Polynomial Factorization 1982-1986*

Erich Kaltofen

Rensselaer Polytechnic Institute
Department of Computer Science
Troy, New York 12181

1. Introduction

In late 1981 I wrote the survey article [41] on the problem of factoring polynomials for the Computer Algebra book edited by B. Buchberger, G. Collins, and R. Loos. In the conclusion of that survey, I raised six opens problems, all of which were related to finding polynomial-time solutions to problems in polynomial factorization. During the tremendous research effort of the last five years, all six problems have been satisfactorily solved. This paper complements my previous survey and summarizes the major advances that now make polynomial factoring a fine example of a classical computational question for which efficient and polynomial time algorithms have been designed.

Before we describe the new results, however, I want to make a few additions to my discussion on the earlier developments [41]. Von Schubert's finite step algorithm of 1793 for factoring univariate integral polynomial is merely a generalization of Newton's method published in the chapter ''invention of divisors'' in his Arithmetica Universalis in 1707. The idea is simple. In order to find a factor of degree d, one evaluates the polynomial at $d+1$ integers and interpolates all possible combinations of integer factors of the resulting values. Van der Waerden [88] points out that this method is valid over any unique factorization domain with finitely many units, and that then the decidability of factoring in the domain implies the decidability of polynomial factoring. He suggests, however, that over infinite fields the factoring problem may be undecidable given only effective procedures for the field arithmetic (including zero testing). This is indeed so, and Fröhlich and Shepherdson [26] show the existence of such ''explicit'' fields over which irreducibility testing is undecidable. Of course this undecidability result stands in the shadow of the polynomial-time solution for most natural fields such as the rational numbers.

From the work by Zassenhaus [93] and Lenstra [62] in 1981 it became clear that the polynomial-time resolution of univariate integer polynomial factorization hinged on progress for certain diophantine approximation questions, whereas the author's early results on Hilbert

* This material is based upon work supported by the National Science Foundation under Grant No. DCR-85-04391 and by an IBM Faculty Development Award. This article was written for a tutorial on polynomial factorization given at the conference ''Computers and Mathematics,'' at Stanford University on August 2, 1986.

Irreducibility (cf [42]. and [47], §7) indicated that the question of multivariate factoring may be polynomial-time reducible to univariate factoring. The missing link in the univariate polynomial factoring algorithm was finally supplied by Lovász in late 1981, who constructed a remarkable algorithm for finding relatively short vectors in multidimensional integer lattices. The resulting polynomial factoring algorithm by A. K. Lenstra, H. Lenstra, and Lovász [67] could decompose an integer polynomial of degree n with coefficients absolutely bounded by B into irreducible non-constant integer polynomials in $O(n^{12} + n^9(\log B)^3)$ binary steps, this when using classical long integer arithmetic procedures. Polynomial-time reductions from multivariate to univariate integer polynomial factorization were discovered at about the same time. The algorithm by the author [47], is based on Zassenhaus's root approximation scheme, which in those circumstances happens in the power series domain and then leads to a linear system rather than a diophantine approximation problem. This algorithm only requires field arithmetic and a univariate factoring oracle, provided the field has sufficiently many elements. Another algorithm by Christov and Grigoryev [18] uses the Hilbert irreducibility results mentioned earlier and a polynomial-time bivariate polynomial factoring algorithm over finite coefficient fields, which is derived from the lattice approach. In the multivariate case, the number of possible monomial coefficients can grow exponentially with the number of variables. A densely represented, or shortly a dense multivariate polynomial is one where all coefficients are counted towards the input size, even if a large proportion of them is zero. The algorithms just described have polynomial running time in the dense size of the input polynomial.

Other dense multivariate polynomial factoring algorithms over finite fields were found by von zur Gathen and the author [31] and by A. K. Lenstra [65]. For large coefficient fields, the algorithms are randomized in the Las Vegas sense, ''always correct and probably fast.'' Only recently has it been established that multivariate irreducibility testing over large finite fields can be accomplished in polynomial time without probabilistic choices [51]. There are also several generalizations of the integer lattice algorithm to dense multivariate factoring over the rational numbers [64] and [39].

Factoring polynomials over algebraic extensions is polynomial-time reducible to factoring over the ground field by virtue of the classical Kronecker reduction, and the timing analysis with Trager's improvement can be found in [59]. This approach is also described in [18]. An alternate way to factor over algebraic number fields in polynomial-time, again based on lattice reduction, is given by A. K. Lenstra [63], [66]. It turns out that the reduction from multivariate to univariate polynomial factorization also gives an absolute irreducibility test [44].

The high exponents in the running time bounds may indicate that the polynomial-time solutions are quite impractical at present. This is probably true for the original algorithms, but the short lattice vector algorithm, aside form many intriguing diophantine approximation applications, also has its function in polynomial factorization. A. K. Lenstra's lattice procedure to recover algebraic numbers from their modular images [62] seems an efficient way to eliminate the exponentially complex Chinese remaindering step in the Weinberger-Rothschild [91] factoring algorithm over algebraic number fields [1]. There is also the possibility now to factor univariate integer polynomials via complex root approximation and lattice reduction [82], [55]. In particular, Schönhage's algorithm has asymptotic running time $O(n^8 + n^5(\log B)^3)$, n = input degree, B = coefficient bound. This approach may be superior in practice on hard-to-factor inputs [53]. M. Monagan's experiments [74] also indicate that the Adleman-Odlyzko irreducibility test [6] combined with a probabilistic integer primality test is quite practical. The new absolute irreducibility criteria [44], [86], Chapter 3, and [23] are also very useful in practice.

If the number of variables is allowed to grow with the input size, then the dense representation often consumes too much space. The first less expensive representation studied is the sparse one, in which only monomials with non-zero coefficients are stored. Zippel's sparse interpolation [94] and lifting algorithms [95] were the first methods based on randomization rather than heuristics. However, a rigorous probability analysis of the sparse lifting process required an effective probabilistic version of the Hilbert irreducibility theorem. Already in 1981, Heintz and Sieveking [37] provided such a theorem for algebraically closed fields, and in 1983 von zur Gathen proved a suitable version for arbitrary coefficient fields. In retrospect, the author's effective Hilbert irreducibility theorem also lends itself to an even simpler probabilistic version [45]. In [32] sparse polynomials are described that possess irreducible factors with super-polynomially more terms. These examples imply that any sparse Hensel Lifting scheme can have more than polynomial running time on certain inputs. In order to deal with this phenomenon it became clear that the sparse representation had to be replaced by a more powerful one.

The usage of ''straight-line programs'' as a means to compute certain polynomials has been developed in the framework of complexity theory in the past decade, refer for example to [83] and [84]. In 1983 von zur Gathen [28] combined his probabilistic Hilbert irreducibility theorem with the probabilistic method of straight-line program evaluation [80] and [40] to find the factor degree pattern of polynomials defined by straight-line computations. A previously known operation on polynomials in straight-line representation is that of taking first order partial derivatives [9]. Although there is evidence that other operations such as higher partial derivatives are inherently complex [87] (cf. also §5), the greatest common divisor problem of polynomials is straight-line program representation could shown by the author in late 1984 to be in probabilistic polynomial-time [52]. In 1985 the author could also show that straight-line programs for the irreducible factors of a polynomial given by a straight-line program can be found in probabilistic polynomial-time [49]. With Zippel's 1979 sparse polynomial interpolation algorithm [94] our factorization result resolves all problems left open in [95], [32], and [46]. We note that, unlike the randomized solutions for factorization of univariate polynomials over large finite fields, the probabilistic solutions are of the Monte-Carlo kind, ''probably correct and always fast.'' The failure probability can, of course, be made arbitrarily small.

2. The Lattice Reduction Algorithm

In this section we present a key algorithm in diophantine optimization, which enabled A. K. Lenstra, H. Lenstra, and L. Lovász [67] to prove that polynomial factoring is in polynomial time. Given a set of n linearly independent m-dimensional integer vectors, the algorithm finds a vector in the **Z**-span of these vectors, the Euclidean length of which is within a factor of $2^{(n-1)/2}$ of the length of the shortest vector in this lattice. Aside from polynomial factorization, where the algorithm has at least two applications, one for factoring primitive integer polynomials and one for recovering algebraic numbers from their modular images, this short vector algorithm has found many other applications. The algorithm can be used to find simultaneous rational approximations to a set of reals [67], Proposition 1.39, and [57], to break some variants of the Merkle-Hellman knapsack based encryption scheme [3], [75], to solve certain subset sum problems [58], to determine integer relations among reals [36] (see also §3), to find a lattice point close to a given point [8], to name but a few.

We now introduce the notion of reduced basis. Let $b_1,\ldots,b_n \in \mathbf{Z}^m$, $m \geq n$, be linearly independent over **Q**. By $b_1^*,\ldots,b_n^*$ we denote the orthogonalization of this basis, namely

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j}\, b_j^*,\ 1 \leq i \leq n, \tag{2.1}$$

$$\mu_{i,j} = \frac{(b_i, b_j^*)}{\|b_j^*\|^2},\ 1 \leq j < i \leq n, \tag{2.2}$$

where (,) denotes the scalar product and $\|\ \ \|$ the square norm. The basis $b_1,\ldots,b_n$ is called *reduced* if

$$\|b_k^*\|^2 \geq \tfrac{1}{2}\|b_{k-1}^*\|^2 \text{ for } 1 < k \leq n. \tag{2.3}$$

Lemma ([67], Proposition 1.11): Let $b_1,\ldots,b_n \in \mathbf{Z}^m$ form a reduced basis. Then for any non-zero vector $x \in \mathbf{Z}b_1+\cdots+\mathbf{Z}b_n$

$$2^{n-1}\|x\|^2 \geq \|b_1\|^2.$$

Hence $\|b_1\|$ is within a factor of $2^{(n-1)/2}$ of the norm of the shortest vector in the lattice spanned by $b_1,\ldots,b_n$.

Proof: Let $x = \sum_{i=1}^{l} r_i b_i = \sum_{i=1}^{l} r_i^* b_i^*$ with $r_i \in \mathbf{Z}$, $r_i^* \in \mathbf{Q}$ and $r_l \neq 0$, $1 \leq l \leq n$. Since b_l^* is orthogonal to b_i for $i = 1,\ldots,l-1$, $(x, b_l^*) = r_l(b_l, b_l^*) = r_l^*(b_l^*, b_l^*)$. But $(b_l, b_l^*) = (b_l^*, b_l^*) \neq 0$. Therefore, $r_l^* = r_l$, which is an integer $\neq 0$. It follows that $\|x\|^2 = \sum_{i=1}^{l}(r_i^*)^2\|b_i^*\|^2 \geq (r_l^*)^2\|b_l^*\|^2 \geq \|b_l^*\|^2 \geq \|b_1^*\|^2/2^{l-1}$, the last inequality by using (2.3) inductively. Since $b_1^* = b_1$ the lemma is proven. □

For reference we shall give the reduction algorithm, incorporating a slight improvement of [43].

Algorithm *Lattice Basis Reduction*

Input: n linearly independent m dimensional integer vectors $b_1, \ldots, b_n$.

Output: $b_1, \ldots, b_n$ forming a reduced basis for the integer lattice spanned by the input vectors. By the above lemma, b_1 must therefore be a short vector.

Step I (Gram-Schmidt Initialization): The arrays μ and β are initialized such that $\mu_{i,j}$, $1 \le j < i \le n$, and $\beta_l = \|b_l^*\|^2$, $1 \le l \le n$, satisfy (2.1) and (2.2):

FOR $i \leftarrow 1$ TO n DO

(The following loop is skipped for $i = 1$.)
FOR $j \leftarrow 1$ TO $i-1$ DO

$$\mu_{i,j} \leftarrow \frac{1}{\beta_j} \left((b_i, b_j) - \sum_{l=1}^{j-1} \mu_{j,l} \mu_{i,l} \beta_l \right).$$

$$\beta_i \leftarrow \|b_i\|^2 - \sum_{l=1}^{i-1} \mu_{i,l}^2 \beta_l.$$

$k \leftarrow 2$. (Used as counter in next step.)

Step R (Reduction Loop): At this point μ and β correspond to the orthogonalization of $b_1, \ldots, b_n$. Moreover $b_1, \ldots, b_{k-1}$ is reduced, i.e.

$$\|b_l^*\|^2 \ge \tfrac{1}{2}\|b_{l-1}^*\|^2,\ 1 < l \le k-1, \text{ and } |\mu_{i,j}| \le \tfrac{1}{2},\ 1 \le j < i \le k-1. \tag{2.4}$$

IF $k = n+1$ THEN RETURN $(b_1, \ldots, b_n)$.
IF $k = 1$ THEN $k \leftarrow 2$; *incremented* $\leftarrow$ *true*; GOTO step R.
IF *incremented* THEN perform step A given below.

Now (2.4) is also valid for $i = k$, i.e.

$$|\mu_{k,j}| \le \tfrac{1}{2} \text{ for } 1 \le j < k. \tag{2.5}$$

IF $\beta_k \ge \frac{1}{2}\beta_{k-1}$ THEN $k \leftarrow k+1$; *incremented* $\leftarrow$ *true*; GOTO step R.

At this point

$$\|b_k^*\|^2 < \tfrac{1}{2}\|b_{k-1}^*\|^2 \le \left(\tfrac{3}{4} - \mu_{k,k-1}^2\right) \|b_{k-1}^*\|^2, \tag{2.6}$$

the last inequality because of (2.4) for $j = k-1$.

Interchange b_{k-1} and b_k and update the arrays μ and β such that (2.1) and (2.2) is satisfied for the new order of basis vectors. The only entries which change are β_{k-1}, β_k and $\mu_{i,j}$, $i = k-1, k$, $1 \le j < i$, as well as $\mu_{i,k-1}$, $\mu_{i,k}$, $k < i \le n$. Let γ and ν denote the updated contents of β and μ, then, according to [67], 1.22 and Figure 1,

$$\gamma_{k-1} = \beta_k + \mu_{k,k-1}^2 \beta_{k-1}, \quad \nu_{k,k-1} = \frac{\mu_{k,k-1}\beta_{k-1}}{\gamma_{k-1}}, \quad \gamma_k = \frac{\beta_{k-1}\beta_k}{\gamma_{k-1}},$$

$$\left.\begin{array}{l}\nu_{i,k-1} = \mu_{i,k-1}\nu_{k,k-1} + \mu_{i,k}\beta_k/\gamma_{k-1}\\ \nu_{i,k} = \mu_{i,k-1} - \mu_{i,k}\mu_{k,k-1}\end{array}\right\}\text{ for } k<i\le n,$$

$$\nu_{k-1,j} = \mu_{k,j},\ \nu_{k,j} = \mu_{k-1,j} \text{ for } 1\le j<k-1.$$

$k \leftarrow k-1$; *incremented* $\leftarrow$ *false*; GOTO step R.

Step A (Adjust $\mu_{k,j}$): This step replaces b_k by $b_k - \sum_{l=1}^{k-1}\lambda_l b_l$, $\lambda_l \in \mathbf{Z}$, such that the new $\mu_{k,j}$ satisfy (2.4). The replacements of the entries in b_k are carried out modulo M, where M is chosen so large that the final b_k agrees with its modular image.

Substep AM (Initialize the modulus):
$M \leftarrow \left\lceil \sqrt{(k+3)\max\{\beta_1,\ldots,\beta_k\}}\right\rceil$. (For efficiency in modular operations, M can also be chosen the least power of the radix larger than that.)

FOR $l \leftarrow k-1$ DOWNTO 1 DO substep AL.

Substep AL (Make $\mu_{k,l}$ absolutely smaller than 1/2):

IF $|\mu_{k,l}| \ge \frac{1}{2}$ THEN

> $r \leftarrow$ ROUNDED($\mu_{k,l}$). ROUNDED(x) denotes the largest integer z s.t. $|z-x| \le 1/2$.
> Replace b_k by $(b_k - r\, b_l)$ modulo M.
> (Notice that in this case $\lambda_l = r$, otherwise $\lambda_l = 0$.)
>
> FOR $j \leftarrow 1$ TO $l-1$ DO $\mu_{k,j} \leftarrow \mu_{k,j} - r\mu_{l,j}$.
>
> $\mu_{k,l} \leftarrow \mu_{k,l} - r$.

Substep AB: Balance the residual entries of b_k mod M such that each modulus $> \frac{1}{2}M$ is replaced by its negative equivalent $\le \frac{1}{2}M$. □

We will not prove the basis reduction algorithm correct or analyze its complexity, but refer to [67]. However, for later reference we shall state its binary running time in a more refined way. Let us assume that we carry out the arithmetic on the rationals in β_k and $\mu_{i,j}$ by representing their integer numerators and denominators. Let d_l denote the Gramian of the vectors $\{b_1,\ldots,b_l\}$, that is

$$d_l = \det\Big((b_i,b_j)\Big)_{1\le i,j\le l} = \prod_{i=1}^{l}\|b_i^*\|^2 \quad \text{for } 1\le l\le n.$$

Then $\|b_l^*\|^2 = d_l/d_{l-1}$, $2 \le l \le n$. Moreover,

$$d_j\mu_{i,j} \in \mathbf{Z} \text{ and } \mu_{i,j}^2 \le d_{j-1}\|b_i\|^2 \text{ for } 1 \le j < i \le n.$$

These relationships is always true at the top of step R. Throughout the algorithm $\|b_i\|^2 < nB$, where B is the the square of the length of the longest vector in the original basis.

Furthermore, the Gramians only change when b_k is exchanged with b_{k-1}, in which case the new value of d_{k-1} is at least 3/4 times smaller than the old value. Therefore we have the following theorem.

Theorem: Let B be as above, and let d_l be the Gramian of the first l input vectors. Then the Lattice Basis Reduction Algorithm requires

$$O(n^2 m + nm \ \log(\prod_{l=1}^{n} d_l))$$

integer arithmetic operations. The integers on which these operations are performed have binary length

$$O(\log(\max\{n, B, d_1, \ldots, d_n\})).$$

In general, the number of arithmetic operations and the integer lengths can be bounded by $O(n^3 m \ \log(B))$, $O(n \log(B))$, respectively.

In practice, the β's and μ's should be computed in big floating point arithmetic. Roundoff errors can be corrected by checking whether the final lattice is reduced, but at the moment we have no well-tested recommendations what mantissa length should be used (perhaps $2n+\log_2(B)$ bits). Schnorr [77] has theoretically justified the use of floating point arithmetic and cut the integer lengths in the algorithm to $O(n+\log(B))$. For particular lattices, Schönhage's improvements [82] speed up the short vector construction further.

Although computational solutions for finding short lattice vectors have been carried out earlier, e.g. by Dieter [22] and by Ferguson and Forcade [24], the Basis Reduction algorithm is the first guaranteed polynomial-time solution. Aside from improving the running time, one can ask to improve the ratio (length of computed short vector)/(length of shortest vector) while retaining polynomial running time. The Basis Reduction algorithm can be improved to produce a ratio $\le \gamma^{(n-1)/2}$, where $\gamma > 4/3$. All one has to change is the comparison in step R to $\beta_k \ge \beta_{k-1}/\gamma$. Schnorr [78] constructs for every $\varepsilon > 0$ a polynomial-time basis reduction algorithm such that the ratio is $(1+\varepsilon)^n$. The running time of these algorithm are in fact only a constant times slower the than Basis Reduction algorithm, where the constant depends on ε.

3. Root Approximation Algorithms

In this section, we will describe two algorithms with which we can establish that dense multivariate rational polynomials can be factored in polynomial time. The first algorithm splits univariate integer polynomials, and the second reduces the problem of multivariate factoring to univariate factoring. Both algorithms make use the following idea. Let $f(x) \in F[x]$ be the polynomial to be factored. First, we find an approximation $\hat{\zeta}$ to a root $f(\zeta)=0$. Then we try to find another polynomial $\hat{g}(x) \in F[x]$, $\deg(\hat{g}) < \deg(f)$, whose coefficients are small and for which $\hat{g}(\hat{\zeta})$ is approximately zero. Clearly, the minimal polynomial g of ζ is a candidate for $\hat{g}$. The key argument will show that for sufficiently good approximation $\hat{\zeta}$ the only polynomial $\hat{g}$ with small enough coefficients for which $\hat{g}(\hat{\zeta})$ remains bounded is $\hat{g} = g$.

The notions of "small" will be different for $F = \mathbf{Q}$ and $F = \mathbf{Q}(y_1, \ldots, y_r)$. For $F = \mathbf{Q}$, we use the distance in the complex plane as our valuation and for $F = \mathbf{Q}(y_1, \ldots, y_r)$ we use the order of the multivariate Taylor series approximation. We now present the details for the algorithm in $\mathbf{Q}[x]$.

Let $f(x) \in \mathbf{Z}[x]$ be a primitive polynomial, $n = \deg(f)$. Let $\zeta \in \mathbf{C}$ with $f(\zeta) = 0$ and let $g(x) = g_0 + g_1 x + \cdots + g_m x^m \in \mathbf{Z}[x]$ be the minimal polynomial of ζ, $m \le n$. Furthermore, for $k > 0$, let $\hat{\alpha}_i, \hat{\beta}_i \in \mathbf{Z}$, $0 \le i \le m$, satisfy

$$| 2^k \mathrm{Re}(\zeta^i) - \hat{\alpha}_i | \le \frac{1}{2}, \quad | 2^k \mathrm{Im}(\zeta^i) - \hat{\beta}_i | \le \frac{1}{2}.$$

Now consider the $m+3$ dimensionals lattice spanned by the $m+1$ columns of

$$L_m = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & \vdots \\ 0 & 0 & 1 & & \\ \vdots & \vdots & \vdots & & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ 2^k (=\hat{\alpha}_0) & \hat{\alpha}_1 & \hat{\alpha}_2 & \cdots & \hat{\alpha}_m \\ 0\,(=\hat{\beta}_0) & \hat{\beta}_1 & \hat{\beta}_2 & \cdots & \hat{\beta}_m \end{bmatrix}.$$

The vector

$$L_m \times \begin{vmatrix} g_0 \\ \vdots \\ g_m \end{vmatrix} = \begin{vmatrix} g_0 \\ \vdots \\ g_m \\ \sum_{i=0}^m g_i \hat{\alpha}_i \\ \sum_{i=0}^m g_i \hat{\beta}_i \end{vmatrix} = \begin{vmatrix} g_0 \\ \vdots \\ g_m \\ \hat{\alpha} \\ \hat{\beta} \end{vmatrix}$$

is an element in this lattice. We first observe that

$$\|g\| \le \binom{2m}{m}^{\frac{1}{2}} \|f\| =: B_1(m, \|f\|),$$

§4.6.2, Exercise 20. Now with $\mathbf{i} = \sqrt{-1}$

$$\hat{\alpha}^2 + \hat{\beta}^2 = \left| \sum_{i=0}^m g_i (\hat{\alpha}_i + \mathbf{i}\hat{\beta}_i) \right|^2 = \left| \sum_{i=0}^m g_i (\hat{\alpha}_i + \mathbf{i}\hat{\beta}_i - 2^k \zeta^i) \right|^2$$

$$\le \left(\sum_{i=0}^m \left| g_i (\hat{\alpha}_i - 2^k \mathrm{Re}(\zeta^i)) + \mathbf{i} g_i (\hat{\beta}_i - 2^k \mathrm{Im}(\zeta^i)) \right| \right)^2$$

$$\le \left(\sum_{i=0}^m | g_i |\right)^2 \le (m+1) \|g\|^2.$$

The last inequality follows from

$$\left(\sum_{i=1}^m w_i\right)^2 \le m \sum_{i=1}^m w_i^2 \quad \text{for } w_i \ge 0. \tag{3.1}$$

Therefore,

$$\sqrt{\hat{\alpha}^2+\hat{\beta}^2+g_0^2+\cdots+g_m^2} \le \sqrt{m+2}\ \|g\| \le \sqrt{m+2}\ B_1 =: B_2(m,\|f\|),$$

which means that no matter how large k is chosen, the lattice L_m contains a vector of length $\le B_2$. Assume now that $[\bar{\alpha}, \bar{\beta}, \bar{g}_0, \ldots, \bar{g}_m]^T \in \mathbf{Z}^{m+3}$ is a short vector in L_m, that is

$$\sqrt{\bar{\alpha}^2+\bar{\beta}^2+\bar{g}_0^2+\cdots+\bar{g}_m^2} \le 2^{m/2}B_2 =: B_3(m,\|f\|). \tag{3.2}$$

We show that for sufficient large k, $\bar{g}(x) = \sum_{i=0}^{m}\bar{g}_i x^i$ is a scalar multiple of g. Let $s(x)$, $t(x) \in \mathbf{Z}[x]$ such that

$$\rho = \mathrm{resultant}(g(x), \bar{g}(x)) = s(x)g(x)+t(x)\bar{g}(x).$$

Since the coefficients of s and t are minors of the Sylvester matrix [12], we get from Hadamard's determinant inequality that

$$|t| \le \|g\|^{\deg(\bar{g})}\|\bar{g}\|^{\deg(g)} \le (B_1B_3)^m =: B_4(m,\|f\|),$$

where $|t|$ is the height (infinity norm) of the polynomial t. Therefore,

$$|t(\zeta)| \le B_4\frac{|\zeta|^m-1}{|\zeta|-1} =: B_5(m,\|f\|,|\zeta|).$$

Moreover by (3.1) and (3.2)

$$\begin{aligned} 2^k|\bar{g}(\zeta)| &\le \left|\sum_{i=0}^{m}\bar{g}_i(2^k\zeta^i-\hat{\alpha}_i-\mathbf{i}\hat{\beta}_i)\right| + \left|\sum_{i=0}^{m}\bar{g}_i\hat{\alpha}_i\right| + \left|\sum_{i=0}^{m}\bar{g}_i\hat{\beta}_i\right| \\ &\le \sum_{i=0}^{m}|\bar{g}_i| + |\bar{\alpha}| + |\bar{\beta}| \le \sqrt{m+3}\ B_3. \end{aligned}$$

Hence,

$$|\rho| = |t(\zeta)\bar{g}(\zeta)| \le \frac{\sqrt{m+3}\ B_3B_5}{2^k} =: \frac{B_6(m,\|f\|,|\zeta|)}{2^k}. \tag{3.3}$$

If we choose k so large that the RHS of (3.3) is less than 1, then by the integrality of ρ we get $\rho = 0$. Hence $\bar{g}$ must divide g, and since g is irreducible, our claim is established.

From the lattice reduction algorithm and complex root approximation procedures, we can now conclude that factoring $f(x)$ can be done in polynomial-time. The powers ζ^i of the root ζ are needed to precision $2^{-k_{\max}}$, where

$$k_{\max} = \log_2(B_6(n-1,\|f\|,|\zeta|)) = O(n^2+n\log\|f\|).$$

There are many simple ways of showing that this complex root approximation problem is in polynomial-time. We refer to Schönhage's monograph [81] for an exhaustive study on the computational complexity of the fundamental theorem of algebra. Once the $\hat{\alpha}_i$ and the $\hat{\beta}_i$ are known with respect to $k_{\max}$, we compute a short vector in $L_2, L_3, \ldots, L_{n-1}$ until a factor g is found. Notice that if we use the basic reduction algorithm, we will always get $\bar{g} = \pm g$. Also the reduction of L_{i+1} implicitly performs the reduction of L_i. Since g is irreducible, we can continue the process with another root of f/g.

The running time of this method is dominated by the lattice reductions. Let g_r be a factor of f with degree $m_r = \deg(g_r)$. It turns out that the Gramians of L_{m_r} can be explicitly computed. As in §2, we denote by d_i the Gramian spanned by the first i columns of L_{m_r}, $1 \le i \le m_r$. It can be shown [82], Lemma 6.1, that

$$d_i = A_i B_i - C_i^2 \quad \text{where} \quad A_i = 1 + \sum_{l=0}^{i-1} \hat{\alpha}_l^2, \;\; B_i = 1 + \sum_{l=0}^{i-1} \hat{\beta}_l^2, \;\; C_i = \sum_{l=0}^{i-1} \hat{\alpha}_l \hat{\beta}_l .$$

Not only does this formula show that $\log(d_i) = O(k_{\max})$, but it also provides a quick way to compute the initial $\beta_i = d_i / d_{i-1}$, $1 \le i \le m+1$. Incidentally, a similar formula yields the initial $\mu_{i,j}$, which we give for the convenience to implementors:

$$\mu_{i,j} = \frac{1}{d_j} (\hat{\alpha}_{i-1} \hat{\alpha}_{j-1} B_{j-1} + \hat{\beta}_{i-1} \hat{\beta}_{j-1} A_{j-1} - (\hat{\alpha}_{i-1} \hat{\beta}_{j-1} + \hat{\alpha}_{j-1} \hat{\beta}_{i-1}) C_{j-1}),$$

$1 \le j < i \le m+1$. By the theorem in §2, the reduction of L_{m_r} now costs $O(m_r^3 k_{\max})$ arithmetic steps (including divisions with remainder) on integers of size $O(k_{\max})$. Since $\sum_r m_r = n$, the total requirement is $O(n^5 + n^4 \log|f|)$ steps on integers of size $O(n^2 + n \log|f|)$. This root approximation algorithm is due to Schönhage [82], who also presents a modification of the reduction algorithm itself to obtain the following theorem. An asymptotically slower version is described in [55].

Theorem (Schönhage): For any $\varepsilon > 0$, the irreducible factors of a primitive polynomial $f \in \mathbf{Z}[x]$, $n = \deg(f)$, can be found in

$$O(n^{6+\varepsilon} + n^4 (\log|f|)^{2+\varepsilon})$$

binary steps.

Once the $\hat{\alpha}_i$ and $\hat{\beta}_i$ are known, the dependency on f is only in the bound B_1. Assume $2^k > B_6$ and the length of the short vector found in L_m is $> B_3$. Then no polynomial of degree $\le m$ and Euclidean norm $\le B_1$ can have α as its root. The method therefore can be used to provide evidence that certain numbers are transcendental in the spirit of Ferguson and Forcade [24]. One can employ the algorithm also to find the defining equations for certain fields. Yui and the author, for instance, use it to find equations for Hilbert class fields [54]. Those polynomials possess exactly one real root, which is easily computed. Finally, Kannan, A. K. Lenstra, and Lovász [55] show that the bits of certain transcendental numbers are not computationally unpredictable as defined by Micali and Blum [11].

Our algorithm essentially finds an integer relation among $\alpha, \ldots, \alpha^m$ or proves that none below a given norm bound exists. It lies at hand to generalize this problem to a set of complex numbers $\gamma_1, \ldots, \gamma_m$. The difficulty is to establish the non-existence of a norm-bounded relationship, which is resolved in [36].

We now come to the multivariate case. We shall restrict ourselves to bivariate polynomials. The full multivariate case is dealt with in [47]. Let $\bar{f}(y, x) \in F[y, x]$, F a field. Then by transforming and squarefree decomposing the polynomial we can reduce the problem to factoring $f(y, x) = x^n + f_{n-1}(y)x^{n-1} + \cdots + f_0(y) \in F[y, x]$ with $f(0, x)$ squarefree. We shall be more specific about that. Let $\tilde{f}(y, x) = \sum_{i=0}^{n} \sum_{j=0}^{d} f_{i,j} y^j x^i$ be a squarefree factor of $\bar{f}(y, x)$, which can be found by multivariate GCD computations or Hensel lifting as described in [41]. We now choose $a, b \in F$ such that $f(y, x) = \tilde{f}(y+bx+a, x)$ has $\deg_x(f) = \deg(\tilde{f})$ and $f(0, x)$ is squarefree. The first condition implies that the leading coefficient of f in x is a field element, by which we can divide the polynomial f. Notice that b has to satisfy

$$\sum_{i+j=\deg(\tilde{f})} f_{i,j} b^j \neq 0,$$

and a must not be the zero of a certain discriminant (cf [47],. Lemma 1). If $g(y, x)$ is now an irreducible factor of f, then $\tilde{g}(y, x) = g(y-bx-a, x)$ is one for $\tilde{f}$. If the field F has at least $2(n+d)d$ elements, suitable a and b can always be found. The only problem occurs if $F = \mathbf{F}_q$, the field with q elements, with q being too small. Then we perform our transformations in $\mathbf{F}_{q^p}$, p a prime $> \max(n, d)$. The irreducible factors of $\tilde{f}(y, x)$ in $\mathbf{F}_{q^p}[y, x]$ actually lie in $\mathbf{F}_q[y, x]$, due to the following useful lemma.

Lemma (von zur Gathen [28], Theorem 7.1): Let $f(x_1, \ldots, x_r) \in \mathbf{F}_q[x_1, \ldots, x_r]$ be irreducible, p a prime $> \deg_{x_i}(f)$ for all $1 \le i \le r$. Then f is irreducible in $\mathbf{F}_{q^p}[x_1, \ldots, x_r]$.

We now describe our factorization algorithm [47], which works for arbitrary fields.

Algorithm *Bivariate Factoring*

Input: $f(y, x) \in F[y, x]$ monic in x such that $f(0, x)$ is squarefree.

Output: $g(y, x) \in F[y, x]$ irreducible that divides $f(y, x)$.

Step N (Newton iteration): We compute the approximation to a root of f in $\bar{F}[[y]]$.

Find an irreducible factor $h(x)$ of $f(0, x)$. Now there exists a root $\alpha = \sum_{j=0}^{\infty} a_j y^j \in G[[y]]$, where $G = F[z]/(h(z))$ and $a_0 = z \bmod h(z)$. Let $n = \deg_x(f)$, $d = \deg_y(f)$, $l = \deg(h)$. The maximum needed precision for the root is $k = \lceil (2n-1)d/l \rceil$. By Newton iteration (cf. Lipson [68], §IX.3.3) we find $a_0 = z, a_1, \ldots, a_k$ such that

$$f(y, a_0 + a_1 y + \cdots + a_k y^k) \equiv 0 \bmod y^{k+1}.$$

Step M (Minimal polynomial determination): Compute powers $\alpha_k^{(i)}(y) \in G[y]$ of the root approximation. $\alpha_k^{(0)}(y) \leftarrow 1$.

FOR $i \leftarrow 1, \ldots, l-1$ DO $\alpha_k^{(i)}(y) \leftarrow (a_0 + \cdots + a_k y^k)\, \alpha_k^{(i-1)}(y) \bmod y^{k+1}$.
FOR $m \leftarrow l, \ldots, n-1$ DO Step L.
At this point, the polynomial f is known to be irreducible.

Step L: First compute the next power of the root approximation.

$$\alpha_k^{(m)}(y) \leftarrow (a_0+\cdots+a_k y^k)\alpha_k^{(m-1)}(y) \bmod y^{k+1}.$$

The precision needed to find a minimal polynomial of degree m is $\kappa \leftarrow \lceil (n+m)d/l \rceil$. Try to solve the equation

$$\alpha_\kappa^{(m)}(y) + \sum_{i=0}^{m-1} g_i(y)\alpha_\kappa^{(i)}(y) \equiv 0 \bmod y^{\kappa+1} \tag{3.4}$$

for polynomials $g_i(y) \in F[y]$, $\deg(g_i) \le d$. Let $g_i(y) = \sum_{0\le j\le d} g_{i,j} y^j$ and let

$$\alpha_\kappa^{(i)}(y) =: \sum_{j=0}^{\kappa} \left(\sum_{\lambda=0}^{l-1} a_{j,\lambda} z^\lambda \right) y^j \in F[z,y], \quad a_{j,\lambda}=0 \text{ for } j<0.$$

Then (3.4) leads to the linear system over F

$$a_{j,\lambda}^{(m)} + \sum_{i=0}^{m-1} \sum_{\mu=0}^{d} a_{j-\mu,\lambda}\, g_{i,\mu} = 0 \tag{3.5}$$

for $j=0,\ldots,\kappa$, $\lambda=0,\ldots,l-1$ in the variables $g_{i,j}$, $i=0,\ldots,m-1$, $j=0,\ldots,d$. If this linear system has a solution, which then must be unique, we return the irreducible polynomial

$$g(x) \leftarrow x^m + \sum_{i=0}^{m-1} g_i(y)x^i. \quad \square$$

The proof that a solution to the system (3.4) corresponds to the minimal polynomial of α is similar to the univariate case and can be found in [47], Theorem 1. In order to prove that the algorithm works in polynomial-time for $F = \mathbf{Q}$, the size of the numerators and denominators needs to be bounded. We will not present the fairly intricate analysis here but refer to [47], §6. For $f(x, y) \in \mathbf{Z}[y, x]$ the intermediate integers can be shown to be no more than $O(n^4 d^3 \log|f|)$ bits in length. Since we know that $|g| \le \sqrt{6}^{\,n+d} |f| =: B_7(n, d, |f|)$ [34], Chapter III, §4, Lemma II, a randomized approach becomes asymptotically significantly faster. For we can choose a random prime p with $2B_7 < p \le 4B_7$. Then for sufficiently large n and d with high probability Steps N, M, and L, when carried out in the homomorphic image $\mathbf{F}_p$ of $\mathbf{Z}$, result in the polynomial $g \bmod p$, from which g is readily recovered. Although the justification of the following theorem needs to be pieced together from the above, the transformation to monicity [56], §4.6.2, Exercise 18, and probability estimates derived according to [44], Theorem 4, we nonetheless have:

Theorem: Let $T(\deg(f), \log|f|)$ be a function dominating the binary running time for factoring $f \in \mathbf{Z}[x]$, and let ω be the exponent for matrix multiplication (classically $\omega=3$, at the moment the best is $\omega=2.376 + o(1)$ [21]). Then for any $\varepsilon>0$ the irreducible factors of a coefficient primitive polynomial $f \in \mathbf{Z}[y, x]$, $\delta = \deg(f)$, can be found by randomization in

$$T(\delta, \delta^{1+\varepsilon} + \log|f|) + O(\delta^{2\omega+2+\varepsilon} + \delta^{2\omega+1}(\log|f|)^{1+\varepsilon})$$

expected binary steps.

Algorithm Bivariate Factoring generalizes to an arbitrary number of variables [47]. The main advantage of this algorithm over other polynomial-time solutions [18], [64], and [39] is that it is field independent. Nonetheless, it is of little practical significance since combinational explosion in the multivariate Hensel algorithm is unlikely to occur because of theory of Hilbert irreducibility as described in the survey [41] (see also §5). In conclusion to this section, we can state that dense multivariate polynomials over the prime fields $\mathbf{F}_p$ and $\mathbf{Q}$ can be factored in polynomial-time.

4. Factoring over Finite Fields

Historically, it should be added to our survey [41] and Knuth's book [56], §4.6.2, that the ‘‘Q-matrix’’ construction, which is the basis of Berlekamp's algorithm, was first discovered by Butler in 1954 [14]. A nice generalization of that construction can be found in [35]. Using asymptotically fast polynomial arithmetic procedures, linear algebra algorithms, and multipoint polynomial evaluation with Zassenhaus's improvement [56], §4.6.2, Exercise 14, the running time of Berlekamp's algorithm in $\mathbf{F}_p[x]$ is

$$O(n^{\omega} + \log(p)n^{1+\varepsilon} + \max(p, n)(\log n)^{2+\varepsilon})$$

deterministic arithmetic steps in $\mathbf{F}_p$. The first two of the three terms correspond to the complexity of Butler's irreducibility test, but for that particular problem the distinct degree factorization is asymptotically faster. The usage of randomization in order to get the expected running time polynomial in $\log(p)$ is already considered a classic in the theory of randomized algorithms. A beautifully simple approach is due to Cantor and Zassenhaus [17], §3, in conjunction with Rabin's [76] analysis, or Ben-Or's refinement [10]. Its expected arithmetic running time is $O(\log(p)n^{2+\varepsilon})$, but more importantly it requires only to store $O(n)$ field elements at a time.

For large p, the probabilistic algorithms are of course the only practical choice, although it might not be clear which of several randomization schemes [76], [17], [61], [16], or [92], is preferable.

As a consequence of our Bivariate Factoring algorithm, dense multivariate factoring over finite fields also becomes a polynomial-time problem. The analysis can be found in [31], different algorithms are discussed in [18] and [65]. For large p, the algorithms are probabilistic in the Las Vegas sense, but again irreducibility testing can be done polynomially in $\log(p)$ without random choices [51].

New progress towards the removal of random choices in the univariate case can be reported. If one allows preprocessing depending on the field $\mathbf{F}_p$ only and with unlimited computational resources, a so-called splitting set of cardinality $\leq 2\log_2 p$ can be found, such that the random choices in certain probabilistic algorithms can be restricted to this set [2], [16]. A special problem is that of taking squareroots. Then the Tonelli-Shanks method [56], §4.6.2, Exercise 15, requires as its splitting set a single quadratic non-residue. This algorithm has been generalized to k-th roots, the splitting set being d-th non-residues, d all primes dividing both $p-1$ and k [5]. Assuming an extended Riemann hypothesis such splitting sets are deterministically constructible, in fact under such an assumption any polynomial $f \in \mathbf{Z}[x]$ with Abelian Galois group can be factored mod p in $(\deg(f)\log(p))^{O(1)}$ deterministic steps [38]. If $p-1$ is a ''smooth'' integer, i.e. $p-1$ only has prime factors of order $(\log p)^{O(1)}$, Moenck's algorithm [73] requires a primitive root and this requirement can even be shown to be necessary to polynomial-time factoring [30].

Finally, the theory of elliptic curves also has made its entry into factoring in $\mathbf{F}_p[x]$. Schoof [79] uses this theory to show that squareroots of a mod p can be taken deterministically in $O(\sqrt{|a|}(\log p)^8)$ steps.

A somewhat different question is the generation of irreducible polynomials of degree n in $\mathbf{F}_p[x]$. The probability that a randomly picked monic n-degree polynomial is irreducible in $\mathbf{F}_p[x]$ is asymptotically $1/n$ and Rabin's probabilistic generation uses such an estimate [76], Lemma 2. An improvement to this probability is reported in [15]. Recently, deterministic algorithms have been invented under hypothesis by von zur Gathen [29] and by Adleman and H. Lenstra [4].

5. Multivariate Factoring

As we have discussed already in §3, factoring of dense multivariate polynomials over the usual coefficient fields can be accomplished in polynomial-time. However, if the number of indeterminates is high, the dense representation causes exponential expression swell compared to more compact representations such as sparse ones. Note that there are $\binom{n+d}{d}$ monomials of total degree $\leq d$ in n indeterminates, although in the sparse representation only a few may be non-zero. Sparse lifting procedures strive to preserve the sparseness of input and output. Additional insight has been gained towards the well-known complication [41] arising during this process.

a) *The leading coefficients problem*: There are two new techniques for dealing with it. One is to use Viry's translation

$$\tilde{f}(x_1, x_2, \ldots, x_n) = f(x_1, x_2+b_2x_1, \ldots, x_n+b_nx_1),$$

where the b_i are random elements such that $\tilde{f}$ becomes monic in x_1 [71]. The drawback of this method is that $\deg_{x_1}(\tilde{f})$ might be substantially larger than $\deg_{x_1}(f)$ making the univariate factoring step costly. Another method by the author [48] finds the leading coefficients by lifting from a single univariate factorization and appears to be the algorithm of choice within a sparse lifting procedure. In the univariate case Wang's [89] idea appears quite useful for predicting integer leading coefficients.

b) *The extraneous factors problem*: Controlling this problem by randomization and Hilbert irreducibility has been theoretically justified. An effective theorem reads like that.

Theorem (cf [45]. and [50]): Let $f \in F[x_1, \ldots, x_n]$, F a perfect field, $d = \deg(f)$, $R \subset F$. The factor degree pattern of f is a lexicographically ordered vector $((d_i, e_i))_{i=1,\ldots,r}$ such that for $f = \prod_{i=1}^{r} h_i^{e_i}$, $h_i \in F[x_1, \ldots, x_n]$,

$$h_i \text{ irreducible}, \quad d_i = \deg(h_i) \geq 1, \quad h_i/h_j \notin F, \quad \text{for } 1 \leq i \neq j \leq r.$$

Let $a_1, a_3, \ldots, a_n, b_3, \ldots, b_n \in R$ be randomly selected elements,

$$f_2 = f(x_1+a_1, x_2, b_3x_1+a_3, \ldots, b_nx_1+a_n).$$

Then

$$\text{Prob}(f \text{ and } f_2 \text{ have the same factor degree pattern}) \geq 1 - \frac{4d\,2^d + d^3}{\text{card}(R)}.$$

Although this and all other known effective theorems [37], [28] only reduce to bivariate factoring, in practice one maps directly to the univariate case by letting $x_2 = b_2x_1+a_2$. The similarity of the used evaluations with those of Viry's can be taken advantage of for controlling the leading coefficients problem as well. Again, the classical mapping

$$f(x_1, x_2, \ldots, x_n) \to f(x_1, a_2, \ldots, a_n), \quad a_i \in F,$$

leads to a more efficient lifting procedure [48], although effective bounds for the probabilities for getting extraneous factors are not known.

c) *The bad zeros problem*: During the lifting process, the coefficients of monomials $x_1^{e_1} \cdots x_i^{e_i}$ need to be computed. In order to combat inefficiency, one should lift variable by variable, i.e. compute the factorization of $f(x_1, \ldots, x_i, a_{i+1}, \ldots, a_n)$ explicitly. Even then, in the presence of many factors, the problem of collecting like terms has exponential complexity [32]. We refer to Lugiez's lifting scheme [71] and to Luck's heuristics [70] for suggestions to control this problem in practice. We note that the reference as ''bad zeros problem'' is somewhat a misnomer chosen here for historial reasons.

Another newly discovered issue is that sparse polynomials can have dense factors [32]. For instance, the first factor in

$$(d + \prod_{i=1}^{n}(1+x_i+\cdots+x_i^{d-1})) \prod_{i=1}^{n}(x_i-1), \quad d \text{ prime} \tag{5.1}$$

is irreducible over **Q** and contains d^n non-zero monomials, whereas the product has $t < 4^n$ non-zero monomials. Since $d^n > t^{1/2 \log_2 d}$, the number of monomials in the first factor grows by more than $t^{O(1)}$. Therefore, any sparse lifting procedure will need more than polynomial running time with respect to the input size.

Furthermore, consider the Vandermonde determinant

$$\det\left(\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix}\right) = \prod_{i>j}(x_i - x_j).$$

If the corresponding Vandermonde matrix were multiplied with a unimodular matrix, the determinant of the resulting matrix would be inaccessible to a sparse factorization procedure by virtue of its $n!$ non-zero monomials. In general, symbolic objects represented by formulas (cf. (5.1)) or determinants cannot be dealt with by sparse techniques alone.

A computational model for evaluating polynomials and rational functions is that by a straight-line program. In order to deal with the issue of denseness we have adopted straight-line programs as a representation for polynomials. Let us give a small example for this representation, as it would appear in our system [25].

```
(c1) matrix([1,x12,x13],[x21,2,x23],[0,x32,3]);
```

```
                              [  1    x12   x13  ]
                              [                  ]
(d1)                          [ x21    2    x23  ]
                              [                  ]
                              [  0    x32    3   ]
```

The next instruction converts the determinant of d1 to straight-line format and then optimizes the resulting program

```
(c2) straightopt3(polytostraight('determinant(%)));
7(26%) instructions saved.

          v1  := 0
          v2  := 1
          v3  := x12
          v4  := x13
          v5  := x21
          v6  := 2
          v7  := x23
          v8  := x32
          v9  := 3
          v10 := v3 * v5
          v11 := v6 - v10
          v12 := v4 * v5
          v13 := v7 - v12
          v14 := v13 / v11
          v15 := v14 * v3
          v16 := v4 - v15
          v17 := v14 * v8
          v18 := v9 - v17
          v19 := v18 * v11
```

The above 19 instruction program computes the determinant of d1 in the variable v19. It should be pointed out that in the internal representation the variables are pointers to the corresponding instructions as in a directed acyclic graph.

It is not obvious at all that the GCD and factorization problems are feasible for polynomials in straight-line representation. To prove our point, consider a seemingly easier operation, that of computing partial derivatives. Letting

$$f(x_{1,1},\ldots,x_{n,n},y_1,\ldots,y_n) = \prod_{i=1}^{n}\Big(\sum_{j=1}^{n} x_{i,j}y_j\Big)$$

Valiant [87] observes that

$$\frac{\partial^n f}{\partial y_1 \cdots \partial y_n} = \text{permanent}\left(\begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{bmatrix}\right).$$

Clearly, f can be computed by a straight-line program of length $O(n^2)$, whereas the computation of the permanent is by Valiant's results #P-hard. Therefore it is believed that no straight-line program of length $n^{O(1)}$ exists that computes the permanent, and hence the intermediate expression swell for iterated partial derivatives is inherent even for the straight-line representation.

Aside from the just mentioned negative result, several efficient straight-line program transformations have been developed in the context of computational algebraic complexity. Most notably are the method by Strassen [83] for eliminating divisions from computations for polynomials, the method by Baur and Strassen [9] for computing all first partial derivatives, and the probabilistic equivalence test of straight-line programs [40]. One of the first results for polynomials represented by straight-line programs is the efficient computation of their factor degree pattern by von zur Gathen [28].

We now give the I/O specifications for the two main algorithms connected to straight-line factorization [50].

Algorithm *Factorization*

Input: $f \in F[x_1, \ldots, x_n]$, F a field, given by a straight-line program P of length l, a bound $d \geq \deg(f)$, and an allowed failure probability $\varepsilon \ll 1$.

Output: Either ''failure'', that with probability $< \varepsilon$, or $e_i \geq 1$ and irreducible $h_i \in F[x_1, \ldots, x_n]$, $1 \leq i \leq r$, given by a straight-line program Q of length

$$\mathrm{len}(Q) = O(d^2 l + d^6)$$

such that with probability $> 1 - \varepsilon$, $f = \prod_{i=1}^{r} h_i^{e_i}$. In case $p = \mathrm{char}(F)$ divides any e_i, that is $e_i = p^{\hat{e}_i}\,\bar{e}_i$ with $\bar{e}_i$ not divisible by p, we return $\bar{e}_i$ in place of e_i and Q will compute $h_i^{p^{\hat{e}_i}}$.

Algorithm *Sparse Conversion*

Input: $f \in F[x_1, \ldots, x_n]$ given by a straight-line program P of length l. Furthermore, a bound $d_0 \geq \max_{1 \leq i \leq n} \{\deg_{x_i}(f)\}$, the allowed failure probability $\varepsilon \ll 1$, and an upper bound $t \leq (d_0+1)^n$ for the number of monomials permitted in the answer.

Output: Either ''failure'' (that with probability $< \varepsilon$), or the representation of a sparse polynomial with no more than t monomials, or the message ''f has (probably) more than t monomials.'' The latter two outputs are correct with probability $> 1 - \varepsilon$.

Both algorithms work over an abstract field and are randomized. Probabilistic choices are interpreted as picking random elements from a sufficiently large subset of the field. A major theoretical fact is that both algorithms have polynomial complexity in a certain natural and precise sense [50]. In particular, for $F = \mathbf{Q}$ and $F = \mathbf{F}_q$ the algorithms have polynomial running time in bit steps. We therefore get the following theorem, which resolves all problems with sparse lifting mentioned above.

Theorem: If in addition to the input parameters of the Factorization algorithm we are given $t > 0$, for $F = \mathbf{Q}$ or $F = \mathbf{F}_q$ we can find in polynomially many binary steps and random bit choices in

$$l,\ d,\ \log(\frac{1}{\varepsilon}),\ \text{el–size}(P),\ \text{cc–size}(f),\ \text{and } t$$

sparse polynomials that with probability $> 1 - \varepsilon$ constitute all irreducible factors of f with no more than t monomials. Here el–size(P) is the binary size of the scalars in P, and cc–size(f) is the binary size of the coefficients of f, which in the case of the rationals are considered with a common denominator.

We mention that the degree bound d can be probabilistically determined [52], §5, and that d can be exponential in the length of the input programs. As it turns out, the length of the shortest straight-line program for a factor can then become exponential in the input length (or the input degree in binary) [69]. We conclude this section with a non-trivial straight-line factoring example, executed on our system.

```
(c1) p : 'determinant(matrix([w+x+y+z,a+b+c,u+v,0],
                              [(a-x-y-z)^2,(u-b-c)^2,(d-w)^2,0],
                              [(a+b+c+d)^3,(x+y+z)^3,(u+v)^3,0],
                              [(u+z)^5,(x+d)^5,(a+w)^5,x^2+y^2+z^2]));
Time= 250.0 msecs.

             [   z + y + x + w      c + b + a     v + u          0       ]
             [                                                           ]
             [                   2              2          2             ]
             [(- z - y - x + a)   (u - c - b)   (d - w)          0       ]
(d1)determinant([                                                           ])
             [                  3              3          3              ]
             [ (d + c + b + a)    (z + y + x)   (v + u)          0       ]
             [                                                           ]
             [             5              5          5   2     2     2]
             [        (z + u)        (x + d)      (w + a)   z   + y   + x ]

(c2) sf : straightfactor(polytostraight(p),1000)$
Time= 37100.0 msecs.
```

Determine length of straight-line program for first factor

```
(c3) straightlength(sf[1][1]);
Time= 100.0 msecs.
(d3)                                    11565
```

Optimize the straight-line program for first factor

```
(c4) sfo : straightopt3(sf[1][1])$
1811(15%) instructions saved.
Time= 110000.0 msecs.
```

Convert first factor to sparse

```
(c5) straighttosparse(sfo,10,terms=3);
Time= 111000.0 msecs.
                                      2     2     2
(d5)                                 z  +  y  +  x
```

Convert second factor to sparse unless it has more than 3 terms

```
(c6) straighttosparse(sf[2][1],10,terms=3);
Term bound exceeded.
Time= 28900.0 msecs.
(d6)                                   false
```

Use sparse lifting algorithm to obtain factorization (takes 1.7 hours)

```
(c7) factor(d1)$
Time= 6120000.0 msec.
```

6. Conclusion

Uni- and multivariate polynomial factorization is not only a classical problem, but efficient procedures also have important applications. An also classical one is that for determining the Galois group of a polynomial [60], [72], [13], etc. Multivariate polynomial factorization can help speed up the ubiquitous Gröbner basis construction [33], and some of the largest test cases have been successfully factored in this setting. Factorization over algebraic extensions is a key subroutine in the Cylindrical Algebraic Decomposition algorithm [7], and the references there, in integration in closed form [86], and takes part in Chou's method for geometrical theorem proving [19].

The question arises what major unresolved problems in the subject of polynomial factorization remain. One theoretical question is to remove the necessity of random choices from any of the problems known to lie within probabilistic polynomial-time, say factorization of univariate polynomials over large finite fields. Another problem is to investigate the parallel complexity of polynomial factorization, say for the **NC** model [20]. Kronecker's reduction from algebraic number coefficients [85], [59], Berlekamp's factorization algorithm over small finite fields [27], Kaltofen's deterministic Hilbert irreducibility theorem [47], §7, and Weinberger's irreducibility test for $\mathbf{Q}[x]$ [90] all lead to **NC** solutions by simply applying known **NC** methods for linear algebra problems. It is open whether factoring in $\mathbf{Q}[x]$ or irreducibility testing in $\mathbf{F}_p[x]$, p large, or in $\mathbf{Q}[x, y]$ can be accomplished in **NC**. We remark that testing a rational dense multivariate polynomial for absolute irreducibility can be shown to be in **NC** [44].

In connection with the Factorization algorithm presented in §5, we mention an open question. Assume that a straight-line program computes a polynomial whose degree is exponential in the length of the program. Do then at least the factors of polynomially bounded degree have feasible straight-line computations? A positive answer to this question would show that testing a polynomial for zero in a suitable decision-tree model is polynomial-time related to computing that polynomial. In general the theory of straight-line manipulation of polynomials may be extendable in part to unbounded input degrees, but even for the elimination of divisions problem [83] the answer is not known.

References

1. Abbott, J. A., Bradford, R. J., and Davenport, J. H., ''The Bath algebraic number package,'' *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 250-253, 1986.

2. Adleman, L. M., ''Two theorems on random polynomial time,'' *Proc. 19th IEEE Symp. Foundations Comp. Sci.*, pp. 75-83, 1978.

3. Adleman, L. M., ''On breaking generalized knapsack public key crypto systems,'' *Proc. 15th Annual ACM Symp. Theory Comp.*, pp. 402-412, 1983.

4. Adleman, L. M. and Lenstra, H. W., ''Finding irreducible polynomials over finite fields,'' *Proc. 18th ACM Symp. Theory Comp.*, pp. 350-355, 1986.

5. Adleman, L. M., Manders, K., and Miller, G. L., ''On taking roots in finite fields,'' *Proc. 18th IEEE Symp. Foundations Comp. Sci.*, pp. 175-178, 1977.

6. Adleman, L. M. and Odlyzko, A. M., ''Irreducibility testing and factorization of polynomials,'' *Math. Comp.*, vol. 41, pp. 699-709, 1983.

7. Arnon, D. S., Collins, G. E., and McCallum, S., ''Cylindrical algebraic decomposition I: The basic algorithm,'' *SIAM J. Comp.*, vol. 13, pp. 865-877, 1984.

8. Babai, L., ''On Lovász' lattice reduction and the nearest lattice point problem,'' *Combinatorica*, vol. 6, pp. 1-13, 1986.

9. Baur, W. and Strassen, V., ''The complexity of partial derivatives,'' *Theoretical Comp. Sci.*, vol. 22, pp. 317-330, 1983.

10. Ben-Or, M., ''Probabilistic algorithms in finite fields,'' *Proc. 22nd IEEE Symp. Foundations Comp. Sci.*, pp. 394-398, 1981.

11. Blum, M. and Micali, S., ''How to generate cryptographically strong sequences of pseudo-random bits,'' *SIAM J. Comp.*, vol. 13, pp. 850-864, 1984.

12. Brown, W. S. and Traub, J. F., ''On Euclid's algorithm and the theory of subresultants,'' *J. ACM*, vol. 18, pp. 505-514, 1971.

13. Bruen, A. A., Jensen, C. U., and Yui, N., ''Polynomials with Frobenius groups of prime degree as Galois groups II,'' *J. Number Theory*, vol. 24, pp. 305-359, 1986.

14. Butler, M. C. R., ''On the reducibility of polynomials over a finite field,'' *Quart. J. Math., Oxford Ser. (2)*, vol. 5, pp. 102-107, 1954.

15. Calmet, J. and Loos, R., ''An improvement of Rabin's probabilistic algorithm for generating irreducible polynomials over GF(p),'' *Inf. Proc. Lett.*, vol. 11, pp. 94-95, 1980.

16. Camion, P., ''A deterministic algorithm for factorizing polynomials of $F_{p[x]}$,'' *Ann. Discrete Math.*, vol. 17, pp. 149-157, 1983.

17. Cantor, D. G. and Zassenhaus, H., ''A new algorithm for factoring polynomials over finite fields,'' *Math. Comp.*, vol. 36, pp. 587-592, 1981.

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

19. Chou, S.-C., ''Proving elementary geometry theorems using Wu's algorithm,'' in *Theorem Proving: After 25 Years*, ed. Bledsoe, W. W. Loveland, D. W., Contemporary Mathematics, vol. 29, pp. 243-286, AMS, Providence, RI, 1984.

20. Cook, S. A., ''A taxonomy of problems with fast parallel algorithms,'' *Inf. Control*, vol. 64, pp. 2-22, 1985.

21. Coppersmith, D. and Winograd, S., ''Matrix multiplication via arithmetic progressions,'' *Proc. 19th Annual ACM Symp. Theory Comp.*, pp. 1-6, 1987.

22. Dieter, U., ''How to calculate the shortest vector in a lattice,'' *Math. Comp.*, vol. 29, pp. 827-833, 1975.

23. Duval, D., ''Une méthode géométrique de factorisation des polynomes en deux indéterminées,'' Tech. Report, Institut Fourier, Université de Grenoble I, 1983.

24. Ferguson, R. P. and Forcade, R. W., ''Multidimensional Euclidean algorithms,'' *J. reine angew. Math.*, vol. 334, pp. 171-181, 1982.

25. Freeman, T. S., Imirzian, G., Kaltofen, E., and Yagati, Lakshman, ''DAGWOOD: A system for manipulating polynomials given by straight-line programs,'' Tech. Report 86-15, Dept. Comput. Sci., RPI. Preliminary version in *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 169-175, 1986.

26. Fröhlich, A. and Shepherdson, J. C., ''Effective procedures in field theory,'' *Phil. Trans. Roy. Soc., Ser. A*, vol. 248, pp. 407-432, 1955/56.

27. Gathen, J. von zur, ''Parallel algorithms for algebraic problems,'' *SIAM J. Comp.*, vol. 13, pp. 802-824, 1984.

28. Gathen, J. von zur, ''Irreducibility of multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

29. Gathen, J. von zur, ''Irreducible polynomials over finite fields,'' Manuscript, 1986.

30. Gathen, J. von zur, ''Factoring polynomials and primitive elements for special primes,'' *Theoretical Comput. Sci.*, vol. 52, pp. 77-89, 1987.

31. Gathen, J. von zur and Kaltofen, E., ''Factoring multivariate polynomials over finite fields,'' *Math. Comp.*, vol. 45, pp. 251-261, 1985.

32. Gathen, J. von zur and Kaltofen, E., ''Factoring sparse multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 265-287, 1985.

33. Gebauer, R., Private communication, April 1986.

34. Gelfond, A. O., *Transcendental and Algebraic Numbers,* Dover Publ., New York, 1960.

35. Gunji, H. and Arnon, D., ''On polynomial factorization over finite fields,'' *Math. Comp.*, vol. 36, pp. 281-287, 1981.

36. Hastad, J., Just, B., Lagarias, J. C., and Schnorr, C. P., ''Polynomial time algorithms for finding integer relations among reals,'' *Proc. STACS '86, Springer Lec. Notes Comp. Sci.*, vol. 210, pp. 105-118, 1986.

37. Heintz, J. and Sieveking, M., ''Absolute primality of polynomials is decidable in random polynomial-time in the number of variables,'' *Proc. ICALP '81, Springer Lec. Notes Comp. Sci.*, vol. 115, pp. 16-28, 1981.

38. Huang, M.-D. A., ''Riemann hypothesis and finding roots over finite fields,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 121-130, 1985.

39. Hulst, M.-P. van der and Lenstra, A. K., ''Factorization of polynomials by transcendental evaluation,'' *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.*, vol. 204, pp. 138-145, 1985.

40. Ibarra, O. H. and Moran, S., ''Probabilistic algorithms for deciding equivalence of straight-line programs,'' *J. ACM*, vol. 30, pp. 217-228, 1983.

41. Kaltofen, E., ''Polynomial factorization,'' in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 95-113, Springer Verlag, Vienna, 1982.

42. Kaltofen, E., ''A polynomial reduction from multivariate to bivariate integral polynomial factorization,'' *Proc. 14th Annual ACM Symp. Theory Comp.*, pp. 261-266, 1982.

43. Kaltofen, E., ''On the complexity of finding short vectors in integer lattices,'' *Proc. EUROCAL '83, Springer Lec. Notes Comp. Sci.*, vol. 162, pp. 236-244, 1983.

44. Kaltofen, E., ''Fast parallel absolute irreducibility testing,'' *J. Symbolic Computation*, vol. 1, pp. 57-67, 1985.

45. Kaltofen, E., ''Effective Hilbert irreducibility,'' *Information and Control*, vol. 66, pp. 123-137, 1985.

46. Kaltofen, E., ''Computing with polynomials given by straight-line programs II; Sparse factorization,'' *Proc. 26th IEEE Symp. Foundations Comp. Sci.*, pp. 451-458, 1985.

47. Kaltofen, E., ''Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization,'' *SIAM J. Comp.*, vol. 14, pp. 469-489, 1985.

48. Kaltofen, E., ''Sparse Hensel lifting,'' *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.*, vol. 204, pp. 4-17, 1985.

49. Kaltofen, E., ''Uniform closure properties of p-computable functions,'' *Proc. 18th ACM Symp. Theory Comp.*, pp. 330-337, 1986.

50. Kaltofen, E., ''Factorization of polynomials given by straight-line programs,'' *Math. Sci. Research Inst. Preprint*, vol. 02018-86, Berkeley, CA, 1986. To appear in: ''Randomness in Computation,'' Advances in Computing Research, S. Micali ed., JAI Press Inc., Greenwich, CT, January 1987.

51. Kaltofen, E., ''Deterministic irreducibility testing of polynomials over large finite fields,'' *J. Symbolic Comp.*, vol. 3, pp. 77-82, 1987.

52. Kaltofen, E., ''Greatest common divisors of polynomials given by straight-line programs,'' *J. ACM*, vol. 35, no. 1, pp. 231-264, 1988.

53. Kaltofen, E., Musser, D. R., and Saunders, B. D., ''A generalized class of polynomials that are hard to factor,'' *SIAM J. Comp.*, vol. 12, pp. 473-485, 1983.

54. Kaltofen, E. and Yui, N., ''Explicit construction of the Hilbert class field of imaginary quadratic fields with class number 7 and 11,'' *Proc. EUROSAM '84, Springer Lec. Notes Comp. Sci.*, vol. 174, pp. 310-

320, 1984.

55. Kannan, R., Lenstra, A. K., and Lovász, L., ''Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers,'' *Proc. 16th Annual Symp. Theory Comp.*, pp. 191-200, 1984.

56. Knuth, D. E., *The Art of Programming, vol. 2, Semi-Numerical Algorithms, ed. 2,* Addison Wesley, Reading, MA, 1981.

57. Lagarias, J. C., ''The computational complexity of simultaneous Diophantine approximation problems,'' *SIAM J. Comp.*, vol. 14, pp. 196-209, 1985.

58. Lagarias, J. C. and Odlyzko, A. M., ''Solving low-density subset sum problems,'' *J. ACM*, vol. 32, pp. 229-246, 1985.

59. Landau, S., ''Factoring polynomials over algebraic number fields,'' *SIAM J. Comp.*, vol. 14, pp. 184-195, 1985.

60. Landau, S. and Miller, G. L., ''Sovability by radicals,'' *J. Comp. System Sci.*, vol. 30, pp. 179-208, 1985.

61. Lazard, D., ''On polynomial factorization,'' *Proc. EUROCAM '82, Springer Lec. Notes Comp. Sci.*, vol. 144, pp. 126-134, 1982.

62. Lenstra, A. K., ''Lattices and factorization of polynomials over algebraic number fields,'' *Proc. EUROCAM '82, Springer Lec. Notes Comp. Sci.*, vol. 144, pp. 32-39, 1982.

63. Lenstra, A. K., ''Factoring polynomials over algebraic number fields,'' *Proc. EUROCAL '83, Springer Lec. Notes Comp. Sci.*, vol. 162, pp. 245-254, 1983.

64. Lenstra, A. K., ''Factoring multivariate integral polynomials,'' *Theoretical Comp. Sci.*, vol. 34, pp. 207-213, 1984.

65. Lenstra, A. K., ''Factoring multivariate polynomials over finite fields,'' *J. Comput. System Sci.*, vol. 30, pp. 235-248, 1985.

66. Lenstra, A. K., ''Factoring multivariate polynomials over algebraic number fields,'' *SIAM J. Comp.*, vol. 16, pp. 591-598, 1987.

67. Lenstra, A. K., Jr., H. W. Lenstra, and Lovász, L., ''Factoring polynomials with rational coefficients,'' *Math. Ann.*, vol. 261, pp. 515-534, 1982.

68. Lipson, J., *Elements of Algebra and Algebraic Computing,* Addison-Wesley Publ., Reading, Mass., 1981.

69. Lipton, R. and Stockmeyer, L., ''Evaluations of polynomials with superpreconditioning,'' *Proc. 8th ACM Symp. Theory Comp.*, pp. 174-180, 1976.

70. Lucks, M., ''A fast implementation of polynomial factorization,'' *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 228-232, 1986.

71. Lugiez, D., ''A new lifting process for multivariate polynomial factorization,'' *Proc. EUROSAM '84, Springer Lec. Notes Comp. Sci.*, vol. 174, pp. 297-309, 1984.

72. McKay, J., ''Some remarks on computing Galois groups,'' *SIAM J. Comp.*, vol. 8, pp. 344-347, 1979.

73. Moenck, R. T., ''On the efficiency of algorithms for polynomial factoring,'' *Math. Comp.*, vol. 31, pp. 235-250, 1977.

74. Monagan, M. B., ''A heuristic irreducibility test for univariate polynomials,'' *J. Symbolic Comp.*, vol. submitted, 1986.

75. Odlyzko, A. M., ‘‘Cryptoanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir’s fast signature scheme,’’ *IEEE Trans. Inf Theory*, vol. IT-30/4, pp. 584-601, 1984.

76. Rabin, M. O., ‘‘Probabilistic algorithms in finite fields,’’ *SIAM J. Comp.*, vol. 9, pp. 273-280, 1980.

77. Schnorr, C. P., ‘‘A more efficient approach for lattice basis reduction,’’ *Proc. ICALP ’86, Springer Lec. Notes Comp. Sci.*, vol. 226, pp. 359-369, 1986.

78. Schnorr, C. P., ‘‘A hierarchy of polynomial time basis reduction algorithms,’’ in *Theory of Algebra*, ed. L. Lovász and E. Semerédi, Coll. Math. Soc. Janos Bolyai, vol. 44, pp. 375-386, North Holland Publ., Amsterdam, 1986.

79. Schoof, R. J., ‘‘Elliptic curves over finite fields and the computation of square roots mod p,’’ *Math. Comp.*, vol. 44, pp. 483-494, 1985.

80. Schwartz, J. T., ‘‘Fast probabilistic algorithms for verification of polynomial identities,’’ *J. ACM*, vol. 27, pp. 701-717, 1980.

81. Schönhage, A., ‘‘The fundamental theorem of algebra in terms of computational complexity,’’ Tech. Report, Univ. Tübingen, 1982.

82. Schönhage, A., ‘‘Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm,’’ *Proc. ICALP ’84, Springer Lec. Notes Comp. Sci.*, vol. 172, pp. 436-447, 1984.

83. Strassen, V., ‘‘Vermeidung von Divisionen,’’ *J. reine u. angew. Math.*, vol. 264, pp. 182-202, 1973. (In German).

84. Strassen, V., ‘‘Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten,’’ *Numer. Math.*, vol. 20, pp. 238-251, 1973. (In German).

85. Trager, B. M., ‘‘Algebraic factoring and rational function integration,’’ *Proc. 1976 ACM Symp. Symbolic Algebraic Comp.*, pp. 219-228, 1976.

86. Trager, B. M., ‘‘Integration of algebraic functions,’’ Ph.D. Thesis, MIT, 1984.

87. Valiant, L., ‘‘Reducibility by algebraic projections,’’ *L’Enseignement mathématique*, vol. 28, pp. 253-268, 1982.

88. Waerden, B. L. van der, *Modern Algebra*, F. Ungar Publ. Co., New York, 1953.

89. Wang, P. S., ‘‘Early detection of true factors in univariate polynomial factorization,’’ *Proc. EUROCAL ’83, Springer Lec. Notes Comp. Sci.*, vol. 162, pp. 225-235, 1983.

90. Weinberger, P. J., ‘‘Finding the number of factors of a polynomial,’’ *J. Algorithms*, vol. 5, pp. 180-186, 1984.

91. Weinberger, P. J. and Rothschild, L. P., ‘‘Factoring polynomials over algebraic number fields,’’ *ACM Trans. Math. Software*, vol. 2, pp. 335-350, 1976.

92. Yokoyama, K. and Takeshima, T., ‘‘Factorization of univariate polynomials over finite fields,’’ Manuscript, 1986.

93. Zassenhaus, H., ‘‘Polynomial time factoring of integral polynomials,’’ *SIGSAM Bulletin*, vol. 15, no. 2, pp. 6-7, 1981.

94. Zippel, R. E., ‘‘Probabilistic algorithms for sparse polynomials,’’ *Proc. EUROSAM ’79, Springer Lec. Notes Comp. Sci.*, vol. 72, pp. 216-226, 1979.

95. Zippel, R. E., ''Newton's iteration and the sparse Hensel algorithm,'' *Proc. '81 ACM Symp. Symbolic Algebraic Comp.*, pp. 68-72, 1981.