma(\ell)$$

again using Lemma 4.12. □

Until now, we have obtained for each pair of distinct valuations (v_1, v_2) a partition of f with the desired properties. We aim to invert the quantifiers, that is to prove that there exists a partition that has the desired properties with respect to any pair of valuations. Or otherwise stated, we aim to prove that one can partition $f = f_1 + \cdots + f_s$ such that any multidimensional factor g of bidegree at most (d_X, d_Y) satisfy $\mathrm{mult}_g(f) = \min_t(\mathrm{mult}_g(f_t))$.

Lemma 4.14. *Let $f \in \mathbb{K}[X,Y]$ of sparsity k and total degree D, d_X, $d_Y \in \mathbb{N}$. There exists a partition $f = f_1 + \cdots + f_s$ such that for any multidimensional polynomial g of bidegree at most (d_X, d_Y),*

$$\mathrm{mult}_g(f) = \min_{1 \le t \le s} (\mathrm{mult}_g(f_t)).$$

Furthermore, for $1 \le t \le s$

$$\deg_Y(f_t) - \mathrm{val}_Y(f_t) \le k d_X d_Y \gamma(k).$$

This partition can be computed in time $O(k \log D)$.

Proof. We give an algorithm to compute the partition $f = f_1 + \cdots + f_s$ and then prove it has the required properties.

Let us write $f = \sum_{j=1}^{k} c_j X^{\alpha_j} Y^{\beta_j}$ such that $\beta_j \le \beta_{j+1}$ for $1 \le j < k$.

$t \leftarrow 1$;
$j_m \leftarrow 1$;
for $\ell = 1$ to $k-1$ **do**
  if $\beta_{\ell+1} - \beta_\ell > d_X d_Y \gamma(k)$ **then**
    $f_t \leftarrow \sum_{j=j_m}^{\ell} c_j X^{\alpha_j} Y^{\beta_j}$;
    $j_m \leftarrow \ell + 1$;
    $t \leftarrow t + 1$;
  end if
end for
return $\{f_1, \ldots, f_t\}$.

Let us first note that the bound on $\deg_Y(f_t) - \mathrm{val}_Y(f_t)$ is straightforward since the sparsity of f_t is bounded by k. The complexity is dominated by the cost of sorting the exponents, and this cost is $O(k \log D)$.

Let s be the final value of t in the above algorithm. We claim that at the end of the algorithm, the partition $f = f_1 + \cdots + f_s$ satisfies $\mathrm{mult}_g(f) = \min_t(\mathrm{mult}_g(f_t))$ for all multidimensional polynomial g of bidegree at most (d_X, d_Y).

First, if the lower hull of $\mathrm{Newt}(g)$ has two non-parallel edges, g has two roots of distinct valuations v_1 and v_2 in $\overline{\mathbb{K}}\langle\langle X \rangle\rangle$. One can call $\textsc{Bipartition}(f, d_X, d_Y, v_1, v_2)$ to compute a partition $f = f_1^1 + \cdots + f_{s^1}^1$. This partition satisfies $\deg_Y(f_t^1) - \mathrm{val}_Y(f_t^1) \le d_X d_Y \gamma(\ell_t^1)$ where ℓ_t^1 is the sparsity of f_t^1 and $\mathrm{mult}_g(f) = \min_t(\mathrm{mult}_g(f_t^1))$. We aim to show that if two monomials $X^{\alpha_p} Y^{\beta_p}$ and $X^{\alpha_q} Y^{\beta_q}$ belong to a same f_t^1 in this partition, they also belong to a same polynomial f_t in the partition computed by the algorithm. Indeed, given the bound on $\deg_Y(f_t^1) - \mathrm{val}_Y(f_t^1)$, we have $|\alpha_p - \alpha_q| \le d_X d_Y \gamma(\ell_t^1) \le d_X d_Y \gamma(k)$ since γ is an increasing function. Let us assume $\alpha_p \ge \alpha_q$. Then, for $q < r \le p$, $\alpha_r - \alpha_{r-1} \le \alpha_p - \alpha_q \le d_X d_Y \gamma(k)$ and α_p, α_{p+1}, …, α_q belong to a same polynomial in the partition $f = f_1 + \cdots + f_t$.

This proves that each f_t is a sum of $f^1_{t'}$'s, hence $\min_t(\mathrm{mult}_g(f_t) = \min_t(\mathrm{mult}_g(f^1_t))$.

A second case concerns the polynomials g such that the upper hull of $\mathrm{Newt}(g)$ has two non-parallel edges. For these polynomials, one can consider $f^X(X,Y) = X^{\deg_X(f)} f(1/X, Y)$ and $g^X(X,Y) = X^{\deg_X(g)} g(1/X, Y)$. It is classical that $\mathrm{mult}_g(f) = \mathrm{mult}_{g^X}(f^X)$. The lower hull of $\mathrm{Newt}(g^X)$ is the symmetric of the upper hull of $\mathrm{Newt}(g)$, thus it has two non-parallel edges. One can thus apply the previous proof to this case too. Since the arguments only use differences of exponents, they are equally valid with f^X.

The last case is a polynomial g whose Newton polygon has at most one edge in both its lower and upper hulls. In such case, $\mathrm{Newt}(g)$ contains two vertical edges (and exactly one edge in both its upper and lower hulls). Let v be the valuation of the root of g in $\overline{\mathbb{K}}\langle\langle X\rangle\rangle$ corresponding to the edge in its lower hull. We can first partition $f = f^1_1 + \cdots + f^1_{s^1}$ using $\textsc{Partition}(f, d_X, d_Y, v, \sigma)$ where σ is a permutation such that $j \mapsto \alpha_j + v\beta_j$ is non-decreasing. Then, we exchange the roles of X and Y and consider $\bar{f}(X,Y) = f(Y,X)$ and $\bar{g}(X,Y) = g(Y,X)$. Clearly, $\mathrm{mult}_{\bar{g}}(\bar{f}) = \mathrm{mult}_g(f)$. Since $\mathrm{Newt}(\bar{g})$ has now an horizontal edge in its lower hull, for each f^1_t we can partition $\bar{f}^1_t$ by calling $\textsc{Partition}(\bar{f}^1_t, d_X, d_Y, 0, \sigma_0)$ where σ_0 is another permutation, adapted to this case. This partition of $\bar{f}^1_t$ corresponds to a partition of f^1_t, and we can again as in $\textsc{Bipartition}$ try to find gaps until we get $f = f^2_1 + \cdots + f^2_{s^2}$ such that no f^2_t contains any gap. Using the Gap Theorem, $\mathrm{mult}_g(f) = \min_t(\mathrm{mult}_g(f^2_t))$. As above, all it remains to show is that if two monomials appear in the same f^2_t, they appear in the same f_t in the partition we compute. For, we aim to bound $\deg_Y(f^2_t) - \mathrm{val}_Y(f^2_t)$ for each t. Let $X^{\alpha_p}Y^{\beta_p}$ and $X^{\alpha_q}Y^{\beta_q}$ two monomials of some f^2_t. Since the valuation is 0 is the call to $\textsc{Partition}$ on $\bar{f}^1_t$, we get that $|\beta_p - \beta_q| \leq \gamma(\ell^2_t)$ where ℓ^2_t is the sparsity of f^2_t. The end of the proof is similar to the previous cases: This bound on $|\alpha_p - \alpha_q|$ implies that the two monomials belong to a same f_t. □

Finally, we get to our simple algorithm. First note that we can replace the bound $\gamma(k)d_Xd_Y$ in the algorithm by any larger value and obtain the same result, but of course with a larger value for $\deg_Y(f_t) - \mathrm{val}_Y(f_t)$. Our aim is to use the above partitioning algorithm with respect to all the variables, sequentially. For ease of presentation, let us reformulate the above algorithm in the settings of a multivariate polynomial (Algorithm 3) before presenting the general algorithm (Algorithm 4).

Algorithm 3 UNIVARIATEPARTITION(f, δ, i)

Input: $f = \sum_{j=1}^{k} c_j X_1^{\alpha_{1,j}} \cdots X_n^{\alpha_{n,j}}$, δ and i;
Output: $\{f_1, \ldots, f_s\}$ such that $\deg_{X_i}(f_t) - \mathrm{val}_{X_i}(f_t) \leq k\delta$.

$\sigma \leftarrow$ permutation such that $j \mapsto \alpha_{\sigma(j)} + v\beta_{\sigma(j)}$ is non-decreasing;
$t \leftarrow 1$;
$j_m \leftarrow 1$;
for $\ell = 1$ to $k-1$ **do**
  if $\alpha_{i,\sigma(\ell+1)} - \alpha_{i,\sigma(\ell)} > \delta$ **then**
    $f_t \sum_{j=j_m}^{\ell} c_{\sigma(j)} X_1^{\alpha_{1,\sigma(j)}} \cdots X_n^{\alpha_{n,\sigma(j)}}$;
    $j_m \leftarrow \ell + 1$;
    $t \leftarrow t + 1$;
  end if
end for
$f_t \leftarrow \sum_{j=j_m}^{k} c_{\sigma(j)} X_1^{\alpha_{1,\sigma(j)}} \cdots X_n^{\alpha_{n,\sigma(j)}}$;
return $\{f_1, \ldots, f_t\}$.

Algorithm 4 MULTIVARIATEPARTITION$(f, d_1, \ldots, d_n)$

Input: $f \in \mathbb{K}[X_1, \ldots, X_n]$, $d_1, \ldots, d_n \in \mathbb{N}$;
Output: $\{f_1, \ldots, f_s\}$.

$\mathcal{S} \leftarrow \{f\}$;
$s \leftarrow 0$;
while $s < |\mathcal{S}|$ **do**
  $s \leftarrow |\mathcal{S}|$;
  for $i = 0$ to n **do**
    for all $h \in \mathcal{S}$ **do**
      $k \leftarrow$ sparsity of h;
      $\delta \leftarrow \gamma(k) d_i \max_{i' \neq i}(d_{i'})$;
      $\mathcal{S}_h \leftarrow$ UNIVARIATEPARTITION(f, δ, i);
    end for
    $\mathcal{S} \leftarrow \bigcup_{h \in \mathcal{S}} \mathcal{S}_h$;
  end for
end while
return $\{h / \boldsymbol{X}^{\mathrm{mval}(h)} : h \in \mathcal{S}\}$.

Theorem 4.15. *If $f \in \mathbb{K}[X_1, \ldots, X_n]$ has sparsity k and $d_1, \ldots, d_n$ are positive integers, the algorithm* MULTIVARIATEPARTITION$(f, d_1, \ldots, d_n)$ *runs in time $O(nk^2 \log D)$ and outputs a partition $f_1 + \cdots + f_s$ of f such that each f_t has degree at most $O(d^4 k^3)$ in each variable where $d = \max_i(d_i)$ and for any multidimensional polynomial $g \in \mathbb{K}[X_1, \ldots, X_n]$ of*

multidegree at most $(d_1, \dots, d_n)$,

$$\mathrm{mult}_g(f) = \min_{1 \le t \le s} (\mathrm{mult}_g(f_t)).$$

Proof. The correctness of the algorithm and the degree bound follow from Lemma 4.14. For the complexity, note that at each iteration of the while loop, the size of $\mathcal{S}$ increases by at least 1, and the final size is bounded by k. This proves that this loop terminates in at most k iterations. The global complexity follows from the complexity of UnivariatePartition, given in Lemma 4.14. □

References

[1] M. Avendano, T. Krick, and M. Sombra. Factoring bivariate sparse (lacunary) polynomials. *Journal of Complexity*, 23(2):193–216, 2007.

[2] E. R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46(8):1853–1859, 1967.

[3] J. Berthomieu and G. Lecerf. Reduction of bivariate polynomials from convex-dense to dense, with application to factorizations. *Mathematics of Computation*, 81(279):1799–1821, 2012.

[4] J. Bi, Q. Cheng, and J. M. Rojas. Sub-Linear Root Detection, and New Hardness Results, for Sparse Polynomials Over Finite Fields. In *Proc. ISSAC'13*. ACM, 2013. arXiv:1204.1113.

[5] A. Bostan and P. Dumas. Wronskians and linear independence. *Am. Math. Mon.*, 117(8):722–727, 2010.

[6] A. Chattopadhyay, B. Grenet, P. Koiran, N. Portier, and Y. Strozecki. Computing the multilinear factors of lacunary polynomials without heights. Manuscript (submitted), 2013. arXiv:1311.5694.

[7] A. Chattopadhyay, B. Grenet, P. Koiran, N. Portier, and Y. Strozecki. Factoring bivariate lacunary polynomials without heights. In *Proc. ISSAC'13*, pages 141–158, 2013. arXiv:1206.4224.

[8] G. Chèze and A. Galligo. Four lectures on polynomial absolute factorization. In A. Dickenstein and I. Z. Emiris, editors, *Solving Polynomial Equations*, volume 14 of *Algorithms Comput. Math.*, pages 339–392. 2005.

[9] F. Cucker, P. Koiran, and S. Smale. A polynomial time algorithm for Diophantine equations in one variable. *J. Symb. Comput.*, 27(1):21–30, 1999.

[10] A. Fröhlich and J. Shepherdson. On the factorisation of polynomials in a finite number of steps. *Mathematische Zeitschrift*, 62(1):331–334, 1955.

[11] M. Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009.

[12] B. Grenet. Computing low-degree factors of lacunary polynomials: a Newton-Puiseux approach. In *Proc. ISSAC'14*, pages 224–231. ACM, 2014.

[13] D. Harvey, J. van der Hoeven, and G. Lecerf. Even faster integer multiplication. arXiv:1407.3360, 2014.

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

[15] E. Kaltofen and P. Koiran. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In *Proc. ISSAC'05*, pages 208–215. ACM, 2005.

[16] E. Kaltofen and P. Koiran. Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields. In *Proc. ISSAC'06*, pages 162–168. ACM, 2006.

[17] E. Kaltofen, J. P. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *J. Symb. Comput.*, 43(5):359–376, 2008.
[18] M. Karpinski and I. Shparlinski. On the computational hardness of testing square-freeness of sparse polynomials. In *Proc. AAECC-13*, volume 1719 of *LNCS*, pages 731–731. Springer, 1999.
[19] P. Koiran, N. Portier, and S. Tavenas. On the intersection of a sparse curve and a low-degree curve: A polynomial version of the lost theorem. arXiv:1310.2447, 2013.
[20] S. Landau. Factoring polynomials over algebraic number fields. *SIAM Journal on Computing*, 14(1):184–195, 1985.
[21] A. Lenstra, J. Lenstra, H.W., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
[22] A. K. Lenstra. Factoring polynomials over algebraic number fields. In *Computer algebra*, pages 245–254. Springer, 1983.
[23] A. K. Lenstra. Factoring Multivariate Polynomials over Algebraic Number Fields. *SIAM J. Comput.*, 16(3):591–598, 1987.
[24] H. Lenstra Jr. On the factorization of lacunary polynomials. In *Number theory in progress*, pages 277–291. De Gruyter, 1999.
[25] V. Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *Journal of Symbolic Computation*, 33(5):701–733, 2002.
[26] D. Plaisted. Sparse complex polynomials and polynomial reducibility. *J. Comput. Syst. Sci.*, 14(2):210–221, 1977.
[27] M. Sagraloff. A near-optimal algorithm for computing real roots of sparse polynomials. In *Proc. ISSAC'14*, pages 359–366. ACM, 2014.
[28] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7(3-4):281–292, 1971.

LIRMM – Université Montpellier II, UMR 5506 CNRS
E-mail address: Bruno.Grenet@lirmm.fr