HENSEL LIFTING AND BIVARIATE POLYNOMIAL FACTORISATION OVER FINITE FIELDS

SHUHONG GAO AND ALAN G.B. LAUDER

Abstract. This paper presents an average time analysis of a Hensel lifting based factorisation algorithm for bivariate polynomials over finite fields. It is shown that the average running time is almost linear in the input size. This explains why the Hensel lifting technique is fast in practice for most polynomials.

1. Introduction

It is well known that the Hensel lifting technique provides practical methods for factoring polynomials over various fields. Such methods are known to run in exponential time in the worst case, but seem fast for most polynomials. The latter phenomenon has not been fully understood and calls for an average running time analysis. The only analysis we know of is that of Collins 1979 [4] for univariate integral polynomials (factoring over the rational numbers). He shows, under some reasonable number theoretic conjectures, that the average running time is indeed polynomial. In this paper, we present a rigorous analysis for bivariate polynomials over finite fields. We show that the average running time is almost linear in the input size. More precisely, for all bivariate polynomials of total degree n over a fixed finite field, the average running time is $\mathcal{O}(N)$ using fast polynomial arithmetic, and $\mathcal{O}(N^{1.5})$ using standard polynomial arithmetic where $N = n^2$ represents the input size and we ignore the logarithmic factors in our running times. This explains why the Hensel lifting technique is fast in practice for bivariate polynomials over finite fields.

Date: May 30, 2000.

The first author was supported in part by NSF grant #DMS9970637, NSA grant #MDA904-00-1-0048 and ONR grant #N00014-00-1-0565. The second author gratefully acknowledges the support of the Marr Educational Trust and Wolfson College, Oxford.

2000 Mathematics Subject Classification: Primary 11Y16; Secondary 11T06, 11Y05, 68Q25.

Key words and phrases: bivariate polynomial, finite field, Hensel lifting, factorisation, average-case complexity.

Our paper is organised in the following way. In Section 2 we discuss different ways of ordering bivariate polynomials, and the probabilities that polynomials chosen uniformly at random with respect to these orderings are irreducible or absolutely irreducible. These put our estimations on average running times in perspective, and are also of independent interest. Section 3 contains a discussion of the basic ideas behind Hensel lifting, and then in Section 4 we present our Hensel lifting based algorithm, which is in essence the standard one. Section 5 contains an analysis of the algorithm's expected running time; this is the main result of the paper. Finally in Section 6 we present a randomised version of the algorithm.

2. Distribution of reducible polynomials

In this section we discuss different natural ways of "ordering" bivariate polynomials, and the distribution of irreducible and absolutely irreducible polynomials under these "orderings'.

For an integer $n \geq 1$, let $T(n,q)$ denote the set of all polynomials in $\mathbb{F}_q[x,y]$ of total degree n that are monic in x and have degree n in x. Let $t(n,q) = |T(n,q)|$.

Proposition 2.1. *Let $r(n,q)$ be the number of reducible polynomials in $T(n,q)$. Then, for $n \geq 6$,*

$$\frac{3}{4} \cdot \frac{1}{q^{n-1}} \leq \frac{r(n,q)}{t(n,q)} \leq \frac{4}{3} \cdot \frac{1}{q^{n-1}}.$$

Proof. Observe that $t(n,q) = q^{n(n+3)/2}$ as $n(n+3)/2$ is the number of coefficients for a polynomial in $T(n,q)$. If a polynomial in $T(n,q)$ is reducible then one of its factors must have total degree i between 1 and $n/2$. Hence $r(n,q)$ is at most

$$\sum_{1 \leq i \leq n/2} t(i,q)t(n-i,q) = \sum_{1 \leq i \leq n/2} q^{\frac{i(i+3)}{2} + \frac{(n-i)(n-i+3)}{2}}.$$

It follows that

$$\frac{r(n,q)}{t(n,q)} \leq \sum_{1 \leq i \leq n/2} \frac{1}{q^{i(n-i)}}.$$

Since $i(n-i)$ is a convex function of i (concave down), we have

$$i(n-i) \geq n - 1 + \frac{n-2}{2}(i-1), \quad 1 \leq i \leq n/2.$$

(The linear function on the right agrees with the quadratic on the left when $i = 1$ and $i = n/2$.) Hence

$$\frac{r(n,q)}{t(n,q)} \le \sum_{1\le i\le n/2} \frac{1}{q^{n-1+\frac{n-2}{2}(i-1)}} \le \frac{1}{q^{n-1}} \cdot \frac{1}{1-1/q^{(n-2)/2}},$$

which is at most $\frac{1}{q^{n-1}} \cdot \frac{4}{3}$ for $q \ge 2$ and $n \ge 6$.

A trivial lower bound for $r(n,q)$ is the number of polynomials in $T(n,q)$ that are products of a linear polynomial in $T(1,q)$ and a polynomial in $T(n-1,q)$ with no linear factors. Hence

$$r(n,q) \ge t(1,q)\big(t(n-1,q) - t(1,q)t(n-2,q)\big)$$

and

$$\frac{r(n,q)}{t(n,q)} \ge \frac{q^2\left(q^{\frac{(n-1)(n+2)}{2}} - q^2 q^{\frac{(n-2)(n+1)}{2}}\right)}{q^{\frac{n(n+3)}{2}}} = \frac{1}{q^{n-1}}\left(1 - \frac{1}{q^{n-2}}\right),$$

which is at least $\frac{1}{q^{n-1}} \cdot \frac{3}{4}$ for $n \ge 4$ and $q \ge 2$. □

The next proposition will not be needed for our analysis but seems interesting by itself.

Proposition 2.2. *Let $r_0(n,q)$ be the number of polynomials in $T(n,q)$ that are not squarefree. Then for $n \ge 5$,*

$$\frac{3}{4} \cdot \frac{1}{q^{2n-1}} \le \frac{r_0(n,q)}{t(n,q)} \le \frac{4}{3} \cdot \frac{1}{q^{2n-1}}.$$

Proof. The proof is similar to that of the previous proposition. We have

$$\frac{r_0(n,q)}{t(n,q)} \le \frac{\sum_{1\le i\le n/2} t(i,n)^2 t(n-2i,q)}{t(n,q)} = \sum_{1\le i\le n/2} \frac{1}{q^{i(4n-5i+3)/2}}.$$

Note that $i(4n-5i+3)$ is a convex function of i, we have

$$i(4n-5i+3) \ge 4n-2+(i-1)(3n-4)/2, \quad 1 \le i \le n/2,$$

where equality holds for $i = 1$ and $i = n/2$. Hence

$$\frac{r_0(n,q)}{t(n,q)} \le \sum_{1\le i\le n/2} \frac{1}{q^{2n-1+(i-1)(3n-4)/4}} \le \frac{1}{q^{2n-1}} \cdot \frac{1}{1-1/q^{(3n-4)/4}},$$

which is at most $\frac{1}{q^{2n-1}} \cdot \frac{4}{3}$ for $n \ge 4$ and $q \ge 2$.

For the lower bound,

$$\begin{aligned}\frac{r_0(n,q)}{t(n,q)} &\geq \frac{t(1,q)^2\big(t(n-2,q)-t(1,q)^2t(n-4,q)\big)}{t(n,q)}\\ &\geq \frac{1}{q^{2n-1}}\cdot\left(1-\frac{1}{q^{2n-7}}\right),\end{aligned}$$

which is at least $\frac{1}{q^{2n-1}}\cdot\frac{4}{3}$ for $n\geq 5$ and $q\geq 2$. □

Remark. The above arguments actually prove more: for $n\geq 4$,

$$\begin{aligned}\frac{1}{q^{n-1}}\cdot\left(1-\frac{1}{q^{n-2}}\right)\leq \ \tfrac{r(n,q)}{t(n,q)} \ \leq \frac{1}{q^{n-1}}\cdot\frac{1}{1-1/q^{(n-2)/2}},\\ \frac{1}{q^{2n-1}}\cdot\left(1-\frac{1}{q^{2n-7}}\right)\leq \ \tfrac{r_0(n,q)}{t(n,q)} \ \leq \frac{1}{q^{2n-1}}\cdot\frac{1}{1-1/q^{(3n-4)/4}}.\end{aligned}$$

So $\frac{r(n,q)}{t(n,q)}$ and $\frac{r_0(n,q)}{t(n,q)}$ are asymptotically $1/q^{n-1}$ and $1/q^{2n-1}$, respectively, when q or n is large.

The upper bound in Proposition 2.1 means that most polynomials in $T(n,q)$ are irreducible. Thus a polynomial picked uniformly at random from the set $T(n,q)$ is unlikely to have any proper factorisations over the defining field $\mathbb{F}_q$. Any good general algorithm for factoring bivariate polynomials must perform well on most irreducible polynomials, that is, it must detect most irreducible polynomials as soon as possible. Our analysis below indicates that Hensel lifting based algorithms do seem to have this property so perform well on average, even though very badly on some polynomials.

The lower bound in Proposition 2.1 means that there are still a significant fraction of polynomials in $T(n,q)$ that are reducible. This shows that our model of polynomials is not "trivial". Certainly, our model is not trivial also because any polynomial of total degree n can be transformed into a polynomial in $T(n,q)$ that has the same factorisation pattern (provided $q>n$). This can be seen as follows. Let $h(y)=\sum_{i=0}^{n}c_iy^i$ where $\sum_{i=0}^{n}c_ix^{n-i}y^i$ is the homogeneous part of f of degree n. Then $g=f(x,y+\alpha x)$ still has total degree n and the coefficient of x^n is $h(\alpha)$. Since h is nonzero and has degree at most n, we only need to pick $\alpha\in\mathbb{F}_q$ such that $h(\alpha)\neq 0$; this is always possible provided $q>n$. If q is too small, one needs to go an extension of $\mathbb{F}_q$ to have enough elements. When $h(\alpha)\neq 0$, g can be made monic in x so can be viewed as belonging to $T(n,q)$. Certainly, the factors of f can be easily obtained from those of g by the inverse transformation.

To see what we mean by "trivial", we give below a model of polynomials that has a simple description similar to $T(n,q)$, yet we consider

it "trivial" for factoring purpose. Note that a polynomial f in $T(n,q)$ can be written as

$$f = f_n(y) + f_{n-1}(y)x + \cdots + f_1(y)x^{n-1} + x^n \in \mathbb{F}_q[x,y] \tag{1}$$

with

$$\deg f_i(y) \le i, \quad 1 \le i \le n. \tag{2}$$

Let us slightly modify this degree condition as follows

$$\deg f_i(y) \le n, \quad 2 \le i \le n, \quad \deg f_1(y) = n. \tag{3}$$

Let $\bar{T}(n,q)$ be the set of all polynomials f in (1) satisfying (3).

Proposition 2.3. *For any $f \in \bar{T}(n,q)$ as in (1), rewrite f as $f = a_0(x) + a_1(x)y + \cdots + a_n(x)y^n$ and let*

$$h = \gcd(a_0(x), a_1(x), \ldots, a_n(x)) \in \mathbb{F}_q[x].$$

Then f/h is an absolutely irreducible factor of f.

Thus factoring a bivariate polynomial $f \in \bar{T}(n,q)$ is easily reduced to factoring a univariate polynomial h which is 1 almost all the time! So polynomials in $\bar{T}(n,q)$ are indeed quite trivial to factor.

To prove Proposition 2.3, one just considers the Newton polytope of f, a polygon in the Euclidean plane formed by the convex hull of the exponent vectors (i,j) of all nonzero terms x^iy^j in f. The degree condition in (3) implies that the polygon has a long indecomposable edge determined by the terms x^n and $x^{n-1}y^n$ and so one of the summands in any Minkowski decomposition of the polygon must be a horizontal line segment, which corresponds to a factor of f that only involves x (no y). Then the proposition follows easily. For more details on this argument and on Newton polytopes and factorisation of polynomials, the reader is referred to the recent papers [6, 7].

3. Motivation

This section contains a discussion of the motivation behind the algorithm we present following in part the exposition in [13]. In particular, our discussion will justify the correctness of the algorithm and elucidate some of its subtler features which are of importance in the analysis of the average running time. However, the reader familiar with Hensel lifting based factorisation algorithms may safely move directly onto Section 4 and refer back when required.

Let $f \in T(n,q)$ with $f = gh$ where $g, h \in \mathbb{F}_q[x][[y]]$ are non-constant power series. We call $f = gh$ a (proper) *analytic factorisation* of f at the prime ideal generated by y. If both of g and h lie in the subring $\mathbb{F}_q[x,y]$ then we further refer to $f = gh$ as a *polynomial factorisation*

of f. All analytic factorisations of f may in principle be found using Newton polygons and a form of Hensel lifting with respect to the prime ideal (y).

Suppose that $f = gh$ for some power series $g, h \in \mathbb{F}_q[x][[y]]$. We shall first of all examine how the coefficients in the y-adic expansions of f,g and h are related. So let $f = \sum_{k=0}^{n} f_k y^k$ denote the finite y-adic expansion of f, and $g = \sum_{k\geq 0} g_k y^k$ and $h = \sum_{k\geq 0} h_k y^k$ denote the, possibly infinite, expansions of g and h. Here $f_k, g_k, h_k \in \mathbb{F}_q[x]$. Since $f \equiv gh \bmod y$ we have that $f_0 = g_0 h_0$. Equating the coefficients of y^k for $k \geq 1$ on both sides of $f = gh$ we see that

$$\begin{array}{rcl} f_1 & = & g_0 h_1 + g_1 h_0 \\ f_2 & = & g_0 h_2 + g_1 h_1 + g_2 h_0 \\ \vdots & \vdots & \vdots \\ f_k & = & \sum_{i=0}^{k} g_i h_{k-i} \\ \vdots & \vdots & \vdots \end{array}$$

Thus for $k \geq 1$ we have

$$g_0 h_k + g_k h_0 = f_k - \sum_{i=1}^{k-1} g_i h_{k-i} \tag{4}$$

Now let $d = \gcd(g_0, h_0)$ with u and v chosen so that $ug_0 + vh_0 = d$ and $\deg u < \deg h_0$, $\deg v < \deg g_0$. Then d divides the righthand side of Equation (4) and we see that g_k and h_k must be of the form

$$g_k = v\frac{f_k - \sum_{i=1}^{k-1} g_i h_{k-i}}{d} + w_k \frac{g_0}{d} \tag{5}$$

$$h_k = u\frac{f_k - \sum_{i=1}^{k-1} g_i h_{k-i}}{d} - w_k \frac{h_0}{d} \tag{6}$$

for some polynomial $w_k \in \mathbb{F}_q[x]$. Thus we have obtained equations which relate the coefficients f_k, g_k and h_k of the y-adic expansions of f,g and h respectively.

Consider now the situation in which we are given a polynomial $f = \sum_{k=0}^{n} f_k y^k$ and a factorisation $f_0 = g_0 h_0$ for some polynomials $g_0, h_0 \in \mathbb{F}_q[x]$. Is it possible to use Equations (5) and (6) to *define* a sequence of polynomials $\{g_k\}_{k\geq 0}$ and $\{h_k\}_{k\geq 0}$ such that $g = \sum_{k\geq 0} g_k y^k$ and $h = \sum_{k\geq 0} h_k y^k$ satisfy $f = gh \bmod y^{n+1}$? The answer is positive, provided that at each stage w_k is chosen so that d, the greatest common divisor of g_0 and h_0, divides the polynomial $f_k - \sum_{i=1}^{k-1} g_i h_{k-i}$. If $d = \gcd(g_0, h_0) \neq 1$ then the choice we make of w_k may not be unique, so resulting in exponentially many choices for g_k's and h_k's. If

$d = \gcd(g_0, h_0) = 1$, however, the equation (4) uniquely determines g_k and h_k when $\deg g_k < \deg h_0$ and $\deg h_k < \deg g_0$. This means that the "lifting" can be carried out uniquely as high as one wishes.

We are interested in polynomial factorisations rather than arbitrary analytic factorisations and so a few more observations can be made. Suppose we have been given a factorisation $f = gh$. Let us further assume that f,g and h all lie in $\mathbb{F}_q[x, y]$ and $n = \deg(f), r = \deg(g)$ and $s = \deg(h)$. Then we have $r + s = n$. Hence for $0 \leq k \leq n$ we have $\deg(g_k) \leq r - k$ and $\deg(h_k) \leq s - k$. Here we interpret this to mean g_k and h_k should be zero in the cases when $r - k$ and $s - k$ are less than zero, respectively.

Turning this observation around, suppose now we have been given a polynomial $f \in \mathbb{F}_q[x, y]$ and a factorisation $f_0 = g_0 h_0$ where $f = \sum_{k=0}^{n} f_k y^k$ and g_0 and h_0 are polynomials in $\mathbb{F}_q[x]$ with $\deg(g_0) = r$, $\deg(h_0) = s$. We wish now to lift this to a factorisation in $\mathbb{F}_q[x, y]$. When using Equations (5) and (6) to define the polynomials g_k and h_k we must choose w_k so that appropriate conditions on the degrees are met. In the case that $\deg(f_0) = n$ the restrictions are $\deg(g_k) \leq r - k$ and $\deg(h_k) \leq s - k$. When $\gcd(g_0, h_0) = 1$ there will be at most one way of doing this. One defines g_k and h_k by the equations

$$g_k = v\big(f_k - \sum_{i=1}^{k-1} g_i h_{k-i}\big) \bmod g_0 \tag{7}$$

$$h_k = u\big(f_k - \sum_{i=1}^{k-1} g_i h_{k-i}\big) \bmod h_0. \tag{8}$$

and then checks whether $\deg(g_k) \leq r-k$ and $\deg(h_k) \leq s-k$. (Observe that since $f_k = \sum_{i=0}^{k} g_i h_{k-i}$, if $\deg(g_i) \leq r - i$ for all $i \leq k$, and we further assume that $\deg(h_i) \leq s - i$ for all $i < k$ then $\deg(h_k) \leq s - k$. Thus we need only check the degrees of the polynomials g_k.)

It is these recursion equations we use in Algorithm 4.1 which is presented in the next section. The check which must be made on the degree of g_k at each step is crucial to our analysis of the running time.

For more information on Hensel lifting based algorithms for factoring polynomials, see the textbook [9], particularly [16] for univariate polynomials over rationals and [11, 14, 15] for multivariate polynomials.

4. The Algorithm

For $n \geq 1$, let $M(n, q) \subseteq T(n, q)$ denote the subset of all polynomials whose reduction modulo y is squarefree. The previous section shows that Hensel lifting works for all polynomials in $M(n, q)$. In this section

and the next one, we will analyse the average running time of this method. In Section 6, we show how to factor polynomials in $T(n, q)$.

Let us first state the algorithm explicitly as follows.

Algorithm 4.1. Hensel Factorisation
Input: A polynomial $f = \sum_{k=0}^{n} f_k y^k$ *in* $M(n, q)$, *where* $f_k \in \mathbb{F}_q[x]$.
Output: All monic factors of f *with total degree between 1 and* $\lfloor n/2 \rfloor$.

Step 1: Use a univariate polynomial factorisation algorithm to factor f_0, *a squarefree polynomial.*

If f_0 *is irreducible then halt the algorithm.*

Hence assume f_0 *is reducible. List all pairs* (g_0, h_0) *of monic factors with* $f_0 = g_0 h_0$ *and* $1 \leq \deg g_0 \leq \deg h_0$. *For each pair* (g_0, h_0), *do the following steps 2–4 where* $r = \deg g_0$ *so* $1 \leq r \leq \lfloor n/2 \rfloor$.

Step 2: Compute polynomials u *and* v *with* $ug_0 + vh_0 = 1$ *and* $\deg(u) < \deg(h_0)$, $\deg v < \deg(g_0)$.

Step 3: For k *from* 1 *to* $\lfloor n/2 \rfloor$, *compute*

$$g_k = v\{f_k - \sum_{i=1}^{k-1} g_i h_{k-i}\} \bmod g_0, \tag{9}$$

and

$$h_k = u\{f_k - \sum_{i=1}^{k-1} g_i h_{k-i}\} \bmod h_0. \tag{10}$$

In the case that $r \geq k$ *check whether* $\deg(g_k) \leq r - k$, *and in the case that* $k > r$ *check whether* $g_k = 0$. *If the appropriate one of these two conditions is not satisfied halt the computation for this pair.*

Step 4: Check whether $g := \sum_{k=0}^{r} g_k y^k$ *divides* f. *If so then output* g.

This is in essence the standard Hensel lifting technique for factoring polynomials, and a proof of its correctness follows easily from the discussion in Section 3. It is of interest to note that the check that the polynomial g_k has suitably bounded degree in Step 3, which is crucial for our estimate of the average running time, appears to originate in Wan [13]. Also, in Step 3 one needs only lift a maximum of r steps rather than $\lfloor n/2 \rfloor$ steps; however, we include these redundant extra lifting steps so that Algorithm 4.1 ties in precisely with the slightly modified version which we present shortly. Note that these extra steps do not adversely affect the average running time since they are performed so seldomly.

Our main concern is to determine the average running time. We have the following result.

Theorem 4.2. *For $f \in M(n,q)$, the average number of $\mathbb{F}_q$-field operations used by Algorithm 4.1 is $\tilde{\mathcal{O}}(n^{1+\alpha}+d(n,q))$. Here $d(n,q)$ denotes a bound on the worst-case number of $\mathbb{F}_q$-field operations required to factor univariate polynomials of degree n over $\mathbb{F}_q$. Also, $\alpha = 1$ or 2 according to whether we are using ordinary or fast polynomial multiplication and division, respectively.*

Here and hereafter we adopt the soft-$\mathcal{O}$ notation:

$$\tilde{\mathcal{O}}(p(n)) = \mathcal{O}(p(n)(\log n \log q)^{O(1)}),$$

meaning that we ignore the logarithmic factors.

To prove the above theorem it is convenient to present the algorithm in a slightly different manner. Let $(g_0^{(j)}, h_0^{(j)})$, $1 \le j \le t$, denote all the pairs which are computed in Step 1, and $u^{(j)}$ and $v^{(j)}$ the corresponding polynomials computed in Step 2. Let $r^{(j)} = \deg(g_0^{(j)})$ for $1 \le j \le t$. We shall now give an equivalent but alternative description of Step 3 in which all liftings are performed in parallel, as opposed to in series as in the above algorithm. This does not affect the average running time and aids analysis.

We shall replace Step 3 by

Step 3 ′: For each $1 \le k \le \lfloor n/2 \rfloor$ define a subset $C_k \subset \{1, 2, \ldots t\}$. Firstly let $C_1 = \{1, 2, \ldots t\}$. For $k \ge 1$, and for each $j \in C_k$ we compute

$$g_k^{(j)} = v^{(j)}\{f_k - \sum_{i=1}^{k-1} g_i^{(j)} h_{k-i}^{(j)}\} \bmod g_0, \tag{11}$$

$$h_k^{(j)} = u^{(j)}\{f_k - \sum_{i=1}^{k-1} g_i^{(j)} h_{k-i}^{(j)}\} \bmod h_0. \tag{12}$$

For $1 \le k \le \lfloor n/2 \rfloor - 1$ define

$$\begin{aligned} C_{k+1} &= \{j \in C_k \,|\, r^{(j)} \ge k \text{ and } \deg(g_k^{(j)}) \le r^{(j)} - k\} \\ &\cup \{j \in C_k \,|\, r^{(j)} < k \text{ and } g_k^{(j)} = 0\}. \end{aligned}$$

(Thus the set C_k just contains the indices j such that in Algorithm 4.1 Step 3, starting with the factors $g_0^{(j)}$ and $h_0^{(j)}$, Equations (9) and (10) are performed at least k times.)

We also replace Step 4 with

Step 4′: For each $j \in C_{\lfloor n/2 \rfloor}$ determine whether $g^{(j)} := \sum_{k=0}^{r} g_k^{(j)} y^k$ divides f. If so then output $g^{(j)}$.

Our modified algorithm thus comprises Steps $1, 2, 3'$ and $4'$, and we will refer to this version as Algorithm 4.1′. It is clearly sufficient to determine the average running time of Algorithm 4.1′ to prove Theorem 4.2. The main challenge in doing this is to determine the expected cardinality of the sets C_k for randomly selected input. We shall do this, and prove Theorem 4.2, in the next section.

5. An analysis of the algorithm

5.1. **Polynomial arithmetic and the distribution of factors.** Our algorithm uses basic polynomial arithmetic such as multiplication and factorisation of univariate polynomials, and in Section 6 we shall consider gcd computations for bivariate polynomials over $\mathbb{F}_q$. We measure the time complexity of an algorithm by the number of operations used in $\mathbb{F}_q$, which is easily transformed into the number of bit operations. A product, division or gcd of two univariate polynomials of degree at most n over $\mathbb{F}_q$ can be computed in $\mathcal{O}(n^2)$ operations in $\mathbb{F}_q$ using "classical" arithmetic, or in $\mathcal{O}(n \log^2 n) = \tilde{\mathcal{O}}(n)$ operations in $\mathbb{F}_q$ using fast algorithms (Schönhage and Strassen 1971 [12], Cantor and Kaltofen 1991 [2]). So a product of two polynomials in $\mathbb{F}_q[x, y]$ of bidegree at most (m, n) can be computed in $\mathcal{O}(mn \log^2(mn)) = \tilde{\mathcal{O}}(mn)$ operations in $\mathbb{F}_q$. Factoring a univariate polynomial of degree n over $\mathbb{F}_q$ can be done in time $\tilde{\mathcal{O}}(n^2 + n \log q)$ (von zur Gathen and Shoup 1992 [8]) or $\mathcal{O}(n^{1.815} \log q)$ (Kaltofen and Shoup 1998 [10]). To compute gcd of bivariate polynomials, we use a modular approach (Brown 1971 [1], Geddes et al 1992 [9]). For any two polynomials in $\mathbb{F}_q[x, y]$ of total degree n, their gcd can be found in time $O(n^4)$ (using "classical" arithmetic).

We also need the following lemma.

Lemma 5.1. *The average number of unordered, non-trivial pairs of monic factors $\{g, h\}$ of a squarefree monic polynomial $f \in \mathbb{F}_q[x]$ (so $f = gh$) of degree $n(\geq 3)$ over the field $\mathbb{F}_q$ is at most $n/2$.*

Proof. Let $SF(n, q)$ denote the set of all squarefree monic polynomials of degree n over $\mathbb{F}_q$. Then $|SF(1, q)| = q$ and, for $n \geq 2$, $|SF(n, q)| = (q-1)q^{n-1}$, due to L. Carlitz [3] (see also[5]). Now let $f \in SF(n, q)$. We need to find the average number of monic factors of f whose degree is at least 1 and not greater than $\lfloor n/2 \rfloor$. This is $|SF(n, q)|$ divided into

the following expression:

$$\begin{aligned}
& \sum_{f\in SF(n,q)} \sum_{i=1}^{\lfloor n/2\rfloor} \sum_{g\in SF(i,q),\, g|f} 1 \\
= & \sum_{i=1}^{\lfloor n/2\rfloor} \sum_{g\in SF(i,q)} \sum_{h\in SF(n-i,q),\, \gcd(g,h)=1} 1 \\
\leq & \sum_{i=1}^{\lfloor n/2\rfloor} \sum_{g\in SF(i,q)} \sum_{h\in SF(n-i,q)} 1 \\
= & \; q(q-1)q^{n-2} + \sum_{i=2}^{\lfloor n/2\rfloor} q^{i-1}(q-1)q^{n-i-1}(q-1) \\
= & \; q^{n-1}(q-1) + (\lfloor n/2\rfloor - 1)(q-1)^2 q^{n-2}.
\end{aligned}$$

Finally $1 + (1-1/q)(\lfloor n/2\rfloor - 1) \leq n/2$, so the lemma is proved. □

5.2. **Affine maps.** Let $U(m,q)$ denote the set of all univariate polynomials over $\mathbb{F}_q$ of total degree bounded by m. Throughout this section we shall consider random variables on the sets $M(n,q)$ and $U(m,q)$, where $m \leq n$, with respect to the uniform distribution. We use the notation $\mathbf{E}(.)$ to denote the expectation of a random variable. This is of course just the average, but it is convenient to use the formalism of probability theory in our proofs.

We wish to obtain an estimate of the likelihood that the conditions on the degrees of the polynomials "g_k" in Step 3′ of Algorithm 4.1′ meet the required restrictions. This will allow us to estimate the expected cardinality of the sets C_k for input polynomials chosen uniformly at random from $M(n,q)$. We do this after first presenting a necessary result on affine maps.

Recall that any affine map L from $\mathbb{F}_q^m$ to $\mathbb{F}_q^w$ may be represented uniquely, with respect to the natural bases, as $L(x) = Ax + b$ where A is an $w \times m$ matrix over $\mathbb{F}_q$ and $b \in \mathbb{F}_q^w$. In the case that $m \geq w$, we shall say that L has *full rank* if the corresponding matrix A has rank w. Thus a full rank affine map L maps $\mathbb{F}_q^m$ surjectively "q^{m-w} to 1" onto the space $\mathbb{F}_q^w$.

Lemma 5.2. *Let $m \geq w$ and $L_1, L_2, \ldots, L_t$ be full rank affine maps from $\mathbb{F}_q^m$ to $\mathbb{F}_q^w$. For z selected uniformly at random from $\mathbb{F}_q^m$ the expected number of L_j such that $L_j(z) = 0$ is t/q^w.*

Proof. Observe firstly that for each L_j the cardinality of the preimage $L_j^{-1}(0)$ is exactly q^{m-w}. Now consider the array A with q^m rows and

t columns with entries from $\mathbb{F}_q^w$ defined as follows. Order the elements of $\mathbb{F}_q^m$ as $z_1, z_2, \ldots, z_{q^m}$. The (i,j)th entry of A is $L_j(z_i)$. For z chosen uniformly at random from $\mathbb{F}_q^m$ the expected number of L_j such that $L_j(z) = 0$ is just the number of zero w-tuples in this array divided by q^m. Now the jth column has $|L_j^{-1}(0)| = q^{m-w}$ zero elements of $\mathbb{F}_q^w$. As there are t columns the required expected value is therefore $tq^{m-w}/q^m = t/q^w$. This completes the proof. □

We now consider again Equation (9) in $\mathbb{F}_q[x]$. Here $\deg(f_k) \leq n-k$, $\deg(g_0) = r$ and we have $n - k \geq r$ (since $k \leq \lfloor n/2 \rfloor$). By interpreting f_k and g_k as vectors in $\mathbb{F}_q^{n-k}$ and $\mathbb{F}_q^r$ respectively, Equation (9) defines a map, which we denote M, from $\mathbb{F}_q^{n-k}$ to $\mathbb{F}_q^r$. Specifically $M(f_k) = g_k$.

Lemma 5.3. *The map M is a full rank affine map.*

Proof. This follows from the observation that the map M can be decomposed as $M = Q \circ R \circ S$. Here $S : \mathbb{F}_q^{n-k} \to \mathbb{F}_q^r$ is the full rank linear map $z \mapsto z \bmod g_0$, $R : \mathbb{F}_q^r \to \mathbb{F}_q^r$ is the full rank affine map $z \bmod g_0 \mapsto z - \sum_{i=1}^{k-1} g_i h_{k-i} \bmod g_0$ and $Q : \mathbb{F}_q^r \to \mathbb{F}_q^r$ is the full rank linear map $z \bmod g_0 \mapsto vz \bmod g_0$ (recall that v is invertible $\bmod g_0$ so this map is indeed a bijection). □

For $n - k \geq r \geq k$ we know that at the kth lift, if we are to find a polynomial factor, the polynomial g_k must satisfy $\deg(g_k) \leq r - k$. Similarly, for $n - k \geq r$ and $r < k$, at the kth lift, if we are to find a polynomial factor, then g_k must equal the zero polynomial. Define $w = \min\{k-1, r\}$ and let P denote the map from $\mathbb{F}_q^r$ to $\mathbb{F}_q^w$ which projects onto the last w coordinates. Thus for $g_k \bmod g_0$ we have that $\deg(g_k) \leq r - k$ in the case $r \geq k$, or $g_k = 0$ in the case $r < k$, if and only if $P(g_k) = 0 \in \mathbb{F}_q^w$. Now let $L = P \circ M$ denote composition of our full rank affine map M with the projection P. Then $L(f_k) = (P \circ M)(f_k) = P(g_k)$, and moreover, L is still a full rank affine map, but now of rank w whereas M was of rank r. So we have

Lemma 5.4. *Define the map L and integers n, k and w as above. Then L is a full rank affine map from $\mathbb{F}_q^{n-k}$ to $\mathbb{F}_q^w$. Moreover, the appropriate condition in Step 3 is met — that is $\deg(g_k) \leq r - k$ in the case $r \geq k$ or $g_k = 0$ in the case $r < k$ — if and only if $L(f_k) = 0$.*

The above lemma may now be used to prove

Lemma 5.5. *For $f \in M(n,q)$ and $1 \leq k \leq n$, let $c_k (= c_k(f))$ denote the cardinality of the set C_k when f is input to Algorithm 4.1′. With respect to the uniform distribution on $M(n,q)$ denote by $\mathbf{E}(c_k)$*

the expectation of c_k. Then for $2 \leq k \leq \lfloor n/2 \rfloor$ we have

$$\mathbf{E}(c_{k+1}) \leq \mathbf{E}(c_k)/q.$$

Proof. For each $j \in E_k$ associate with Equation (11) an affine map L_j, as described in the paragraphs preceding Lemma 5.4. This gives c_k full rank affine maps $\{L_j\}_{1 \leq j \leq c_k}$ from $\mathbb{F}_q^{n-k}$ to $\mathbb{F}_q^{w^{(j)}}$ where $w^{(j)} = \min\{k-1, r^{(j)}\}$. Now let $w = \min_{j \in C_k}\{w^{(j)}\}$ and $L'_1, L'_2, \ldots, L'_{c_k}$ be defined as $L'_j = P_j \circ L_j$, where P_j is the projection of the first w coordinates from $\mathbb{F}_q^{w^{(i)}}$ onto $\mathbb{F}_q^w$. Then we have a set of c_k full rank (rank w) affine maps L'_j from $\mathbb{F}_q^{n-k}$ to $\mathbb{F}_q^w$. Observe that if $L_j(f_k) = 0$ then $L'_j(f_k) = 0$.

Because of the uniform distribution on $M(n,q)$, we see that f_k in Equation (11) is chosen uniformly at random from the set $U(n-k,q)$ of all polynomials over $\mathbb{F}_q$ of degree not greater than $n-k$. Thus by Lemma 5.2 the expected number of L'_j with $L'_j(f_k) = 0$ is c_k/q^w. Hence by our observation at the end of the preceding paragraph, the expected number of L_j such that $L_j(f_k) = 0$ is not greater than c_k/q^w. It follows from Lemma 5.4 that the expected number of $g_k^{(j)}$ which meet the required condition — $\deg(g_k^{(j)}) \leq r-k$ in the case $r \geq k$ and $g^{(j)} = 0$ in the case $r < k$ — cannot be greater than c_k/q^w. Hence the expectation, with respect to the uniform distribution on $U(n-k,q)$, of c_{k+1} is not greater than c_k/q^w. Thus $\mathbf{E}(c_{k+1})$, the expected value of c_{k+1} with respect to the uniform distribution on $M(n,q)$, is not greater than $\mathbf{E}(c_k)/q^w$. The result now follows since trivially $w \geq 1$. □

5.3. **Proof of the main theorem.** We now prove Theorem 4.2.

Proof. Throughout this proof we shall ignore logarithmic factors in n and q in our estimates on the expected number of $\mathbb{F}_q$-field operations. Also these estimates are only true for sufficiently large n and q.

Let $\mathbf{E}(c_k)$ denote the expected cardinality of the set C_k over the uniform distribution on $M(n,q)$. By Lemma 5.5 we have that

$$\mathbf{E}(c_k) \leq \mathbf{E}(c_1)/q^{k-1}. \tag{13}$$

Moreover, by Lemma 5.1 we see that $\mathbf{E}(c_1) \leq n$, this is just the expected number of suitably normalised pairs of factors of a squarefree univariate polynomial of degree n. We claim now that the number of $\mathbb{F}_q$-field operations in the algorithm has expected value not greater than a constant times

$$d(n,q) + \mathbf{E}(c_1)n^\alpha + \left(\sum_{k=1}^{\lfloor n/2 \rfloor} (\sum_{j=1}^{k} j) n^\alpha \mathbf{E}(c_k) \right) + \mathbf{E}(c_{\lfloor n/2 \rfloor}) n^{2+\alpha}. \tag{14}$$

The first term in expression (14) corresponds to the univariate factorisation in Step 1, and the second term to the computations in Step 2. The $\lfloor n/2 \rfloor$ terms in the outer summation correspond to the computations performed in the parallel lifting up to the $\lfloor n/2 \rfloor$th stage in Step 3′. In Step 4′ the polynomial of smaller degree in each pair of polynomials corresponding to the indices in $C_{\lfloor n/2 \rfloor}$ is then divided into f. This accounts for the last term in the expression; note that the factor $n^{2+\alpha}$ is either n^3 or n^4 depending upon whether we are using fast or standard polynomial division in Step 4′.

Substituting (13) into (14) we find the expected number of field operations is not greater than a constant times

$$d(n,q) + \mathbf{E}(c_1)n^\alpha + \sum_{k=1}^{\lfloor n/2 \rfloor} \frac{k(k+1)}{2q^{k-1}} n^\alpha \mathbf{E}(c_1) + \frac{n^{2+\alpha}\mathbf{E}(c_1)}{q^{\lfloor n/2 \rfloor - 1}}.$$

As observed before we have $\mathbf{E}(c_1) \leq n$ and thus the overall expression is not greater than

$$d(n,q) + n^{\alpha+1} + n^{\alpha+1} \left(\sum_{k=1}^{\lfloor n/2 \rfloor} \frac{k(k+1)}{2q^{k-1}} + \frac{n^2}{q^{\lfloor n/2 \rfloor - 1}} \right).$$

It is easily seen that this expression is less than

$$d(n,q) + n^{\alpha+1} + 2n^{\alpha+1} \sum_{k=1}^{\lfloor n/2 \rfloor} \frac{k(k+1)}{q^{k-1}}.$$

Finally observe that

$$\sum_{k=1}^{\lfloor n/2 \rfloor} \frac{k(k+1)}{q^{k-1}} \leq \sum_{k=1}^{\infty} \frac{k(k+1)}{q^{k-1}} = \frac{2}{(1-1/q)^3} \leq 16,$$

as $q \geq 2$. Thus we have shown that the expected number of $\mathbb{F}_q$-field operations in the algorithm is bounded by a constant times $d(n,q) + n^{\alpha+1}$, ignoring logarithmic factors in n and q, and for suitably large n and q. This completes the proof of Theorem 4.2. □

6. A randomised version of the algorithm

Let $f \in T(n,q)$ be squarefree in $\mathbb{F}_q[x,y]$. In general, $f_0 = f \bmod y$ may not be squarefree in $\mathbb{F}_q[x]$. We show how to transform f into a member of $M(n,q)$ so that it can be factored by the Hensel lifting algorithm.

Lemma 6.1. *Let S be a subset of $\mathbb{F}_q$ and $f \in T(n,q)$ squarefree. For random $\beta \in S$, we have $g = f(x, y+\beta) \in M(n,q)$ with probability at least $1 - n(2n-1)/|S|$.*

Proof. We need to determine how likely $g_0 = g \bmod y$ is squarefree for random $\beta \in S$. Note that $g_0 = g(x,0) = f(x,\beta)$. First let us view β as a variable and $g_0 \in \mathbb{F}_q[x,\beta]$. Then g_0 and f determine each other by simple substitutions. Since f is squarefree, we see that g_0 is squarefree in $\mathbb{F}_q[x,\beta]$ so squarefree in $\mathbb{F}_q(\beta)[x]$. Hence the resultant

$$R = \mathrm{Res}_x(g_0, \frac{\partial g_0}{\partial x}) \in \mathbb{F}_q[\beta]$$

is nonzero and has degree (in β) at most $n(2n-1)$. Now we let β take random values in S. With probability at least $1 - n(2n-1)/|S|$, we have $R \neq 0$ so g_0 is squarefree in $\mathbb{F}_q[x]$. □

If $q > 4n^2$ then we can take $S = \mathbb{F}_q$ and the probability in the lemma will be at least $1/2$. If q is small, one needs to go to an extension of $\mathbb{F}_q$ of suitable size and factor f over there and then combine the factors to go down to $\mathbb{F}_q$. For simplicity, we will assume that q is already large enough to have any required probability of success.

Now one may easily obtain the following randomised version of Algorithm 4.1.

Algorithm 6.2. Randomised Hensel Factorisation
Input: A polynomial $f \in T(n,q)$.
Output: A proper factor of f, "Irreducible" or "Failure"

Step 1: Choose $\beta \in \mathbb{F}_q$ uniformly at random and define $\bar{f} = f(x, y+\beta)$. Check whether $\bar{f}_0 = \bar{f} \bmod y$ is squarefree.

Step 2: If $\bar{f}_0$ is not squarefree then compute $h = \gcd(\bar{f}, \frac{\partial \bar{f}}{\partial x})$ in $\mathbb{F}_q[x,y]$. If $h \neq 1$ then output h, otherwise output "Failure".

Step 3: If $\bar{f}_0$ is squarefree then input $\bar{f}$ to Algorithm 4.1. If Algorithm 4.1 has no output then output "Irreducible". Otherwise output $g(x, y-\beta)$ for any polynomial g output by Algorithm 4.1.

Theorem 6.3. *Suppose $q > 4n^2$. For $f \in T(n,q)$, the average running time of Algorithm 6.2 is $\mathcal{O}(n^{1+\alpha} + d(n,q))$, where α and $d(n,q)$ are*

defined as in Theorem 4.2, and the probability of failure is less than $1/2$.

Proof. The algorithm fails only if f is squarefree in $\mathbb{F}_q[x,y]$ but $\bar{f}_0 = f(x,\beta)$ is not squarefree in $\mathbb{F}_q[x]$. By Lemma 6.1, the probability of this happening is less than $1/2$.

On the running time, we assume that f is chosen from $T(n,q)$ uniformly at random. Then for any $\beta \in \mathbb{F}_q$, $\bar{f}$ is still uniform random in $T(n,q)$ (since the transform is a bijection). Particularly, $\bar{f}_0$ is a uniform at random monic polynomial in $\mathbb{F}_q[x]$ of degree n and $\bar{f}$ is uniform at random in $M(n,q)$. The probability of $\bar{f}_0$ being squarefree is $q^{n-1}(q-1)/q^n = 1 - 1/q$, so the probability of not being squarefree is $1/q$.

Now Step 1 costs $\mathcal{O}(n^2)$, Step 2 costs $\mathcal{O}(n^4)$ and Step 3 costs on average $\tilde{\mathcal{O}}(n^{1+\alpha} + d(n,q))$. So the average cost for the whole algorithm is

$$\tilde{\mathcal{O}}\left(n^2 + \frac{1}{q}n^4 + (1-\frac{1}{q})(n^{1+\alpha} + d(n,q))\right) = \tilde{\mathcal{O}}(n^{1+\alpha} + d(n,q)),$$

as $n^4/q \leq n^2$. The theorem is proved. □

7. Conclusion

We presented a modified version of the Hensel lifting method for factoring bivariate polynomials over finite fields. The average running was shown to be almost linear in the input size. Compared to Collins' analysis for univariate integral polynomials, our proof was unconditional. Our success relies on the fact that almost all polynomials are irreducible and so presumably can not be lifted too high. It may be interesting to give a more sensitive analysis that yields the "variance" of the number of field operations required in our Hensel lifting based factorisation algorithm.

In practice, polynomials to be factored may be known to be reducible in advance. Is it possible to find the average time for all reducible polynomials?

Acknowledgement. We thank Daniel Panario for his helpful disscusion on the number of factors of polynomials and for bringing Carlitz's result to our attention.

References

[1] W. S. Brown, "On Euclid's algorithm and the computation of polynomial greatest common divisors", *J. ACM* **18** (1971), 478–504.

[2] D.G. Cantor and E. Kaltofen, "On fast multiplication of polynomials over arbitrary algebras", *Acta Inform.* **28** (1991), 693-701.

[3] L. Carlitz, "The arithmetic of polynomials in a Galois field", *Amer. J. Math.* **54** (1932), 39–50.

[4] G. E. Collins, "Factoring univariate integral polynomials in polynomial average time", *Symbolic and algebraic computation* (EUROSAM '79, Internat. Sympos., Marseille, 1979), pp. 317–329, Lecture Notes in Comput. Sci., 72, Springer, Berlin-New York, 1979.

[5] P. Flajolet, X. Gourdon and D. Panario, "Random polynomials and polynomial factorization", *Automata, languages and programming* (Paderborn, 1996), 232–243, Lecture Notes in Comput. Sci., 1099, Springer, Berlin, 1996.

[6] S. Gao, "Absolute irreducibility of polynomials via Newton polytopes," preprint, 1998 (16 pages).
(Available at URL: http://www.math.clemson.edu/faculty/Gao)

[7] S. Gao and A.G.B. Lauder, "Decomposition of polytopes and polynomials," preprint, 1999 (17 pages).
(Available at URL: http://www.math.clemson.edu/faculty/Gao)

[8] J. von zur Gathen and V. Shoup, "Computing Frobenius maps and factoring polynomials", *Computational Complexity* **2** (1992), 187–224.

[9] K. O. Geddes, S. R. Czapor and G. Labahn, *Algorithms for Computer Algebra*, Kluwer, Boston/Dordrecht/London, 1992.

[10] E. Kaltofen and V. Shoup, "Subquadratic-time factoring of polynomials over finite fields", *Math. Comp.* **67** (1998), no. 223, 1179–1197.

[11] D.R. Musser, "Multivariate polynomial factorization", *J. ACM* **22** (1975), 291–308.

[12] A. Schönhage and V. Strassen, "Schnelle Multiplikation großer Zahlen", *Computing* **7** (1971), 281-292.

[13] D. Wan, "Factoring polynomials over large finite fields", *Math. Comp.* **54** (1990), No. 190, 755–770.

[14] P. S. Wang, "An improved multivariate polynomial factorization algorithm", *Math. Comp.* **32** (1978), 1215–1231.

[15] P. S. Wang and L. P. Rothschild, "Factoring multivariate polynomials over the integers," *Math. Comp.* **29** (1975), 935–950.

[16] H. Zassenhaus, "On Hensel factorization I", *J. Number Theory* **1** (1969), 291–311.

Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-0975 USA *E-mail address*: sgao@math.clemson.edu

Mathematical Institute, Oxford University, Oxford OX1 3LB, UK
E-mail address: lauder@maths.ox.ac.uk