Polynomial time factorization of univariate polynomials over the integers using LLL lattice reduction*

Jeffrey De Fauw

January 10, 2014

1 Introduction

Most of the well-performing factorization algorithms for polynomials over the integers make heavy use of the quotient rings $\mathbb{Z}/m$, for $m \in \mathbb{N}$, by doing (efficient) factorizations there and then trying to "lift" them to $\mathbb{Z}$.[vH13][vH13s] To expand on this idea, we start by giving a rough overview of the algorithm we implemented.

Let f be a (univariate) polynomial over the integers (i.e., in $\mathbb{Z}[x]$) of degree N.[1]

(Step 0.)
Determine some constants. In particular, generate a pseudo-random pair (p, l), p prime and $l \in \mathbb{N}$.

Step 1.
Factorize f as $f \equiv h_1 \cdots h_r \pmod{p}$, with h_i irreducible polynomials over $\mathbb{Z}/p$.

Step 2.
Factorize f as $f \equiv g_1 \cdots g_r \pmod{p^l}$, with g_i irreducible over $\mathbb{Z}/p^l$ and such that $g_i \equiv h_i \pmod{p}$, by Hensel lifting the factorization over $\mathbb{Z}/p$.

Step 3.
Recombine the factors using a "short vector" obtained by LLL[2] (lattice basis) reduction.

[We focus mainly on Step 3. For Step 1 we use the built-in methods of SAGE.]

Because f has more roots in the quotient rings of $\mathbb{Z}$ – since a factorization over $\mathbb{Z}$ implies one over $\mathbb{Z}/m$ but not the other way around – irreducible factors over $\mathbb{Z}$ *split* over some $\mathbb{Z}/m$. Step 3 tries to find all the (irreducible) *modular factors* g_i in $\mathbb{Z}/p^l[x]$ which belong to the same irreducible factor f_j of f in $\mathbb{Z}[x]$ ("(re)combining the factors") and lifts them to f_j.

This third step, in general, uses certain bounds (in our case Mignotte's bound) obtained for norms on the coefficient vectors of polynomials to determine if we can lift an equality over $\mathbb{Z}/p^l$ to one over $\mathbb{Z}$. It is an extension to polynomials of the following simple idea: if we have $a \equiv 0 \pmod{p^l}$, for some integer a, and we can bound $|a| < p^l$, then we can conclude $a = 0$. Although this sounds trivial, it is essential in the recombining of the factors.

A naive implementation of this idea is given by Zassenhaus' algorithm which simply tries every combination of modular factors until the right one is found (see [vzGG13]). As such, when f is irreducible

*Project Computeralgebra – Ghent University – 2013-2014

[1] Throughout we will assume the polynomials are all square-free, primitive, have leading coefficient 1 and satisfy any other assumption that is not very essential for the idea of the algorithm.

[2] Lenstra-Lenstra-Lovász

(over $\mathbb{Z}$), the algorithm has exponential complexity. Our implemented algorithm tries to improve on this exhaustive search by employing the LLL lattice basis reduction algorithm to find, for a certain modular factor g_i and a certain degree d higher than the degree of g_i, one polynomial $g^\star$ divisible by g_i over $\mathbb{Z}/p^l$ and of degree lower than d. If d is greater than or equal to the degree of the irreducible factor f_j of f corresponding with g_i, it will satisfy our bound and we can lift it to f_j in $\mathbb{Z}[x]$. We then remove all the modular factors g_k of f_j and repeat the search on the remaining ones. If it does not satisfy the bound we keep looking for $g^\star$ of higher degrees until we have found one or can conclude the irreducibility of the polynomial still to be factored.

In this implementation the complexity is completely determined by the effectiveness of finding a polynomial $g^\star$ of maximum degree d with the LLL algorithm. In the next section we go into more detail about this algorithm.

2 LLL lattice basis reduction

1 DEFINITION
A ($\mathbb{Z}$-)**lattice** L is a $\mathbb{Z}$-module which has an embedding into a vector space $\mathbb{R}^n$. Let L be generated by the vectors $f_1, \ldots, f_n \in \mathbb{R}^n$ (as a $\mathbb{Z}$-module). We say $(f_i)_i$ is a **basis** for L when they are linearly independent in $\mathbb{R}^n$ (and are thus a basis for the vector space spanned by $(f_i)_i$).

This is the definition from [vzGG13] which could be somewhat confusing. Notice that, even though L is a $\mathbb{Z}$-module, the definition of a basis in Definition 1 is with respect to the vector space structure over $\mathbb{R}$. In particular, when L is a free module, and hence, has a $\mathbb{Z}$-module basis B, it does not follow that B is a basis by Definition 1 (since B could be $\mathbb{R}$-linearly dependent). When, however, $f_i \in \mathbb{Z}^n$, the definitions agree.

Now, suppose we are given a lattice L with basis $b_1, \ldots, b_n \in \mathbb{Z}^n$. Since we are only allowed to take $\mathbb{Z}$-linear combinations in L, a "good" basis for L should be able to give a more clearer representation of L. In a way, we would like to reduce the lattice to its "essential components". For vector spaces Gram-Schmidt orthogonalization (GSO) is such a reduction or decomposition (others are eigenvalue decomposition, singular value decomposition ...). It transforms a basis for a vector space to an orthogonal one, such that the roles of these new basis vectors are much more distinguishable.

Naturally we cannot simply use the orthogonal basis returned by applying GSO to our basis $(b_i)_i$ for L since, most likely, the orthogonal basis $B^\star$ (for the associated vector space) will not be a basis for our lattice. But when it is, it provides us with an interesting (and useful) object: the *shortest vector in the lattice* as measured by the standard (2-)norm. This is quite intuitive; the norm of an element in L is the sum of the squares of the components, integers which in the basis $B^\star$ represent $\mathbb{Z}$-linear combinations of the orthogonal basis vectors (which can be seen as the standard basis for $\mathbb{R}^n$ but with the basis vectors individually rescaled). Hence, the shortest vector in L is then one of the n basis vectors in $B^\star$.

When we define some (relatively weak) constraints *on the corresponding GSO-basis* of our lattice basis $(b_i)_i$:

2 DEFINITION
Let $(b_1, \ldots, b_n)$ be a basis for the lattice L and $(b_1^\star, \ldots, b_n^\star)$ its corresponding Gram-Schmidt orthogonal basis. Then $(b_i)_i$ is **reduced** if $(b_i^\star)_i$ satisfies $\|b_i^\star\|^2 \leqslant 2\|b_{i+1}^\star\|^2$ for $1 \leqslant i < n$.

we immediately obtain a weaker version for lattices of the result for vector spaces:

3 Theorem
Let $(b_1, \ldots, b_n)$ *be a reduced basis for the lattice* L. *Then* $\|b_1\| \leqslant 2^{(n-1)/2}\|f\|$ *for all non-zero* f *in* L. *We call* b_1 *a* **short vector** *in* L.

The **LLL algorithm** transforms a basis $b_1, \ldots, b_n \in \mathbb{Z}^n$ of a lattice L to a reduced one simply by doing GSO but with the extra step of checking after each new orthogonal basis vector $b^\star_{i+1}$ if it satisfies the constraint in Definition 2. If not, it exchanges the last two vectors b_i and b_{i+1} and starts again with constructing $b^\star_i$.

However:

Problem 1
It is possible to keep exchanging vectors and thus, never finishing.

Problem 2
Even when the process finishes, and thus we have an orthogonal basis $(b^\star_i)_i$ that satisfies the constraints in Definition 2, it does not follow that $(b^\star_i)_i$ is the GSO-basis for the (potentially) reordered basis $(b_i)_i$ of L! That is, we still have no reduced basis.

The LLL algorithm solves these two problems simultaneously by replacing, for each new basis vector $b^\star_i$, the basis vector b_i with a "vector in L close enough to $b^\star_i$": we simply use the integers closest to the coefficients in the GSO procedure. It is not so difficult to see that the new $(b_i)_i$ form a reduced basis for L. That it also solves Problem 1 is not immediately clear and requires some work.[vzGG13]

As such, the reduced basis returned by LLL is also *nearly orthogonal* as a sort of "integer approximation" of the GSO-basis.

3 LLL lends a (short) hand

We return to the factorization algorithm to give a more thorough explanation of the idea of the most important step, the recombining of the factors. We first state the lemma that makes LLL useful in this step (it uses the same principle of lifting an equality over a modulo ring to one over $\mathbb{Z}$ as explained in the introduction).

4 Lemma ([vzGG13])
Let $F, G \in \mathbb{Z}[x]$ *with positive degrees* n, k *respectively, and suppose* $u \in \mathbb{Z}[x]$ *is non-constant, monic, and divides both* F *and* G *modulo* m*, for some* $m \in \mathbb{N}$ *with* $\|F\|^k \|G\|^n < m$*. Then* $\gcd(F, G) \in \mathbb{Z}[x]$ *is non-constant.*

We show its application in the algorithm by expanding on the general principle of the recombining step for a more concrete example.

Suppose $f = f_1 \cdots f_n$ for $f_i \in \mathbb{Z}[x]$ and $f_1 \equiv h_1 h_2 h_3 \pmod p$. Thus also $f_1 \equiv g_1 g_2 g_3 \pmod{p^l}$. ($g_i, h_i$ as in Step 1 and Step 2 from the introduction.) *How can we find* f_1 *from* $g_1 \in \mathbb{Z}/p^l[x]$*?*

Step 3.1
Define a lattice L that contains all (coefficient vectors of) polynomials divisible by g_1 over $\mathbb{Z}/p^l$. Well, not *all* the polynomials, of course, we have to choose some upper bound d on the degree of polynomials to consider. And the upper bound should be at most N, the degree of f. (Setting $d = N$ immediately would have too much of a negative impact on the performance.)

Step 3.2
Use LLL on a basis for L to get a reduced basis $B^\star$.

Step 3.3*
Let $g^\star$ be the first basis vector in $B^\star$, a *short vector* of L, and let $F = f_1$ and $G = g^\star$, with the natural embedding of $g^\star$ in $\mathbb{Z}[x]$. Then g_1 divides both F and G modulo p^l and Lemma 4 says that when also $\|F\|^{\deg(G)} \|G\|^{\deg(F)} < p^l$, we have that $\gcd(F, G) = \gcd(f_1, g^\star) = f_1$ (since f_1 is irreducible over $\mathbb{Z}$). Hence, in that case we have found f_1 as $\pm g^\star$!

Now, we know, from Theorem 3, that $\|g^\star\| \leqslant 2^{(d-1)/2} \|g\|$ for all non-zero g in L. When the upper bound d on the degree of the polynomials in L is big enough, we have that $f_1 \in L$ and thus $\|g^\star\| \leqslant$

$2^{(d-1)/2}\|f_1\|$. The main point is now that with these bounds we can choose p^l big enough *at the start of the algorithm* to be sure that the inequality from Lemma 4 will be satisfied when $f_1 \in L$ (precisely when $d \geqslant \deg(f_1)$). (This can easily be extended to work for arbitrary factors.)

Step 3.4

Use Mignotte's bound to check if $g^\star$ (as embedded in $\mathbb{Z}[x]$) is an irreducible factor of f (this is precisely when $\pm g^\star = f_1$ or thus when $d \geqslant \deg(f_1)$, but $\deg(f_1)$ is, of course, a priori unknown).

4 Complexity can be deceiving

Even though our algorithm has polynomial (worst-case) complexity whereas Zassenhaus' algorithm has exponential (worst-case) complexity, the latter is more practical in many cases.[vH02][vH13][vH13s][HvHN] The point being that the standard complexity of an algorithm is the worst case complexity, which may only be achieved rarely. In our case the LLL algorithm contributes certain "fixed costs" to our algorithm which give it a relatively high average complexity.

These problems have been discussed more thoroughly in the recent decade: see [vH13][vH13s] for an overview of the latest (anno 2013) factorization algorithms for univariate polynomials over $\mathbb{Z}$ or $\mathbb{Q}$, and [vH02] for a (very popular) improvement of our algorithm addressing these issues.[3]

References

[HvHN] Hart, W. and van Hoeij, M. and Novocin, A. *Practical polynomial factoring in polynomial time*, ACM, 2011.

[vH02] van Hoeij, M. *Factoring polynomials and the knapsack problem*, 2002.

[vH13] van Hoeij, M. *The complexity of factoring univariate polynomials over the rationals*, 2013.

[vH13s] van Hoeij, M. *The complexity of factoring univariate polynomials over the rationals (slides)*, 2013.

[vzGG13] von zur Gathen, J., and Gerhard, J. *Modern Computer Algebra*, Cambridge University Press, 2013.

[3] On `http://magma.maths.usyd.edu.au/magma/handbook/text/217` Magma claims to use the algorithm from [vH02] for factoring over $\mathbb{Z}$ or $\mathbb{Q}$.