Chapter 8
The van Hoeij Algorithm for Factoring Polynomials

Jürgen Klüners

Abstract In this survey, we report about a new algorithm for factoring polynomials due to Mark van Hoeij. The main idea is that the combinatorial problem that occurs in the Zassenhaus algorithm is reduced to a very special knapsack problem. In case of rational polynomials, this knapsack problem can be very efficiently solved by the LLL algorithm. This gives a polynomial time algorithm, which also works very well in practice.

Introduction

Let $f \in \mathbb{Z}[x]$ be a polynomial of degree n with integral coefficients. One of the classical questions in computer algebra is how to factorize f in an efficient way. About 40 years, ago Hans Zassenhaus [1] developed an algorithm, which was implemented in almost all computer algebra systems until 2002. This algorithm worked very well for many examples, but his worst case complexity was exponential. In the famous LLL–paper [2], it was proved that it is possible to factor polynomials in polynomial time in the degree and the (logarithmic) size of the coefficients. Despite the fact that the new lattice reduction algorithm was very good in theory and in practice, the new polynomial factorization was not used in implementations. For most practical examples, the Zassenhaus algorithm was more efficient than the new algorithm based on LLL.

In 2002, Mark van Hoeij [3] developed a new algorithm, which, for practical examples, was much more efficient than the Zassenhaus algorithm. This new algorithm is also based on the LLL reduction, but it uses a different type of lattices compared to the ones in the original LLL paper [2]. Unfortunately, Mark van Hoeij gave no running time estimates in his original paper. He was only able to show that his algorithm terminates, but he gave very impressive practical examples of factorizations, which have not been possible to compute before. Together with Karim

J. Klüners
Mathematisches Institut, Universität Paderborn, Warburger Str. 100, 30098 Paderborn, Germany.
e-mail: klueners@math.uni-paderborn.de

Belabas, Mark van Hoeij, and Allan Steel [4], the author of this survey simplified the presentation of this algorithm and introduced a variant for factoring bivariate polynomials over finite fields. Furthermore, we have been able to show that the new factoring algorithm runs in polynomial time. The worst case estimate is better than the one given in the original LLL paper. Let us remark that we believe that this estimate is still pessimistic.

In this survey, we will study the cases $f \in \mathbb{Z}[x]$ and $f \in \mathbb{F}_p[t][x]$ in parallel, where $\mathbb{F}_p$ denotes the finite field with p elements. We study the second case because the presentation is easier. In the bivariate case, it is not necessary to use LLL reduction. The corresponding step can be solved by computing kernels of systems of linear equations.

The Zassenhaus Algorithm

Before we are able to explain the van Hoeij algorithm, we need to understand the Zassenhaus algorithm. We can assume that our given polynomial is squarefree. Note that multiple factors of f divide the greatest common divisor of f and the derivative f', which can be computed efficiently using the Euclidean algorithm. We remark that we have to be careful in characteristic p since the derivative of f may be 0, e.g., $f(x) = x^p - t$. In this case, all monomials are p-th powers, and we can take the p-th root or we switch the role of t and x.

Let $f \in \mathbb{Z}[x]$ be a squarefree and monic polynomial, i.e., a polynomial with integer coefficients and leading coefficient one. In the Zassenhaus algorithm, we choose a prime number p such that f modulo p has no multiple factors. It is possible to choose every prime that does not divide the discriminant of f. We denote by $\bar{f} \in \mathbb{F}_p[x]$ the polynomial that can be derived out of f by reducing each coefficient modulo p. Using well known algorithms, e.g., see [5, Chap. 14], we can compute the following factorization:

$$\bar{f}(x) = \bar{f}_1(x) \cdots \bar{f}_r(x) \in \mathbb{F}_p[x].$$

Using the so-called Hensel lifting (by solving linear systems of equations), we can efficiently compute for all $k \in \mathbb{N}$ a factorization of the following form: (e.g., see [5, Chap. 15])

$$f(x) \equiv \tilde{f}_1(x) \cdots \tilde{f}_r(x) \bmod p^k,$$

where (using a suitable embedding) $\tilde{f}_i \equiv \bar{f}_i \bmod p$. Let us explain the Zassenhaus algorithm using the example $f(x) = x^4 - 11$. Using $p = 13$, we get that $\bar{f} \in \mathbb{F}_{13}[x]$ is irreducible. Certainly, this implies that $f \in \mathbb{Z}[x]$ is irreducible. When we choose (a little bit unlucky) $p = 5$, then we only get linear factors modulo p, which will be lifted using Hensel lifting:

$$f(x) \equiv (x + 41)(x - 38)(x + 38)(x - 41) \bmod 125.$$

To proceed, we need a bound for the size of the coefficients of a factor of f. The following theorem can be found in [5, p.155ff].

Theorem 1 (Landau-Mignotte). *Let g be a factor of a monic polynomial $f \in \mathbb{Z}[x]$ with*

$$f(x) = \sum_{i=0}^{n} a_i x^i \text{ and } g(x) = \sum_{i=0}^{m} b_i x^i.$$

Then: $|b_i| \leq \binom{m}{i} ||f||_2$, where $||f||_2 := \sqrt{\sum_{i=0}^{n} a_i^2}$ denotes the 2–norm.

In our example, this means that all coefficients of a factor g of f must be less than or equal to 33 in absolute value. Therefore, we see that f has no linear factor, because modulo 125 all linear factors contain a coefficient in the symmetric residue system $\{-62, \dots, 62\}$, which is bigger than 33 in absolute value. In the next step, we try if the product of two modulo 125 factors corresponds to a true factor of f in $\mathbb{Z}[x]$. We get:

$$\begin{aligned}
(x+41)(x-38) &\equiv x^2 + 3x - 58 \mod 125,\\
(x+41)(x+38) &\equiv x^2 - 46x + 58 \mod 125,\\
(x+41)(x-41) &\equiv x^2 - 56 \mod 125.
\end{aligned}$$

All these quadratic polynomials contain a coefficient that is bigger than 33 in absolute value. This means that the "modular factor" $(x + 41)$ is no divisor of a linear or quadratic factor $g \in \mathbb{Z}[x]$ of f. This implies that the polynomial $f \in \mathbb{Z}[x]$ is irreducible. In case that our given polynomial is reducible, we find modular factors of f such that all coefficients in the symmetric residue system are smaller than the Landau–Mignotte bound. Now, we can use trial division in $\mathbb{Z}[x]$ to check if we have found a factor or not. Since trial divisions are expensive, it is a good idea in actual implementations to choose p^k much bigger than twice the Landau–Mignotte to increase the probability that wrong candidates will be found without a trial division.

The choice of $p = 5$ in the above example was very artificial. We remark that it is easy to construct irreducible polynomials such that for all primes p, we have many modular factors. For example, we can take a polynomial f of degree $n = 2^\ell$ such that the Galois group is isomorphic to the elementary abelian group $(\mathbb{Z}/2Z)^n$. In this case, f is irreducible, but for every prime p, we have at least $2^{\ell-1} = n/2$ modular factors.

If we analyze the Zassenhaus algorithm, we figure out that most parts of the algorithm are very efficient. Since we are able to choose a small prime p, it is not difficult to factor $\bar{f} \in \mathbb{F}_p[x]$. The Hensel lifting can be solved using linear systems of equations, and the Landau-Mignotte bound is sufficiently small. The drawback of the algorithm is the number of tests that have to be performed when the number r of modular factors is big. In this case, we have to perform more or less 2^r tests.

The Knapsack Lattice

This is the place where the new van Hoeij algorithm starts. It reduces the combinatorial problem to a so-called knapsack problem. The resulting knapsack problem can be efficiently solved using lattices and the LLL algorithm. We remark that we use different type of lattices compared to the original LLL paper.

We fix the following notation:

$$f = g_1 \cdots g_s \in \mathbb{Z}[x] \text{ and } f = \tilde{f}_1 \cdots \tilde{f}_r \in \mathbb{Z}_p[x].$$

The factorization over the p-adic numbers $\mathbb{Z}_p$ can only be determined modulo p^k. For the understanding of the following, it is possible to interpret the p-adic numbers as modulo p^k approximations. We write:

$$g_v := \prod_{i=1}^{r} \tilde{f}_i^{v_i} \text{ for } v = (v_1, \ldots, v_r) \in \{0,1\}^r$$

and get a new

Problem 1. For which $v \in \{0,1\}^r$, do we have: $g_v \in \mathbb{Z}[x]$?

To linearize our problem, we consider (more or less) the logarithmic derivative, where $\mathbb{Q}_p(x) := \{\frac{a(x)}{b(x)} \mid a, b \in \mathbb{Q}_p[x]\}$:

$$\Phi : \mathbb{Q}_p(x)^*/\mathbb{Q}_p^* \to \mathbb{Q}_p(x), g \mapsto \frac{fg'}{g}.$$

It is immediately clear that Φ is additive, i.e., $\Phi(g_{v_1}) + \Phi(g_{v_2}) = \Phi(g_{v_1+v_2})$. Furthermore, we have for $v \in \mathbb{Z}^r$ that $\Phi(g_v)$ is a polynomial and therefore an element of $\mathbb{Z}_p[x]$.

The next step is to translate everything into a lattice problem. Let us define vectors $w_1, \ldots, w_s \in \{0,1\}^r$ such that for the true factors $g_1, \ldots, g_s \in \mathbb{Z}[x]$, we have

$$g_i = \prod_{1 \le j \le r} \tilde{f}_j^{w_{ij}}.$$

These vectors generate a lattice (the knapsack lattice) $W = \langle w_1, \ldots, w_s \rangle \subseteq \mathbb{Z}^r$. The lattice $\mathbb{Z}^r$ is generated by the standard basis vectors, which correspond to the local factors $\tilde{f}_i$. An important fact for the new method is the property that $v \in \mathbb{Z}^r$ is an element of W only if $\Phi(g_v) \in \mathbb{Z}[x]$ (even in the case that v has negative coefficients). We remark that it is easy to construct the canonical basis vectors $w_1, \ldots, w_s$ of W if we know some generating system of W. As soon as we know the w_i and p^k is at least twice larger than the Landau-Mignotte bound, we are able to reconstruct the corresponding factors g_i like in the Zassenhaus algorithm.

The idea of the algorithm is as follows. We start with the lattice $L = \mathbb{Z}^r$ and know that $W \subseteq L$. Then, we construct a sublattice $L' \subset L$ still containing W. The hope is that after finitely many steps, we will reach $L' = W$.

At this place, we change to the situation $f \in \mathbb{F}_p[t][x]$, because the following part of the algorithm is easier here. The Landau-Mignotte bound simplifies to

$$g \mid f \in \mathbb{F}_p[t][x] \Rightarrow \deg_t(g) \leq \deg_t(f),$$

where $\deg_t(f)$ is the t–degree of the polynomial f. In order to simplify the situation, we assume that $\bar{f}(x) := f(0,x) \in \mathbb{F}_p[x]$ is squarefree. This is a real restriction because it might be the case that $f(a,x)$ has multiple factors for all $a \in \mathbb{F}_p$. For the solution of this problem, we refer the reader to [4]. Using Hensel lifting, we get from the factorization $\bar{f} = \bar{f}_1 \cdots \bar{f}_r \in \mathbb{F}_p[x]$ a factorization $f(t,x) = \tilde{f}_1 \cdots \tilde{f}_r$ in the power series ring $\mathbb{F}_p[[t]][x]$. In practice, we can approximate the power series in t modulo t^k. Now, we define the function Φ in the following way:

$$\Phi : \mathbb{F}_p[[t]](x)^*/\mathbb{F}_p[[t]](x^p)^* \to \mathbb{F}_p[[t]](x), g \mapsto \frac{fg'}{g}.$$

The lattices L and W are defined analogously as in the situation for $\mathbb{Z}[x]$. Assume that we have an element $v \in L \setminus W$. Then, we have:

$$\begin{aligned}\mathrm{Pol}(v) := \Phi(g_v)(x) &= \sum_{i=1}^{r} v_i \Phi(f_i) \\ &= \sum_{i=0}^{n-1} b_i x^i \in \mathbb{F}_p[[t]][x] \setminus \mathbb{F}_p[t][x].\end{aligned}$$

Additionally, we have for $g_v \in \mathbb{F}_p[t][x]$ the estimate $\deg_t(b_i) \leq \deg_t(f)$. Now, we choose a $k > \deg_t(f)$, and we compute for $v \in L$ the corresponding polynomial

$$g_v \equiv \sum_{i=0}^{n-1} b_i(t)x^i \mod t^k.$$

Here, modulo t^k means that all $b_i(t)$ are reduced modulo t^k, i.e., $\deg_t(b_i) < k$. In case that one of the polynomials b_i has a t-degree that is bigger than $\deg_t(f)$, we know that the corresponding v is not an element of W. In the following, we avoid the combinatorial approach.

Denote by $e_1, \ldots, e_r \in \mathbb{F}_p^r$ the standard basis of $\mathbb{F}_p^r$, and identify the elements of $\mathbb{F}_p$ with $\{0, \ldots, p-1\} \subseteq \mathbb{Z}$. We define $m := \deg_t(f)$ and

$$A_i := \begin{pmatrix} b_{i,m,1} & \cdots & b_{i,m,r} \\ b_{i,m+1,1} & \cdots & b_{i,m+1,r} \\ \vdots & \ddots & \vdots \\ b_{i,k-1,1} & \cdots & b_{i,k-1,r} \end{pmatrix} \in \mathbb{F}_p^{(k-m)\times r},$$

where the $b_{i,j,\ell}$ are given by

$$\mathrm{Pol}(e_\ell) \equiv \sum_{i=0}^{n-1}\sum_{j=0}^{k-1} b_{i,j,\ell} t^j x^i \bmod t^k \quad (1 \leq \ell \leq r).$$

All $v \in W$ have the property that $A_i v^{\mathrm{tr}} = 0$. Using iterative kernel computation, we are able to determine lattices $L' \supseteq W$, which (hopefully) become smaller.

The Polynomial Complexity Bound

In a slightly improved version of this algorithm, we show in [4] that we finally get $L' = W$:

Theorem 2. *Let $f \in \mathbb{F}_p[t][x]$ be a polynomial of x-degree n and assume $k > (2n-1)\deg_t(f)$. Then, W is the kernel of $A_1, \ldots, A_{n-1}$.*

In the following, we give a sketch of the proof of this theorem. Let $v \in L \setminus W$ be chosen such that g_v is not a p-th power. Then, it is possible to change v using $w_1, \ldots, w_s$ such that the following holds:

1. $f_i \mid \mathrm{Pol}(v)$ for some $1 \leq i \leq r$.
2. $g_j \nmid \mathrm{Pol}(v)$ for all $1 \leq j \leq s$.

Take this new v and define $H := \mathrm{Pol}(v) \bmod t^k$ interpreted as a polynomial in $\mathbb{F}_p[t][x]$. Using well known properties of the resultant, we immediately get:

$$\mathrm{Res}(f, \mathrm{Pol}(v)) = 0 \text{ and } \mathrm{Res}(f, H) \neq 0.$$

This implies that $t^k \mid \mathrm{Res}(f, H)$. Choosing k large enough is a contradiction to the definition of the resultant via the Sylvester matrix.

Let us come back to our original problem over $\mathbb{Z}$. We cannot apply the same algorithm because we have overflows when we add, e.g., $(3 + 1 \cdot 5^1) + (3 + 1 \cdot 5^1) = (1 + 3 \cdot 5^1) \neq 1 + 2 \cdot 5^1$ in $\mathbb{Z}_5$. Fortunately, we can show that the errors coming from overflows are small. Instead of solving linear systems of equations, we define a suitable lattice and look for vectors of small length in that lattice. Since finding shortest vectors in lattices is an NP-complete problem, it is important to choose the lattices in a very clever way. For those lattices, we apply LLL reduction and can guarantee that the first basis vectors of the LLL reduced basis will be sufficient to derive a basis of W. We use the analogous definitions as in the bivariate case and get:

$$\mathrm{Pol}(e_\ell) \equiv \sum_{i=0}^{n-1} b_{i,\ell} x^i \bmod p^k \quad (1 \le \ell \le r).$$

Now, we define a lattice Λ, which is defined by the columns of the following matrix:

$$A := \begin{pmatrix} I_r & 0 \\ \tilde{A} & p^k I_n \end{pmatrix} \text{ with } \tilde{A} := \begin{pmatrix} b_{0,1} & \cdots & b_{0,r} \\ \vdots & \ddots & \vdots \\ b_{n-1,1} & \cdots & b_{n-1,r} \end{pmatrix}.$$

If we project a vector from Λ to the first r rows, we get a vector in L. Assuming that we choose the precision p^k large enough, we are able to prove that all vectors in Λ, such that the last n entries are smaller than the Landau-Mignotte bound, correspond to a vector in W. We compute an LLL reduced basis of the above lattice and are able to prove that the first s vectors correspond to a basis of W. It is easy to give an upper bound B for the norm for the vectors in Λ, which correspond to $w_1, \dots, w_s$. For the LLL approach and practical implementations, the following lemma is very useful, since it allows to have some progress, i.e., a new lattice $W \subseteq L' \subseteq L$, if the precision p^k was not large enough to derive $L' = W$.

Lemma 1. *Let Λ be a lattice with basis $b_1, \dots, b_m$ and Gram–Schmidt–basis $b_1^*, \dots, b_m^*$. Define $t := \min\{i \mid \forall i < j \le m : ||b_j^*||_2 > B\}$. Then, all vectors b, such that $||b||_2 \le B$, are contained in $\mathbb{Z}b_1 + \cdots + \mathbb{Z}b_t$.*

We remark that this lemma is already contained in the original LLL paper [2]. Analogous to the bivariate case, we need an estimate for the precision, which guarantees that the algorithm terminates, i.e., that finally we have $L' = W$. If we use this precision, our algorithm terminates in one (LLL reduction) step and we get the polynomial running time. We remark that, in practice, we do not need this (explicit) estimate, because we start with some precision and increase it until the algorithm terminates.

Theorem 3. *Let $f \in \mathbb{Z}[X]$ of degree n. Then, the above described algorithm terminates if*

$$p^k > c^n \cdot 4^{n^2} ||f||_2^{2n-1} \tag{8.1}$$

holds, where c is an explicit computable constant.

If we determine the running time for this algorithm, we get the same running time as the one in the original LLL paper.

The Original LLL Factoring Method

In order to understand the difference between those two algorithms, we need to understand the original LLL factoring method, at least roughly. As before, let $f \in \mathbb{Z}[x]$ of degree n be the monic polynomial; we would like to factor and assume that we have chosen a prime p such that f mod p has no multiple factors. As before, using Hensel lifting, we get the factorization of the form:

$$f(x) \equiv \tilde{f}_1(x) \cdots \tilde{f}_r(x) \bmod p^k.$$

The idea of the original LLL factorization algorithm is to compute an irreducible factor $g \in \mathbb{Z}[x]$ such that $\tilde{f}_1 \mid g \in \mathbb{Z}_p[x]$. To compute such a g, they write down a tricky lattice that allows to check if there exists such a g of degree $m < n$. After an LLL reduction of this lattice, the first basis vector corresponds to such a g if it exists. If no such g exists, we have to use the theoretical lifting bound (similar to the one we have given in the van Hoeij algorithm) to get a proof that such a g does not exist. This is the practical drawback of the original algorithm. In case that our polynomial is reducible, we might be lucky to find g using a small precision. In case that f is irreducible, we have to use the full theoretical precision in order to get a proof that no nontrivial factor exists. We remark that after some recursive application of this algorithm (we find at most one factor at each LLL-step), we run into this situation. By taking irreducible polynomials, which have many modular factors, we get examples that really need the worst case running time we computed before.

Usually, in the van Hoeij algorithm, such a polynomial will be detected using a very low precision. What we say here is just heuristics and practical experience. We expect that the worst case running time we have given before is just a very bad upper estimate and will never be attained. We remark that we cannot prove such a statement.

One important fact for practical implementations of the van Hoeij algorithm is the fact that it is possible to make partial progress. If we choose a precision that was too small to derive the full factorization, it is very often the case that we are able to compute a smaller lattice L', which means that this "try" was not worthless. If in the original LLL factorization algorithm we use a too small precision and do not succeed, we have nothing.

There is another advantage of the new van Hoeij algorithm compared to the original one. In the original algorithm, we try to compute a factor directly. This means if the coefficients of the given polynomial are big, we need to compute a factor that has big coefficients as well. Therefore, we want to find a short(est) vector in a lattice, which is already huge. In the knapsack approach of van Hoeij, we are looking to find a zero-one combination. The shortest vector we are looking for is really very short (only zeroes and ones and some small errors coming from overflows). In some sense, those lattices are independent on the size of the coefficients of the given polynomial.

In the meantime, the van Hoeij algorithm is implemented in all big computer algebra systems. As already remarked, it is possible to factor polynomials in a few minutes, for which it was impossible to factor those in month before.

References

1. H. Zassenhaus, *On Hensel factorization I*, Journal of Number Theory, **1**, 291–311 (1969)
2. A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovász, *Factoring polynomials with rational coefficients*, Annals of Mathematics, **261**(4), 515–534 (1982)
3. M. van Hoeij, *Factoring polynomials and the knapsack problem*, Journal of Number Theory, **95**, 167–189 (2002)
4. K. Belabas, M. van Hoeij, J. Klüners and A. Steel, *Factoring polynomials over global fields*, Journal de théorie des nombres de Bordeaux, **21**(1), 15–39 (2009)
5. J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge University Press (1999)