

Polynomial Factorization: Sharp Bounds, Efficient Algorithms¹

BERNARD BEAUZAMY [§] VILMAR TREVISAN ^{†,‡} and PAUL S. WANG [‡]

[§] *Institut de Calcul Mathématique, Université de Paris 7, Paris, France*

[†] *Instit. de Matemática, Univ. Federal do Rio Grande do Sul, Porto Alegre, Brazil*

[‡] *Dept. of Maths and Computer Science, Kent State Univ., Kent, OH 44242, USA*

Dedicated to the memory of Hans Zassenhaus.

A new coefficient bound is established for factoring univariate polynomials over the integers. Unlike an overall bound, the new bound limits the size of the coefficients of at least one irreducible factor of the given polynomial. The single-factor bound is derived from the weighted norm introduced in Beuzamy *et al.* (1990) and is almost optimal. Effective use of this bound in p-adic lifting results in a more efficient factorization algorithm. A full example and comparisons with known coefficient bounds are included.

1. Introduction

Polynomial factorization is one of the most striking successes of Symbolic Computation. Take a polynomial, $f(x)$ over the integers \mathbf{Z} , and ask a computer to factor it. The result, a list of irreducible factors, comes back within a few seconds, provided that the polynomial is not too big. Modern symbolic computation systems factor equally well over algebraic extension fields (cf. Wang 1976), and factor multivariate polynomials (cf. Wang 1978), if the number of variables is not too high.

But, this success does not mean at all that the problem is easy. In fact, the basic idea one may try (obtaining values of $f(x)$ at some points and deducing from them the values of the coefficients in the factors) leads to an algorithm which does not

¹Work reported contains part of a Ph.D. preparation of V. Trevisan supervised by P. S. Wang and B. Beuzamy. It has been partially supported by contract ETCA/CREA/20367/91, Ministry of Defense, France, by research contract EERP-FR 22, Digital Equipment Corporation, by the C.N.R.S. and by the National Science Foundation under grants CCR-8714836 and CCR-9201800.

work well at all : it is far too slow in practice. The idea which led to the present approach is due to Zassenhaus (1969). It uses elaborate tools from Number Theory whose connection with the factorization problem, routine now, was far from obvious at that time.

It is not our intention to give here a complete overview of polynomial factorization and its history. We refer the reader to Kaltofen (1982) and Knuth (1981). Described in vague terms, however, modern factorization algorithms work as follows. Assume, without loss of generality, that the given polynomial $f(x) \in \mathbf{Z}[x]$ is *primitive* (content is 1) and *squarefree* (no repeated factors). Take a suitable prime p and reduce f modulo p to a polynomial $f_0 \in \mathbf{Z}_p[x]$. Factor f_0 over \mathbf{Z}_p into distinct irreducible factors (cf. Berlekamp (1967), Cantor-Zassenhaus (1981), Trevisan (1992)). Then “lift” these factors modulo increasingly larger modulus (p^2, p^3, \dots). Stop lifting when the modulus p^k is large enough. Finally, recover the irreducible factors of f over \mathbf{Z} from the factors mod p^k .

How large should p^k be taken? This is one of the topics of this paper. Existing algorithms use a stopping point for lifting based on estimates of a bound B for the coefficient size of any divisor of f . An a priori estimate for B can be computed from the degree and from the coefficients of f (see section 2.1). The smaller the estimate, the sooner the lifting can be stopped. Previously known estimates bound all factors of f ; the one we present here bounds only one factor of f , and is much smaller. The introductions of Beauzamy (1992) and of Trevisan (1992) also give a short history of some of the bound estimates.

Before we enter technicalities, let us observe that a complete factoring algorithm has to satisfy two aims, which are indeed contradictory: *recognize that a polynomial is irreducible and if not, factor it into irreducible factors.*

Since most polynomials are irreducible (take a polynomial at random and it will be irreducible), it is important for a factoring algorithm to be very efficient in detecting irreducibility. On the other hand, the polynomials scientists and engineers wish to factor are often reducible. Therefore, an algorithm must also be effective in finding the factors. The performance of a factoring algorithm will always be judged by these two different requirements.

Factoring f by first factoring $f_0 \bmod p$ detects irreducibility very quickly: if f_0 is irreducible mod p then f is irreducible over \mathbf{Z} . However, if f_0 is reducible, the number of its factors is at least the same as that of f : it cannot have fewer factors. For instance, if we take $f = 4x^2 + 4x + 3$, and $p = 3$, then $f_0 = x^2 + x = x(x + 1) \bmod 3$, whereas f is irreducible over the integers.

A factor of f_0 is *extraneous* if it does not correspond to an irreducible factor of f . The number of extraneous factors may depend on the value of p . For instance if we use $p = 5$ in the above example, f_0 will be irreducible. Thus we may try to minimize the number of extraneous factors by using multiple values for p (cf. Musser (1978),

Wang (1990)). But there are polynomials such as $x^4 + 1$, which are irreducible over \mathbf{Z} and become reducible modulo any prime, so extraneous factors generally cannot be avoided.

All factors of f_0 , extraneous or not, will be lifted until p^k is big enough. This can be unnecessarily costly if there are many extraneous factors. A sharper coefficient bound helps reduce this cost. Additionally, the technique of “early detection of true factors”, (cf. Wang (1983)) can also be very helpful. After presenting the *single-factor* bound, we build a lifting algorithm that takes advantage of the new bound and performs early detection of true factors.

Another complication caused by the computation modulo p^k is that coefficients are unique only up to units in the coefficient domain. If f is *monic* (with leading coefficient 1), we can make sure that all factors are also monic. But if f is not monic, a procedure must be applied in order to recover the leading coefficient of a factor before detecting whether it is a true factor or not. The stopping point of lifting also depends on the size of the leading coefficient and which method is used for coefficient recovery. We summarize known techniques for the effective handling of this *leading coefficient problem* and incorporate the most effective ones in our new factoring scheme using the single-factor bound.

A complete example is found in Appendix A. Comparisons with known coefficient bounds are included in Appendix B.

Now, let’s begin by considering estimates for coefficient bounds.

2. Overall Bounds and Partial Bounds

Let $f = \sum_0^n a_i x^i$ be a polynomial with complex coefficients. We define

$$|f|_\infty = \max_{1 \leq i \leq n} |a_i|,$$

called the *height* of f by number theorists. We say that B is an *overall bound* for the factorization of f if, in any factorization of f ,

$$f = f_1 \times \dots \times f_j, \tag{1}$$

each of the factors f_1, \dots, f_j satisfies $|f_j|_\infty \leq B$.

We say that b is a *single-factor bound* for the factorization of f if, in any factorization of f , one of the factors, say f_1 , satisfies

$$|f_1|_\infty \leq b.$$

Note that in (1) the f_1, \dots, f_j need not be irreducible. They can be any polynomials as long as their product is f . In fact, it does not make much sense to talk about factoring $f \in \mathbf{C}[x]$ into irreducible polynomials because it becomes root finding.

These general bound definitions can be used in the factorization of an integral polynomial f . Basically we lift the factors of f_0 to a large enough p^k . The choice of the integer k can be made according to the following observation : we stop when we are sure that the modulus is above all coefficients of all factors. Allowing the representation of both positive and negative integers, the coefficient range modulo p^k is

$$\left[-\frac{p^k - 1}{2}, \frac{p^k - 1}{2} \right].$$

Therefore, k must be such that $p^k \geq 2B$, where B is the overall bound defined above. This stopping point for lifting is used by all existing factoring algorithms.

But a better strategy is to use the single-factor bound b . After lifting to $p^k \geq 2b$, one factor is already present if f is not irreducible. This factor can be found and removed (see section 3).

The difference between these two strategies is significant if (and only if) B and b are significantly different in size. To show that this is the case, we turn to precise estimates. It is interesting to note that all existing bounds, as well as the new one we present here, are valid for polynomials with *complex* coefficients, and that they are derived with tools from complex analysis, not number theory on which the factoring algorithm is based.

2.1 Overall Bounds

The first bound, due to Zassenhaus, was :

$$B \leq |a_n| \max_{1 \leq i \leq \lfloor n/2 \rfloor} \binom{\lfloor n/2 \rfloor}{i} C^i, \quad (2)$$

where

$$C = \frac{1}{(\sqrt[n]{2} - 1)} \max_{1 \leq i \leq n} \left(\frac{|a_{n-i}|}{|a_n| \binom{n}{i}} \right)^{1/i}.$$

From Mahler's work, Mignotte (1974) deduced :

$$B \leq \binom{n}{\lfloor n/2 \rfloor} \|f\|_2, \quad (3)$$

where $\|f\|_2 = (\sum |a_j|^2)^{1/2}$.

This formula admits slight refinements : see for instance Knuth (1981), section 4.6.2, exercise 20, and Ph. Glesser (1990).

More recently, using the weighted norm

$$[f]_2 = \left(\sum_{i=0}^n \frac{|a_i|^2}{\binom{n}{i}} \right)^{1/2} \quad (4)$$

introduced by Bombieri in Beauzamy *et al.* (1990), Beauzamy (1992) showed that

$$B \leq \frac{3^{3/4}}{2\sqrt{\pi}} \frac{3^{n/2}}{\sqrt{n}} [f]_2. \quad (5)$$

Comparisons between (3) and (5) were made by Trevisan (1992) ; they show that (5) is always better than (3), except in the special cases of polynomials with coefficients only at extremities, such as $1 - z^n$.

2.2 Single-factor Bounds

If we are interested in one of the factors only, we can either take the one with smallest degree, or the one with smallest norm. Using the first idea, Mignotte deduced from the estimate (3) above that one of the factors must satisfy

$$b \leq \binom{d}{\lfloor d/2 \rfloor} \|f\|_2, \quad (6)$$

where $d = \lfloor n/2 \rfloor$.

The same type of idea does not work well with the weighted norm $[f]_2$, because the norm depends on the degree. However, its behavior with respect to products is very satisfactory, and we can easily find, among the factors, the one with smallest norm :

Theorem 1 *Let f be a univariate polynomial with complex coefficients and degree $n \geq 2$. Let $f = f_1 \cdots f_r$ be any factorization of f ($1 \leq r \leq n$). Then one of these factors, say f_1 , satisfies*

$$|f_1|_\infty \leq \frac{\Gamma(n+1)^{1/(2r)}}{\Gamma(\frac{n}{2r}+1)} [f]_2^{1/r}. \quad (a)$$

If $r \geq 2$, this implies :

$$|f_1|_\infty \leq \frac{\Gamma(n+1)^{1/4}}{\Gamma(\frac{n}{4}+1)} [f]_2^{1/r}, \quad (b)$$

which gives numerically :

$$|f_1|_\infty \leq \frac{2^{5/8}}{\pi^{3/8}} e^{1/4n} \frac{2^{n/2}}{n^{3/8}} [f]_2^{1/r}. \quad (c)$$

Proof : We need the following result by Bombieri (cf. Beauzamy *et al.* (1990)).

Lemma 2 *If g and h are univariate polynomials of degrees m and l respectively, then*

$$[gh]_2 \geq \sqrt{\frac{m! l!}{(m+l)!}} [g]_2 [h]_2 \quad (7)$$

If $n = 2$ in Theorem 1, estimate (7) shows that

$$[f]_2 \geq \frac{1}{\sqrt{2}} [f_1]_2 [f_2]_2 .$$

Therefore, one of the factors, say f_1 , satisfies

$$|f_1|_\infty \leq 2^{1/4} [f]_2^{1/2} . \quad (8)$$

Thus, (a), (b), and (c) are true for $n = 2$. We now consider the case $n \geq 3$.

Lemma 3 *Let f be a polynomial of degree n , with complex coefficients. Then*

$$[f]_2 \geq \frac{1}{\binom{n}{\lfloor n/2 \rfloor}^{1/2}} |f|_\infty .$$

Proof : Indeed, we have :

$$[f]_2^2 = \sum_{i=0}^n \frac{|a_i|^2}{\binom{n}{i}} \geq \frac{1}{\binom{n}{\lfloor n/2 \rfloor}} \sum_{i=0}^n |a_i|^2 \geq \frac{1}{\binom{n}{\lfloor n/2 \rfloor}} \max_i |a_i|^2 ,$$

which proves our Lemma.

Applying inequality (7) to the factorization $f = f_1 \cdots f_r$, we obtain immediately

$$[f]_2 \geq \sqrt{\frac{m_1! \cdots m_r!}{n!}} [f_1]_2 \cdots [f_r]_2 , \quad (9)$$

where $m_1 = \deg f_1, \dots, m_r = \deg f_r$.

Now, using Lemma 3 and writing $m'_1 = \lfloor m_1/2 \rfloor, \dots, m'_r = \lfloor m_r/2 \rfloor$, we get

$$[f]_2 \geq \sqrt{\frac{(m_1 - m'_1)! m'_1! \cdots (m_r - m'_r)! m'_r!}{n!}} |f_1|_\infty \cdots |f_r|_\infty .$$

So at least one of the factors, say f_1 , satisfies $|f_1|_\infty \leq A [f]_2^{1/r}$, where

$$A = \left(\frac{n!}{(m_1 - m'_1)! m'_1! \cdots (m_r - m'_r)! m'_r!} \right)^{1/(2r)} .$$

The $2r$ numbers $m_1 - m'_1, m'_1, \dots, m_r - m'_r, m'_r$ sum to n . Therefore (cf. Mitrinovic (1970), pp. 285, formula 3.6.48)

$$(m_1 - m'_1)! m'_1! \cdots (m_r - m'_r)! m'_r! \geq \Gamma\left(\frac{n}{2r} + 1\right)^{2r}$$

and

$$A \leq \frac{\Gamma(n+1)^{1/(2r)}}{\Gamma\left(\frac{n}{2r} + 1\right)} ,$$

which proves part (a) of the Theorem 1.

To prove part (b), we are going to show that the above quantity, for $r = 2, \dots, n$, takes its maximum for $r = 2$. For this, we let

$$h_n(s) = \frac{\Gamma(n+1)^s}{\Gamma(ns+1)},$$

and want to show that $h_n(s)$ is increasing for

$$\frac{1}{2n} \leq s \leq \frac{1}{4}.$$

Let $g_n(s) = \log h_n(s) = s \log \Gamma(n+1) - \log \Gamma(ns+1)$. Then the derivative is

$$g'_n(s) = \log \Gamma(n+1) - n \frac{\Gamma'(ns+1)}{\Gamma(ns+1)}.$$

We use the standard notation

$$\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)}$$

for the Digamma-function and we obtain

$$g'_n(s) = \log \Gamma(n+1) - n\psi(ns+1).$$

In order to show that $g'_n(s) \geq 0$ it is enough to show that

$$\frac{1}{n} \log \Gamma(n+1) \geq \psi\left(\frac{n}{4} + 1\right), \quad (10)$$

since ψ is increasing for real values of z (cf. Abramowitz & Stegun (1964), pp. 258, formula 6.3.1).

Using the rough estimates $\log z \geq \psi(z)$ (cf. Abramowitz & Stegun (1964), pp. 259, formula 6.3.18), and $\log \Gamma(n+1) \geq n \log\left(\frac{n}{e}\right)$ (cf. Abramowitz & Stegun (1964), pp. 257, formula 6.1.41), we see that (10) holds when

$$\log \frac{n}{e} \geq \log\left(\frac{n}{4} + 1\right),$$

which holds when $n \geq \frac{4e}{4-e}$, or $n \geq 9$.

The fact that (10) holds for $n = 3, \dots, 8$ is checked numerically. This completes the proof for part (b) of Theorem 1.

To show part (c), we use the inequality (cf. Mitrinovic (1970), p. 288, formula 3.6.55)

$$\left(x - \frac{1}{2}\right) \log x - x + \frac{1}{2} \log 2\pi < \log \Gamma(x) < \left(x - \frac{1}{2}\right) \log x - x + \frac{1}{2} \log 2\pi + \frac{1}{x}.$$

For the Gamma function we have $\Gamma(x+1) = x\Gamma(x)$, thus

$$(x + \frac{1}{2}) \log x - x + \frac{1}{2} \log 2\pi < \log \Gamma(x+1) < (x + \frac{1}{2}) \log x - x + \frac{1}{2} \log 2\pi + \frac{1}{x}$$

which gives

$$\frac{1}{4} \log \Gamma(n+1) < (\frac{n}{4} + \frac{1}{8}) \log n - \frac{n}{4} + \frac{1}{8} \log 2\pi + \frac{1}{4n},$$

and

$$\log \Gamma(\frac{n}{4} + 1) > (\frac{n}{4} + \frac{1}{2}) \log \frac{n}{4} - \frac{n}{4} + \frac{1}{2} \log 2\pi.$$

So if

$$C = \frac{\Gamma(n+1)^{1/4}}{\Gamma(\frac{n}{4} + 1)},$$

we find

$$\log C \leq -\frac{3}{8} \log n + (\frac{n}{4} + \frac{1}{2}) \log 4 - \frac{3}{8} \log 2\pi + \frac{1}{4n},$$

and

$$C \leq \frac{2^{5/8}}{\pi^{3/8}} \frac{2^{n/2}}{n^{3/8}} e^{1/4n},$$

as we announced.

2.3 Sharpness

The estimate (8) is best possible. Indeed, for $f = (Nx+1)(x+N)$, we have

$$[f]_2^{1/2} \approx N/2^{1/4}, \quad |f_1|_\infty = |f_2|_\infty = N.$$

The estimate (c) of Theorem 1 is almost best possible. Indeed, consider

$$f = (1-z^2)^d, \quad f_1 = (1-z)^d, \quad f_2 = (1+z)^d.$$

Then

$$[f]_2 = \frac{2^d}{\sqrt{\binom{2d}{d}}} \approx (\pi d)^{1/4}$$

(see Beauzamy (1992)). Also :

$$|f_1|_\infty = |f_2|_\infty = \binom{d}{\lfloor d/2 \rfloor} \approx \frac{2^d}{\sqrt{\pi d/2}}$$

Therefore

$$\frac{|f_1|_\infty}{[f]_2^{1/2}} \approx \frac{2^{d+1/2}}{\pi^{5/8} d^{5/8}}.$$

But the degree of f is $2d$, so with $n = \deg f$,

$$\frac{|f_1|_\infty}{[f]_2^{1/2}} \approx \frac{2^{9/8}}{\pi^{5/8}} \frac{2^{n/2}}{n^{5/8}}.$$

This shows that in Theorem 1 the exponential term $2^{n/2}$ cannot be improved; only the power term $n^{3/8}$ might be improved to some n^α , $3/8 \leq \alpha \leq 5/8$. However, in this example, the factors are not irreducible, so quite possibly Theorem 1 might be improved for irreducible factors.

2.4 Coefficient Cancellation

So we see that our estimates for B and b are significantly different, and that it is worth designing an algorithm which uses b instead of B . We see also that b , and a fortiori B , are much bigger than $|f|_\infty$ itself : some factors of f may have coefficients larger than the coefficients of f itself. This means term cancellations may occur in such a way that a product polynomial may have maximum coefficient size smaller than one or more of its factors. As a simple example consider:

$$f_1 = 1 + x + x^2 + 2x^3 + 2x^4 + x^5 + x^6 + x^7$$

$$f_2 = 1 - x + x^2 - 2x^3 + 2x^4 - x^5 + x^6 - x^7$$

for which the product is

$$f = 1 + x^2 + x^4 - x^{10} - x^{12} - x^{14}.$$

Another example is the polynomial $1 - z^n$, which is factored into irreducible factors, called cyclotomic polynomials. The size of the coefficients in these polynomials is well-known (cf. Paul Bateman (1981)) ; it depends on the number of prime factors in the integer n .

Finally, let us observe that, starting with *any* polynomial, we can multiply it by *some* polynomial, so as to produce a lot of cancellation. This is Bombieri-Vaaler's sharpening of Siegel's lemma (quoted by Mignotte (1988)) :

Lemma. *Let P be a polynomial with integer coefficients, irreducible, of degree $d \geq 2$. Let N be an integer, $N \geq d$. There exists a polynomial Q , such that the product $f = PQ$ has degree N and satisfies*

$$|f|_\infty \leq (N + 2)^{\frac{d}{2(N+1-d)}} M^{\frac{N}{N+1-d}},$$

where $M = M(P)$ is Mahler's measure of P .

This quantity tends to M when $N \rightarrow +\infty$. So we see that, if we start with P , with $M(P) = 1$ (for instance), no matter how large its coefficients may be, we can multiply it out by some other polynomial (with degree $\sim (\deg P)^2$), so as to finish with a polynomial, all coefficients of which are smaller than 2.

The advantage of tools such as Mahler's measure or the weighted norm $[\cdot]_2$ is that they allow general estimates, without any specific study of the cancellation phenomena, which are far from being well-understood. For a partial investigation, the reader is referred to Beauzamy-Enflo (1985).

3. Applying the Single-factor Bound

Overall bounds are general and work for any divisor of a polynomial. Their application in polynomial factoring is well understood. The smaller quantities given in Theorem 1, however, bound only one of the factors in an arbitrary decomposition of the given polynomial. We now show why such a bound suffices in practice.

Let $c(n) = 2^{5/8} e^{1/4n} / \pi^{3/8}$. For $n \geq 3$, we have $e^{1/4n} \leq e^{1/12}$ and

$$c(n) \leq c = \frac{2^{5/8} e^{1/12}}{\pi^{3/8}} \approx 1.091198478740277. \quad (11)$$

As an obvious consequence of Theorem 1 we have the following

Corollary 4 *Let f be a polynomial with integer coefficients and degree $n \geq 3$. Then if f is reducible over the integers, one of its irreducible factors, say f_1 , satisfies*

$$|f_1|_\infty \leq b = c \frac{2^{n/2}}{n^{3/8}} [f]_2^{1/2}. \quad (12)$$

Trevisan (1992) describes an efficient binomial coefficient computation using the *Pascal triangle* and a straightforward procedure for the weighted norm. Let's see how the single-factor bound b given by (12) is used in factoring.

We assume $n \geq 3$ because degree-two polynomials are easily factored with the quadratic formula. For simplicity, we also assume in this section that f and all its factors are monic. The leading coefficient problem will be treated in section 4.

After lifting to $p^k > 2b$, factors of $f(x)$, namely f_1, f_2, \dots, f_r , are individually tested by trial division, producing none, some, or all true factors of f . If all factors are found, we are done. If no factors are found (case A), then some of the r factors are extraneous, and others may not be lifted far enough (because we are using a single-factor bound, not an overall bound).

If only some factors are found, we let F be f with the factors found removed. We then compute the single-factor bound $b(F)$ for F . It is possible, but rare, that $2b(F) > p^k$ because coefficients in F may be bigger than those in f . In this case we continue to lift the factors of F to $p^k > 2b(F)$, and try division again (previous paragraph). But if $2b(F) \leq p^k$ (more likely), we can continue to search for true factors by setting $f = F$ (case B).

Now we deal with the remaining factors (from case A or B) by a systematic search of factor combinations through multiplication. If the combinatorial search produces no true factors, then f is irreducible and we have found all factors. The reason is that with the single-factor bound b we should find at least one factor if f is reducible.

Otherwise, the combinatorial search has produced a set of true factors $f = g_1 \cdots g_s$. Each g_i is a product of two or more distinct factors f_i . For example $g_1 = f_1 f_2 \bmod p^k$ or $g_2 = f_3 f_5 f_7 \bmod p^k$. At this stage, each g_i is *not necessarily irreducible*. Let's see why (the reducible ones will be dealt with further).

Consider, for example, a true factor $g_1 = f_1 f_2 \bmod p^k$. It need not be irreducible, despite the fact that neither f_1 nor f_2 were found to be a true factor of f (at p^k). Indeed, upon further lifting, f_1 and f_2 may each lead to a different irreducible factor of f . In this case it also means that g_1 has coefficients smaller than those of its factors over \mathbf{Z} , a possible but rare condition.

To determine whether g_i is reducible, the algorithm computes the single-factor bound $b(g_i)$. If $2 b(g_i) \leq p^k$ then g_i is irreducible. Otherwise, the constituent factors of $g_i \bmod p^k$ will be lifted further to satisfy this new bound before searching for its true factors. The algorithm works this way on each g_i separately to find all factors of f .

We have modified the basic combinatorial search (in Collins (1979) it is shown that on average this procedure has polynomial time complexity). Now true factors can be found earlier and the single-factor bound can help test the irreducibility of the factors found.

Note that whenever an irreducible factor is found and removed, the lifting process is affected in three different ways :

1. The degree of f decreases.
2. The list of factors shrinks.
3. The leading coefficient may become smaller.
4. The coefficient bound changes.

The first three items will clearly simplify the lifting procedure. For the fourth item, the single-factor bound, though infrequently, may become larger. In any case, a lifting problem must be reformulated after some factors are removed. See Wang (1992) for an efficient reformulation algorithm that involves minimal recomputation.

If the polynomial being factored is non-monic, there are further complications due to non-uniqueness of the factors modulo p^k . The leading coefficient problem will be discussed in Section 5.

When we have a polynomial to factor, the number of irreducible factors over the integers is, in general, not known. Quite often there are more factors modulo p than actual integral factors. To cope with this problem, a well known technique is to factor the polynomial f modulo several primes. Then we can choose the factorization that gives the least number of factors. Furthermore, this enables degree consistency tests (cf. Musser (1978)) and degree reconciliation (cf. Wang (1990)) to further reduce the number of factors and trial divisions. Most polynomials require only five primes to determine its irreducibility (Musser (1978)). If the number of irreducible factors is known (or expected) to be more than two, part (a) of Theorem 1 can be applied to obtain a much better bound.

4. The Leading Coefficient Problem

The purpose of a coefficient bound B in factoring is to obtain a stopping point for the p-adic lifting process. After lifting is done, the lifted factors are used to find the actual irreducible factors of the given polynomial f . The exact stopping point, $q = p^k$, for lifting depends on B , the leading coefficient a_n of f , and the method employed to recover the true factors. These are the subjects of this section.

There are two operations involved in the recovery of true factors from lifted factors : recombining extraneous factors and recovering the right coefficients. The former requires a systematic search of combinations of the lifted factors, as we just discussed. We shall concentrate on the latter operation here.

Let's first look at an example. Consider the polynomial

$$f(x) = 45x^4 - 78x^3 + 165x^2 + 64x - 26$$

This polynomial can be factored over \mathbf{Z}

$$f(x) = A(x) \times B(x)$$

where

$$\begin{aligned} A(x) &= 3x^2 - 6x + 13 \\ B(x) &= 15x^2 + 4x - 2 \end{aligned} \tag{13}$$

are both irreducible over \mathbf{Z} .

If we factor f modulo $p = 7$ and then lift the two factors to the modulus 7^4 we may end up with the following factors :

$$\begin{aligned} f(x) &= a(x) \times b(x) \pmod{7^4} \\ a(x) &= 671x^2 + 1059x - 1094 \\ b(x) &= 340x^2 + 891x + 755 \end{aligned} \tag{14}$$

The modulus $7^4 = 2401$ is big enough but the factors $a(x)$ and $b(x)$ look nowhere near the actual factors $A(x)$ and $B(x)$ that we want to find. These factors $a(x)$ and $b(x)$ are in fact equal to $A(x)$ and $B(x)$ when multiplied by the right units in \mathbf{Z}_{2401} .

In general, factors are only unique up to units in the coefficient domain, and most elements of the ring of integers modulo p^k are invertible (the only non-invertible elements are the multiples of p). Two techniques already exist in the literature in bits and pieces, in order to recover the correct coefficients ; they are presented here. Their relation with the stopping point q is also clearly stated.

One method to transform $a(x)$ to $A(x)$ involves multiplying $a(x)$ by the inverse of its leading coefficient, and then by the leading coefficient of f .

$$45 \times (671^{-1}) \times a(x) = 45x^2 - 90x + 195 \quad \text{mod } 7^4 \quad (15)$$

As mentioned before, we use the *symmetric range* $-(q-1)/2, (q-1)/2$ so both positive and negative coefficients may be recovered. Now we simply take the principal part of the right-hand side of equation (15) to recover $A(x)$.

If the given polynomial f is monic (with leading coefficient 1), then simply keeping all factors monic in the lifting process will recover the correct coefficients. It is possible to first apply a *monic transformation* to a polynomial before factoring it. For any given polynomial $f(x)$ with a leading coefficient $a_n \neq 1$ the transformation

$$h(x) = a_n^{n-1} f(x/a_n)$$

produces a monic $h(x)$ whose factors lead directly to those of $f(x)$. The problem with this approach is that $h(x)$ will have much larger coefficients, especially when a_n and/or the degree n are not small. The coefficient bound with $h(x)$ can therefore be very much worse. The problem is so severe that we recommend against performing this monic transformation.

When factoring a monic polynomial, we can keep the factors unique by insisting that each factor be also monic. The question is what to do when the given polynomial is non-monic.

If the constant coefficient a_0 of f is ± 1 , we will “reverse” the polynomial, that is consider $x^n f(1/x)$ instead of f : this polynomial has a_0 instead of a_n as leading coefficient, and is now monic. More generally, we shall perform this transformation if $|a_0| < |a_n|$. This will be assumed throughout the sequel, so when we speak of the “leading coefficient”, it is either a_n or a_0 : whichever is smaller.

If the usual p-adic lifting is applied without regard to the leading coefficient situation, the factors being lifted can take on arbitrary leading coefficients subject only to the condition that their product be congruent to a_n , as we saw in the example above. This makes recovering actual factors harder. We recommend keeping the leading coefficient of one of the factors being lifted congruent to a_n . The other factors are kept monic throughout the lifting process.

Let's say that the first factor $f_1(x)$ always carries the non-trivial leading coefficient throughout the p-adic lifting process. This is arranged by *replacing* the leading coefficient of $f_1(x)$ by a_n modulo the next higher modulus at the beginning of each lifting step. Following this strategy, the correction terms added to each factor at each lifting stage will be of lower degree and will not modify the leading term of the factor.

When attempting to derive actual factors from the lifted factors, we first form a *candidate* for test division. The *candidate* can be a single lifted factor or a product of several lifted factors. Consider a candidate $a(x)$ that actually corresponds to a true factor $A(x)$ of f . As mentioned before, $A(x) = u \times a(x) \pmod q$ for the right unit u when q is big enough. The question is : What is the right unit u ? The answer is apparently simple : $u = lc(A) \times lc(a)^{-1}$. But we don't know $lc(A)$ because we are trying to find $A(x)$. However, we do know a multiple of $lc(A)$, namely a_n . So we apply the following two-step procedure to get $A(x)$.

$$\begin{aligned} g(x) &= a_n \times lc(a)^{-1} \times a(x) \pmod q \\ A(x) &= pp(g(x)) \quad (\text{computed over } \mathbf{Z}) \end{aligned} \tag{16}$$

Let's be more specific about recovering the correct coefficients. Recall that we keep all but one factors being lifted monic. For a candidate $a(x)$ that is also monic, the procedure of algorithm **LC** is applied.

Algorithm 1 LC

- LC-1** Let $g(x)$ be $a_n a(x)$ modulo the final modulus q .
- LC-2** Normalize the coefficients so they fall between $-q/2$ and $q/2$ inclusive.
- LC-3** Let $g(x)$ be the principal part of $g(x)$, discarding its content (performed over \mathbf{Z}).
- LC-4** Test $g(x)$ by trial division into $f(x)$ over the integers.

For a candidate, such as f_1 , that is non-monic, the multiplication by a_n in Step **LC-1** is omitted. If $a(x)$ is not an extraneous factor, this routine will produce a true factor $g(x)$ provided that q is big enough.

So how large should q be? It is not enough to have q just bigger than the largest size coefficient in $g(x)$. Because coefficients of $g(x)$ can be positive or negative, q should be at least bigger than twice the maximum coefficient size. However, even that is not enough because this procedure multiplies $g(x)$ by an extra integer constant (Eq. (16)) which, in the worst case, can be as big as a_n . Therefore, the stopping point has to be adjusted accordingly. Specifically, we must require $q > 2a_n b$.

The procedure **LC** is just one way of recovering the true factor. Another, often more advantageous, approach is to think about the congruence

$$A(x)/lc(A) = lc(a)^{-1} \times a(x) \pmod{q}$$

The left-hand side is a polynomial with rational coefficients. Wang (1983) gave an algorithm to recover the rational coefficients from $a(x)$ provided q is big enough. We will not go into details here but will simply outline this alternative procedure. So instead of algorithm **LC** we put a candidate $a(x)$ through algorithm **RC** :

Algorithm 2 RC

RC-1 $u(x) \leftarrow lc(a)^{-1} \times a(x) \pmod{q}$

RC-2 For each coefficient c_i of $u(x)$, replace it by the rational number it represents modulo q , i.e. by n_i, d_i such that $n_i = c_i \times d_i \pmod{q}$.

RC-3 Now $u(x)$ is a monic polynomial with rational coefficients. Clear the denominators of $u(x)$.

RC-4 Test $u(x)$ by trial division into $f(x)$ over the integers.

This procedure will find a true factor if $a(x)$ is not an extraneous factor provided, as before, that q is big enough. Let s be any numerator or denominator of any rational coefficient of the target monic (rational) factor. The modulus q is big enough if $s^2 \leq q/2$ for any s (cf. Wang (1983)). An efficient method to compute the rational representation can also be found in Wang (1983). As in the first method it may not be enough for q to be larger than $2b$. Since we must have $s^2 \leq q/2$, an upper bound for q is twice the square of the maximum coefficient.

To summarize, we have given two different methods for recovering true coefficients from lifted factors. Algorithm **LC** requires $q > 2a_n b$ while algorithm **RC** needs $q > 2b^2$. The method to choose should be the one requiring a smaller q

5. An Efficient Factorization Algorithm

We have presented the single-factor bound b , its theoretic derivation, and efficient computation. We also discussed how b is used within p -adic lifting to help reduce the problem size and to obtain irreducible factors early. Depending on which algorithm, **LC** or **RC**, is used to handle the leading coefficient problem, the bound b is also adjusted differently to determine the final modulus q .

We now present an algorithm outline that summarizes the ideas discussed. The algorithm factors univariate integral polynomial $f(x)$ into irreducible factors. Many details have been omitted because most steps already appeared elsewhere.

Experimental data are provided in the appendices.

Algorithm 3 FACTOR

- F-1** Cast out any repeated factors (cf. Wang & Trager (1978)) and any content of f .
- F-2** Factor f over several (up to 5) prime fields (cf. Trevisan & Wang (1991)). If f is irreducible over a prime field then it is irreducible over \mathbf{Z} (terminate).
- F-3** Apply the degree consistency algorithm (cf. Musser (1978), Wang (1990)). If f is irreducible, terminate. Otherwise, the largest prime that gives the smallest number of factors (after combining factors through degree consistency) is used for the rest of the steps.
- F-4** If the number of irreducible factors is known (through some other independent means), compute coefficient bound b using part (a) of Theorem 1. Otherwise compute $b = 2 c \frac{2^{n/2}}{n^{3/8}} \sqrt{[f]_2}$.
- F-5** If the constant term is smaller in size than the leading coefficient, reverse the polynomial and the factors. If $lc(f) \neq 1$, let $\beta = \min \{2b^2, 2b lc(f)\}$.
- F-6** Use EEZ (cf. Wang (1978)) to lift the factorization from modulo p to p^k until $p^k > \beta$.
- F-7** By trial division of single factors, test for true factors. For the non-monic case, Algorithm RC or Algorithm LC is used, depending on how bound β is computed.
- F-8** If no true factors are found in step **F-7**, go to step **F-9**. If all factors are found, terminate. Otherwise, remove the true factors found from f and compute a new bound $b(f)$ which gives a new stopping point p^K . If $p^K \leq p^k$ then goto step **F-7** (Note lc may have changed so one cannot go to F-9). Otherwise ($p^K > p^k$), reformulate the lifting problem (cf. Wang (1992)), continue the lifting, then goto step **F-7**.
- F-9** Find true factors by trial division of combinations of the lifted factors. (For the nonmonic case, Algorithm RC or Algorithm LC is used, depending on how bound β is computed.)
- F-10** If no factors are found then f is irreducible (terminate). Otherwise, a decomposition $f = g_1 \cdots g_s$ over the integers is found. But some g_i may be reducible over \mathbf{Z} .
- F-11** For each g_i compute b_i , the single-factor bound for g_i . If $2 b_i a_n \leq p^k$ (or $2 b_i^2 \leq p^k$ whichever is appropriate) then g_i is irreducible. Otherwise, perform steps **(a)** and **(b)** on g_i .
- (a)** Continue to lift the factors of g_i until the $p^k > 2lc(g_i)b_i$.

(b) Apply steps F-7 through F-11 to g_i and its lifted factors.

Conclusion

The present paper shows that, in order to realize fast algorithms, all we need is to control the size of coefficients in one irreducible factor. On this topic, all we know in theory is given by Theorem 1, but we have not met any case where some of the factors, say f_1 , did not satisfy $|f_1|_\infty \leq |f|_\infty$. Let's state a specific question :

Given f , polynomial with integer coefficients, what is

$$\inf\{|g|_\infty ; g \text{ is an irreducible factor of } f \}.$$

Are there cases where all irreducible factors satisfy $|g|_\infty > |f|_\infty$?

So, as we see again here, Computer Science challenges Mathematics, the kind of challenge Zassenhaus liked.

Acknowledgment

The authors are indebted to Bruce Reznick for his suggestions regarding the proof of Theorem 1 and to K. Weber for his numerous helpful comments on earlier versions of this paper. We are also grateful for the careful reading by a referee and helpful suggestions by the editor Erich Kaltofen.

References

- Abramowitz, M. and Stegun, I. E. (1964), (Ed.): *Handbook of Mathematical Functions*. National Bureau of Standards.
- Bateman, P.T., Pomerance, C., Vaughan, R.C. (1981) On the size of the coefficients of the cyclotomic polynomial. *Colloquia Mathematica Societatis Janos Bolyai*, 34, Topics in Classical Number Theory, Budapest (Hungary), pp 171-202.
- Beauzamy, B. (1992) Products of polynomials and a priori estimates for coefficients in polynomial decompositions : a sharp result. *Journ. of Symb. Comp.*, 13, pp. 463–472.
- Beauzamy, B., Bombieri, E., Enflo, P. and Montgomery, H. (1990), Products of polynomials in many variables. *Journ. of Number Theory*, vol. 36, no. 2, 219–245.
- Beauzamy, B., Enflo, P. (1985), Estimations de produits de polynômes. *Journ. of Number Theory*, vol. 21, 3, 390–412.
- Berlekamp E. R. (1967), Factoring Polynomials over Finite Fields. *Bell System Tech. J.*, vol. 46, 1853-1859.

- Cantor D. and Zassenhaus H. (1981), A New Algorithm for Factoring Polynomials over Finite Fields. *Math. Comp.*, 36, 587-592.
- Cerlienco, L., Mignotte, M. and Piras, F. (1987), Computing the measure of a polynomial, *Journ. of Symb. Comp.*, vol. 4, no. 1, 21-34.
- Collins, G. E. (1979), Factoring univariate integral polynomials in polynomial average time, *Proceedings EUROSAM'79* (Marseille), Springer LNCS 72, 317-329.
- Glessner, P. (1990), Majoration de la norme des facteurs d'un polynôme, *Ann. Fac. Sci. Toulouse*, vol. XI, 67-74.
- Kaltofen, E. (1982) Factorization of Polynomials, *Computer Algebra-Symbolic and Algebraic Computation*, Computing, Suppl. 4, Springer-Verlag, 95-113.
- Knuth, D. E.(1981), *The Art of Computer Programming*, vol. 2, Addison-Wesley, Reading, Mass.
- Mignotte, M. (1974), An inequality about factors of polynomials, *Math. Comp.*, vol. 28, no. 128, 1153-1157.
- Mignotte, M. (1988), An inequality about irreducible factors of integer polynomials. *Journ. of Number Theory*, vol. 30, 156-166.
- Mitrinovic, D. S. (1970), *Analytic Inequalities*, Springer Verlag, Berlin.
- Musser, D. R. (1978), On the efficiency of a polynomial irreducibility test. *Journ. of the A.C.M.*, vol. 25, no. 2, 271-282.
- Trevisan, V., (1992), Univariate Polynomial Factorization, *Ph.D. Thesis*, Dept. of Math. and Comp. Sci., Kent State University.
- Trevisan, V. and Wang, P. S. (1991), Practical factorization of univariate polynomials over finite fields, *Proceedings of ISSAC'91*, Bonn, July 15-18.
- Trevisan, V. (1990), Recognition of Hurwitz polynomials, *SIGSAM Bulletin*, Vol. 24, No. 4.
- Wang, P. S. (1976) Factoring Multivariate Polynomials over Algebraic Number Fields, *Math. Comp.*, vol 30, Apr. 1976, 324-336.
- Wang, P. S. (1978), An Improved Multivariate Polynomial Factoring Algorithm, *Math. Comp.*, vol. 32, no. 144, 1215-1231.
- Wang, P. S. (1983), Early detection of true factors in univariate polynomial factorization, *Proceedings ACM EUROCAL'83*, London, March 28-30.
- Wang, P. S. (1990), Parallel Univariate Polynomial Factorization on Shared-Memory Multiprocessors, *Proceedings of the ISSAC'90*, Aug. 1990, 145-151.
- Wang, P. S. (1992), Parallel Univariate p -adic Lifting on Shared-Memory Multipro-

cessors, *Technical Report*, Institute for Computational Mathematics, Kent State University.

Zassenhaus, H. (1969), On Hensel Factorization I, *Journ. of Number Theory*, vol. 1, no. 1, 291–311.

Appendix A: An Example

To clarify the algorithm presented and to see how things work exactly we present some details in the factorization of a specific polynomial :

$$f(x) = 904050 x^7 + 1479450 x^6 - 2336817 x^5 - 3403088 x^4 + 2021847 x^3 + 1477766 x^2 - 1006566 x + 170694.$$

Over \mathbf{Z}_{11} , we obtain the factorization

$$f(x) = (4x^2 + 5x - 2)(x^2 - x - 4)(x + 4)(x + 5)(x + 3) \pmod{11}.$$

We now use the lifting algorithm and the recovering procedure to find the true integral factorization.

The single-factor bound is $b = 14351$. Using this value as a heuristic bound, we require an accuracy of $11^4 = 14641$. First we lift to 11^2 , and we obtain, modulo 11^2 :

$$f(x) = (59x^2 + 60x + 42)(x^2 + 21x - 4)(x + 4)(x - 50)(x + 36),$$

and the correction factors

$$\begin{aligned} \alpha_1(x) &= 58x - 53, & \alpha_2(x) &= -39x + 6, \\ \alpha_3(x) &= 43, & \alpha_4(x) &= -21, & \alpha_5(x) &= -25 \end{aligned}$$

One more linear lifting produces the factorization modulo 11^4

$$F_3(x) = (-3692x^2 - 6111x + 1373)(x^2 - 6755x - 5449)(x - 4957)(x + 2249)(x - 2626).$$

These adjusted factors are now put through the leading coefficient recovering algorithm **RC**. This procedure is chosen because $904050b > 2b^2$.

After monicizing the factor $(-3692x^2 - 6111x + 1373)$ modulo 11^4 , we obtain $(x^2 - 4880x - 5231)$. Because $-4880 = 1/3 \pmod{11^4}$ and $-5231 = -29/14 \pmod{11^4}$, after clearing the denominators, algorithm **RC** generates $(42x^2 + 14x - 87)$, which is a true irreducible factor over the integers. No other factor is found in this procedure. Notice that algorithm **LC** would not produce any true factor at this stage.

After removing the factor found, we have reduced the problem to factoring over \mathbf{Z} the integral polynomial

$$f(x) = 21525 x^5 + 28050 x^4 - 20401 x^3 - 16122 x^2 + 11254 x - 1962,$$

and we know the factorization mod 11^4 :

$$f(x) = (6884 x^2 - 1604 x - 674)(x - 4957)(x + 2249)(x - 2626).$$

Notice how the leading coefficient of the first factor has been recomputed as $6884 = 21525 \bmod 11^4$.

Products of two or more lifted factors are now tested for divisibility over the integers. The product $(6884x^2 - 1604x - 674)(x + 2249)$, after going through algorithm **RC**, generates the integral factor $(15x^3 + 12x^2 - 10x + 2)$. We conclude that this factor is irreducible, since the single-factor bound for this factor is 18, and $2 \times 18^2 < 11^4$ the current modulus.

The problem is now reduced to factoring

$$f(x) = 1435 x^2 + 722 x - 981$$

over the integers. Its single-factor bound is $b = 134$. Because the polynomial is not monic and the adjusted bound is $2 \times 134^2 = 35912 > 11^4$, we need to lift further. At this point, we have the factorization mod 11^4 :

$$f(x) = (1435 x + 6295)(x - 2626),$$

with correction factors $\alpha_1(x) = 38$, $\alpha_2(x) = 35$. The recomputation of the correction factors is done according to Trevisan (1992). One more linear lifting produces the factorization

$$f(x) = (1435 x + 181987)(x + 656219) \bmod 11^6.$$

None of these two factors produces a true integral factor. Since there are two single factors, they have to be multiplied and we conclude that $f(x)$ is irreducible. This terminates the algorithm with complete factorization over the integers :

$$F_3(x) = (15 x^3 + 12 x^2 - 10 x + 2)(42 x^2 + 14 x - 87)(1435 x^2 + 722 x - 981).$$

Appendix B: Bound Comparisons

We mentioned in Section 2.1 several coefficient bounds, some of them have been known since the early 60's (for a historical view of the coefficient bound problem, we refer to Trevisan (1992)). We will now compare several of these bounds :

- (a) $\frac{3^{3/4}}{2\sqrt{\pi}} \frac{3^{n/2}}{\sqrt{n}} [f]_2$ – the recent overall bound given by Beuzamy (1992)
- (b) $\binom{d}{\lfloor d/2 \rfloor} M(f)$, $d = \lfloor n/2 \rfloor$ – the single-factor bound deduced from Mahler’s measure
- (c) $\binom{d}{\lfloor d/2 \rfloor} \|f\|_2$, $d = \lfloor n/2 \rfloor$ – the single-factor bound given by Mignotte (1974)
- (d) $c \frac{2^{n/2}}{n^{3/8}} \sqrt{[f]_2}$, $c \sim 1.1$ – the single-factor bound introduced here.

The polynomials used in Table 1 are:

$$\begin{aligned}
F_1(x) &= x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8 \\
F_2(x) &= x^{15} + 30x^{14} + 5x^{13} + 2x^{12} + 5x + 2 \\
F_3(x) &= 904050x^7 + 1479450x^6 - 2336817x^5 - 3403088x^4 \\
&\quad + 2021847x^3 + 1477766x^2 - 1006566x + 170694. \\
F_4(x) &= x^{18} + 9x^{17} + 45x^{16} + 126x^{15} + 189x^{14} + 27x^{13} - 540x^{12} \\
&\quad - 1215x^{11} + 1377x^{10} + 15444x^9 + 46899x^8 + 90153x^7 + 133893x^6 \\
&\quad + 125388x^5 + 29160x^4 - 32076x^3 + 26244x^2 - 8748x + 2916 \\
F_5(x) &= P_{40} = a_1(x) \times a_2(x) \times a_3(x) \times a_4(x)
\end{aligned}$$

The polynomial P_{40} is the well known *SIGSAM Problem # 7* and can be found, for example in Wang (1990).

f	a	b	c	d
F_1	122	80	109	23
F_2	2745	1046	1083	204
F_3	10^8	8535109	15×10^6	7175
F_4	8.2×10^6	16730116	27×10^6	12927
F_5	2.8×10^{25}	5.387×10^{26}	9.7×10^{27}	1.446×10^{14}

Table 1: Comparison between Different Bounds

These results indicate that the single-factor bounds are smaller than the overall bound (which means it is worth trying to detect a single factor), and that among the single-factor bounds, the one presented here is by far the smallest.