MATHEMATICS OF COMPUTATION
VOLUME 54, NUMBER 190
APRIL 1990, PAGES 755–770

FACTORING MULTIVARIATE POLYNOMIALS OVER LARGE FINITE FIELDS

DAQING WAN

ABSTRACT. A simple probabilistic algorithm is presented to find the irreducible factors of a bivariate polynomial over a large finite field. For a polynomial $f(x, y)$ over F_q of total degree n, our algorithm takes at most

$$n^{4.89} \log^2 n \log q$$

operations in F_q to factor $f(x, y)$ completely. This improves a probabilistic factorization algorithm of von zur Gathen and Kaltofen, which takes

$$O(n^{11} \log n \log q)$$

operations to factor $f(x, y)$ completely over F_q. The algorithm can be easily generalized to factor multivariate polynomials over finite fields. We shall give two further applications of the idea involved in the algorithm. One is concerned with exponential sums; the other is related to permutational polynomials over finite fields (a conjecture of Chowla and Zassenhaus).

1. INTRODUCTION

The factorization of polynomials over finite fields has been studied for a long time. However, an efficient algorithm for factoring univariate polynomials over finite fields was not presented until the late 1960's. Berlekamp [2] then devised a deterministic algorithm which factors a univariate polynomial of degree n over F_q in $O(n^3 q)$ field operations. This running time is polynomial both in n and q. Soon after, Berlekamp [3] made the running time polynomial in the input size, i.e., using $\log q$ rather than q, at the expense of introducing a probabilistic rather than a deterministic method. The Cantor-Zassenhaus [5] probabilistic version of the algorithm factors a univariate polynomial of degree n in $O(n^3 + n^2 \log n \log q)$ operations. Various other algorithms for this problem are due to Rabin [17], Ben-Or [1], and Camion [4]. Unfortunately, there is no known deterministic method for factoring univariate polynomials over finite fields in time polynomial in the input size.

The corresponding problem of factoring multivariate polynomials over finite fields has recently been studied by numerous authors. There are several approaches. Lenstra [14] and Chistov-Grigoryev used the short vector algorithm for lattices from Lenstra, Lenstra, and Lovász [13]. Their algorithms

Received August 26, 1988.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 11T06, 68Q40.

are deterministic, and require $O(n^8 + n^3 q)$ field operations to factor a bivariate polynomial of total degree n. By establishing an effective version of the Hilbert Irreducibility Theorem, von zur Gathen [7] gave a probabilistic algorithm to compute the factorization pattern of a multivariate polynomial (i.e., the degrees and the multiplicities of the irreducible factors of a given multivariate polynomial). Utilizing similar ideas, von zur Gathen and Kaltofen [8] presented a probabilistic algorithm to factor a sparse multivariate polynomial.

In [9] von zur Gathen and Kaltofen described a complete probabilistic multivariate factorization method which uses Newton's iteration and linear equation systems. It takes $O(n^8 \log n \log q)$ field operations to factor a nice bivariate polynomial of total degree n. In general, it takes $O(n^{11} \log n \log q)$ field operations to factor a bivariate polynomial of total degree n over a large finite field F_q.

In this paper, we present a new probabilistic algorithm to factor bivariate polynomials over a finite field, which is simpler and faster than the above methods. This algorithm takes at most $O(n^{4.89} \log^2 n \log q)$ field operations to factor a bivariate polynomial of total degree n over F_q. It can also be generalized to factor multivariate polynomials over finite fields in time polynomial in the total degree of the polynomial. However, we shall concentrate on the bivariate case.

We remark that our algorithm can be adapted to many other fields instead of finite fields, for example, the rational number field. In this case, an interesting phenomenon is that, whenever the Berlekamp-Hensel algorithm succeeds in factoring the related univariate polynomial over the integers, our algorithm succeeds in factoring the bivariate polynomial over the integers. See the remarks at the end of §3 for more details. In view of this, we describe our main result in the more general setting as follows.

Let F be a field (arbitrary effectively computable) and $f(x, y)$ be a bivariate polynomial of degree n over F. By preprocessing $f(x, y)$, we may suppose $f(x, y)$ is nonsingular at infinity. We write

$$f(x, y) = f_n(x, y) + f_{n-1}(x, y) + \cdots + f_0(x, y), \tag{1}$$

where $f_k(x, y)$ is the homogeneous part of degree k in $f(x, y)$. Let

$$f_n(x, y) = p_1^{e_1}(x, y) \cdots p_t^{e_t}(x, y) \tag{2}$$

be the prime factorization of $f_n(x, y)$ over F. Then we can deterministically find all irreducible factors of f over F with at most $O(2^t n^3 \log^2 n)$ field operations in F.

To apply the above result in a particular field, one needs to know a preprocessing algorithm and a univariate factorization algorithm. In the case of finite fields, this does not present a problem. In §6, we shall show that a random univariate polynomial over F_q has small t, i.e., $t \leq e \log n$. Combining this with Berlekamp's probabilistic algorithm gives our algorithm, which takes at most $O(n^{3+e \log 2} \log^2 n \log q) = O(n^{4.89} \log^2 n \log q)$ field operations.

2. Motivation for the algorithm

Before we give the algorithm in the next section, we would like to see what we can extract from the assumption that $f(x, y)$ is reducible. This will produce the algorithm more naturally and serve as part of the proof of our algorithm in the next section. F is an arbitrary field in this section.

For simplicity, we first suppose that $f_n(x, y)$ is squarefree, i.e., $e_j = 1$ $(j = 1, \ldots, t)$. Now, let $f(x, y)$ be reducible over the field F. Then there are two nonconstant polynomials $g(x, y)$ and $h(x, y)$ over F such that

$$f(x, y) = g(x, y)h(x, y). \tag{3}$$

Let $\deg(g(x, y)) = r$ and $\deg(h(x, y)) = s$. Then, $s = n - r$. As in (1), we write

$$\begin{aligned} g(x, y) &= g_r(x, y) + g_{r-1}(x, y) + \cdots + g_0(x, y), \\ h(x, y) &= h_s(x, y) + h_{s-1}(x, y) + \cdots + h_0(x, y). \end{aligned} \tag{4}$$

Substituting (4) into (3) and comparing the homogeneous part of degree k, we get

$$\begin{aligned} f_n &= g_r h_s, \\ f_{n-1} &= g_r h_s \left(\frac{g_{r-1}}{g_r} + \frac{h_{s-1}}{h_s} \right), \\ &\cdots \\ f_{n-k} &= g_r h_s \left(\frac{\sum_{i=0}^{k} g_{r-i} h_{s-k+i}}{g_r h_s} \right), \end{aligned} \tag{5}$$

where we define $g_k = h_k = 0$ for $k < 0$.

We claim that $g(x, y)$ and $h(x, y)$ are uniquely determined by their highest-degree parts g_r and h_s, i.e., all the polynomials g_i and h_j are uniquely determined from f, g_r, and h_s. In fact, dividing (5) by $g_r h_s = f_n$, we have

$$\begin{aligned} \frac{f_{n-1}}{f_n} &= \frac{g_{r-1}}{g_r} + \frac{h_{s-1}}{h_s}, \\ \frac{f_{n-2} - g_{r-1}h_{s-1}}{f_n} &= \frac{g_{r-2}}{g_r} + \frac{h_{s-2}}{h_s}, \\ &\cdots \\ \frac{f_{n-k} - \sum_{i=1}^{k-1} g_{r-i} h_{s-k+i}}{f_n} &= \frac{g_{r-k}}{g_r} + \frac{h_{s-k}}{h_s}, \\ &\cdots \end{aligned} \tag{6}$$

where $1 \le k \le n$.

Since g_r and h_s are relatively prime (f_n is squarefree), from the first expression of (6) we find that g_{r-1} and h_{s-1} are uniquely determined and can be obtained, for example, by expanding the partial fraction of f_{n-1}/f_n and

then taking suitable partial sums (a more practical way is to use the Euclidean algorithm, as we will do in the algorithm). Similarly, the second expression of (6) implies that g_{r-2} and h_{s-2} can also be uniquely determined, and so on. In this way, we prove the claim.

Above we showed that if $f(x, y) = g(x, y)h(x, y)$, then the factors $g(x, y)$ and $h(x, y)$ can be constructively recovered using only $g_r(x, y)$ and $h_s(x, y)$. So, by trying all possible pairs g_r, h_s with $g_r h_s = f_n$, we can find all pairs g, h with $gh = f$. This naturally suggests an algorithm for factoring f. More precisely, for each given pair g_r, h_s with $g_r h_s = f_n$, by the above constructive method we can produce two polynomial sequences g_i $(i = r, r-1, \ldots, r-n)$ and h_j $(j = s, s-1, \ldots, s-n)$. If $g_i = h_j = 0$ for all $i, j < 0$, then we do find a pair of factors g, h such that $g = g_r + g_{r-1} + \cdots + g_0$, $h = h_s + h_{s-1} + \cdots + h_0$. Otherwise, f simply has no such factors, and we need to try another pair g_r, h_s with $g_r h_s = f_n$. This forms the basic part of the deterministic algorithm. It clearly works for any field for which an algorithm for factoring univariate polynomials is given.

There is an alternative in the construction of the above polynomial sequences. Without loss of generality, assume $r < s$. We may first construct only g_i $(i = r, r-1, \ldots, 0)$, h_j $(j = s, s-1, \ldots, s-r)$, and then check whether $g = g_r + \cdots + g_0$ divides f or not. If g divides f, we get a factor; otherwise, we need to try another pair g_r, h_s with $g_r h_s = f_n$.

We assumed that $f_n(x, y)$ is squarefree in the above discussion. If f_n has repeated factors, but any of these repeated factors is relatively prime to f_{n-1}, then the above argument still works. In this case, we claim that $(g_r, h_s) = 1$ (the crucial point). Otherwise, let $p(x, y)$ be a nontrivial prime factor of (g_r, h_s). The first two expressions of (5) would imply that $p(x, y) | f_{n-1}$ and $p^2(x, y) | f_n$. This is a contradiction. Now, the crucial condition $(g_r, h_s) = 1$ makes the rest of the above discussion work well.

The above weaker condition can be formulated using singularities of a projective curve. Let $f(x, y)$ be given as in (1). Then $f(x, y)$ defines a projective curve (possibly reducible) in $P^2(F)$ by its homogenization:

$$f(x, y, z) = f_n(x, y) + f_{n-1}(x, y)z + \cdots + f_0(x, y)z^n. \tag{7}$$

It is easy to verify that f is nonsingular at infinity $(z = 0)$ if and only if any repeated factor of f_n is relatively prime to f_{n-1}. Motivated by this, we introduce the following

Definition 2.1. f is called "nice" with respect to z if the curve $f(x, y, z) = 0$ has no singular points at infinity $(z = 0)$.

Note that our nice polynomials are similar to the nice polynomials defined by von zur Gathen and Kaltofen [9]. In their paper, a nice polynomial f satisfies: $f(x, 0)$ is squarefree and f is monic with respect to x. This later condition corresponds to our "normalized" polynomials (to be introduced).

We have seen that if f is nice with respect to z, an algorithm can be constructed to factor f. In the expressions (6), all the g_i and h_j are bivariate homogeneous polynomials. By dehomogenization, they are equivalent to univariate polynomials. This will simplify calculations in practice. To carry out this simplification, we can normalize $f(x, y)$ by requiring $f_n(x, y)$ to be monic with respect to x. Under this assumption, dehomogenization of g_i and h_j and their homogenization with respect to y are simply inverse processes to each other.

Definition 2.2. f is called "normalized" with respect to x if $f_n(x, y)$ is monic with respect to x.

3. The Basic Algorithm

In this section, F is still an arbitrary field. We give the basic algorithm for nice normalized polynomials. In many cases, a general polynomial can be preprocessed to become a nice normalized one, so the basic algorithm here can be applied in general. We also suppose an algorithm is given for factoring univariate polynomials over F. The algorithm will be allowed to be probabilistic, so that it will either return the correct answer or else fail, the latter with small probability. In the rest of this section, we assume that f is nice and normalized with respect to z and x, respectively. f has been given as in (1).

Basic Algorithm.

Input: A nice and normalized polynomial $f(x, y) \in F[x, y]$.

Output: A complete factorization of $f(x, y)$ over F or failure.

Step 1. Use the given algorithm for univariates to factor f_n completely over F:

$$f_n(x, y) = p_1^{e_1}(x, y) \cdots p_t^{e_t}(x, y),$$

where $p_i(x, y)$ are distinct prime factors of f_n over F, monic with respect to x. If the probabilistic univariate procedure returns failure, then return failure. Otherwise, go to Step 2.

Step 2. Consider all proper divisors of f_n of the form $p_{i_1}^{e_{i_1}} \cdots p_{i_k}^{e_{i_k}}$ $(1 \le i_1 < \cdots < i_k \le t,\ 1 \le k \le [t/2])$. List them in a table, say, $d_1, \ldots, d_N$, where

$$N = \binom{t}{1} + \binom{t}{2} + \cdots + \binom{t}{[t/2]} \le 2^{t-1}.$$

Step 3. Choose any element d in the table of Step 2. Let $\deg(d) = r$, $g_r(x) = d(x, 1)$, $h_s(x) = f_n(x, 1)/d(x, 1)$. Use the Euclidean algorithm to find univariate polynomials $u(x)$ and $v(x)$ in $F[x]$ such that

$$u(x)g_r(x) + v(x)h_s(x) = 1, \tag{8}$$

where $\deg(u) < n - r = s$ and $\deg(v) < r$.

Step 4. Inductively define the univariate polynomial sequences $g_i(x)$ $(r-n \le i \le r)$ and $h_j(x)$ $(s-n \le j \le s)$ as follows:

$$
\begin{aligned}
g_{r-k} &= v(x)\left(f_{n-k} - \sum_{i=1}^{k-1} g_{r-i}h_{s-k+i}\right) \bmod g_r(x), \\
h_{s-k} &= u(x)\left(f_{n-k} - \sum_{i=1}^{k-1} g_{r-i}h_{s-k+i}\right) \bmod h_s(x),
\end{aligned} \tag{9}
$$

where $f_{n-k} = f_{n-k}(x, 1)$ and $1 \le k \le n$.

If $\deg(g_i) > i$ or $\deg(h_j) > j$, then f has no such factors, and we go back to Step 3 and choose another d in the table.

Step 5. Let $G_i(x, y)$ and $H_j(x, y)$ be the homogenization of g_i and h_j with respect to y such that $\deg(G_i) = i$ and $\deg(H_j) = j$ $(0 \le i \le r,\ 0 \le j \le s)$. Return

$$
\begin{aligned}
g &= G_r + G_{r-1} + \cdots + G_0, \\
h &= H_s + H_{s-1} + \cdots + H_0.
\end{aligned}
$$

Then g and h are two proper factors of f satisfying $f = gh$. In this case, g and h are still nice and normalized with respect to z and x. We repeat the above algorithm until we find all irreducible factors of f.

As we pointed out in §2, there is an alternative in Step 4 and Step 5. Hence, Step 4 and Step 5 can be replaced by the following Step 4′.

Step 4′. Let $r < s$ (if $s < r$, interchange the positions of g and h). Inductively define the univariate polynomial sequences $g_i(x)$ $(0 \le i \le r)$ and $h_j(x)$ $(s-r \le j \le s)$ as in (9). Let $G_i(x, y)$ be the homogenization of g_i such that $\deg(G_i) = i$ $(0 \le i \le r)$ (note that for $0 \le i \le r$ one always has $\deg(g_i) \le i$). Put

$$
g = G_r + G_{r-1} + \cdots + G_0.
$$

Check whether g divides f or not. If g divides f, we find a factor. Otherwise, go back to Step 3 and choose another d in the table.

Proof of correctness of the algorithm. Recall the argument in §1. Instead of using partial fractions, we have now used the Euclidean algorithm. Clearly, we only need to prove that the G_i and H_j are the same as the $g_i(x, y)$ and $h_j(x, y)$ in §1. Now, (8) implies

$$
\frac{v(x)}{g_r(x)} + \frac{u(x)}{h_s(x)} = \frac{1}{f_n}. \tag{10}
$$

Thus, for any univariate polynomial $q(x)$ with $\deg(q) < n$, we have

$$
\begin{aligned}
\frac{q(x)}{f_n(x)} &= \frac{v(x)q(x)}{g_r(x)} + \frac{u(x)q(x)}{h_s(x)} \\
&= \frac{v(x)q(x) \bmod g_r(x)}{g_r(x)} + \frac{u(x)q(x) \bmod h_s(x)}{h_s(x)}.
\end{aligned} \tag{11}
$$

Now, comparing (11) and (6) (noting that f is normalized), we find that $G_i = g_i$ and $H_j = h_j$. This proves the correctness of the Basic Algorithm. □

For the analysis of running time, we assume that the factorization procedure used in Step 1 to factor a univariate polynomial of degree n takes at most $\tau(n)$ operations in F. We will allow a probabilistic procedure, which either correctly returns an irreducible factor or else fails.

Theorem 3.1. *Let $f(x, y) \in F[x, y]$ be a nice and normalized polynomial of degree n, and assume that Step 1 of the Basic Algorithm does not return failure. Then the Basic Algorithm deterministically finds all irreducible factors of f in $O(2^t n^3 \log^4 n) + \tau(n)$ operations in F, where $\log^4 n$ can be replaced by $\log^2 n$ if F has more than $2n$ elements.*

Lemma 3.2 [9]. *Let F be an arbitrary field, and $r(x) \in F[x]$ of degree d. Then an arithmetic operation ($+$, $-$, multiplication, and division by an invertible element) in $F[x]/(r(x))$ can be performed in $O(d \log^4 d)$ operations in F. If the cardinality of F is at least $2d$, then it can be performed in $O(d \log^2 d)$ operations.*

Proof of Theorem 3.1. Lemma 3.2 shows that Step 3 and Step 4 take $O(n^3 \log^4 n)$ operations in F. By the inequality for N in Step 2, we deduce that the algorithm can be performed in $O(2^t n^3 \log^4 n) + \tau(n)$ operations. The theorem is proved. □

Remarks. (i) The above algorithm has a bottleneck quite similar to that of the Berlekamp-Hensel algorithm for factoring a univariate polynomial over the integers, see [11]. In Step 2, we may have to test as many as 2^{t-1} potential factors $d(x)$ of f_n. If t is large, this presents an exponentially bad case. However, if we combine the Berlekamp-Hensel algorithm and our algorithm to factor a bivariate polynomial f of degree n over the integers, we find the following remarkable fact: If the Berlekamp-Hensel algorithm succeeds in factoring the univariate polynomial f_n over the integers (this forces t to be relatively small), then our algorithm will succeed in factoring f over the integers. In view of Landau's comments in her survey article [12], the Berlekamp-Hensel algorithm is considered to be a practical one for factoring a univariate polynomial over the integers. Thus, we expect that our algorithm is practical for factoring bivariates over the integers.

(ii) In Step 2, we order the elements $d_1, \ldots, d_N$ in such a way that d_i has no more distinct prime factors than d_j $(i < j)$ does. If f is reducible and the ordering is fortunate, then one can find a factor of f in only one step! On the other hand, if f is irreducible, then no best ordering exists; any ordering takes the same amount of time. It is an interesting but seemingly difficult problem to obtain an optimal ordering method. This question is equivalent to finding a factor d of f_n such that f has a factor of the form $d(x, y) +$ lower degree terms.

(iii) The above algorithm (in Step 2) takes at most N steps to factor f completely. However, it will take exactly N steps to prove that an irreducible polynomial is irreducible. Hence, this algorithm is more suitable for factoring a polynomial with many factors. On the other hand, if f has only a few factors, then the Hilbert Irreducibility Theorem shows that t is often small. By preprocessing f again, as we shall do in §5, we can expect to obtain smaller t, and then the algorithm can be applied.

Definition 3.3. Let c be a positive constant. We define the Basic Algorithm C to be the Basic Algorithm given above except that in Step 1 of the Basic Algorithm we return failure whenever $t > c \log n$.

Corollary 3.4. *Basic Algorithm* C *can be performed in* $O(n^{3+c\log 2}\log^4 n)$ *operations, where* $\log^4 n$ *can be replaced by* $\log^2 n$ *if* F *has more than* $2n$ *elements.*

In the case of finite fields or number fields, we shall show in §6 that a random univariate polynomial of degree n has $t \leq e \log n$. Hence, we can take $c \geq e$ in Basic Algorithm C.

4. Bivariate factoring over large finite fields

In this section, we give a probabilistic algorithm which completely factors an arbitrary bivariate polynomial of degree n over a finite field.

For the convenience of time analysis, we assume that $F = F_q$ is a large finite field compared to the degree of f. More precisely, we suppose $q > n^2$. If F_q is a small finite field, then we can first extend F_q to get a larger field, and apply the algorithm to the bigger field, as von zur Gathen and Kaltofen did in [9]. We note that the preprocessing stage (Step 1–Step 5) of the algorithm given here also works for many other fields; in particular, it works over algebraic number fields.

Let $f(x, y)$ be any bivariate polynomial over F_q. The following algorithm converts f into a nice normalized polynomial and gives a complete factorization of f over F_q. Note that in Step 5, we have a constant $c \geq e$ to be determined. How to choose the value of c depends on the requirement for failure probability.

Bivariate Factoring C.

Input: A polynomial $f(x, y) \in F_q[x, y]$ of total degree n.

Output: A complete factorization of f over F_q or failure.

Step 1. Check squarefreeness. Compute $f_x = \partial f/\partial x$ and $f_y = \partial f/\partial y$. If $f_x = f_y = 0$, then $f = \sum_{i,j\leq n} f_{ij} x^{ip} y^{jp}$, where p is the characteristic of F_q. Let $g = \sum_{i,j\leq n} f_{ij}^{q/p} x^i y^j$; then g is a factor of f and $f = g^p$. In this case, replace f by g and factor g.

If $f_x \neq 0$, or $f_y \neq 0$, compute the greatest common divisor $d = (f, f_x)$ or (f, f_y), respectively. Replace f by $\frac{f}{d}$ and factor $\frac{f}{d}$ ($\frac{f}{d}$ is squarefree). Thus, in the following steps we assume f is squarefree.

Step 2. Normalize f with respect to y. Let $f = f(x, y) = f_n + \cdots + f_0$. Choose an element b_0 in F satisfying $f_n(b_0 y, y) = y^n f_n(b_0, 1) \neq 0$ (this can be done if $q > n$). Then

$$f_*(x, y) = \frac{f(x + b_0 y, y)}{f_n(b_0, 1)}$$

is a normalized polynomial with respect to y.

Step 3. Transform to a nice polynomial. Compute the g.c.d. of the resultants

$$R_y(x) = \left(\mathrm{Res}_y\left(f_*, \frac{\partial f_*}{\partial x}\right), \mathrm{Res}_y\left(f_*, \frac{\partial f_*}{\partial y}\right)\right).$$

This is a univariate polynomial in x of degree $\leq n(n-1) < n^2$. Since f_* is squarefree, $R_y(x) \neq 0$. Choose a nonzero element a in F satisfying $R_y(a) \neq 0$ (this can be done if $q > n^2$). Using the same symbol f_* for the homogenized polynomial $f_*(x, y, z)$ as for $f_*(x, y)$, then

$$f^*(x, y, z) = f_*\left(x, y, \frac{x - z}{a}\right)$$

is a nice polynomial with respect to z.

Step 4. Normalize f^* with respect to x. Let $f^* = f^*(x, y, 1) = f_n^* + \cdots + f_0^*$. Choose an element b in F satisfying $f_n^*(x, bx) = x^n f_n^*(1, b) \neq 0$. Then

$$f_*^*(x, y) = \frac{f^*(x, bx + y)}{f_n^*(1, b)}$$

is a normalized polynomial with respect to x.

Step 5. Call procedure Basic Algorithm C with input $f_*^* \in F[x, y]$, to return a complete factorization of f_*^* with factors $P_i(x, y) \in F[x, y]$ $(i = 1, \ldots, l)$, where l is the number of factors of f_*^*.

Step 6. Set $n_i = \deg(P_i)$; return the following complete factorization of $f(x, y) \in F[x, y]$:

$$f(x, y) = \frac{f(a + ab_0 b, ab)}{a^n} \prod_{i=1}^{l} (x - by - a)^{n_i} P_i\left(\frac{x - by}{x - by - a}, \frac{y - bx + bb_0 y}{x - by - a}\right).$$

Proof. The correctness of Steps 1, 2, and 4 is obvious. For Step 3, suppose $(x, y, 0)$ is a singular point of f^* at infinity. Since f_* is normalized with respect to y, $(0, 1, 0)$ is not a point of the curve $f_* = 0$. We must have $x \neq 0$. This implies $(a, ay/x, 1)$ would be a singular point of f_*. This forces $R_y(a) = 0$, contradicting the choice of a. Step 6 is simply the inverse transformation. □

For a concrete estimate of the running time, as in [9] we have to implement Step 1 of the procedure Basic Algorithm C. The probabilistic version of Berlekamp's univariate algorithm, due to Cantor and Zassenhaus [5], factors a polynomial of degree n in

$$O(n^3 + n^2 \log n \log q)$$

operations in F_q. This algorithm can be written as a Las Vegas procedure, so that it either returns an irreducible factor or failure—the latter with probability at most $1/2$. The cost of the Las Vegas univariate factoring procedure in Step 1 of the Basic Algorithm is dominated by the cost of other steps. So we can apply that procedure several times, say n times, to obtain failure probability at most 2^{-n}.

Theorem 4.1. *Let $q > n^2$ and $c \geq e$, and let $f(x, y) \in F_q[x, y]$ be a polynomial of total degree n over F_q. Then the algorithm Bivariate Factoring C with input f can be performed in $O(n^{3+c\log 2}\log^2 n \log q)$ operations in F_q, with failure occurring with probability at most $n^{c(1-\log c)}/\sqrt{2\pi \log n} + 1/2q$.*

Proof. In Step 1, the pth root and the g.c.d. can be computed respectively in $O(n \log q/p)$ and $O(n^2 \log^2 n)$ operations in F_q. (See the proof of Theorem 3.2 in [9].) Lemma 3.2 shows that Step 2, Step 4, and Step 6 all take $O(n^2 \log^2 n)$ operations in F_q. In Step 3, the resultant and the g.c.d. algorithms can be performed in $O(n^3)$ operations. (See the proof of Theorem 3.2 in [9].) The cost of the whole algorithm is dominated by the running time for Step 5, which is $O(n^{3+c\log 2}\log^2 n + n(n^3 + n^2 \log n \log q))$. By Theorem 6.4 in §6, the failure probability of the algorithm is bounded by $n^{c(1-\log c)}/2\pi\sqrt{\log n} + 1/2q$. □

Taking $c = e$ in Theorem 4.1, we deduce

Corollary 4.2. *Let $q > n^2$. Any bivariate polynomial over F_q of total degree n can be completely factored in $O(n^{4.89}\log^2 n \log q)$ operations, with failure probability at most $1/\sqrt{\pi \log n}$.*

5. An improved probabilistic variant

The algorithm given in §4 is fast if we choose c to be small. Compared to the algorithm in [9], however, it has the disadvantage that the failure probability is not very small. There are two ways to improve this. The first is to increase the value of c. This will increase the running time significantly, but the improvement in the failure probability is not very great. In the following, we give another way which seems to be efficient.

An Improved Probabilistic Algorithm.

Step 1. Check squarefreeness as in Step 1 of the Bivariate Factoring C.

Step 2. Use the preprocessing stage (Steps 2–4 of the Bivariate Factoring C) to convert f into a normalized nice polynomial. Use the Basic Algorithm to extract all small factors of f, i.e., in Step 3 of the Basic Algorithm we only consider those elements d of the form $p_{i_1}^{e_{i_1}}(x, y)\cdots p_{i_k}^{e_{i_k}}(x, y)$ with $k \leq c$.

Step 3. Check whether or not $t(f_n) \leq c \log n$. If yes, apply the Basic Algorithm to factor f completely. Otherwise, go back to Step 2 and repreprocess f in different ways (that is, randomly choose b_0, a, and b satisfying the nec-

essary requirements in Steps 2–4 of the Bivariate Factoring C). One can repeat this process for up to $O(n)$ times.

Remarks. (i) In the above algorithm, one can take $c = 3, 4, 5, 6$, or even 7, because $n^{7\log 2} < n^{4.9}$.

(ii) We give a heuristic argument that the above algorithm should be efficient. First, if a squarefree bivariate $f(x, y)$ has many factors, then it is likely we can find some of the factors in Step 2; this would decrease the degree of f and the number of factors. If, on the other hand, $f(x, y)$ has only a few factors, then by repreprocessing f in different ways several times as in Step 3, one can expect that t (the number of distinct factors of f_n) is small. In fact, this could be justified by using the effective Hilbert Irreducibility Theorem developed in von zur Gathen [7] if q is large.

6. Distribution of polynomials over finite fields

As we have just seen, we need to study the probability that a polynomial f_n has small t. This question is clearly equivalent to the distribution problem of irreducible univariate polynomials.

We consider the case when $F = F_q$ is a finite field. For example, in the case of a number field K, we can take a prime ideal P with sufficiently large norm q, in which case the residue class field is a large finite field F_q. It is well known that if a given polynomial f over K has a certain factorization pattern over K, then there exists a large residue class field over which its reduction has the same factorization pattern. Therefore, the probability $P(\deg(f) = n | t(f) \leq c \log n)$ over a large finite field F_q measures in some sense the corresponding probability over the global field K. In the following, we assume that $F = F_q$ is the field of q elements.

Let $N_r(n)$ be the number of monic univariate polynomials of degree n over F_q with no repeated factors and with exactly r distinct prime factors. We study the distribution of $N_r(n)$ as r varies $(1 \leq r \leq n)$. We want to show that a typical polynomial of degree n has no repeated factors and has no more than $e \log n$ factors.

It is a classical result that

$$N_1(n) = \frac{1}{n} \sum_{d|n} \mu\left(\frac{n}{d}\right) q^d \leq \frac{1}{n} q^n . \tag{12}$$

Let $M_n(q)$ be the number of monic univariate polynomials of degree n over F_q with at least one repeated prime factor. Then, by definition, we have

$$M_n(q) \leq \sum_{2i+j=n,\, i>0} q^i q^j \leq 2q^{n-1} . \tag{13}$$

Let $1 \le r \le n$. From equation (12), we have

$$\begin{aligned} N_r(n) &\le \frac{1}{r!} \sum_{i_1+\cdots+i_r=n,\, i_j \ge 1} N_1(i_1)\cdots N_1(i_r) \\ &\le \frac{1}{r!} \sum_{i_1+\cdots+i_r=n,\, i_j \ge 1} \frac{1}{i_1 \cdots i_r} q^n . \end{aligned} \tag{14}$$

Let

$$b_r(n) = \sum_{i_1+\cdots+i_r=n,\, i_j \ge 1} \frac{1}{i_1 \cdots i_r}$$

and $a_r(n) = b_r(n)/r!$. We have proved the following

Lemma 6.1. *Let* $1 \le r \le n$; *then* $N_r(n) \le a_r(n) q^n$.

Lemma 6.2. *For* $2 \le r \le n$, *we have*

$$a_r(n) \le \frac{1}{n} \left(\frac{\log^{r-1} n}{(r-1)!} + \frac{\log^{r-2} n}{(r-2)!} \right) .$$

Proof. If we define $b_0(0) = 1$, we have the following generating function for $b_r(n)$:

$$f_r(x) = \sum_{n=r}^{\infty} b_r(n) x^n = \left(x + \cdots + \frac{x^n}{n} + \cdots \right)^r = (-\log(1-x))^r .$$

It is easy to see that

$$f_r'(x) = r f_{r-1}(x)(1-x)^{-1} .$$

Hence, we have the following recursive relations:

$$n b_r(n) = r \sum_{k=r-1}^{n-1} b_{r-1}(k)$$

and

$$a_r(n) = \frac{1}{n} \sum_{k=r-1}^{n-1} a_{r-1}(k) . \tag{15}$$

By definition of $a_r(n)$, we have $a_1(n) = 1/n$. Now, (15) gives

$$a_2(n) = \frac{1}{n} \sum_{k=1}^{n-1} \frac{1}{k} \le \frac{1}{n} (\log n + 1) ,$$

$$a_3(n) \le \frac{1}{n} \sum_{k=2}^{n-1} \frac{1}{k} (\log k + 1) \le \frac{1}{n} \left(\frac{\log^2 n}{2!} + \log n \right) .$$

In general, suppose Lemma 6.2 is true for some $r-1$ (≥ 3). From (15), and by induction, we deduce that

$$\begin{aligned} a_r(n) &\leq \frac{1}{n}\sum_{k=r-1}^{n-1}\left(\frac{\log^{r-2}k}{k(r-2)!}+\frac{\log^{r-3}k}{k(r-3)!}\right) \\ &\leq \frac{1}{n}\int_{r-2}^{n}\left(\frac{\log^{r-2}x}{x(r-2)!}+\frac{\log^{r-3}x}{x(r-3)!}\right)dx \\ &\leq \frac{1}{n}\left(\frac{\log^{r-1}n}{(r-1)!}+\frac{\log^{r-2}n}{(r-2)!}\right). \end{aligned}$$

The second inequality is true because we extended the limits of integration, and the functions in the integrand have only one local minimum. The proof is complete. □

Lemma 6.3. *Let* $c_r(n) = \frac{1}{n}\log^{r-1} n/(r-1)!$; *then for* $n > 1$ *and* $c \geq e$ *we have*

$$\sum_{c\log n<r\leq n} c_r(n) \leq \frac{1}{\sqrt{2\pi\log n}}n^{c(1-\log c)}.$$

Proof. In the inequality

$$\left|e^x - \sum_{r=0}^{m}\frac{x^r}{r!}\right| \leq \frac{e^x x^{m+1}}{(m+1)!} \qquad (x \geq 0)$$

we put $x = \log n$ and $m = [c\log n]$; then

$$\sum_{c\log n<r\leq n} c_r(n) \leq \left|1 - \frac{1}{n}\sum_{r=0}^{m}\frac{\log^r n}{r!}\right| \leq \frac{(\log n)^{m+1}}{(m+1)!}.$$

Now, Stirling's approximation formula for factorials implies that the rightmost term is

$$\leq \sqrt{\frac{e}{2\pi(m+1)}}\left(\frac{e\log n}{m+1}\right)^{m+1} \leq \frac{1}{\sqrt{2\pi\log n}}\left(\frac{e}{c}\right)^{c\log n} = \frac{1}{\sqrt{2\pi\log n}}n^{c(1-\log c)}.$$

So Lemma 6.3 is correct. □

Lemmas 6.1–6.3 and (13) together imply the following

Theorem 6.4. *For any* $n \geq 2$ *and* $c \geq e$, *one has*

$$P(f_n \in F_q[x],\ \deg(f_n) = n | t(f_n) \leq c\log n) \geq 1 - \frac{1}{\sqrt{2\pi\log n}}n^{c(1-\log c)} - \frac{1}{2q}.$$

The theorem shows that the number of factors of a typical univariate polynomial of degree n is less than $e\log n$.

7. Two applications

The basic idea involved in our algorithm has theoretical interest. It can be used to prove certain polynomials are absolutely irreducible (irreducible over

an algebraic closure of the ground field). Here we shall give only two examples. One is concerned with exponential sums; the other is related to permutational polynomials (the Chowla-Zassenhaus conjecture). We first discuss exponential sums.

Let $f(x_1, \ldots, x_n)$ $(n \geq 2)$ be a polynomial of degree d with integral coefficients. Let p be a prime number and F_{p^m} be the finite field of p^m elements. Define the exponential sum

$$S(p^m, f) = \sum_{x_1, \ldots, x_n \in F_{p^m}} \exp\left(\frac{Tr_{F_{p^m}/F_p}(f(x_1, \ldots, x_n))}{p}\right),$$

where $\exp(x) = e^{2\pi i x}$. It is a classical problem of number theory to give a good estimate for $|S(p^m, f)|$. For a nice account of this subject, see [10 and 18]. A deep theorem of Katz [10, Theorem 2.3.1, p. 41] shows that if $f(x_1, \ldots, x_n) - T$ is irreducible over an algebraic closure of the rational function field $Q(T)$, then one has

$$|S(p^m, f)| \ll p^{m(n-1)} \tag{16}$$

for all sufficiently large primes p (depending on f) and all m, where the implied constant depends only on $\deg(f)$. This result is best possible under the given conditions.

We shall give a large class of such polynomials.

Theorem 7.1. *Let F be any field and f be any polynomial over F of degree $d > 0$ with n $(n > 1)$ variables. We write f in the form*

$$f = f_d + f_{d-1} + \cdots + f_0, \tag{17}$$

where f_i $(i = 0, \ldots, d)$ is the homogeneous part of degree i in f. Suppose that g.c.d(f_d, f_{d-1}^2) *is squarefree (for example, this is true when f_d is squarefree). Then, among the polynomials $f_a = f(x_1, \ldots, x_n) - a$ $(a \in \overline{F})$, at most 2^d of them are reducible in the algebraic closure $\overline{F}$.*

Remark. In the case of algebraic number fields or finite fields with large characteristic, we can improve the bound 2^d to $O(d^2)$. However, the proof is not simple.

Proof of Theorem 7.1. Similar to the discussion of §2. Let $X = (x_1, \ldots, x_n)$. If $f_a(X)$ is reducible in $\overline{F}$, then we can write

$$\begin{aligned} f_a(X) &= g(X)h(X), \\ g(X) &= g_r(X) + \cdots + g_0(X), \\ h(X) &= h_s(X) + \cdots + h_0(X), \\ f_d(X) &= g_r(X)h_s(X), \end{aligned} \tag{18}$$

where $0 < \deg(g) = r < d$.

The discussion of §2 shows that, if g.c.d(f_d, f_{d-1}^2) is squarefree, then $g(X)$ and $h(X)$ (hence also a) are uniquely determined by $g_r(X)$, $h_s(X)$, and $f_d(X) + \cdots + f_1(X)$. Now, by (18), we know there are at most 2^d such pairs $g_r(X)$, $h_s(X)$, so there are at most 2^d values $a \in \overline{F}$ such that $f_a(X)$ is reducible in $\overline{F}$. □

Corollary 7.2. *Under the assumptions of Theorem* 7.1, *we have that* $f(X) - T$ *is irreducible over any algebraic closure of* $F(T)$, *hence irreducible over* $\overline{F}$.

Proof. Suppose $f(X) - T$ is reducible in some algebraic closure of $F(T)$; then for any value $a \in \overline{F}$, $f(x) - a$ is reducible. This contradicts Theorem 7.1. □

Corollary 7.3. *Let* $f(X)$ *be an integral polynomial given in the form* (17). *If* g.c.d(f_d, f_{d-1}^2) *is squarefree, then* (16) *holds.*

This corollary follows from Katz's theorem and Corollary 7.2.

We now turn to permutational polynomials over finite fields. Let F_p be the finite field of p elements. A polynomial $f(x) \in F_p(x)$ is called a permutational polynomial over F_p if the values $f(a)$ $(a \in F_p)$ are distinct. For the general theory of permutational polynomials, see [15]. Chowla and Zassenhaus [6] conjectured that if $f(x)$ is an integral polynomial of degree ≥ 2 and p is a sufficiently large prime for which $f(x)$ is a permutational polynomial over F_p, then for no $a \in F_p^*$ is $f(x) + ax$ a permutational polynomial over F_p. Mullen and Niederreiter [16] have recently shown that the Chowla-Zassenhaus conjecture is true for a special class of polynomials (the so-called Dickson polynomials). Using Theorem 7.1, we have

Corollary 7.4. *Let* $f(x)$ *be an integral polynomial of degree* $d > 1$ *and* p *be a sufficiently large prime. Then* $f(x) - ax$ *can be a permutational polynomial over* F_p *for at most* 2^d *values of* $a \in F_p$.

Proof. By Theorem 7.1, we know that

$$F_a(x, y) = \frac{(f(x) + ax) - (f(y) + ay)}{x - y} = \frac{f(x) - f(y)}{x - y} + a$$

is absolutely irreducible over F_p except for at most 2^d values a. If $F_a(x, y)$ is absolutely irreducible over F_p, then for large p the Riemann Hypothesis for curves over finite fields shows that $F_a(x, y) = 0$ has a solution with $x \neq y$ in F_p, i.e., $f(x) + ax$ is not a permutational polynomial over F_p. This proves the corollary. □

As we remarked before, the bound 2^d in Corollary 7.4 can be improved to d^2. But the proof would not be elementary.

ACKNOWLEDGMENT

I would like to thank Professors Neal Koblitz, Andrew Odlyzko, and Susan Landau for their comments on the original manuscript, especially Neal Koblitz for his encouragement and many valuable corrections.

BIBLIOGRAPHY

1. M. Ben-Or, *Probabilistic algorithms in finite fields*, Proc. 22nd Sympos. Foundations Comp. Sci., IEEE, New York, 1981, pp. 394–398.
2. E. R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Tech. J. **46** (1967), 1853–1859.
3. ———, *Factoring polynomials over large finite fields*, Math. Comp. **24** (1970), 731–735.
4. P. F. Camion, *Improving an algorithm for factoring polynomials over a finite field and constructing large irreducible polynomials*, IEEE Trans. Inform. Theory **29** (1978), 378–385.
5. D. G. Cantor and H. Zassenhaus, *On algorithms for factoring polynomials over finite fields*, Math. Comp. **36** (1981), 587–592.
6. S. Chowla and H. Zassenhaus, *Some conjectures concerning finite fields*, Norske Vid. Selsk. Forh. (Trondheim) **41** (1968), 34–35.
7. J. von zur Gathen, *Irreducibility of multivariate polynomials*, J. Comput. System Sci. **31** (1985), 225–264.
8. J. von zur Gathen and E. Kaltofen, *Factoring sparse multivariate polynomials*, J. Comput. System Sci. **31** (1985), 265–287.
9. ———, *Factorization of multivariate polynomials over finite fields*, Math. Comp. **45** (1985), 251–261.
10. N. M. Katz, *Sommes exponentielles*, Astérisque **79** (1980).
11. D. Knuth, *The art of computer programming*, Vol. 2, Seminumerical Algorithms, Addison-Wesley, Reading, Mass., 1981.
12. S. Landau, *Factoring polynomials quickly*, Notices Amer. Math. Soc. **34** (1987), 3–8.
13. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), 515–534.
14. A. K. Lenstra, *Factoring multivariate polynomials over finite fields*, J. Comput. System Sci. **30** (1985), 235–248.
15. R. Lidl and H. Niederreiter, *Finite fields*, Encyclopedia of Math. and Its Appl., Vol. 10, Addison-Wesley, Reading, Mass., 1983.
16. G. L. Mullen and H. Niederreiter, *Dickson polynomials over finite fields and complete mappings*, Canad. Math. Bull. **30** (1987), 19–27.
17. M. O. Rabin, *Probabilistic algorithms in finite fields*, SIAM J. Comput. **9** (1980), 273–280.
18. J. P. Serre, *Majoration de sommes exponentielles*, J. Arithmétiques de Caen, Astérique **41–42** (1977), 111–126.
19. H. Zassenhaus, *On Hensel factorization.* I, J. Number Theory **1** (1969), 291–331.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF WASHINGTON, SEATTLE, WASHINGTON 98195