1. Introduction

In 1982 a polynomial-time algorithm for factoring polynomials in one variable with rational coefficients was published [II]. This *L^3-algorithm* came as a rather big surprise: hardly anybody expected that the problem allowed solution in polynomial time. The purpose of this introductory part is to present an informal description of the L^3-algorithm.

To measure the complexity of our algorithms we have to specify the encoding of the polynomials to be factored. Two *encoding schemes* for polynomials can be distinguished, a *dense* encoding scheme and a *sparse* encoding scheme. If a polynomial is densely encoded, all its coefficients, including the zeros, are listed; in a sparse encoding only the non-zero coefficients are listed. Here we use the dense encoding scheme. This implies that an algorithm to factor polynomials runs in polynomial time if for any polynomial f to be factored, the running time is bounded by a fixed polynomial function of the degrees and the size of the coefficients of f.

After the introduction of some basic tools in Section 2, we describe in Section 3 a well-known older algorithm to factor polynomials, the Berlekamp-Hensel algorithm, and we will indicate why this algorithm is not polynomial-time. Roughly speaking, the reason is that the irreducible factors we are looking for (which will frequently be called the *true factors*) are determined by a combinatorial search among other, p-adic factors.

A true factor can also be regarded as a short vector in a certain integral lattice, a concept that was introduced in [I]. Therefore we consider the problem of computing short vectors in a lattice in Section 4, and thereafter we explain the L^3-algorithm in Section 5.

This same technique of looking for short vectors can be applied to other polynomial factoring problems as well. Some of these generalizations of the

L^3-algorithm are presented in Section 6. We conclude in Section 7 with some remarks about the relative merits of these polynomial-time algorithms for the factorization of polynomials.

2. Preliminaries

In the subsequent sections the following three notions will play an important role: Berlekamp's algorithm, Hensel's lemma, and Mignotte's bound. These are the basic tools for most of the polynomial factoring algorithms. We will briefly explain here what they stand for.

Berlekamp's algorithm is an algorithm to determine the irreducible factors of a polynomial in one variable with coefficients in a finite field. Let $\mathbb{F}_q$ denote a finite field containing q elements, for some prime power $q = p^m$, and let f be a polynomial in $\mathbb{F}_q[X]$ of degree n. To factor f, the maximal number of additions, multiplications, and divisions in $\mathbb{F}_q$ to be carried out by Berlekamp's algorithm is $O(p\,m\,n^3)$. This is the best worst-case running time that is known for an algorithm to factor polynomials in $\mathbb{F}_q[X]$. There exist *probabilistic* algorithms for which the *expected* running time is linear in $\log p$ rather than linear in p, as is the case in Berlekamp's algorithm. Although such methods are usually much faster in practice, no upper bound can be given for their worst-case running time, and therefore they are irrelevant for our purposes. For a description of Berlekamp's algorithm we refer to [1; 11].

The Hensel lemma can be formulated as follows. Let p be a prime number, and let k be a positive integer. By $\mathbb{Z}/p^k\mathbb{Z}$ we will denote the ring of integers modulo p^k. Suppose that a polynomial $f \in \mathbb{Z}[X]$ and a factor $h \in (\mathbb{Z}/p^k\mathbb{Z})[X]$ of $f \bmod p^k$ in $(\mathbb{Z}/p^k\mathbb{Z})[X]$ are given, such that $h \bmod p$ and $(f \bmod p)/(h \bmod p)$ are relatively prime in $(\mathbb{Z}/p\mathbb{Z})[X]$ and such that h has leading coefficient equal to one (notice that, because p is prime, $\mathbb{Z}/p\mathbb{Z}$ is a finite field containing p elements). Hensel's lemma guarantees the existence of a unique polynomial $a \in (\mathbb{Z}/p\mathbb{Z})[X]$ of degree smaller than degree(h), such that $h + p^k a \in (\mathbb{Z}/p^{k+1}\mathbb{Z})[X]$ is a factor of $f \bmod p^{k+1}$ in $(\mathbb{Z}/p^{k+1}\mathbb{Z})[X]$. Furthermore, its proof gives an algorithm to construct this a, and this algorithm needs a number of bit operations that is bounded by a polynomial function of degree(f) and $\log p^k$ (cf. [11: exercise 4.6.2.22; 22; 23]). Thus, Hensel's lemma enables us to extend or *lift* a factorization of $f \bmod p$ to a factorization of $f \bmod p^k$ for any $k \in \mathbb{Z}_{>0}$ that we want. This computation can be done in polynomial time as long as k is polynomially bounded. Such a factorization of $f \bmod p^k$ will be called a *p-adic factorization of precision* p^k.

Finally, Mignotte's bound is an upper bound for the coefficients of a factor of a polynomial in one variable with integral coefficients. Let $f = \sum_{i=0}^{n} f_i X^i \in \mathbb{Z}[X]$ be a polynomial of degree n, and let $g = \sum_{i=0}^{m} g_i X^i \in \mathbb{Z}[X]$ be a factor of degree m of f in $\mathbb{Z}[X]$. Mignotte has proved in [14] that

$$|g_i| \leq \binom{m}{i} |f|, \tag{2.1}$$

where $|f|$ denotes the *length* $(\sum_{i=0}^{n} f_i^2)^{1/2}$ of the polynomial f. It follows that

$$|g| \leq \binom{2m}{m}^{1/2} |f|. \tag{2.2}$$

Notice that $\log|g| = O(n + \log|f|)$, so that the length of a dense encoding of a factor is polynomially bounded by the length of a dense encoding of the polynomial itself. Similar bounds for polynomials in more than one variable can be found in [6].

In the next section we will see that Berlekamp's algorithm, Hensel's lemma, and Mignotte's bound together give rise to an important algorithm to factor polynomials in $\mathbb{Z}[X]$, the Berlekamp-Hensel algorithm.

3. The Berlekamp-Hensel algorithm

The Berlekamp-Hensel algorithm was the first practical algorithm to factor polynomials in one variable with integral coefficients. In this section we present one of the simplest versions of this algorithm and we discuss its most important properties. Although many improvements of the algorithm have been suggested by several authors, the basic ideas remained the same, and hence we will ignore these variants. Also we will not discuss the generalizations of the Berlekamp-Hensel algorithm to polynomials in more than one variable.

The Berlekamp-Hensel algorithm essentially works as follows. First, a sufficiently precise p-adic factorization of the polynomial to be factored is

computed. Next, the true factors are determined by combining these p-adic factors in the proper way. We now present a somewhat more detailed description of the algorithm.

Let $f \in \mathbb{Z}[X]$ of degree n be the polynomial to be factored. For simplicity we assume that f is *monic*, i.e. f has leading coefficient one. The first step of the Berlekamp-Hensel algorithm is to remove the multiple factors from f. Because a factor of multiplicity $k \geq 1$ in f has multiplicity $k-1$ in the derivative f' of f, this can be done by dividing f by $\gcd(f,f')$, where this gcd can be computed by means of one of the subresultant algorithms (cf. [2]). So, from now on we may assume that f is *square-free*, i.e. f does not contain multiple factors.

Next, we determine a prime number p such that the polynomial $f \bmod p$ in $(\mathbb{Z}/p\mathbb{Z})[X]$ is square-free in $(\mathbb{Z}/p\mathbb{Z})[X]$. This condition on p is equivalent to the condition that p does not divide the *discriminant* $\mathrm{discr}(f) \in \mathbb{Z}_{\neq 0}$ of f (notice that $\mathrm{discr}(f) \neq 0$ because f is square-free). This implies that such a prime number p indeed exists, and that p can be bounded by a polynomial function of n and $\log|f|$.

In the third step we apply Berlekamp's algorithm to compute the complete factorization of $f \bmod p$ in $(\mathbb{Z}/p\mathbb{Z})[X]$. We may assume that the factors of $f \bmod p$ are monic. Clearly the irreducible factors of f in $\mathbb{Z}[X]$ are also factors of $f \bmod p$, but these factors of $f \bmod p$ are not necessarily irreducible in $(\mathbb{Z}/p\mathbb{Z})[X]$. So, the set of irreducible factors of $f \bmod p$ can be partitioned into a number of subsets such that each subset corresponds to an irreducible factor of f in $\mathbb{Z}[X]$. That is, the product of the elements of such a subset is just an irreducible factor of f in $\mathbb{Z}[X]$, reduced modulo p. Thus, these subsets will in general not be sufficient to reconstruct the factors of f in $\mathbb{Z}[X]$, because the coefficients of the resulting product are only integers modulo p. Therefore, before we look for the proper combinations of the p-adic factors, we first have to compute the p-adic factorization of f up to a higher precision.

This is what we do in the next step, where we apply Hensel's lemma, which is possible because $f \bmod p$ is square-free and because we assumed that the factors of $f \bmod p$ are monic. We modify each irreducible factor $\tilde{h}$ of $f \bmod p$ in $(\mathbb{Z}/p\mathbb{Z})[X]$ into a factor h of $f \bmod p^k$ in $(\mathbb{Z}/p^k\mathbb{Z})[X]$, for a value of $k \in \mathbb{Z}_{>0}$ that we will specify below. The polynomials h are monic, so that the factorization in $(\mathbb{Z}/p^k\mathbb{Z})[X]$ that we find in this way is unique.

The value of k has to be chosen in such a way that the coefficients of the combinations that we will have to consider are not too small. Therefore, if we represent $\mathbb{Z}/p^k\mathbb{Z}$ by $\{-[(p^k-1)/2], \ldots, -1, 0, 1, \ldots, [(p^k)/2]\}$, then k has to be chosen such that $(p^k-1)/2$ is greater than the largest possible coefficient of any factor of f in $\mathbb{Z}[X]$. From (2.1) it follows that we can take k minimal such that $(p^k-1)/2 > \binom{n}{n/2}|f|$.

Now that we have a sufficiently precise p-adic factorization of f, we are ready for the last step of the Berlekamp-Hensel algorithm, the determination of the true factors of f. As explained above, this can be done by looking at combinations of p-adic factors. So, for all subsets of our set of p-adic factors, we test whether the product of the p-adic factors in a subset is a true factor of f in $\mathbb{Z}[X]$, until all irreducible factors of f in $\mathbb{Z}[X]$ are found. Of course, we have to arrange the subsets in order of increasing cardinality, to guarantee that the factors of f that we find are irreducible.

This completes the description of the Berlekamp-Hensel algorithm. Let us consider its running time. It is not difficult to verify that, up to the last step, everything can be done in polynomial time. Unfortunately, this is not the case for the last step, the search for the true factors.

Namely, suppose that f is irreducible in $\mathbb{Z}[X]$ and that the number of irreducible factors of $f \bmod p$ is r. Then we have to look at all subsets of cardinality at most $r/2$, before we are sure that none of them yields a factor of f in $\mathbb{Z}[X]$. This implies that the number of subsets to be considered is exponential in r, and because the degree n of f is the only a priori upper bound that we can give for r, we get a bound on the running time of the last step, that is exponential in n.

In [7] a method is given to generate infinite classes of irreducible polynomials in $\mathbb{Z}[X]$, such that, for some $c \in \mathbb{R}_{>0}$ and for every prime number p, the number of p-adic factors is at least cn. For these polynomials the number of subsets to be considered is indeed exponential in n, so that the exponential-time bound that we derived is the best possible.

In practice however, the situation is not so bad as it seems, and the algorithm usually has no problems to factor high-degree polynomials with large coefficients. Also, in [4] it is made plausible that, under certain assumptions concerning the distribution of the degrees of the factors of f, the expected number of subsets to be considered in the last step is at most n^2. This is in accordance with the practical experience that the last step

usually takes much less time than the computation of the p-adic factorization.

Clearly, to obtain a polynomial-time algorithm to factor polynomials in $\mathbb{Z}[X]$, we have to invent another method to reconstruct the factors in $\mathbb{Z}[X]$ from the p-adic factors. In Section 5 we will see that only one sufficiently precise p-adic factor suffices to reconstruct the corresponding irreducible factor in $\mathbb{Z}[X]$, so that we do not have to look for the proper combination of p-adic factors anymore. Before we can explain this construction and show that it gives a polynomial-time factoring algorithm, we have to present an important result concerning integral lattices; this will be done in the next section.

4. Short vectors in lattices

Let $b_1, b_2, \ldots, b_n \in \mathbb{Z}^n$ be linearly independent. The n-dimensional *lattice* $L \subset \mathbb{Z}^n$ with basis $b_1, b_2, \ldots, b_n$ is defined as the set of integral linear combinations of the vectors $b_1, b_2, \ldots, b_n$:

$$L = \{\Sigma_{i=1}^{n} r_i b_i : r_i \in \mathbb{Z}\}.$$

In Section 5 we will be interested in determining short vectors in a lattice, where we use the ordinary Euclidean norm for vectors. We will not give any detailed algorithms here; we will only briefly sketch what is known about computing short vectors in a lattice, and mention an important recent result.

Until now, no polynomial-time algorithm is known to compute a shortest non-zero vector in a lattice (polynomial-time means here polynomial-time in n and the size of the entries of the vectors b_i). In fact, the problem is widely conjectured to be NP-hard, but this has not yet been proved. (If we replace the L_2-norm by the L_∞-norm, then the shortest vector problem is known to be NP-hard (cf. [16]).) At several places more or less practical algorithms to calculate shortest vectors are presented [5; 11; 15]. Although the running times of these algorithms are not analyzed, they are certainly not polynomial-time. Also, in general they perform quite poorly for high-dimensional lattices (say $n \geq 15$). If we fix the dimension n of the lattice however, then a shortest vector can be found in polynomial time. This is a consequence of the

polynomial-time algorithm to solve integer linear programming problems with a fixed number of variables [13].

In 1981, L. Lovász invented an important algorithm, the so-called *basis reduction algorithm*, which made it possible to compute reasonably short vectors in a lattice in polynomial time. More precisely, this algorithm computes a non-zero vector b, belonging to a basis for L, such that $|b| \leq 2^{(n-1)/2}|x|$, for every $x \in L$, $x \neq 0$, where $||$ denotes the Euclidean length. So, in polynomial time we can find a vector in the lattice that is no more than $2^{(n-1)/2}$ times longer than a shortest vector in the lattice. In fact the basis reduction algorithm does not only compute an approximation of a shortest vector. It also computes a so-called *reduced basis* for the lattice, which is, roughly speaking, a basis that is reasonably orthogonal (cf. Section 7). A detailed description of the algorithm and a careful analysis of its running time are given in [11].

There are special cases in which the basis reduction algorithm computes a shortest vector. This will happen for instance if all vectors that are linearly independent of the shortest vector, are more than $2^{(n-1)/2}$ times longer than the shortest vector. This situation will occur in Section 5.

Unfortunately, the fact that the basis reduction algorithm runs in polynomial time does not imply that it is very fast in practice, although it is much better than the algorithms mentioned above. If $B \in \mathbb{Z}_{>0}$ bounds the number of bits in the coordinates of the basis $b_1, b_2, \ldots, b_n$ for L, then a theoretical bound on the number of bit operations to be performed is $O(n^6 B^3)$ (cf. [11]; a slightly better bound, namely $O(n^6 B^2 + n^5 B^3)$, was derived in [9]). Experiments by A.M. Odlyzko showed that in practice the running time is proportional to nB^3. To give an impression of actual running times, we conclude this section with some results from Odlyzko's implementation on a Cray-1 computer:

n	B	average running time in minutes
31	55	0.5
31	65	0.75
41	70	1.2
51	88	3
51	180	18
71	140	14
81	160	23

5. The L^3-algorithm

We return to the problem of factoring polynomials in $\mathbb{Z}[X]$. We will show that the basis reduction algorithm enables us to formulate a polynomial-time algorithm to factor polynomials in $\mathbb{Z}[X]$. Again, let $f \in \mathbb{Z}[X]$ of degree n be the polynomial to be factored. We will assume that the gcd of the coefficients of f is 1 (i.e. f is *primitive*), and that f is square-free.

From Section 2 we know that it is possible to compute a p-adic factorization of f up to any precision that we want, where the prime number p is chosen in such a way that $f \bmod p$ is square-free in $(\mathbb{Z}/p\mathbb{Z})[X]$. Let, for some positive integer k, the polynomial $h \in (\mathbb{Z}/p^k\mathbb{Z})[X]$ be a monic p-adic factor of $f \bmod p^k$, such that $h \bmod p$ is irreducible in $(\mathbb{Z}/p\mathbb{Z})[X]$. It follows that there exists a unique irreducible factor h_0 of f in $\mathbb{Z}[X]$, such that h divides $h_0 \bmod p^k$ in $(\mathbb{Z}/p^k\mathbb{Z})[X]$. We will see that, if k is chosen sufficiently large, then the p-adic factor h suffices to determine the factor h_0 of f (of course, if f is irreducible, we will find $h_0 = f$).

The L^3-algorithm proceeds roughly as follows. First, we construct a lattice such that the factor h_0 that we are looking for is contained in this lattice, and such that h_0 is much shorter than the other vectors in the lattice. Next, we determine h_0 by means of the basis reduction algorithm, which is possible because h_0 is very short.

We will explain how to construct a basis for such a lattice. Denote by ℓ the degree of h; so $0 < \ell \le n$. Let us suppose that the degree of h_0 is m, with $\ell \le m \le n$. To start with, we know that h divides $h_0 \bmod p^k$ in $(\mathbb{Z}/p^k\mathbb{Z})[X]$. This implies that h_0 belongs to the set L of polynomials of degree at most m which have h as a factor, when taken modulo p^k. Because h was chosen to be monic, we have that

$$(5.1) \qquad L = \{p^k r + h q : r, q \in \mathbb{Z}[X],\ \text{degree}(r) < \ell,\ \text{degree}(q) \le m-\ell\}.$$

We define the polynomials $b_0, b_1, \ldots, b_m \in \mathbb{Z}[X]$ as $b_i = p^k X^i$ for $0 \le i < \ell$ and $b_i = h X^{i-\ell}$ for $\ell \le i \le m$. With (5.1) this gives

$$L = \mathbb{Z} b_0 + \mathbb{Z} b_1 + \ldots + \mathbb{Z} b_{\ell-1} + \mathbb{Z} b_\ell + \mathbb{Z} b_{\ell+1} + \ldots + \mathbb{Z} b_m,$$

so that we can rewrite (5.1) as $L = \{\Sigma_{i=0}^{m} r_i b_i : r_i \in \mathbb{Z}\}$.

Regarding the polynomials $b_0, b_1, \ldots, b_m$ as $(m+1)$-dimensional integral vectors (where the coefficient of X^i is identified with the $(i+1)$-th coordinate), we see that the vectors $b_0, b_1, \ldots, b_m \in \mathbb{Z}^{m+1}$ are linearly independent, because they form an upper-diagonal matrix. It follows that L is an $(m+1)$-dimensional lattice in $\mathbb{Z}^{m+1}$. So, we have constructed an $(m+1)$-dimensional lattice containing h_0. We will show that k can be chosen in such a way that all elements of L that are linearly independent of h_0, are more than $2^{m/2}$ times longer than h_0, so that h_0 can be computed by means of the basis reduction algorithm (cf. Section 4).

Because h_0 is a factor of f in $\mathbb{Z}[X]$, we find from (2.2) that

$$(5.2) \qquad |h_0| \le \binom{2m}{m}^{\frac{1}{2}} |f| = B.$$

Suppose that k is chosen such that

$$(5.3) \qquad p^k > (2^{m/2} B^2)^m.$$

Let g be an arbitrary non-zero element of L that is linearly independent of h_0; we claim that $|g| > 2^{m/2} B$, so that g is more than $2^{m/2}$ times longer than h_0.

Because g is linearly independent of h_0, and because h_0 is irreducible in $\mathbb{Z}[X]$, we have that $\gcd(h_0, g) = 1$ in $\mathbb{Z}[X]$. This implies that the polynomials $h_0 X^i$ for $0 \le i < \text{degree}(g)$ and $g X^j$ for $0 \le j < \text{degree}(h_0)$, regarded as $(\text{degree}(g)+\text{degree}(h_0))$-dimensional vectors, are linearly independent. The resultant $R \in \mathbb{Z}$ of h_0 and g is defined as the determinant of the matrix M having these vectors as columns. It follows that $R \neq 0$; from Hadamard's inequality and $\text{degree}(g) \le m$, we get

$$(5.4) \qquad |R| \le |h_0|^m |g|^m.$$

The polynomials h_0 and g, reduced modulo p^k, have h as a common divisor in $(\mathbb{Z}/p^k\mathbb{Z})[X]$ (remember that both are elements of L). Therefore, the columns of M cannot be linearly independent when regarded modulo p^k, so that R must be zero modulo p^k. Combined with $R \neq 0$, we conclude that $p^k \le |R|$, which implies, with (5.4), (5.3), and (5.2), that $|g| > 2^{m/2} B$. This proves our claim.

One problem remains to be solved, namely to determine the correct value of the degree m of h_0. This is simply done by applying the above construction for $m = \ell, \ell+1, \ldots, n-1$ in succession, where k is chosen such that (5.3) holds with m replaced by $n-1$:

$$(5.5) \qquad p^k > \left(2^{(n-1)/2}\binom{2(n-1)}{n-1}^{\frac{1}{2}} |f|\right)^{n-1}.$$

It follows from the above reasoning that for values of m smaller than the degree of h_0, the lattice does not contain any sufficiently short vectors, so that the first short vector that we find must be equal to $\pm h_0$ (note that the lattice of dimension $m+1$ contains all elements of the lattices of dimensions $\leq m$). If no short vector is found at all, then apparently $\text{degree}(h_0) > n-1$, so that $h_0 = f$.

Let us consider the running time of the L^3-algorithm. First, we observe that the factor h modulo p^k, with k such that (5.5) holds, can still be found in polynomial time. This follows from the running time estimates for the application of Hensel's lemma. Next, we see from (5.5) and the definition of the lattice, that the maximal number of bits in the coordinates of the basis for L is $O(k \log p) = O(n^2 + n \log|f|)$. Combined with the running time of the basis reduction algorithm (cf. Section 4), this implies that the applications of this algorithm can be done in time polynomial in n and $\log|f|$. We conclude that h_0 can be determined in polynomial time, so that f can also be factored in polynomial time.

A more careful description of the algorithm and analysis of its running time lead to the following theorem (cf. [II: Theorem 3.6]):

Theorem. *A primitive polynomial $f \in \mathbb{Z}[X]$ of degree n can be completely factored in $O(n^{12} + n^9(\log|f|)^3)$ bit operations.*

(Using the result from [9], which was mentioned in Section 4, this can be improved to $O(n^{11} + n^8(\log|f|)^3)$.)

If we apply Odlyzko's empirical result about the running time of the basis reduction algorithm, we get an $O(n^7 + n^4(\log|f|)^3)$ bound for the factoring algorithm. This indicates that the L^3-algorithm will not be of great practical value. This point will be further discussed in Section 7. First, we will consider some generalizations of the L^3-algorithm to other polynomial factoring problems.

6. Generalizations of the L^3-algorithm

In the previous section we have seen that primitive univariate polynomials with integral coefficients can be factored in polynomial time by an algorithm that essentially works as follows:

(i) compute a sufficiently precise, irreducible p-adic factor,

(ii) construct a lattice, such that the corresponding irreducible true factor is very short in this lattice, and

(iii) determine this true factor by means of the basis reduction algorithm.

In [III; V; VII; IV] we have shown that the same scheme can be used to formulate polynomial-time algorithms for various other kinds of polynomial factoring problems:

- polynomials in one variable with coefficients in an algebraic number field (cf. [III]; see [12] for an algorithm using norms, a technique due to Kronecker);
- multivariate polynomials with integral coefficients (cf. [V]);
- multivariate polynomials with coefficients in an algebraic number field (cf. [VII]);
- multivariate polynomials with coefficients in a finite field (cf. [IV]).

The algorithms described in these papers are polynomial-time in the length of a dense encoding of the polynomial to be factored. That is, application of one of our algorithms to a polynomial f of degree n_i in the variable X_i, for $1 \leq i \leq t$, can be done in time polynomial in $\prod_{i=1}^{t} n_i$ and the size of the coefficients of f.

It is well known that this is not a very realistic complexity measure for multivariate polynomials. Theoretically however, our algorithms compare favorably to the straightforward generalization of the Berlekamp-Hensel algorithm. For the latter nothing better can be proved than a bound that is exponential in each of the degrees n_i. To get a realistic complexity measure, the length of a sparse encoding of the input polynomial and its (output) factors has to be considered; Von zur Gathen has shown that in this case algorithms can be given that run in expected polynomial time [18].

This is certainly not the place to go into the numerous details of the generalizations of the L^3-algorithm; these can be found in the papers referred to above. Instead, let us describe some of the most important points of two of our generalizations, namely $f \in \mathbb{Z}[X_1, X_2]$ and $f \in \mathbb{F}_q[X_1, X_2]$.

First, we consider the case $f \in \mathbb{Z}[X_1, X_2]$; this case is treated in detail in [V]. Step (i) can be generalized as follows. Let p be a prime number and let s be an integer. Denote by s_1 the ideal generated by p and $X_2 - s$. Because $f \bmod s_1$ is a polynomial in $(\mathbb{Z}/p\mathbb{Z})[X_1]$, we can find its factorization in $(\mathbb{Z}/p\mathbb{Z})[X_1]$ by means of Berlekamp's algorithm. If p and s are chosen such that some square-freeness conditions are satisfied, then we can apply a generalized version of Hensel's lemma to lift the factorization of $f \bmod s_1$ to a factorization of f modulo the ideal s_k generated by p^k and $(X_2 - s)^{n_2+1}$, for any $k \in \mathbb{Z}_{>0}$ that we want. In this way we compute a sufficiently precise p-adic factor h of f. Denote by ℓ the degree in X_1 of h, and assume that h is monic with respect to the variable X_1. We now turn to step (ii).

As in the L^3-algorithm, assume that the true factor h_0 of f that corresponds to h has degree m_1 in X_1 (and of course degree at most n_2 in X_2). A basis for the set L of multiples of h modulo the ideal s_k, having degree $\leq m_1$ in X_1 and $\leq n_2$ in X_2, is given by the polynomials

$$\{p^k X_2^j X_1^i : 0 \leq j \leq n_2,\ 0 \leq i < \ell\}$$
$$\cup \{(h X_2^j \bmod s_k)\, X_1^{i-\ell} : 0 \leq j \leq n_2,\ \ell \leq i \leq m_1\}.$$

Regarding these polynomials as $((n_2+1)(m_1+1))$-dimensional integral vectors, we see that they are linearly independent, and that the set L is an $((n_2+1)(m_1+1))$-dimensional lattice.

As in the proof of Section 5, consider the resultant $R \in \mathbb{Z}[X_2]$ of h_0 and an arbitrary non-zero element g of L, for which $\gcd(h_0, g) = 1$ (so, $R \neq 0$). An upper bound for the length $|R|$ of the vector R, as a function of $|h_0|$ and $|g|$, can easily be derived. Using this bound and (2.2), we see that $(X_2 - s)^{n_2+1}$ cannot divide R if s is chosen sufficiently large. But $R \bmod s_k$ must be zero, because h divides both $h_0 \bmod s_k$ and $g \bmod s_k$, so that the absolute value of the maximal coefficient of $R \bmod (X_2 - s)^{n_2+1} \in \mathbb{Z}[X_2]$ must be at least p^k. This yields a lower bound for the length of the vector g as a function of p^k, and we conclude that we can get g as long as we want by choosing k sufficiently large. This means that h_0 can be determined by the basis reduction algorithm (notice that $|h_0|$ can be bounded from above by a result from [6], cf. Section 2).

The algorithm sketched here can easily be extended to more than two variables. The initial substitution $X_2 = s$ is then replaced by a substitution $X_2 = s_2$, $X_3 = s_3, \ldots, X_t = s_t$, where t is the number of variables. The details of this construction for a slightly more complicated case, namely with coefficients in an algebraic number field, are given in [VII].

Another way of generalizing the L^3-algorithm to $\mathbb{Z}[X_1, X_2, \ldots, X_t]$ is described in [VI]. There we apply the idea of the L^3-algorithm (i.e. a true factor is a short vector in a lattice) to formulate a polynomial-time reduction from factoring in $\mathbb{Z}[X_1, X_2, \ldots, X_t]$ to factoring in $\mathbb{Z}[X_1]$. For $\mathbb{Z}[X_1, X_2]$ this reduction follows from the above algorithm by replacing the factor modulo p^k and $(X_2 - s)^{n_2+1}$ by a factor modulo $(X_2 - s)^{n_2+1}$ only.

Other polynomial-time algorithms for factoring in $\mathbb{Z}[X_1, X_2, \ldots, X_t]$ are given by Kaltofen in [8] and Chistov and Grigoryev in [3]. As in [VI], Kaltofen reduced the problem of factoring in $\mathbb{Z}[X_1, X_2, \ldots, X_t]$ in polynomial time to factoring in $\mathbb{Z}[X_1]$; his reduction is completely different from ours. Chistov and Grigoryev applied some of Kaltofen's ideas and developed yet another reduction, this time from factoring in $\mathbb{Z}[X_1, X_2, \ldots, X_t]$ to factoring in $\mathbb{Z}[X_1]$ and $(\mathbb{Z}/p\mathbb{Z})[X_1, X_2, \ldots, X_t]$ (for the latter problem see below). All these algorithms can, in some way or another, be extended to polynomials with coefficients in an algebraic number field.

We now come to a different kind of generalization of the L^3-algorithm, an algorithm to factor polynomials in $\mathbb{F}_q[X_1, X_2, \ldots, X_t]$, where $\mathbb{F}_q$ is a finite field containing q elements (cf. [IV]). We will briefly discuss this algorithm for $f \in \mathbb{F}_q[X_1, X_2]$. (An algorithm very similar to the one described here was independently discovered by Chistov and Grigoryev [3].)

In fact, the algorithm follows immediately from the L^3-algorithm by replacing the ring of integers $\mathbb{Z}$ by the ring of polynomials $\mathbb{F}_q[X_2]$. Consequently, the prime number p is replaced by an irreducible polynomial $F \in \mathbb{F}_q[X_2]$, and the factor modulo p^k by a factor modulo the ideal generated by F^k. Instead of a lattice in $\mathbb{Z}^{m+1}$, we now get an $\mathbb{F}_q[X_2]$-module $L \subset \mathbb{F}_q[X_2]^{m+1}$. Of course, as norm for the elements of L, the Euclidean length does not make sense here. Therefore, for an element x of $\mathbb{F}_q[X_2]^{m+1}$, we define the norm as the maximal degree in X_2 of any of the coordinates of x; so, 'short' means 'small degree in X_2'. Using this norm one easily proves that the true factor corresponding to a factor modulo (F^k) is the shortest element of L if k is sufficiently large (the proof follows the lines of

the proof in Section 5, but is much simpler).

Obviously we cannot use the basis reduction algorithm. To determine the shortest vector in this case, we remark that the shortest vector problem in $L \subset \mathbb{F}_q[X_2]^{m+1}$ can be formulated as a system of linear equations over $\mathbb{F}_q$. This gives a polynomial-time solution of the shortest vector problem at hand (cf. [IV; 3]). This concludes our brief explanation of the algorithm for $f \in \mathbb{F}_q[X_1, X_2]$.

Generalization to more than two variables follows in a similar way as we have seen before, namely by means of substitutions $X_3 = X_2^{k_3}, X_4 = X_2^{k_4}, \ldots, X_t = X_2^{k_t}$, for sufficiently large integers k_i (this is quite different from the way in which Chistov and Grigoryev extend their method to more variables [3]). In [10] another polynomial-time algorithm for $\mathbb{F}_q[X_1, X_2, \ldots, X_t]$-factoring is given.

A more general approach to the generalizations of the L^3-algorithm will be presented in [17].

7. Practical algorithms

The algorithms from the previous sections have a worst-case running time that is bounded by a polynomial function of the length of a dense encoding of the input polynomial. Apart from the fact that a dense encoding gives an unrealistic complexity measure, the polynomial bound on the running time also does not imply that the algorithms are practical. Although the algorithms will perform much better than is suggested by their running times as analyzed in [II; V; VI; III; VII], we cannot expect them to be fast. This follows from Odlyzko's empirical running time of the basis reduction algorithm and from the dimension and coordinate-size of the lattices to which the basis reduction algorithm is applied. For instance, to factor $f \in \mathbb{Z}[X_1, X_2, \ldots, X_t]$ of degree n_i in X_i, the running time will be proportional to

$$\Pi_{i=1}^{t} n_i^7 + (\Pi_{i=1}^{t} n_i^4)(\log|f|)^3$$

at least (the theoretical bound from [VI] is $O(n_1^{3t-3}(\Pi_{i=1}^{t} n_i^{12} + (\Pi_{i=1}^{t} n_i^9)(\log|f|)^3)))$. Hence, even for reasonably small values of n_i the running time will become prohibitively long.

The question arises which algorithms should be used in practice. For polynomials in one variable with integral coefficients the Berlekamp-Hensel algorithm usually performs very well, and if it does not, which is quite unlikely, we can apply the L^3-algorithm. For multivariate polynomials with integral coefficients we have the generalizations of the Berlekamp-Hensel algorithm, as mentioned in Section 3 (cf. [19]). These algorithms apply a reduction from the $\mathbb{Z}[X_1, X_2, \ldots, X_t]$-factoring problem to $\mathbb{Z}[X_1]$-factoring, for which nothing better than an exponential-time bound can be proved. This reduction however, appears to be very fast in practice, and the resulting factoring algorithm can be strongly recommended.

The same is true for polynomials with coefficients in an algebraic number field; the exponential-time generalized Berlekamp-Hensel algorithm (cf. [20]) will perform better than the polynomial-time generalized L^3-algorithm. It looks as though these factoring algorithms that use lattices and the basis reduction algorithm have merely theoretical value. Fortunately, this is not the case, as we have shown in [I], where an algorithm is described to factor polynomials with coefficients in an algebraic number field. The algorithm is based on the Berlekamp-Hensel algorithm, and therefore not polynomial-time, but it is much faster than the methods from [20; 21], which are also of the Berlekamp-Hensel type. The reason of the speed of our algorithm is that, up to the search for the true factors, no computations have to be performed on algebraic numbers. Instead, all computations can be done in $\mathbb{Z}/p^k\mathbb{Z}$ for a suitably chosen prime power p^k. We will briefly explain this algorithm.

Suppose that the algebraic number field $\mathbb{Q}(\alpha)$ is given as the field of rational numbers $\mathbb{Q}$ extended by a root α of a prescribed *minimal polynomial* $F \in \mathbb{Z}[T]$, i.e. $\mathbb{Q}(\alpha) \simeq \mathbb{Q}[T]/(F)$. Let $f \in \mathbb{Q}(\alpha)[X]$ be the polynomial to be factored. In the older algorithms (cf. [20; 21]) one tries to find a prime number p such that $F \bmod p$ is irreducible in $(\mathbb{Z}/p\mathbb{Z})[T]$, and such that some other conditions are satisfied. If such a prime number can be found, then the Berlekamp-Hensel approach can immediately be generalized by observing that $(\mathbb{Z}/p\mathbb{Z})[T]/(F \bmod p)$ is a field. Several techniques are developed for the case that such a prime number cannot be found. In [20] the problem is reduced to the problem of factoring a polynomial of much higher degree in $\mathbb{Z}[X]$ (a technique that uses norms, as in [12]). In [21] the p-adic factorization of the minimal polynomial is used to compute several p-adic factorizations of f; the true factors of f are then determined by means of the Chinese remainder

algorithm combined with a combinatorial search. In both cases the algorithms from [20; 21] are rather slow due to the time consuming computations in the algebraic number field.

The algorithm from [I] proceeds as follows. First, a prime number p is chosen such that, among other conditions, $F \bmod p$ has a linear factor $H \bmod p$ in $(\mathbb{Z}/p\mathbb{Z})[T]$. The polynomial f reduced modulo p and $H \bmod p$ is, due to the linearity of $H \bmod p$, contained in $(\mathbb{Z}/p\mathbb{Z})[X]$, and can easily be factored by means of Berlekamp's algorithm. This factorization can be lifted to a factorization of f modulo p^k and $H \bmod p^k$ for a sufficiently large value of k by means of Hensel's lemma, where $H \bmod p^k$ is the lifted linear factor of $F \bmod p^k$ in $(\mathbb{Z}/p^k\mathbb{Z})[T]$.

To find the true factors of f in $\mathbb{Q}(\alpha)[X]$ we look, as usual, at combinations of the p-adic factors of f. These combinations however are polynomials in $(\mathbb{Z}/p^k\mathbb{Z})[X]$, so we must be able to reconstruct the coefficients in $\mathbb{Q}(\alpha)$ from their images modulo p^k and $H \bmod p^k$. Let x be such a coefficient of a true factor, and let $\tilde{x}$ be its image modulo p^k and $H \bmod p^k$, so $\tilde{x} \in \mathbb{Z}/p^k\mathbb{Z}$. For simplicity we will assume that $x \in \mathbb{Z}[\alpha] = \mathbb{Z}[T]/(F)$. We will show how x can be found given $\tilde{x}$. If we regard x and $\tilde{x}$ as (degree(F))-dimensional integral vectors, then they are congruent modulo the (degree(F))-dimensional integral lattice

$$L = \{p^k r + (H \bmod p^k)\, q : r,q \in \mathbb{Z}[\alpha],\ \text{degree}(r) = 0,\ \text{degree}(q) < \text{degree}(F)-1\}.$$

Because x is a coefficient of a true factor, the vector x is of bounded length. From the same proof as in Section 5 we conclude that we can choose k such that all non-zero elements of L are much longer than x. This implies that x is the shortest element of $\mathbb{Z}^{\text{degree}(F)}$ that is congruent to $\tilde{x}$ modulo L, so that x can be found by reducing $\tilde{x}$ modulo a sufficiently orthogonal basis for L. As mentioned in Section 4 such a basis for L can be computed by the basis reduction algorithm.

We conclude that, during the search for the true factors, all algebraic numbers can be reconstructed in a unique way by means of one application of the basis reduction algorithm. This is the only practical application of lattices and the basis reduction algorithm to factoring polynomials that we know of up till now.

References

I. A.K. Lenstra, *Lattices and factorization of polynomials over algebraic number fields*, Proceedings Eurocam '82, European computer algebra conference, LNCS 144, 32-39.

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

III. A.K. Lenstra, *Factoring polynomials over algebraic number fields*, rep. IW 213/82, Mathematisch Centrum, Amsterdam; extended abstract in Proceedings Eurocal '83, European computer algebra conference, LNCS 162, 245-254.

IV. A.K. Lenstra, *Factoring multivariate polynomials over finite fields*, rep. IW 221/83, Mathematisch Centrum, Amsterdam; to appear in the special STOC issue of *Computer and system sciences*.

V. A.K. Lenstra, *Factoring multivariate integral polynomials*, Proceedings 10-th international colloquium on automata, languages and programming, LNCS 154, 458-465; to appear in the special ICALP issue of *Theoretical computer science*.

VI. A.K. Lenstra, *Factoring multivariate integral polynomials, II*, rep. IW 230/83, Mathematisch Centrum, Amsterdam.

VII. A.K. Lenstra, *Factoring multivariate polynomials over algebraic number fields*, rep. IW 233/83, Mathematisch Centrum, Amsterdam.

1. E.R. Berlekamp, *Factoring polynomials over large finite fields*, Math. Comp. 24 (1970), 713-735.

2. W.S. Brown, *The subresultant PRS algorithm*, ACM Transactions on mathematical software 4 (1978), 237-249.

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

4. G.E. Collins, *Factoring univariate integral polynomials in polynomial average time*, Proceedings Eurosam '79, 317-329.

5. U. Dieter, *How to calculate shortest vectors in a lattice*, Math. Comp. 29 (1975), 827-833.

6. A.O. Gel'fond, *Transcendental and algebraic numbers*, Dover Publ., New York 1960.

7. E. Kaltofen, D.R. Musser, B.D. Saunders, *A generalized class of polynomials that are hard to factor*, Proceedings 1981 ACM symposium on symbolic and algebraic computation, 188-194.

8. E. Kaltofen, *On the complexity of factoring polynomials with integer coefficients*, Ph.D. thesis, Rensselaer Polytechnic Institute, August 1982.

9. E. Kaltofen, *On the complexity of finding short vectors in integer lattices*, Proceedings Eurocal '83, European computer algebra conference, LNCS 162, 236-244.

10. J. Von zur Gathen, E. Kaltofen, *Polynomial-time factorization of multivariate polynomials over finite fields*, Proceedings 10-th international colloquium on automata, languages and programming, LNCS 154, 250-263.

11. D.E. Knuth, *The art of computer programming, Vol. 2, Seminumerical algorithms*, Addison Wesley, Reading, second edition 1981.

12. S. Landau, *Factoring polynomials over algebraic number fields*, to appear.

13. H.W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Math. Oper. Res. 8 (1983), 538-548.

14. M. Mignotte, *An inequality about factors of polynomials*, Math. Comp. 28 (1974), 1153-1157.

15. M. Pohst, *On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications*, Sigsam Bulletin Vol. 15, number 1 (1981), 37-44.

16. P. Van Emde Boas, *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*, Rep. Dep. Math. 81-04, Univ. of Amsterdam, April 1981.

17. J. Von zur Gathen, *Hensel and Newton methods in valuation rings*, Math. Comp., to appear.

18. J. Von zur Gathen, *Factoring sparse multivariate polynomials*, Proceedings 24-th annual symposium on foundations of computer science (1983), 172-179.

19. P.S. Wang, L.P. Rothschild, *Factoring multivariate polynomials over the integers*, Math. Comp. 29 (1975), 935-950.

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

21. P.J. Weinberger, L.P. Rothschild, *Factoring polynomials over algebraic number fields*, ACM Transactions on mathematical software 2 (1976), 335-350.

22. D.Y.Y. Yun, *The Hensel-lemma in algebraic manipulation*, MIT, Cambridge 1974; reprint: Garland Publ. Co., New York 1980.

23. H. Zassenhaus, *On Hensel factorization, I*, J. Number Theory 1 (1969), 291-311.