FACTORING SPARSE MULTIVARIATE POLYNOMIALS

Joachim von zur Gathen

Department of Computer Science
University of Toronto

Extended Abstract

1. Introduction

The problem of factoring polynomials has a venerable history, and also has seen an exciting burst of recent activity, overshadowed by the milestone result of Lenstra-Lenstra-Lovász [82]. It is now known that univariate polynomials over the rational numbers and over finite fields can be factored in polynomial time - over finite fields with a probabilistic algorithm. The same is true for multivariate polynomials in the dense encoding, so that the running time of the factoring algorithm is polynomial in the length d^n of the dense representation of a polynomial f of degree less than d in each of n variables (Kaltofen [82], Lenstra [83a] for **Q**, Chistov-Grigoryev [82], Lenstra [83], von zur Gathen-Kaltofen [83] for finite fields). Earlier algorithms (Wang-Rothschild [73], Musser [75], Wang [76], Wang [78], Zippel [81]) often perform well in practice, but may not have polynomial worst-case running time.

In this paper, we present a factoring algorithm whose running time is polynomial in the length of a sparse encoding, where f is represented as a list of M monomials, and each monomial is given by the exponent vector, written in unary, and the nonzero coefficient. This is probably the more important measure from a practical point of view.

The main results of this paper are two general theorems and two probabilistic algorithms based on these. The first algorithm is a test for irreducibility; its running time is polynomial in ndM. The second algorithm factors the polynomial; its running time is polynomial in $nd\hat{M}$, where $\hat{M}$ is the number of nonzero terms in f or any of its irreducible factors, i.e. input plus output size. We show in Example 6.A that the output size can be more than polynomial in the input size; thus $\hat{M}$ cannot be replaced by M.

Encoding the exponents in binary, the length of the encoding of f could conceivably be proportional to $\log d$ rather than d. Then already the simplest questions about univariate polynomials, e.g. whether the *gcd* of several polynomials is 1, become *NP*-hard (Plaisted [77]). Thus we have to disallow this ultra-compact encoding.

The algorithms rely on two theoretical results. The first is an effective version of Hilbert's Irreducibility Theorem. We consider substituting $n-2$ variables in $f \in F[x_1,\ldots,x_n]$ by linear expressions in the remaining two variables. Then for an irreducible f and a randomly chosen substitution, the resulting bivariate polynomial is irreducible with high probability. The proof uses methods of algebraic geometry.

A finite ground field may not have enough elements to make our probabilistic algorithms work. The second theoretical result overcomes this obstacle. In general, when one makes algebraic extensions of fields, irreducible polynomials have a tendency to split. We prove the somewhat surprising fact that for certain extensions - easy to describe and arbitrarily large - this does not happen.

2. An effective Hilbert irreducibility theorem

We start with a theorem of algebraic geometry going back to Bertini [1882]. The Bertini theorems come in several flavors. They usually assert that if an algebraic variety (embedded in some affine or projective space) has a certain property, then - under suitable conditions - also the intersection with a general hyperplane has this property. Properties considered include smoothness, normality and the case of interest to us: irreducibility. In the context of algebraic computations, Bertini's theorem has been used by Heintz-Sieveking [81] for testing whether integer polynomials are irreducible over **C**. We will use substitutions by linear functions of the first two variables frequently, and it is convenient to have a notation for them.

0272-5428/83/0000/0172$01.00 © 1983 IEEE

Definition 2.A. If F is a field, $f \in F[x_1,\dots,x_n]$ and $t=(u,v,w)=(u_3,\dots,u_n,v_3,\dots,v_n,w_3,\dots,w_n)\in F^{3(n-2)}$, then

$$f\{t\} = f(x_1,x_2,u_3x_1+v_3x_2+w_3,\dots,$$
$$u_nx_1+v_nx_2+w_n) \in F[x_1,x_2].$$

Theorem 2.B. (Bertini) Let K be an algebraically closed field, $n\ge 2$ and $f\in K[x_1,\dots,x_n]$ irreducible. Then there exists an algebraically closed field L containing K and $t\in L^{3(n-2)}$ such that $f\{t\}\in L[x_1,x_2]$ is irreducible. □

Hilbert [1892] proved in his famous Hilbert Irreducibility Theorem that in an irreducible $f\in\mathbf{Q}[x_1,\dots,x_n]$ one can substitute integers for $x_2,\dots,x_n$ and preserve irreducibility. However, no effective version of precisely this theorem is known. We now present an effective result, involving a more general linear substitution for the variables. The approach to this effective Hilbert irreducibility theorem is as follows. First we consider algebraically closed fields, so that we can apply the powerful methods of algebraic geometry, in particular the Bertini theorem. Within the set of all bivariate polynomials, the reducible ones form a subvariety of small dimension and small degree. We view the substitutions $x_i = u_ix_1+v_ix_2+w_i$ as a morphism to bivariate polynomials, and then the "unlucky substitutions" are contained in the set of zeros of a polynomial of small degree. Now take an arbitrary field F and an irreducible multivariate polynomial f over F. If f factors as $f_1\cdots f_r$ over an appropriate algebraically closed field K containing F, then by the above almost all substitutions $x_i = u_ix_1+v_ix_2+w_i$ leave each f_i irreducible. The Galois group of K over F provides additional conditions that hold almost everywhere and ensure that the substitutions leave f irreducible, which is the desired result. For the case $F=\mathbf{Q}$, Kaltofen [82] has a similar result.

So for a field K and $d\ge 0$, denote by X_d the vector space of polynomials in $K[x,y]$ of degree at most d. X_d has dimension $\alpha_d = \binom{d+2}{2}$. Also let

$$Y_d = \{f\in X_d : f \text{ is reducible}\}.$$

Lemma 2.C. Let K be algebraically closed. Then $Y_d\subseteq X_d$ is a closed proper (reducible) subvariety, and $\deg Y_d < 2^{\alpha_d}$. □

Let $F\subseteq K$ be a normal field extension, $f\in F[x_1,\dots,x_n]$ squarefree, and $f=f_1\cdots f_r$ an irreducible factorization of f in $K[x_1,\dots,x_n]$. Then $G=Gal(K/F)$ operates coefficientwise on $K[x_1,\dots,x_n]$, and G also operates on $T=\{f_1,\dots,f_r\}$.

Proposition 2.D. In this situation, f is irreducible in $F[x_1,\dots,x_n]$ iff G operates transitively on T. □

Theorem 2.E. Let F be an arbitrary field, $n\ge 0$ and $f\in F[x_1,\dots,x_n]$ irreducible and of total degree d. Then there exists a field K containing F and a nonzero polynomial

$$s\in K[U_3,\dots,U_n,V_3,\dots,V_n,W_3,\dots,W_n]$$

of total degree at most $19\cdot 2^{d^2}$ such that for all $t=(u,v,w)\in F^{3(n-2)}$ with $s(t)\ne 0$ the polynomial

$$f\{t\} = f(x_1,x_2,u_3x_1+v_3x_2+w_3,\dots,$$
$$u_nx_1+v_nx_2+w_n)\in F[x_1,x_2]$$

is irreducible. □

Remark 2.F. It would be nice to have an analogue of Theorem 2.E involving only the simple substitutions $x_i = w_i$ with $w_i\in F$. For algebraically closed F this is not always possible, but for the computationally important case of the rational numbers there is still hope for an effective version of Hilbert's irreducibility theorem. For practical purposes (see Remark 6.E), it would be convenient to use only the simple substitutions, but, alas, it is still consistent with our present knowledge - although conjectured to be false - that for all d there exists some irreducible polynomial in $\mathbf{Q}[x,y]$ of degree d and with small coefficients such that all substitutions $y=w$ with $w\in\mathbf{Z}$ and $|w|\le 2^{2^d}$, say, yield a reducible polynomial. The following result gives some strength to the conjecture that an effective version of the Hilbert Irreducibility Theorem may hold over some infinite fields.

Theorem 2.G. Let F be an infinite field, $n\ge 2$ and $d\ge 1$. Then for almost all $f\in F[x_1,\dots,x_n]$ of degree at most d and almost all $(w_3,\dots,w_n)\in F^{n-2}$, the bivariate polynomial

$$f(x_1,x_2,w_3,\dots,w_n)\in F[x_1,x_2]$$

is irreducible. □

3. The irreducibility test

Based on the results of the previous section, we now present a probabilistic polynomial-time irreducibility test for sparse multivariate polynomials. We assume that a (probabilistic) irreducibility test for bivariate polynomials is known. All polynomials are supposed to be given in the "sparse representation", consisting of a list of coefficients and exponent vectors.

Algorithm IRREDUCIBILITY TEST.

Input: A polynomial $f \in F[x_1,\ldots,x_n]$ and a finite set $A \subseteq F$.

Output: Either "irreducible" or "reducible" or "failure".

1. Choose $t = (u_3,\ldots,u_n,v_3,\ldots,v_n,w_3,\ldots,w_n) \in A^{3(n-2)}$ at random, and set
$$g = f\{t\} = f(x_1,x_2,u_3x_1 + v_3x_2 + w_3, \ldots,u_nx_1 + v_nx_2 + w_n) \in F[x_1,x_2].$$
2. If the total degree of g is less than the total degree of f, then return "failure".
3. Use the irreducibility test for bivariate polynomials with input g, and return the answer "irreducible" or "reducible" or "failure" according to the outcome of that test.

In order to estimate the running time and error probability of this algorithm, we assume that the test for a bivariate polynomial $h \in F[x_1,x_2]$ of degree at most d uses not more than $\tau(d)$ operations in F and either returns the correct answer "irreducible" respectively "reducible", or else "failure"; the latter with probability at most ε. We also assume that a random element of A (with respect to the uniform distribution) can be chosen in $O(log(\#A))$ "random bit choices".

Theorem 3.A. Let $f \in F[x_1,\ldots,x_n]$ have total degree d and M nonzero terms, and $A \subseteq F$ with $\#A = 55n\,2^{2d^2}$. Then algorithm IRREDUCIBILITY TEST has the following properties:

(i) If f is irreducible, it returns "irreducible" with probability at least $1-(\varepsilon+2^{-d^2})$.

(ii) If f is reducible, it either returns "reducible" or "failure". The latter happens with probability at most $\varepsilon + 2^{-d^2}$.

(iii) It uses $O(d^3M) + \tau(d)$ operations in F, and $O(n(d^2 + logn))$ random bit choices. ☐

For a number field F, Theorem 3.A gives a random polynomial-time irreducibility test for sparse multivariate polynomials. Let F be presented as $F = \mathbf{Q}[z]/(h)$ with $h \in \mathbf{Q}[z]$ monic irreducible of degree m. As is usual, we write an element of F as a polynomial in z of degree less than m with rational coefficients, and a rational number as a pair of integers. Assume that the integers occurring in h all have absolute value at most H. Now let $f \in F[x_1,\ldots,x_n]$ have total degree d and M nonzero terms, and let B be the largest absolute value of the integers that appear in the representation of the coefficients of f. We then call
$$s(f) = M(dn + mlogB) + logH$$
the size of f. Clearly, the "sparse representation" of f - given by a list of all nonzero coefficients in f - can be represented by a bit string of length $O(s(f))$, and in general $\Omega(M(nd+mlogB))$ bits are necessary to represent f in this way.

Corollary 3.B. Let F be a number field. Irreducibility of a polynomial f over F can be tested in random polynomial time in $s(f)$. ☐

4. Field extensions and factorization

The irreducibility test for multivariate polynomials in the last section assumes that one can make random choices from a sufficiently large finite subset of the ground field. This may not be possible over a small finite field. This section presents a somewhat surprising result, namely that one can make arbitrarily large algebraic extensions of a field without changing the factorization of a given polynomial.

Theorem 4.A. Let F be an arbitrary field, $f \in F[x_1,\ldots,x_n]$ of total degree d, and $F \subseteq K$ a finite algebraic extension of degree m such that $gcd(m,d) = 1$. Then

(i) f irreducible in $F[x_1,\ldots,x_n] \iff f$ irreducible in $K[x_1,\ldots,x_n]$.

(ii) If each prime factor of m is greater than d, then f has the same irreducible factorization over F and K. ☐

Using this theorem, it is now easy to put the irreducibility test of section 3 to work over a finite field F. In order to check a bivariate polynomial g of total degree at most d in step 3 of IRREDUCIBILITY TEST for irreducibility, we use the probabilistic algorithm BIVARIATE FACTORING from von zur Gathen-Kaltofen [83].

Algorithm IRREDUCIBILITY TEST OVER A FINITE FIELD.

Input: A polynomial $f \in F[x_1,\ldots,x_n]$ of total degree d over a finite field F with q elements.

Output: Either "irreducible" or "reducible" or "failure".

1. Set $a = 55n\,2^{2d^2}$. If $q \geq a$, then set $K = F$ and goto step 3. Else set $m = 2d^2 + \lceil log_q 55n \rceil$, and choose a prime number l with $m < l \leq 2m$.
2. Choose monic polynomials $h_1,\ldots,h_{4l} \in F[z]$ of degree l at random, and test them for irreducibility. If none is irreducible, return "failure". Otherwise choose an irreducible h_i, and set $K = F[z]/(h_i)$.
3. Call algorithm IRREDUCIBILITY TEST with input $f \in K[x_1,\ldots,x_n]$ and some $A \subseteq K$ with $\#A = a$.

Theorem 4.B. Let F be a finite field with q elements, and $f \in F[x_1,\ldots,x_n]$ have total degree d and M nonzero terms. With this input, algorithm IRREDUCIBILITY TEST OVER A FINITE FIELD has the following properties:

(i) If f is irreducible, it returns "irreducible" with probability at least $1-2^{-d}$.

(ii) If f is reducible, it either returns "reducible" or "failure". The latter happens with probability at most 2^{-d}.

(iii) Let $k = max\{d, log n, log q\}$. The algorithm can be performed in $O(k^{16} + k^8 M)$ bit operations, and $O(k^8)$ random bit choices. □

5. The factoring algorithm

Our probabilistic algorithm for factoring sparse multivariate polynomials proceeds as follows. First, one makes a random substitution to obtain a bivariate polynomial. We assume existence of a factoring algorithm for bivariate polynomials, and factor the substituted polynomial completely. Then this factorization is lifted variable by variable via a Hensel technique. At each lifting step, one has to solve a system of linear equations. Two points now are crucial to ensure polynomial running time: one has to keep the number of indeterminates (corresponding to monomials in the factors) and the number of equations small throughout the algorithm. The first is fairly straightforward, along the lines of Zippel [79].

Zippel [79] first used this type of approach for an interpolation problem, and he also proposed to employ it for factoring polynomials (Zippel [81]). Although the algorithm seems to work well in practice, for lack of an effective Hilbert irreducibility theorem of the form needed, no bound at all can be proven for the expected time of his (probabilistic) approach. We start with a definition that allows us to concentrate on certain monomials and on certain equations.

Definition 5.A. We write $\mathbf{x}^e$ for the monomial $x_1^{e_1} \cdots x_n^{e_n}$ with $e \in \mathbf{N}^n$. Given a polynomial

$$f = \sum_{e \in \mathbf{N}^n} f_e \mathbf{x}^e \in F[x_1,\ldots,x_n]$$

with $f_e \in F$, we call

$$\mathrm{Supp}(f) = \{e \in \mathbf{N}^n : f_e \neq 0\}$$

the support of f. In our algorithm, the first two variables play a special role, and we will use

$$\mathrm{Supp}_2(f) = \{e \in \mathbf{N}^n : \sum_{1 \le i \le n} e_i < deg\ f \text{ and}$$

$$\exists\, a_1, a_2 \in \mathbf{N} \text{ such that } (a_1, a_2, e_3, \ldots, e_n) \in \mathrm{Supp}(f)\}.$$

Here $deg\ f$ is the total degree of f. For a set $E \subseteq \mathbf{N}^n$ we write

$$f\,|_E = \sum_{e \in E} f_e \mathbf{x}^e$$

for f restricted to E. Thus

$$f\,|_E \equiv 0 \bmod x_n^{k+1}$$

is equivalent to

$f_e = 0$ for $e = (e_1,\ldots,e_n) \in E$ with $e_n \le k$. □

Procedure LIFTING.

Input: $j, k, m_1, \ldots, m_r \in \mathbf{N}$, $h, g_1, \ldots, g_r \in F[x_1,\ldots,x_j,y]$, and $E \subseteq \mathbf{N}^j$ such that $deg_y g_i \le k$ and

$$(h - \prod_{1 \le i \le r} g_i^{m_i})\,|_{E \times \{0,\ldots,k\}} \equiv 0 \bmod y^{k+1}.$$

Output: Either $g_1^*, \ldots, g_r^* \in F[x_1,\ldots,x_j,y]$ such that

$$(h - \prod (g_i^*)^{m_i})\,|_{E \times \{0,\ldots,k+1\}} \equiv 0 \bmod y^{k+2},$$

and, if $k=0$, also some $E^* \subseteq E$, or "failure".

1. Set $S_i = \mathrm{Supp}_2(g_i(x_1,\ldots,x_j,0)) \subseteq \mathbf{N}^j$.
2. The congruence
$$(h - \prod_{1 \le i \le r} (g_i + \sum_{e \in S_i} g_{ie}^* \mathbf{x}^e y^{k+1})^{m_i})\,|_{E \times \{k+1\}} \equiv 0 \bmod y^{k+2}$$
corresponds to a system of $\#E$ many linear equations over F in the $\sum \#S_i$ many unknowns g_{ie}^*. If $k \ge 1$ and the system is not square (square meaning that $\#E = \sum \#S_i$) and nonsingular, return "failure". If $k=0$, find a set $E^* \subseteq E$ such that the $\#E^*$ many linear equations corresponding to $E^* \times \{k+1\}$ form a square nonsingular system. If no such E^* exists, return "failure".
3. Return $g_i^* = g_i + \sum_{e \in S_i} g_{ie}^* \mathbf{x}^e y^{k+1}$, given by the solution of the square nonsingular system in step 2.

Procedure HENSEL.

Input: $f \in F[x_1,\ldots,x_n]$ with $n \ge 2$, $m_1, \ldots, m_r \in \mathbf{N}$, $g_1, \ldots, g_r \in F[x_1, x_2]$ irreducible and

$$t \in F^{3(n-2)} \text{ such that } f\{t\} = \prod_{1 \le i \le r} g_i^{m_i}.$$

Output: Either $f_1, \ldots, f_r \in F[x_1,\ldots,x_n]$ or "failure". We expect that each f_i is irreducible, $f_i\{t\} = g_i$, and $f = \prod_{1 \le i \le r} f_i^{m_i}$.

1. Let d be the total degree of f, and set

$$t = (u_3,\ldots,u_n, v_3,\ldots,v_n, w_3,\ldots,w_n),$$

$$E_{1d} = \{0,\ldots,d\}.$$

```
2. For $j=2$ to $n-1$ do steps 3 through 5.
3. Set
       $h_j = f(x_1,\ldots,x_j,u_{j+1}x_1 + v_{j+1}x_2 + w_{j+1} - y,$
       $u_{j+2}x_1 + v_{j+2}x_2 + w_{j+2},\ldots,u_nx_1 + v_nx_2 + w_n),$
       $g_{ij0} = g_{i,j-1,d}(x_1,\ldots,x_{j-1},u_jx_1 + v_jx_2 + w_j - x_j)$
           for $1 \le i \le r$ (with $g_{ij0} = g_i$ if $j=2$),
       $E_{j0} = \{e \in \mathbf{N}^j : \sum_{1 \le l \le j} e_l \le d$ and
       $\exists a_1,a_2 \in \mathbf{N}$ such that $(a_1,a_2,e_3,\ldots,e_{j-1}) \in E_{j-1,d}\}$
4. For $k=0$ to $d-1$ do step 5.
5. Call procedure LIFTING with input
       $(j,k,m_1,\ldots,m_r,h_j,g_{1jk},\ldots,g_{rjk},E_{jk})$
   to return either $(g_{1,j,k+1},\ldots,g_{r,j,k+1})$ and, if
   $k=0$, also $E_{j1} \subseteq E_{j0}$, or "failure". If $k \ge 1$, set
   $E_{j,k+1} = E_{jk}$.
6. For $1 \le i \le r$, set
       $f_i = g_{i,n-1,d}(x_1,\ldots,x_{n-1},u_nx_1+v_nx_2+w_n-x_n),$
   and return $(f_1,\ldots,f_r)$.
```

The following algorithm assumes a probabilistic method for factoring bivariate polynomials over F, and a finite set $A \subseteq F$ with a procedure for picking elements of A at random (with respect to the uniform distribution).

Algorithm SPARSE FACTORING.

```
Input:  $f \in F[x_1,\ldots,x_n]$ and a finite subset $A \subseteq F$.
Output: Polynomials $f_1,\ldots,f_r \in F[x_1,\ldots,x_n]$, and
        $m_1,\ldots,m_r \in \mathbf{N}$. With high probability,
        $f = f_1^{m_1} \cdots f_r^{m_r}$ is the irreducible fac-
        torization of $f$ in $F[x_1,\ldots,x_n]$.
1. Choose $t \in A^{3(n-2)}$ at random.
2. Compute the irreducible factorization
   $f\{t\} = g_1^{m_1} \cdots g_r^{m_r}$ of $f\{t\}$ in $F[x_1,x_2]$, where
   $m_i \ge 1$, each $g_i$ is irreducible and
   $gcd(g_i,g_j) = 1$ for $i \ne j$.
3. Call procedure HENSEL with input
   $(f,m_1,\ldots,m_r,g_1,\ldots,g_r,t)$, and return the out-
   put $(f_1,\ldots,f_r)$. If "failure" occurs at any
   stage, goto step 1.
```

For the correctness proof, we have to assume that the finite set A is large enough, and we set $\beta_2 = 560+26n$, $\beta_3 = 720$, $\beta_4 = 103$, and $\beta_d = 9$ for $d \ge 5$. In step 2 of SPARSE FACTORING we call a probabilistic procedure for factoring a bivariate polynomial of degree at most d. We assume that this Las Vegas algorithm correctly returns the complete factorization of the input polynomial, and has an expected running time of $O(d^{10})$ operations in F.

Theorem 5.B. Let F be a field, $n \ge 3$, $f \in F[x_1,\ldots,x_n]$ of total degree d and such that the number of nonzero terms in f or any of its irreducible factors is at most M. Assume that no irreducible factor of f occurs with a multiplicity that is an integer multiple of $charF$. Let $A \subseteq F$ with $\#A = \beta_d n^{5d} 2^{d^2}$. With this input, algorithm SPARSE FACTORING has the following properties.

(i) It correctly computes the irreducible factorization of f with probability at least $1 - n^{-3d}$.

(ii) The expected running time is $O(nd^{10}M^3)$ operations in F.

(iii) The expected number of random bit choices is $O(nd(d + logn))$. □

Remark 5.C. We want to point out a somewhat unsatisfactory aspect of the timing and error behavior. In Example 6.A we show that the output size may be more than polynomial in the input size, so that we do not have an a priori time bound (even though we can estimate explicitly all the constants in the time bounds). Now if the substitution is not lucky, then the bivariate factorization of $f\{t\}$ might not reflect the true factorization of f, and the running time could conceivably be more than polynomial in the input plus output size. (We only know that it is $O(d^4n^{3d+1})$.) And for lack of an a priori time bound, we could not even realize this during execution of the algorithm. All we could do in SPARSE FACTORING was to make the probability of this unlucky event so small that the expected running is polynomial.

Also note that there does not seem to be a feasible way of checking deterministically whether the output is correct, i.e. whether $f = \prod_{1 \le i \le r} f_i^{m_i}$ or not. If the substitution was lucky, then this equation holds. However, if the substitution was unlucky, then the algorithm might produce a wrong output for which this equation does not hold. The two obvious methods for checking the equation - multiplying the product out or checking at evaluation points - both may require more than polynomial time. One may of course run a few checks at randomly chosen evaluation points and get arbitrarily large confidence in the correctness (resp. discover incorrectness with large probability).

We spell out the theorem for the case of a number field F, using the notation from Corollary 3.B. For any $f \in F[x_1,\ldots,x_n]$, let $\tilde{M}$ denote the maximal number of terms in f or any of its irreducible factors, and set $\tilde{s}(f) = nmd\tilde{M}\, logB\, logH$. Let $A = \{0,1,\ldots,\beta_d n^{5d} 2^{d^2}\} \subseteq \mathbf{Z} \subseteq F$.

Corollary 5.D. In the above situation, algorithm SPARSE FACTORING has the following properties:

(i) It correctly computes the irreducible factorization of f with probability at least $1 - n^{-3d}$.

(ii) The expected number of bit operations is polynomial in $\tilde{s}(f)$.

(iii) The expected number of random bit choices is $O(nd(d + \log n))$. □

It is not hard to describe an algorithm FACTORING IN POSITIVE CHARACTERISTIC that handles the two cases in characteristic $p > 0$ that were left open in SPARSE FACTORING: a small finite ground field F, and factors with multiplicity m_i, where p divides m_i.

Theorem 5.E. Let F be a finite field with q elements, $n \geq 3$, $f \in F[x_1,...,x_n]$ of total degree d and such that the number of nonzero terms in f or any of its irreducible factors is at most M. With this input, algorithm FACTORING IN POSITIVE CHARACTERISTIC has the following properties:

(i) It correctly computes the irreducible factorization of f with probability at least $1 - n^{-3d}$.

(ii) Let $k = \max\{d, n\}$. The expected number of bit operations is $O(\log^2 q\ (k^{10} \log k \log q + k^{15} M^3))$.

(iii) The expected number of random bit choices is $O(k^4 \log q)$. □

6. Examples and remarks

We first present a few examples that highlight some of the unpleasant phenomena that may occur when factoring sparse multivariate polynomials, then, in section 6.D, parallel versions of the irreducibility test and the factoring algorithm. We make in 6.E some remarks about factoring algorithms of practical interest.

Example 6.A. *Short polynomials with a long irreducible factor.* Let $n \geq 3$ be prime, $f_n(x) = x^{n-1} + x^{n-2} + \cdots + 1 \in \mathbb{Q}[x]$, and

$$g_n = f_n(x_1) f_n(x_2) \cdots f_n(x_n),$$
$$h_n = (x_1 - 1)(x_2 - 1) \cdots (x_n - 1),$$
$$p_n = h_n(g_n + n)$$

in $\mathbb{Z}[x_1,...,x_n] \subseteq \mathbb{Q}[x_1,...,x_n]$. Then g_n has n^n terms, both h_n and $h_n g_n = (x_1^n - 1) \cdots (x_n^n - 1)$ have 2^n terms, and each coefficient of p_n is at most $n+1$ in absolute value. Note that $n^n = (2^n)^{\log n}$ is not polynomial in the number n of variables, the degree n^2 and the number $\leq 2^{n+1}$ of terms in p_n. Also, $g_n + n$ is irreducible, and hence p_n is a polynomial where the number of terms in the irreducible factorization is not polynomial in the input size.

In view of this example, it may be unfeasible to factor even reasonably short polynomials. Erich Kaltofen has proposed the following question: Do the reducible polynomials at least have short certificates?

Example 6.B. *Short polynomials with a long squarefree part.* Using the notation from 6.A, we set

$$r_n = h_n^2 g_n \prod_{1 \leq i \leq n} f_{n-1}(x_i) = \prod_{1 \leq i \leq n} (x_i^n - 1)(x_i^{n-1} - 1).$$

Then r_n has 4^n terms, and the irreducible factorization of r_n has $O(n^2)$ factors, each with at most n terms. The squarefree part

$$s_n = h_n g_n \prod_{1 \leq i \leq n} f_{n-1}(x_i) = \prod_{1 \leq i \leq n} (x_i^n - 1) f_{n-1}(x_i)$$

of r_n has $(2n-2)^n$ terms. This number is not polynomial in the input plus output size (for factoring r_n), and thus no polynomial-time algorithm for factoring r_n can first compute the squarefree part of r_n.

Example 6.C. *Short polynomials with no short monic version.* In factoring algorithms, life gets easier if one reduces to the case of a monic polynomial, by replacing the input polynomial f by $g^{d-1} f(\frac{x}{g})$, where f has degree d and leading coefficient g with respect to the variable x. This reduction is not possible for sparse polynomials. Let $n \geq 3$, and consider the symmetric polynomial

$$f = \sum_{1 \leq i \leq n} x_i^n \,(1 + \sum_{\substack{1 \leq j \leq n \\ j \neq i}} x_j) \in \mathbb{Q}[x_1,...,x_n].$$

f is irreducible and has n^2 nonzero terms, and the monic version of f with respect to x_1 is

$$g^{n-1} f(\frac{x_1}{g}, x_2, ..., x_n)$$
$$= x_1^n + g^{n-2} \sum_{2 \leq i \leq n} x_i^n (g(g - x_i) + x_1),$$

which has more than $\binom{2n-3}{n-2}$ nonzero terms. This is an exponential number, and by the symmetry of f, its monic version with respect to any variable has exponential size.

Remark 6.D. *A parallel version.* We want to describe parallel variants of our algorithm. For our model of parallel computation, we can take parallel algebraic computation graphs (see von zur Gathen [83]), where at each step each processor can execute one arithmetic operation (+,-,*,/, fetching a constant), or one test ($a = 0$?) in the ground field F, or a boolean operation on those test results. (The algorithms also work on algebraic PRAM's.)

Theorem 6.D. Let $n \geq 3$, $d \geq 1$, $b = max\{\beta_d n^{5d} 2^{d^2}, 55n\, 2^{2d^2}\}$, and let F be an arbitrary field with at least b elements. For polynomials in $F[x_1,\ldots,x_n]$ of total degree d with M nonzero terms, and at most $\hat{M}$ terms in either f or any of its irreducible factors, we have

(i) Testing for irreducibility can be reduced probabilistically to testing a bivariate polynomial of degree at most d for irreducibility using parallel time $O(log^2 d + log M)$. The number of processors is $O(d^3 M)$.

(ii) Factoring can be reduced probabilistically to factoring bivariate polynomials of degree at most d. The reduction can be performed in parallel time $O(n\ log d\ log^2(d\hat{M}))$, and also in parallel time $O(d\ log^2(nd\hat{M}))$. In both cases, the number of processors is polynomial in $nd\hat{M}$. □

Over a finite field F of characteristic p and with $q = p^m$ elements, we use the parallel bivariate factoring algorithm from von zur Gathen-Kaltofen [83], and obtain

Corollary 6.D. In the above notation, we have:

(i) The irreducibility test can be performed in

$$O((log^2 d\ log^2(d\ m\ log n)\ log p + log M)\ log^2 log q)$$

parallel bit operations.

(ii) Let $L = log d\ max\{log d\ log log n, log log q\}^2$. The factoring algorithm can be performed in $O(L\,(n\ log^2(nd\hat{M}) + log d\ log p))$ or in $O(L\ log d\ (d\ log^2(n\hat{M}) + log p\ log^2(dm)))$ parallel bit operations. □

Remark 6.E. From a practical point of view, it would be nice if the following head-on approach worked: To factor a polynomial $f \in F[x_1,\ldots,x_n]$, perform some initial simplifications, then select a special variable, substitute randomly chosen constants for the other variables, factor the resulting univariate polynomial, and lift a factorization via a sparse Hensel technique. This factorization might either be complete or into just two factors. Zippel [81] had proposed such an algorithm and concluded that "for all practical purposes it does not exhibit exponential behavior". However, if one wants the worst-case running time to be truly polynomial in the input size, then the following problems arise:

1. The initial simplifications may yield too large intermediate expressions.
   a) See Example 6.B for the squarefree part.
   b) See Example 6.C for the monic version.
2. The substituted polynomial may not reflect the true factorization.
   a) There might be a factor involving only one, say, variable. Thus clearly each variable has to be selected as the special variable at some point in the algorithm.
   b) As pointed out in remark 2.F, it is still conceivable that for some irreducible polynomial all reasonable substitutions give a reducible result. The various trials of combining the fake factors might require exponential time, as was the case with the original Berlekamp-Zassenhaus procedure for univariate integer polynomials.
3. During the lifting process, one has to keep the number of monomials and the number of linear equations small.
   a) If a factorization into two factors is lifted, one of them may be too large, e.g. the factor g_n of $f_n g_n$, in the notation of 6.A.
   b) Zippel [79, 81] suggested how to keep the number of monomials (= unknowns in the linear equations) small; we followed his approach.
   c) In order to get a small, yet restrictive enough, set of linear equations one might search at certain points of the algorithm systematically through all the available equations. It is not clear how to guarantee polynomial time for this approach. Our method maintains a polynomially-sized set of equations throughout the algorithm, from which a restrictive enough subset is chosen from time to time.
   d) The output size may not be polynomial in the input size (Example 6.A). That's nature's ways, and we will have to live with it.

Acknowledgement

It is a pleasure to acknowledge the many conversations I had with Erich Kaltofen about the factoring problem, and in particular a contribution to the proof of Theorem 4.A.

References

E. Bertini, Sui sistemi lineari. **Rend. Reale Ist. Lomb. 15** (1882), 24-28.

A.L. Chistov and D.Yu. Grigoryev, Polynomial-time factoring of the multivariable polynomials over a global field. LOMI preprint E-5-82, Leningrad, 1982.

J. von zur Gathen, Parallel algorithms for algebraic problems. Proc. 15th ACM Symp. Theory of Computing, Boston, 1983, 17-23. To appear in SIAM J. Comput.

J. von zur Gathen and E. Kaltofen, Polynomial-time factorization of multivariate polynomials over finite fields. Proc. 10th ICALP, Barcelona, 1983.

J. Heintz and M. Sieveking, Absolute primality of polynomials is decidable in random polynomial time in the number of variables. Springer Lecture Notes Comp. Sci. **115**, Springer Verlag, 1981, 16-28.

D. Hilbert, Ueber die Irreduzibilität ganzer rationaler Funktionen mit ganzzahligen Koeffizienten. J. f. d. reine u. ang. Math. **110** (1892), 104-129.

E. Kaltofen, A polynomial-time reduction from bivariate to univariate integral polynomial factorization. Proc. 23rd Annual Symp. FOCS, Chicago 1982, 57-64.

A.K. Lenstra, Factoring polynomials over algebraic number fields. Preprint, Mathematisch Centrum, Amsterdam, 1982.

A.K. Lenstra, Factoring multivariate polynomials over finite fields. Proc. 15th ACM Symp. Theory of Computing, Boston, 1983, 189-192.

A.K. Lenstra [83a], Factoring multivariate integral polynomials. Proc. 10th ICALP, Barcelona, 1983.

A.K. Lenstra, H.W. Lenstra, and L. Lovász, Factoring polynomials with rational coefficients. Math. Ann. **261** (1982), 515-534.

D.R. Musser, Multivariate polynomial factorization. J. ACM **22** (1975), 291-308.

D.A. Plaisted, Sparse complex polynomials and polynomial reducibility. J. Comp. and System Sciences **14** (1977), 210-221.

P.S. Wang and L.P. Rothschild, Factoring multivariate polynomials over the integers, SIGSAM Bull. **28** (1973), 21-29.

P.S. Wang, Factoring multivariate polynomials over algebraic number fields, Math. Comp. **30** (1976), 324-336.

P.S. Wang, An improved multivariate polynomial factoring algorithm. Math. Comp. **32** (1978), 1215-1231.

R. Zippel, Probabilistic algorithms for sparse polynomials. Proc. EUROSAM 79, Marseille, 1979, 216-226.

R. Zippel, Newton's iteration and the sparse Hensel algorithm. Proc. 1981 ACM Symp. Symbolic and Algebraic Computation, Utah, 1981, 68-72.