Exact Polynomial Factorization by Approximate High Degree Algebraic Numbers

Chen Jing-wei [§], Feng Yong [∗ § ¶], Qin Xiao-lin [§] and Zhang Jing-zhong [§ ¶]

[§] Laboratory for Automated Reasoning and Programming, Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu 610041, China

[¶] Laboratory of Computer Reasoning and Trustworthy Computation, University of Electronic Science and Technology of China, Chengdu 610054, China

{velen, yongfeng, qinxl} @casit.ac.cn zjz101@yahoo.com.cn

ABSTRACT

For factoring polynomials in two variables with rational coefficients, an algorithm using transcendental evaluation was presented by Hulst and Lenstra. In their algorithm, transcendence measure was computed. However, a constant c is necessary to compute the transcendence measure. The size of c involved the transcendence measure can influence the efficiency of the algorithm greatly.

In this paper, we overcome the problem arising in Hulst and Lenstra's algorithm and propose a new polynomial time algorithm for factoring bivariate polynomials with rational coefficients. Using an approximate algebraic number of high degree instead of a variable of a bivariate polynomial, we can get a univariate one. A factor of the resulting univariate polynomial can then be obtained by a numerical root finder and the purely numerical LLL algorithm. The high degree of the algebraic number guarantees that this factor corresponds to a factor of the original bivariate polynomial. We prove that our algorithm saves a $(\log^2(mn))^{2+\epsilon}$ factor in bit-complexity comparing with the algorithm presented by Hulst and Lenstra, where (n, m) represents the bi-degree of the polynomial to be factored. We also demonstrate on many significant experiments that our algorithm is practical. Moreover our algorithm can be generalized to polynomials with variables more than two. [1]

∗Corresponding author.

[1]This research was partially supported by a National Key Basic Research Project of China (2004CB318003), the Knowledge Innovation Program of the Chinese Academy of Sciences (KJCX2-YW-S02), and the National Natural Science Foundation of China (10771205).

SNC'09, August 3–5, 2009, Kyoto, Japan.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on polynomials*

General Terms

algorithms, experimentation

Keywords

factorization, polynomial, approximation, algebraic number

1. INTRODUCTION

The factorization of polynomials is a classical problem in computer algebra, which intervenes in many fields of application. Historical surveys about it can be found in [14, 15, 16, 17, 9]. In 1982 the first polynomial-time algorithm for factoring polynomials in one variable with rational coefficients was published (see [22]). The most important part of this factoring algorithm is the so-called basis reduction algorithm, i.e. the famous LLL algorithm due to Lenstra, Lenstra and Lovász. The LLL algorithm has many important applications, such as wireless communication, cryptography (see [25]), GPS (see [1]) and so on. Since then many generalizations of the original algorithm were published, for example [23, 5, 21, 30, 31, 24, 19], which applied the LLL lattice basis reduction technique to obtain polynomial time algorithms for factoring multivariate polynomials over various fields including finite fields, local fields and number fields. While there are also several other approaches for factoring polynomials. For instance E. Kaltofen presented algorithms for reducing the problem of finding the irreducible factors of a bivariate polynomial with integer coefficients in polynomial time to factoring a univariate integer polynomial (see [12, 13]). In recent years, many more efficient methods and algorithms have been introduced to factor polynomials. For univariate polynomials, van Hoeij proposed some algorithms (see [32, 33]) which follow the Berlekamp-Zassenhaus algorithm and use the LLL algorithm to solve a combinatorial problem which has smaller coefficients and dimensions rather than calculate coefficients of a factor of a univariate polynomial with integer coefficients. The latest advances about the van Hoeij algorithm are presented in [28, 29]. For multivariate polynomials, many references can be used. Using Hensel lifting and factor recombination technique, G. Lecerf *et al* presented some efficient factoring algorithms (see [2, 20, 4]). The differential equations was introduced to factor polynomials by S. Gao *et al* in [9, 10]. Chèze and Galligo proposed an algorithm for computing an exact absolute factorization of a bivariate polynomial from an approximate one (see [3]). There are several well-known techniques and variants for factoring bivariate polynomials. A classical and generic one consists in localizing one of the two variables at a suitable value, computing the local analytic factorization, and discovering the recombinations into the rational factors (see [2, 20, 4] and the references therein). In the particular case of integer coefficients, it is desirable to adapt a more direct strategy on the top of LLL.

Another aspect, the symbolic-numeric hybrid computation (see [7, 18, 35]) is at the intersection of applied and traditional mathematics and at the intersection of numerical analysis and computer algebra. Many methods in this field are playing more and more important role in computer algebra including factorization.

In this paper we adopt the both ideas: the lattice reduction and the symbolic-numeric hybrid computation to solve the problem of factoring a polynomial $f(x, y)$ in $\mathbb{Q}[x, y]$. A method to obtain the minimal polynomial of an algebraic number was given in [19]. Using this method, Hulst and Lenstra presented an algorithm for factorization of polynomials in two variables with rational coefficients by transcendental evaluation (see [31]), in which transcendence measure was needed. However, a constant c is necessary to compute the transcendence measure. The size of c involved the transcendence measure can influence the efficiency of the algorithm greatly. In this paper, we overcome the problem arising in [31] and present a new polynomial-time algorithm for the factorization of bivariate polynomials with rational coefficients. Furthermore the running time of our algorithm is not only $(\log^2(mn))^{2+\epsilon}$ times less than the algorithm in [31] , where (n, m) represents the bi-degree of the polynomial to be factored, but also less than or near to the running time of the order `factor()` in Maple 11 for many examples (see subsection 3.3). Moreover our algorithm can be generalized easily to factor polynomials with variables more than two. Although all of the intermediate computations are approximate, the input and output of our algorithm are both polynomials with exact coefficients.

The outline of our algorithm to factor a bivariate polynomial is as follows. First, we convert the bivariate polynomial to be factored to a univariate one by substituting an approximate algebraic number of high degree for a variable. After the substitution we can compute approximations to the complex roots of the resulting univariate polynomial, and look for the minimal polynomial (over some algebraic extension of $\mathbb{Q}$) of one of the approximated roots. Lemma 1 guarantees that this minimal polynomial corresponds to a factor of the original bivariate polynomial and Lemma 6 guarantees this polynomial is an irreducible one.

The rest of this paper is organized as follows. In Section 2 we present the notations and some preliminary lemmas. We describe our algorithm and analyze the correctness and the running time in Section 3, in which we also give several examples in detail. And we draw a conclusion of this paper in Section 4.

2. NOTATIONS AND PRELIMINARIES

In this section, we first give some notations. And then we discuss the process of approximation in subsection 2.2. We introduce lattice and LLL Algorithm in subsection 2.3. In subsection 2.4, we give some preliminary lemmas which implies our main algorithm.

2.1 Notations

For a bivariate polynomial $f(x, y) = \sum_i \sum_j f_{i,j} x^i y^j$ in $\mathbb{Q}[x, y]$, we denote by $\deg_x(f)$ and $\deg_y(f)$ its degree in x and y respectively, $||f||_1 = \sum_i \sum_j |f_{i,j}|$ its one norm, $\|f\| = (\sum_i \sum_j |f_{i,j}|^2)^{1/2}$ its Euclid length, and $height(f) = \max_{i,j} |f_{i,j}|$ its height. Throughout this paper $p_\lambda(x)$ is the minimal polynomial of an algebraic number λ over $\mathbb{Q}$, i.e. $p_\lambda(x) \in \mathbb{Z}[x]$ is the unique primitive polynomial of smallest degree such that $p_\lambda(\lambda) = 0$. The degree and height of an algebraic number are the degree and the height respectively of its minimal polynomial. And we denote the degree of the algebraic number λ by M, i.e. $M = [\mathbb{Q}(\lambda) : \mathbb{Q}] = \deg(p_\lambda)$. The real and imaginary parts of a complex number z will be denoted $Re(z)$ and $Im(z)$ respectively.

As is well known that factoring a polynomial over $\mathbb{Q}[x, y]$ is equivalent to factoring a primitive polynomial over $\mathbb{Z}[x, y]$. Thus we denote by $f(x, y)$ a primitive polynomial to be factored in $\mathbb{Z}[x, y]$ with $\deg_x(f) = n > 0$, $\deg_y(f) = m > 0$ for the rest of this paper.

Now we are ready to describe the idea behind our algorithm. We determine the irreducible factors of f in $\mathbb{Z}[x, y]$ as follows: At first we use an algebraic number λ with degree $M > 2m(n + 1)$ instead of the variable y in $f(x, y)$ and compute an approximation to a root of $f(x, \lambda)$. We denote by α the root. And then we look for the minimal polynomial of α over $\mathbb{Q}(\lambda)$, which is denoted by $h(x, \lambda) \in \mathbb{Z}[\lambda][x]$. From Lemma 1 we know $h(x, y)$ is a factor of $f(x, y)$ in $\mathbb{Z}[x, y]$. According to Lemma 6 we know $h(x, y)$ is irreducible. This is repeated until all the factors are found.

Lemma 1. *Let $f(x, y)$ be an polynomial of $\deg_x(f) = n > 0$ and $\deg_y(f) = m > 0$ in $\mathbb{Z}[x, y]$, λ an algebraic number with degree $M > (n + 2)m$. If $h(x, y) \in \mathbb{Z}[x, y]$ with $\deg_x(h) \le n$ and $\deg_y(h) \le m$ satisfies $h(x, \lambda)$ is a factor of $f(x, \lambda)$ in $\mathbb{Z}[\lambda][x]$, then $h(x, y)$ is a factor of $f(x, y)$ in $\mathbb{Z}[x, y]$.*

Proof. Since $h(x, \lambda)|f(x, \lambda)$, we have

$$f(x, \lambda) = h(x, \lambda)g(x, \lambda). \tag{1}$$

By the successive pseudo division of $f(x, y)$ and $h(x, y)$ with respect to x, we have

$$I(y)^t f(x, y) = q(x, y)h(x, y) + r(x, y), \tag{2}$$

where $I(y) = lc_x(h(x, y))$, $q, h, r \in \mathbb{Z}[x, y]$, $t \in \mathbb{N}$, and $\deg_x(r) < \deg_x(h)$. So $I(\lambda)^t f(x, \lambda) = q(x, \lambda)h(x, \lambda) + r(x, \lambda)$. Together with (1) we have

$$h(x, \lambda)(I(\lambda)^t g(x, \lambda) - q(x, \lambda)) = r(x, \lambda).$$

Comparing the degrees of x in two sides gives $r(x, \lambda) = 0$. Set $r(x, y) = \sum_{i=0}^{\deg_x(r)} \sum_{j=0}^{\deg_y(r)} r_{i,j} x^i y^j$, then

$$\sum_{i=0}^{\deg_x(r)} (\sum_{j=0}^{\deg_y(r)} r_{i,j} \lambda^j) x^i = 0. \tag{3}$$

From (2) we find

$$\deg_y(I(y)) \le \deg_y(h),$$

$$\deg_y(r) \le \deg_y(I(y)^t) + \deg_y(f) \\ \le t \cdot \deg_y(h) + \deg_y(f),$$

and

$$t \le \deg_x(f) - \deg_x(h) + 1 \le n+1,$$

so $\deg_y(r) \le (n+2)m < M$. Thus $r_{i,j} = 0$ from (3), i.e. $r(x,y) = 0$. Simultaneously $q(x,y) = I(y)^t g(x,y)$, so $h(x,y)$ is a factor of $f(x,y)$. □

2.2 How to Approximate

To avoid intermediate expression swell problem we do not work with λ, but work with some approximation $\bar{\lambda}$ to λ. Therefore, we denote by $\bar{\lambda}_j$ for $0 \le j \le m$ approximations to λ^j where $\bar{\lambda}_0 = 1$. We introduce the following notation for the approximate value of $f(x,y)$ at (x,λ): $f_{\bar{\lambda}} = \sum_i \sum_j f_{i,j} x^i \bar{\lambda}_j$. In our algorithm we will work with $f_{\bar{\lambda}}$ instead of $f(x,\lambda)$. We may assume that $f_{\bar{\lambda}}$ has a root with absolute value at most 1, or we consider the polynomial $x^n f(\frac{1}{x}, y)$ instead of $f(x,y)$.

Next we investigate how close λ should be approximated to enable a zero of $f_{\bar{\lambda}}$ to be an approximation to a root of $f(x,\lambda)$.

Lemma 2. *([31], Lemma 1.3) Let $f = \sum_{i=0}^n f_i x^i$, $\bar{f} = \sum_{i=0}^n \bar{f}_i x^i \in \mathbb{C}[x]$ be two polynomials of degree $n > 0$, and let $\Delta = \max_{0 \le i \le n} |f_i - \bar{f}_i|$. Suppose that $\bar{f}$ has a root $\beta \in \mathbb{C}$ satisfying $|\beta| \le 1$. Then there exists a zero $\alpha \in \mathbb{C}$ of f such that*

$$|\beta - \alpha| \le (\frac{(n+1)\Delta}{|f_n|})^{1/n}.$$

Proof. Since $f(x) - \bar{f}(x) = \sum_{i=0}^n (f_i - \bar{f}_i) x^i$, we get $|f(\beta)| \le \Delta \sum_{i=0}^n |\beta|^i$. Also $|f(\beta)| = |f_n| \prod_{i=1}^n |\beta - \alpha_i|$, where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the zeros of f. □

Lemma 3. *Let $f(x,y) = \sum_{i=0}^n \sum_{j=0}^m f_{i,j} x^i y^j$ and λ be an algebraic number. If $f(x,\lambda) = \sum_{i=0}^n f_i x^i$ satisfy $f_n \ne 0$, then*

$$|f_n| \ge ((1+m) height(f))^{1-M} \|p_\lambda\|^{-m},$$

where p_λ is the minimal polynomial of λ and $\deg(p_\lambda) = M > m$.

Proof. For any polynomial $g = \sum_{i=0}^d g_i x^i \in \mathbb{Z}[x]$ of degree d with the complex roots $z_1, z_2, \ldots, z_d$ we define the Mahler measure $M(g)$ by

$$M(g) = |g_d| \prod_{j=1}^d \max\{1, |z_j|\}.$$

The Mahler measure of an algebraic number λ is defined to be the measure of its minimal polynomial, i.e. $M(\lambda) = M(p_\lambda)$.

Since $0 \ne f_n = \sum_{j=0}^m f_{n,j} \lambda^j$. Hence for the polynomial $f_n(y) = \sum_{j=0}^m f_{n,j} y^j \in \mathbb{Z}[y]$, we have $f_n = f_n(\lambda) \ne 0$. So

$$|f_n| = |f_n(\lambda)| \ge \| f_n(y) \|_1^{1-M} \ M(\lambda)^{-m} \quad (4)$$

can be derived from Lemma 4. Since $||f_n(y)||_1 \le (1+m) height(f_n(y))$, $M(\lambda) \le \|p_\lambda\|$ (see [34], Theorem 6.31) and $height(f_n(y)) \le height(f)$, combined with (4) the proof is complete. □

Lemma 4. *([26], lemma 3) Let $\alpha_1, \ldots, \alpha_q$ be algebraic numbers of exact degree of $d_1, \ldots, d_q$ respectively. Define $D = [\mathbb{Q}(\alpha_1, \ldots, \alpha_q) : \mathbb{Q}]$. Let $P \in \mathbb{Z}[x_1, \ldots, x_q]$ have degree at most N_h in $x_h (1 \le h \le q)$. If $P(\alpha_1, \ldots, \alpha_q) \ne 0$, then*

$$|P(\alpha_1, \ldots, \alpha_q)| \ge \| P \|_1^{1-D} \prod_{h=1}^q M(\alpha_h)^{-DN_h/d_h}.$$

Proof. See Lemma 2 of [8]. □

Lemma 5. *Let β be a 2^{-s-1}-approximation[2] of a zero of absolute value at most 1 of $f_{\bar{\lambda}}$. For all k, if*

$$|\lambda^k - \bar{\lambda}_k| < (2^{sn+n}(n+1)((1+m)height(f))^M \ \|p_\lambda\|^m)^{-1}, \quad (5)$$

then β is also a 2^{-s}-approximation of a zero of $f(x,\lambda)$.

Proof. Let $f(x,\lambda) = \sum_{i=0}^n f_i x^i$ and $f_{\bar{\lambda}}(x) = \sum_{i=0}^n \bar{f}_i x^i$. According to Lemma 2, there exists a root α of $f(x,\lambda)$ such that

$$|\beta - \alpha| \le (\frac{(n+1)\max_i |f_i - \bar{f}_i|}{|f_n|})^{1/n}.$$

According to Lemma 3 and

$$\max_i |f_i - \bar{f}_i| \le height(f) \sum_{j=1}^m |\lambda^j - \bar{\lambda}_j|,$$

the proof easily follows from (5). □

***Remark** 1.* Suppose we have computed a 2^{-s-1}- approximation $\bar{\alpha} \in \mathbb{Q}(i)$ with $|\bar{\alpha}| \le 1$, to a root of $f_{\bar{\lambda}}$. According to Lemma 5 $\bar{\alpha}$ is also a 2^{-s}-approximation to $\alpha \in \mathbb{C}$, a root of $f(x,\lambda)$. So

$$|\alpha| \le 1 + 2^{-s} \quad (6)$$

since $|\alpha| - |\bar{\alpha}| \le |\alpha - \bar{\alpha}| \le 2^{-s}$.

2.3 Lattice

In the rest of this paper, we denote by $\beta_{ij} \in \mathbb{Q}(i)$ for $0 \le i \le n$ and $0 \le j \le m$ the approximations to $\alpha^i \lambda^j$, where $\beta_{00} = 1$.

2.3.1 LLL Algorithm

For convenience of our description, we state the following definitions and LLL Algorithm.

***Definition** 1.* A lattice $L \subset \mathbb{R}^n$ is a set of the form

$$\{\sum_{i=1}^k r_i b_i : r_i \in \mathbb{Z}\},$$

where $b_1, b_2, \ldots, b_k$ are independent vectors in $\mathbb{R}^n$. The lattice L is said to be generated by the vectors $b_1, b_2, \ldots, b_k$ which form a *basis* for L, and k is the *rank* or *dimension* of L.

***Definition** 2.* Let $b_1, b_2, \ldots, b_k \in \mathbb{R}^n$ be linearly independent and $(b_1^*, b_2^*, \ldots, b_k^*)$ the corresponding Gram-Schmidt orthogonal basis. Then $b_1, b_2, \ldots, b_k$ is *reduced* if $|b_i^*|^2 \le 2|b_{i+1}^*|^2$ for $1 \le i < n$.

***Remark** 2.* Roughly speaking, a reduced basis is a basis made of reasonably short vectors which are almost orthogonal. There exist many different notions of reduction, such as those of Hermite, Minkowski, Korkine-Zolotarev, Venkov, Lenstra-Lenstra- Lovász, etc (see [27]).

[2]We call b is a 2^{-s}-approximation of a if $|a - b| < 2^{-s}$.

In our algorithm we need the last one, LLL reduction, which is implemented as follows.

Algorithm 1. (LLL, [34] ALGORITHM 16.10)
Input: Linearly independent vectors $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$
Output: A reduced basis $(v_1, v_2, \ldots, v_k)$ of lattice $L = \sum_{i=1}^{k} \mathbb{Z} b_i$.

1. **for** $i = 1$ to k
   (a) $v_i \leftarrow b_i$
   (b) compute the GSO (Gram-Schmidt Orthogonalization)
   (c) $i \leftarrow 2$
2. **while** $i \leq k$
   (a) **for** $j = i-1$ to 1
      i. $v_i \leftarrow v_i - \lceil \mu_{ij} \rfloor v_j$.
      ii. update the GSO.
      endfor
   (b) **if** $i > 1$ and $|v_{i-1}^*|^2 > 2|b_i^*|^2$ **then**
      i. exchange g_{i-1} and g_i
      ii. update the GSO
      iii. $i \leftarrow i - 1$
      else $i \leftarrow i + 1$
3. **return** $(v_1, v_2, \ldots, v_k)$

2.3.2 Relations between LLL Algorithm and Ours'

For each positive integer s, we define the lattice $L_s \subset \mathbb{R}^{(n+1)(m+1)+2}$ generated by $b_{00}, b_{01}, \ldots, b_{0,m}, b_{10}, \ldots, b_{nm}$ which are the rows of the following $[(n+1)(m+1)] \times [(n+1)(m+1)+2]$ matrix

$$B_s = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 2^s Re(\beta_{00}) & 2^s Im(\beta_{00}) \\ 0 & 1 & 0 & \ldots & 0 & 2^s Re(\beta_{01}) & 2^s Im(\beta_{01}) \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 2^s Re(\beta_{nm}) & 2^s Im(\beta_{nm}) \end{pmatrix}.$$

We consider a map $\mathbb{Z}[x,y] \to L_s$. Corresponding to a polynomial $g(x,y) = \sum_i \sum_j g_{i,j} x^i y^j \in \mathbb{Z}[x,y]$ of $\deg_x(g) \leq n$ and $\deg_y(g) \leq m$, we have a vector in L_s defined by $\bar{g} = \sum_i \sum_j g_{i,j} b_{ij}$. We denote $g_\beta = \sum_{i=0}^{n} \sum_{j=0}^{m} g_{i,j} \beta_{ij}$, where $g_{i,j} = 0$ for $\deg_x(g) < i \leq n$ or $\deg_y(g) < j \leq m$. Clearly we have

$$\|\bar{g}\|^2 = \|g\|^2 + 2^{2s} |g_\beta|^2. \tag{7}$$

We run the celebrated LLL algorithm on b_{00}, b_{01}, $\ldots$, $b_{0,m}$, b_{10}, $\ldots$, b_{nm} to get a reduced basis of L_s. Suppose $\bar{v}$ is the first vector of the reduced basis. Then we have (see [22], Proposition (1.11))

$$\|\bar{v}\|^2 \leq 2^{(n+1)(m+1)-1} \|\bar{h}\|^2, \tag{8}$$

where $\bar{h}$ is the corresponding vector of $h(x,y)$ such that $h(x,\lambda) \in \mathbb{Z}[\lambda][x]$ is the minimal polynomial of α over $\mathbb{Q}(\lambda)$.

If $g \in \mathbb{Z}[x,y]$ with $\deg_x(g) \leq n$ and $\deg_y(g) \leq m$ such that $g(\alpha,\lambda) \neq 0$ then we will show that

$$\|\bar{g}\|^2 > 2^{(n+1)(m+1)-1} \|\bar{h}\|^2 \tag{9}$$

for a suitable choice of s (Lemma 9).

By $v(x,y)$ we denote the corresponding polynomial of $\bar{v}$. From (8) and (9) we know $v(\alpha,\lambda) = 0$, thus $h(x,\lambda) | v(x,\lambda)$. In fact $h(x,\lambda)$ must be $\pm v(x,\lambda)$ since $\bar{v}$ belongs to the basis of L_s of which $\bar{h}$ is an element.

So far we have found $h(x,\lambda)$ which is the minimal polynomial of α over $\mathbb{Q}(\lambda)$, where α is a root of $f(x,\lambda)$ with $|\alpha| \leq 1$. Then the polynomial $h(x,y) \in \mathbb{Z}[x,y]$ can be obtained by replacing λ of $h(x,\lambda)$ by y. Obviously $h(x,\lambda)$ divides $f(x,\lambda)$. From Lemma 1 we know $h(x,y)$ is a factor of $f(x,y)$ such that $h(\alpha,\lambda) = 0$, and the following lemma guarantees the irreducibility of $h(x,y)$ in $\mathbb{Z}[x,y]$.

Lemma 6. *Let $h(x,\lambda) \in \mathbb{Z}[\lambda][x]$ be the minimal polynomial over $\mathbb{Q}(\lambda)$ of an algebraic number α and $[\mathbb{Q}(\lambda) : \mathbb{Q}] = M > \deg_y(h(x,y))$, where $h(x,y)$ is obtained by replacing λ of $h(x,\lambda)$ by y. Then $h(x,y)$ is irreducible in $\mathbb{Z}[x,y]$.*

PROOF. Suppose $h(x,y) = h_1(x,y)h_2(x,y)$ where $h_1, h_2 \in \mathbb{Z}[x,y]$. Then $h(x,\lambda) = h_1(x,\lambda)h_2(x,\lambda)$. Since $h(x,\lambda)$ is the minimal polynomial of α in $\mathbb{Q}(\lambda)[x]$, we have $h_1(x,\lambda) = 1$ or $h_2(x,\lambda) = 1$. Without loss of generality we set $h_1(x,\lambda) = 1$. Since $M > \deg_y(h(x,y))$ and by using the same technique in the proof of Lemma 1 we have $h_1(x,y) = 1$. Therefore $h(x,y)$ is irreducible in $\mathbb{Z}[x,y]$. □

2.4 A Lower Bound of $\|\bar{g}\|$

For proving (9) we need some lemmas below.

Lemma 7. *Let n, m, λ, s, α, β_{ij} be as above and $g \in \mathbb{Z}[x,y]$ with $\deg_x(g) \leq n$, $\deg_y(g) \leq m$ such that $g(\alpha,\lambda) \neq 0$. If*

$$|\alpha^i \lambda^j - \beta_{ij}| \leq 2^{-s+1} \tag{10}$$

for $0 \leq i \leq n$ and $0 \leq j \leq m$, then

$$|g(\alpha,\lambda) - g_\beta| \leq 2^{-s+1}(mn+m+n)height(g).$$

PROOF. Obviously. □

The following lemma gives a lower bound for $|g(\alpha,\lambda)|$ when $g(\alpha,\lambda) \neq 0$.

Lemma 8. *Let $f(x,y)$ be a polynomial in $\mathbb{Z}[x,y]$ with $\deg_x(f) = n$ and $\deg_y(f) = m$, $g(x,y) \in \mathbb{Z}[x,y]$ with $\deg_x(g) \leq n$ and $\deg_y(g) \leq m$. Let λ be an algebraic number of degree $M \geq 2mn$ and $|\lambda| \leq 1/2$, α a root of $f(x,\lambda)$, $h(x,\lambda)$ the minimal polynomial[3] of α in $\mathbb{Z}[\lambda][x]$. If $g(\alpha,\lambda) \neq 0$, then*

$$|g(\alpha,\lambda)| \geq \frac{((2mn+1)^{\frac{1}{2}} B)^{1-M} \|p_\lambda\|^{-2mn}}{4nB}, \tag{11}$$

where $B = (2^{m+n} height(f) height(g)(n+1)^{\frac{3}{2}}(m+1)^{\frac{5}{2}})^n$.

PROOF. If $\deg_x(g) = 0$, then $g(\alpha,y) = g(y) \in \mathbb{Z}[y]$. And $g(\lambda) \neq 0$ since $M \geq 2mn > m \geq \deg_y(g)$. According to Lemma 4, we have

$$|g(\lambda)| \geq \| g \|_1^{1-M} \|p_\lambda\|^{-m}. \tag{12}$$

So (11) follows from (12), which can be easily checked.

Now let $\deg_x(g) > 0$. Since $h(x,\lambda)$ is the minimal polynomial of α in $\mathbb{Q}(\lambda)[x]$ and $g(\alpha,\lambda) \neq 0$ we have $\gcd(h,g) = 1$ if

[3]Here, $h(x,\lambda)$ is the minimal polynomial in $\mathbb{Q}(\lambda)[x]$ of α such that $h(x,\lambda)$ is of the minimal degree in λ. So $\deg_\lambda(h(x,\lambda)) \leq m$.

we regard h and g as polynomials in $\mathbb{Z}[x,y]$, and there exist polynomials $a, b \in \mathbb{Z}[x,y]$ such that

$$a \cdot h + b \cdot g = R, \tag{13}$$

where $R = Res_x(g,h) \in \mathbb{Z}[y]$. Since $\deg_x(h) \le n$ and $\deg_y(h) \le m$, we have $\deg_y(R) \le m \deg_x(g) + n \deg_y(g) \le 2mn$, $\deg_x(a) \le \deg_x(g) - 1$, $\deg_y(a) \le m(\deg_x(g)-1) + n\deg_y(g)$, $\deg_x(b) \le n-1$ and $\deg_y(b) \le m\deg_x(g) + (n-1)\deg_y(g)$.

Substituting α for x and λ for y in (13), we get

$$b(\alpha,\lambda) g(\alpha,\lambda) = R(\lambda),$$

hence

$$|g(\alpha,\lambda)| = \frac{|R(\lambda)|}{|b(\alpha,\lambda)|}. \tag{14}$$

Since $R(\lambda) \neq 0$ we have

$$\begin{aligned} |R(\lambda)| &\ge \| R \|_1^{1-M} |p_\lambda|^{-deg_y(R)} \\ &\ge ((1+2mn)^{1/2}\|R\|)^{1-M} \|p_\lambda\|^{-2mn} \\ &\ge ((1+2mn)^{1/2}B)^{1-M}\|p_\lambda\|^{-2mn}. \end{aligned} \tag{15}$$

The first part of (15) is from Lemma 4. The second is from $\|R\|_1 \le (1+\deg_y(R))^{1/2}\|R\|$ and $\deg_y(R) \le 2mn$.

Since $h(x,y) | f(x,y)$, from [31] we have

$$\begin{aligned} height(h) &\le \|h\| \le 2^{m+n}\|f\| \\ &\le 2^{m+n}(1+n)^{\frac{1}{2}}(1+m)^{\frac{1}{2}} height(f), \end{aligned}$$

and from a Hadamard-type bound on the coefficients of a determinant of polynomials [11], we know

$$height(R) \le \|R\| \le B. \tag{16}$$

So the third part of (15) follows.

From (6) we have $|\alpha|^i \le (1+2^{-s})^n \le 2$, and combining with $|\lambda| \le 1/2$ we derive

$$\begin{aligned} |b(\alpha,\lambda)| &\le height(b) \sum_{i=0}^{n-1} \sum_{j=0}^{2mn-1} |\alpha^i||\lambda^j| \\ &\le 4n \cdot height(b) \\ &\le 4nB. \end{aligned} \tag{17}$$

The last part of (17) is from (16) also holds with R replaced by b. Therefore (11) follows from (14), (15) and (17). $\square$

***Remark** 3.* As a matter of fact, (17) holds not only from (16) but also from $M \ge 2mn$. Combined with the condition $M > (n+2)m$ in Lemma 1, we choose the algebraic number λ such that $|\lambda| \le 1/2$ and $M = 2m(n+1)$ in our algorithm.

The following lemma shows how s should be chosen.

Lemma 9. *Let f, g, h, $\bar{g}$, $\bar{h}$, m, n, α, λ, β_{ij}, B, M be as above. If*

$$2^s \ge \frac{2^{\frac{mn+3(m+n)+10}{2}}\|f\|(m+1)^2(n+1)^2}{(1+2mn)^{\frac{1-M}{2}}\|p_\lambda\|^{-2mn}} \cdot A^M, \tag{18}$$

where $A = (2^{\frac{mn+5(m+n)+2}{2}}\|f\|^2(n+1)^{\frac{5}{2}}(m+1)^{\frac{7}{2}})^n$, then the following inequalities hold:

$$\|\bar{h}\| < 2^{m+n+1}\|f\|(n+1)(m+1), \tag{19}$$

$$\|\bar{g}\| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1). \tag{20}$$

Proof. Since h divides f we have $\deg_x(h) \le n$ and $\deg_y(h) \le m$ so that $\|\bar{h}\|$ is well defined, and from [31] we have $height(h) \le \|h\| \le 2^{m+n}\|f\|$. Together with $h(\alpha,\lambda) = 0$ and Lemma 7, the upper bound on $\|\bar{h}\|$ follows:

$$\begin{aligned} \|\bar{h}\|^2 &= \|h\|^2 + 2^{2s}|h_\beta|^2 \\ &\le (2^{m+n}\|f\|)^2 + 2^{2s}(2^{-s+1}2^{m+n}\|f\|(mn+n+m))^2 \\ &= (2^{m+n}\|f\|)^2(1+4(mn+n+m)^2) \\ &< (2^{m+n+1}\|f\|(n+1)(m+1))^2. \end{aligned}$$

We now wish to prove (20). Since $\|\bar{g}\|^2 = \|g\|^2 + 2^{2s}|g_\beta|^2$, we consider two cases:

If $\|g\| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$, so is $|\bar{g}|$.

If $\|g\| < 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$, we then prove

$$2^s|g_\beta| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1). \tag{21}$$

From Lemma 7 and Lemma 8 we have

$$\begin{aligned} |g_\beta| &\ge |g(\alpha,\lambda)| - 2^{-s+1}height(g)(mn+m+n) \\ &\ge \frac{(1+2mn)^{\frac{1-M}{2}}\|p_\lambda\|^{-2mn}}{4nA^M} \\ &\quad - 2^{-s+1}2^{(mn+3(m+n)+2)/2}\|f\|(m+1)^2(n+1)^2 \\ &\ge \frac{(1+2mn)^{\frac{1-M}{2}}\|p_\lambda\|^{-2mn}}{8nA^M}. \end{aligned}$$

So (21) holds from choosing s as (18). $\square$

Lemma 10. *Let f, h, n, m, λ, s, L_s be as above such that (18) holds. If $\bar{h} \in L_s$, then $h = \pm v$ and in particular*

$$\|\bar{v}\| < 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1). \tag{22}$$

Proof. If $\bar{h} \in L_s$ then $\|\bar{v}\| \le 2^{((m+1)(n+1)-1)/2}|\bar{h}|$ by (8). So with (19), (20) and from Lemma 9 we have $\|\bar{v}\| < 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$. This implies h divides v. Since $\bar{h} \in L_s$ and $\bar{v}$ is contained in a basis for L_s we conclude that $h = \pm v$. $\square$

***Remark** 4.* Actually, we try the values for $n_0 = 1, 2, \dots, n$ and for each $m_0 = 0, 1, \dots, m$ in our algorithm, i.e. the rank of L_s is $N = n_0(m+1) + m_0 + 1$. If N is the minimal such that $\bar{h} \in L_s$, then (22) also holds. A factor of $f(x,y)$ has obtained. If $\bar{h} \notin L_s$, then N is too small and we need another lattice whose rank is greater than N. If (22) does not hold for any N, then $h = f$.

3. THE MAIN ALGORITHM

In this section, we first describe our main algorithm, and then analyze the correctness and the cost of the algorithm. We also give some examples to illustrate our algorithm more clearly and generalize our algorithm to more general cases.

3.1 Description of Our Algorithm

Lemma 10 clearly leads to the following algorithm for factoring a bivariate polynomial.

Algorithm 2.

Input: a bivariate primitive polynomial $f(x,y) \in \mathbb{Z}[x,y]$ with $\deg_x(f) = n$ and $\deg_y(f) = m$, an algebraic number λ with degree $M = 2m(n+1)$ and $|\lambda| \le 1/2$.

Output: all the irreducible factors of $f(x,y)$ in $\mathbb{Z}[x,y]$.

1. Let i_0, $j_0 \in \mathbb{N}$ be the maximal degrees of $x^i y^j$ such that $x^i y^j | f(x, y)$. Then $f \leftarrow f(x, y)/x^{i_0} y^{j_0}$.
2. **repeat**
   (a) Compute $\bar{\lambda}_j \in \mathbb{Q}$ for $0 \leq j \leq m$ such that (5) holds.
   (b) $f_{\bar{\lambda}} \leftarrow \sum_{i=0}^{n} \sum_{j=0}^{m} f_{i,j} x^i \bar{\lambda}_j$.
   (c) **If** $f_{\bar{\lambda}}$ has not a root α such that $|\alpha| < 1$, **then** $f_{\bar{\lambda}} \leftarrow x^n f_{\bar{\lambda}}(1/x)$.
   (d) Apply the algorithm from [30] to compute a 2^{-s-1}-approximation $\bar{\alpha}$ to a root α with absolute value at most 1 of $f_{\bar{\lambda}}$.
   (e) Compute β_{ij} for $0 \leq i \leq n$ and $0 \leq j \leq m$ such that (10) holds.
   (f) **for** $n_0 = 1$ to n
       for $m_0 = 0$ to m
       i. Take $s \in \mathbb{Z}$ minimal such that (18) holds and **call Algorithm 1** to compute a reduced basis for L_s of rank N.
       ii. **If** (22) holds, then
           A. Output $h = v$.
           B. $f \leftarrow f/h$.
           C. **If** f is not a univariate polynomial, **then** goto step 2
           else
           call **Factor(f)** and output.
       endfor
       endfor
   (g) Output $h = f$.
   (h) $f \leftarrow f/h$.
3. **until** $f = 1$.

***Remark** 5.* Firstly, Algorithm 2 processes a simple case in the step 1, which guarantees that any root of f_{λ} is not equal to 0. This implies that β_{ij} are also not equal to 0. Thus Algorithm 1 can be applied to L_s correctly.

***Remark** 6.* It is possible that f is a univariate polynomial after running the step 2→(f)→(ii)→B. When this case happened, Algorithm 2 calls an arbitrary algorithm **Factor** which is used to factorize univariate polynomials and then outputs the result directly.

3.2 Correctness and Time Analysis

We now analyze the running time of our algorithm.

From (18) we have

$$s = O(n^3 m^2 + n^2 m \log \|f\|). \tag{23}$$

According to [31] a 2^{-s-1}-approximation $\bar{\alpha}$ to a root of absolute value at most 1 of $f_{\bar{\lambda}}$ can be computed in

$$O(n^2 (\max\{s, n \log \|f_{\bar{\lambda}}\|\})^{1+\epsilon})$$

bit operations. Since $\log \|f_{\bar{\lambda}}\| = O(mn^2(n^3m^2+n^2m \log \|f\|))$ (cf. (5) and (23)), it will cost

$$O(mn^5(n^3m^2 + n^2m \log \|f\|)^{1+\epsilon})$$

bit operations to compute $\bar{\alpha}$.

From [31] we know that $O(\bar{n}n^2m^3(n^3m^2+n^2m \log \|f\|)^{2+\epsilon})$, where $\bar{n} = \deg_x(h)$, bit operations suffice to compute h. Since $\|f/h\| \leq 2^{m+n}\|f\|$, the complete factorization of f can be found in $O(n^3m^3(n^3m^2 + n^2m \log \|f\|)^{2+\epsilon})$ bit operations. So we have

Theorem 1. *Algorithm 2 correctly computes a factorization of $f(x, y) \in \mathbb{Q}[x, y]$ and runs in polynomial time. It uses*

$$O(n^3m^3(n^3m^2 + n^2m \log \|f\|)^{2+\epsilon})$$

bit operations.

PROOF. From Sect. 2 and the analysis above, this proof is obvious. □

3.3 Experiments

The described algorithm has been successfully carried out many times in Maple 11 on the same PC (Pentium IV 2.53G CPU, 384Mb of main memory). Here we only use several simple examples to better illustrate some steps of Algorithm 2 in detail.

In the following examples, we always set $\lambda = (\frac{1}{3})^{1/2m(n+1)}$. Then $p_\lambda(x) = 3x^{2m(n+1)} - 1$, hence $\|p_\lambda\| = \sqrt{10}$. A vector $v = (v_{00}, \ldots, v_{0m}, v_{10}, \ldots, v_{nm})$ we get from LLL algorithm corresponds to a polynomial $\sum_{i=0}^{n} \sum_{j=0}^{m} v_{ij} x^i y^j$.

***Example** 1.* $f := 2\,x^2 + 3\,xy + 2\,x + y + y^2$.

Then $n = 2$, $m = 2$, $height(f) = 3$ and $\|f\| = \sqrt{19}$. We can first compute the minimal s such that (18) holds. In this example, s should be 32, while s should be 148 at least if we adopt the method in [31]. Secondly, we compute the approximate roots of $f_{\bar{\lambda}}$ with Maple 11.

$$[\ -.44254407603573012077377941295814,\ -1.8850881520714602415475588259163\ ]$$

Obviously, we choose

$$\bar{\alpha} = -.44254407603573012077377941295814$$

with absolute value ≤ 1.

By running Algorithm 2 we get a vector (0., 1., 0., 2., 0., 0., 0., 0., 0., 0., 0.) which corresponds to $2y + x$, a factor of f. After that the algorithm updates f by $f \leftarrow f/(2y+x) = 1 + y + x$. Obviously, the irreducible factorization of f is $(2y + x)(1 + y + x)$.

***Example** 2.* $f := 6\,x^3 + 8\,x^2 + 9\,x^2y + 6\,xy + 2\,x + y + 3\,y^2x + y^2$.

s should be chosen to be 76 in this example. This is about 7 times smaller than the method in [31]. After root finding and LLL computing, Algorithm 2 could give us $3x + 1$ which is a factor of f. Updating f and computing another factor $y + 2x$ gives a complete factorization of f as $(1 + y + x)\,(y + 2\,x)\,(1 + 3\,x)$. Worthy of being mentioned, Algorithm 2 has a more acceptable running time in this example, which is slightly faster than `factor()` in Maple 11. `factor()` needs 0.125 s, while Algorithm 2 needs only 0.015 s to factor f completely.

In Table 1, we show the performance of Algorithm 2 for some randomly generated polynomials on the same PC. Here (n, m) represents the bi-degree of the input polynomial, H represents the height of f and s_{HL} is the s that the method in [31] should choose, and s_{al} is the minimal s satisfied (18) in Algorithm 2. t_{fa} is the running time of the built-in function `factor()` in Maple 11. However, as can be seen from Theorem 1, the running time of Algorithm 2 is decided by

i	(n,m)	H	s_{HL}	s_{al}	s_l	$t_{al}(s)$	$t_{fa}(s)$
1	(2,2)	3	148	32	10	0.	0.
2	(3,2)	9	507	76	16	0.015	0.125
3	(4,4)	3	8000	1024	30	0.047	0.032
4	(4,5)	6	10000	2000	38	0.030	0.047
5	(2,2)	79247	4100	1024	71	0.125	0.187
6	(3,2)	782	500	72	63	0.014	0.
7	(4,4)	25382	8000	1024	108	0.047	0.063
8	(3,6)	3542	31104	1944	77	0.046	0.032

Table 1: Performance of Algorithm 2

the scale of s and $\|f\|$. Moreover s from (18) has a scale as in (23) and grows very fast when n, m or $\|f\|$ grows. That can greatly affect the performance of Algorithm 2. By a lot of tests, we find that s can be smaller than that from (18) in most examples. In Table 1, s_l represents the minimal s such that Algorithm 2 correctly gives all the irreducible factors of f. In fact, every t_{al} in Table 1 is the running time, obtained by the corresponding s_l, of Algorithm 2. From Table 1, we can see the running time of the examples above is slightly less than or near to the running time of `factor()` in Maple 11. The performance of Algorithm 2 can be improved if the problem of finding the best choice for the parameter s has been solved. Unfortunately, one such problem is being considered but can not be attempted at this time.

In addition, Algorithm 2 can be applied to factor multivariate polynomials with rational coefficients. By using the Hilbert irreducibility theorem, we can reduce a multivariate polynomial to a bivariate one. The basic idea was described in [6]. After this reduction and running Algorithm 2 we can find the bivariate polynomial's factors, from which the factors of the original multivariate polynomial can be recovered by using Hensel lifting.

4. CONCLUSION

We overcome the problem arising from the algorithm in [31] and present a new algorithm for completely factoring bivariate polynomials in $\mathbb{Q}[x, y]$. The key step of our algorithm is reducing a bivariate polynomial to a univariate polynomial by substituting an algebraic number of high degree for a variable. And then we use the lattice reduction basis algorithm to get the desired factors. The running time of our algorithm is not only $(\log^2(mn))^{2+\epsilon}$ times less than the algorithm in [31] but also less than or near to the running time of the order `factor()` in Maple 11 for some examples. Furthermore, our algorithm can be generalized easily to polynomials with variables more than two.

Acknowledgments

We would like to thank Prof. Arjen K. Lenstra for his valuable comments about Lemma (1.9) in [31]. We also wish to acknowledge the anonymous referees for their many detailed suggestions, which led to an improved version of the paper.

5. REFERENCES

[1] B.Hassibi and H. Vikalo. On the Sphere-decoding Algorithm I. Expected Complexity. *IEEE Trans. Signal Processing*, 53:2806–2818, 2005.

[2] A. Bostan, G. Lecerf, B. Salvy, É. Schost, and B. Wiebelt. Complexity Issues in Bivariate Polynomial Factorization. In *ISSAC '04: Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 42–49, 2004.

[3] G. Chèze and A. Galligo. From an Approximate to an Exact Absolute Polynomial Factorization. *Journal of Symbolic Computation*, 41(6):682–696, 2006.

[4] G. Chèze and G. Lecerf. Lifting and Recombination Techniques for Absolute Factorization. *Journal of Complexity*, 23(3):380–420, 2007.

[5] A. Chistov. An Algorithm of Polynomial Complexity for Factoring Polynomials, and Determination of the Components of a Variety in a Subexponential Time. *(Russian), Theory of the complexity of computations, II., Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov.(LOMI) 137*, pages 124–188, 1984.

[6] R. M. Corless, A. Galligo, I. S. Kotsireas, and S. M. Watt. A Geometric-numeric Algorithm for Absolute Factorization of Multivariate Polynomials. In *ISSAC '02: Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 37–45, 2002.

[7] J. D. Dora, A. Maignan, M. Mirica-Ruse, and S. Yovine. Hybrid Computation. In *ISSAC '01: Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 101–108, 2001.

[8] N. I. Fel'dman. Estimate for a Linear Form of Logarithms of Algebraic Numbers. *Sbornik: Mathematics*, 5(2):291–307, 1968.

[9] S. Gao. Factoring Multivariate Polynomials via Partial Differential Equations. *Math. Comput.*, 72(242):801–822, 2003.

[10] S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate Factorization of Multivariate Polynomials via Differential Equations. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 167–174, 2004.

[11] A. Goldstein and R. Graham. A Hadamard-type Bound on the Coeffcients of a Determinant of Polynomials. *SIAM Review*, 16(3):394–395, 1974.

[12] E. Kaltofen. A Polynomial-time Reduction from Bivariate to Univariate Integral Polynomial Factorization. In *23rd Annual Symposium on Foundations of Computer Science*, pages 57–64, 1982.

[13] E. Kaltofen. A Polynomial Reduction from Multivariate to Bivariate Integral Polynomial Factorization. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 261–266, 1982.

[14] E. Kaltofen. Polynomial Factorization 1982-1986. In *Computers in mathematics (Stanford, CA, 1986). Vol. 125 of Lecture Notes in Pure and Appl. Math. Dekker*, pages 285–309, 1990.

[15] E. Kaltofen. Polynomial Factorization 1987-1991. In *LATIN '92*, pages 294–313, 1992.

[16] E. Kaltofen. Effective Noether Irreducibility Forms and Applications. *Journal of Computer and System Sciences*, 50(2):274–295, 1995.

[17] E. Kaltofen. Polynomial Factorization: a success story. In *ISSAC '03: Proceedings of the 2003 international symposium on Symbolic and algebraic computation*, pages 3–4, 2003.

[18] E. Kaltofen and L. Zhi. Hybrid Symbolic-numeric Computation. In *ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 7–7, 2006.

[19] R. Kannan, A. Lenstra, and L. Lovász. Polynomial Factorization and Nonrandomness of Bits of Algebraic and Some Transcendental Numbers. *Math. Comput.*, 181:235–250, 1988.

[20] G. Lecerf. Sharp Precision in Hensel Lifting for Bivariate Polynomial Factorization. *Math. Comput.*, 75(254):921–934, 2006.

[21] A. Lenstra. Factoring Multivariate Integral Polynomials. *Theoretical Comput. Sci.*, 34:207–213, 1984.

[22] A. Lenstra, H. Lenstra, and L. Lovász. Factoring Polynormials with Rational Coeffcients. *Math. Ann.*, 261:513–534, 1982.

[23] A. K. Lenstra. Factoring Multivariate Polynomials over Finite Fields. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 189–192, 1983.

[24] A. K. Lenstra. Factoring Multivariate Polynomials over Algebraic Number Fields. *SIAM Journal on Computing*, 16(3):591–598, 1987.

[25] A. May. Using LLL-reduction for Solving RSA and Factorization Problems: A Survey. In *LLL+25 Conference in honour of the 25th birthday of the LLL algorithm*, 2007.

[26] M. Mignotte and M. Waldschmidt. Linear Forms in Two Logarithms and Schneider's method. *Math. Ann.*, 231:241–267, 1978.

[27] P. Nguyen and D. Stehlé. Low-dimensional Lattice Basis Reduction Revisited. In *Algorithmic Number Theory*, pages 338–357. 2004.

[28] A. Novocin. Factoring Univariate Polynomials over the Rationals. *Florida State University*, `http://www.math.fsu.edu/~anovocin`, 2008.

[29] A. Novocin and M. v. Hoeij. Factoring Univariate Polynomials over the Rationals. *ACM Commun. Comput. Algebra*, 42(3):157–157, 2008.

[30] A. Schönhage. Factorization of Univariate Integer Polynomials by Diophantine Aproximation and an Improved Basis Reduction Algorithm. In *Proceedings of the 11th Colloquium on Automata, Languages and Programming*, pages 436–447, 1984.

[31] M.-P. van der Hulst and A. Lenstra. Factorization of Polynomials by Transcendental Evaluation. In *EUROCAL '85*, pages 138–145, 1985.

[32] M. van Hoeij. Factoring Polynomials and 0-1 Vectors. In *Cryptography and Lattices*, pages 45–50, 2001.

[33] M. van Hoeij. Factoring Polynomials and the Knapsack Problem. *Journal of Number Theory*, 95(2):167–189, 2002.

[34] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, London, 1999.

[35] L. Zhi. Numerical Optimization in Hybrid Symbolic-numeric Computation. In *SNC '07: Proceedings of the 2007 international workshop on Symbolic-numeric computation*, pages 33–35, 2007.