COMPUTING AUTOMORPHISMS OF ABELIAN NUMBER FIELDS

VINCENZO ACCIARO AND JÜRGEN KLÜNERS

ABSTRACT. Let $L = \mathbb{Q}[\alpha]$ be an abelian number field of degree n. Most algorithms for computing the lattice of subfields of L require the computation of all the conjugates of α. This is usually achieved by factoring the minimal polynomial $m_\alpha(x)$ of α over L. In practice, the existing algorithms for factoring polynomials over algebraic number fields can handle only problems of moderate size. In this paper we describe a fast probabilistic algorithm for computing the conjugates of α, which is based on p-adic techniques. Given $m_\alpha(x)$ and a rational prime p which does not divide the discriminant $\mathrm{disc}(m_\alpha(x))$ of $m_\alpha(x)$, the algorithm computes the Frobenius automorphism of p in time polynomial in the size of p and in the size of $m_\alpha(x)$. By repeatedly applying the algorithm to randomly chosen primes it is possible to compute all the conjugates of α.

1. INTRODUCTION

Let us assume that $L = \mathbb{Q}[\alpha]$ is an abelian number field of degree n over $\mathbb{Q}$, given by the minimal polynomial $m_\alpha(x)$ of α over $\mathbb{Q}$, and without loss of generality let us assume that $\alpha \in \mathcal{O}$, the ring of algebraic integers of L. The computation of the automorphisms of L over $\mathbb{Q}$ is equivalent to the computation of all conjugates of $\alpha \in L$. Clearly, one can compute these conjugates by factoring $m_\alpha(x)$ over L.

The factorization of $m_\alpha(x)$ over L is computed using general purpose algorithms, which do not take into account the Galois structure of L. For example, H. W. Lenstra states [12, Corollary 3.3] that there is a polynomial time algorithm that given a polynomial $f(x)$ decides whether the splitting field of $f(x)$ is abelian and determines the Galois group G if G is abelian and $f(x)$ is irreducible. The proof is based on the observation that a transitive abelian permutation group of degree n has order n. Moreover, if $f(x)$ is reducible, then G is abelian if and only if the Galois group of each irreducible factor is abelian. For monic irreducible $f(x)$, the polynomial factorization algorithm of S. Landau [9] is applied to $f(x)$ over the field $\mathbb{Q}[x]/f(x)\mathbb{Q}[x]$ in order to determine its Galois group.

However, Lenstra's observation does not imply a very efficient algorithm to test whether $f(x)$ is abelian. In fact, our experience in experiments conducted with PARI, Maple and KASH on a workstation Sun 20 suggests that none of the algorithms implemented in these packages for factoring polynomials over finite nontrivial extensions of $\mathbb{Q}$ can be used to solve large problems. This fact should not surprise us, if we consider the complexity of the existing factorization algorithms over algebraic number fields.

Many of the existing algorithms for computing the lattice of subfields of L require the expensive factorization of $m_\alpha(x)$ over L. The subfield algorithm described in

1991 *Mathematics Subject Classification.* Primary 11R37; Secondary 11Y40.

Key words and phrases. computational number theory, abelian number fields, automorphisms.

[7] does not require factorizations of polynomials over number fields. However, in the abelian case efficiency issues still suggest to use the conjugates of α for the computation of the subfields of L.

In this paper we propose a new technique for computing the conjugates of a root α of a monic abelian irreducible polynomial $m_\alpha(x)$, which works well in practice even with polynomials of large size.

Given $m_\alpha(x)$ and a rational prime p which does not divide the discriminant $\mathrm{disc}(m_\alpha(x))$, the algorithm returns an automorphism of $L = \mathbb{Q}[\alpha]$ over $\mathbb{Q}$ known as the Frobenius automorphism of p.

By repeatedly applying the algorithm to randomly chosen primes smaller than some bound b, it is possible to compute all the conjugates of α. In § 2.6 we will show that such a bound b exists, and we will give some estimates for it.

A meaningful analysis of the expected number of times that the algorithms has to be executed, with randomly chosen primes below b, in order to find all the conjugates of α, would require one to assume a very large bound b. For, in this case, Chebotarev Density Theorem tells us that the primes mapping to a fixed element of the Galois group of L are approximately equidistributed in the set of primes smaller than b.

We will restrict ourselves to show that, for a fixed prime p, the algorithm runs in time polynomial in the size of p and in the size of $m_\alpha(x)$.

The algorithm described in this paper has been implemented using the number theory package KASH [5], developed in Berlin by Prof. M. Pohst and his collaborators.

For the terminology and the basic concepts of algebraic number theory used in this paper we refer the reader to [11].

2. Description of the method

Let p be a rational prime which does not divide the discriminant $\mathrm{disc}(m_\alpha(x))$, and let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ denote the prime ideals of $\mathcal{O}$ lying above p. Let $f = n/r$ denote the degree of inertia of $\mathcal{P}_i$ $(1 \le i \le r)$.

Let σ be the Frobenius automorphism of $\mathcal{P}_1$. It is well known that the Frobenius automorphisms of the prime ideals above p are conjugate to each other. Since the Galois group of L over $\mathbb{Q}$ is abelian, it follows that σ turns out to be the Frobenius automorphism of $\mathcal{P}_2, \ldots, \mathcal{P}_r$ as well. For this reason σ is called simply the Frobenius automorphism of p. Now, by definition

$$\sigma(\alpha) \equiv \alpha^p \pmod{\mathcal{P}_1}, \ldots, \sigma(\alpha) \equiv \alpha^p \pmod{\mathcal{P}_r}$$

and therefore

$$\sigma(\alpha) \equiv \alpha^p \pmod{p\mathcal{O}} \tag{1}$$

Let $g_0(x) \in \mathbb{Z}[x]$ with $\deg g_0(x) < n$, such that $g_0(\alpha) \equiv \alpha^p \pmod{p\mathcal{O}}$. The polynomial $g_0(x)$ can be computed efficiently as follows. Let $\overline{x}$ denote the image of x in $(\mathbb{Z}/p\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p\mathbb{Z})[x]$. Take for $g_0(x)$ the representative of $\overline{x}^p$ in $\mathbb{Z}[x]$ whose coefficients are positive and bounded by p.

We know that $m_\alpha(\sigma(\alpha)) = 0$, and it is clear that $m_\alpha(g_0(\alpha)) \in p\mathcal{O}$. In order to find σ we will use p-adic lifting. Thus, for $k = 1, 2, \ldots$ we would like to compute an integral polynomial $g_k(x)$ of degree less than n such that $m_\alpha(g_k(\alpha)) \in p^{2^k}\mathcal{O}$. We will show in §2.5 that, when k is large enough, it is possible to recover $\sigma(\alpha)$ from $g_k(\alpha)$.

2.1. **Automorphisms in $(\mathbb{Z}/p\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p\mathbb{Z})[x]$.** Our aim is to compute all the automorphisms of L. The p-adic lifting of the computed Frobenius automorphisms in $\mathcal{O}/p\mathcal{O}$ is the most expensive part of our algorithm. We want to avoid this lifting if there is no new contribution to the group of automorphisms.

Let σ be an automorphism of L which is represented in the form $\sigma = g(\alpha)$ with $g(x) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{Q}[x]$. We call $g(x)$ the polynomial representation of σ. In the usual way we define for $C, D, U, M \in \mathbb{Z}$ with $(D, M) = 1$: $C/D \equiv U \pmod{M}$ if $C \equiv DU \pmod{M}$. Then $\overline{\sigma}$ is the image of σ in $(\mathbb{Z}/p\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p\mathbb{Z})[x]$. There is a unique representation of $\overline{\sigma}$ by $\overline{g}(x) = \sum_{i=0}^{n-1} \overline{a}_i x^i$, where $\overline{a}_i \equiv a_i \pmod{p}$. This definition makes sense because p does not divide the denominator of a_i. Let A be a list of computed automorphisms of L and $\overline{\sigma}$ an automorphism which is known in $(\mathbb{Z}/p\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p\mathbb{Z})[x]$. We want to check if σ belongs to A. Define $\overline{A} = \{\overline{\tau} \mid \tau \in A\}$.

Lemma 1. *σ belongs to A if and only if the polynomial representation of $\overline{\sigma}$ coincides with one of the polynomial representations of an automorphism of $\overline{A}$.*

Proof. If $\sigma = \tau$ the polynomial representations of $\overline{\sigma}$ and $\overline{\tau}$ coincide. Let $\alpha_1, \dots, \alpha_n$ be the zeros of $m_\alpha(x)$ and $\overline{\alpha}_1, \dots, \overline{\alpha}_n$ be the zeros of $\overline{m}_\alpha(x) \equiv m_\alpha(x) \pmod{p}$. Since $\overline{m}_\alpha(x)$ has no double roots, there is an isomorphism between α_i and $\overline{\alpha_i}$, and it follows that σ and τ coincide if $\overline{\sigma}$ and $\overline{\tau}$ coincide. □

2.2. **Applying automorphisms.** We represent an automorphism σ by its image on α, thus we have $\sigma(\alpha) = \sum_{i=0}^{n-1} a_i \alpha^i$. Let $\tau(\alpha) = \sum_{i=0}^{n-1} b_i \alpha^i$ be another automorphism. We want to compute the image of $\sigma\tau$ on α. We have

$$\sigma\tau(\alpha) = \sigma(\sum_{i=0}^{n-1} b_i \alpha^i) = \sum_{i=0}^{n-1} b_i \sigma(\alpha^i).$$

The most expensive part is the computation of $\sigma(\alpha^i)$ or $\sigma(\alpha)^i$. If we want to apply σ more than once, then we have to save the images of σ on α^i $(0 \le i \le n-1)$. The proof of the next lemma is immediate.

Lemma 2. *In order to apply an automorphism σ to an element τ, we need $n-2$ multiplications of algebraic numbers for the initialization of the automorphism, and n multiplications of a rational number times an algebraic number for the application of the automorphism.*

2.3. **p-adic lifting.** We show now how to perform the p-adic lifting. We want to use quadratic Newton lifting (see [10, pages 308–311] and [6, Appendix B]).

Let $\beta_0 = \sum_{i=1}^{n-1} b_{0,i} \alpha^i$ be an approximation of a zero of $m_\alpha(x)$ modulo p. Since $p \nmid \mathrm{disc}(m_\alpha(x))$ we can compute an element ω_0 which satisfies $\omega_0 m'_\alpha(\beta_0) \equiv 1 \pmod{p\mathcal{O}}$, using the extended-gcd algorithm for polynomials over finite fields.

In the following we construct two sequences of algebraic integers $\{\beta_k\}$ and $\{\omega_k\}$ which satisfy the relations:

(i) $\beta_{k+1} \equiv \beta_k \pmod{p^{2^k}\mathcal{O}}$
(ii) $\omega_{k+1} \equiv \omega_k \pmod{p^{2^k}\mathcal{O}}$
(iii) $m_\alpha(\beta_k) \equiv 0 \pmod{p^{2^k}\mathcal{O}}$
(iv) $\omega_k m'_\alpha(\beta_k) \equiv 1 \pmod{p^{2^k}\mathcal{O}}$

This can be done by the following double iteration:

(i) $\beta_{k+1} \equiv \beta_k - \omega_k m_\alpha(\beta_k) \pmod{p^{2^{k+1}}\mathcal{O}}$

(ii) $\omega_{k+1} \equiv \omega_k[2 - \omega_k m'_\alpha(\beta_{k+1})] \pmod{p^{2^{k+1}}\mathcal{O}}$

We remark that it is possible to compute β_k by the following simple iteration:

$$\beta_{k+1} \equiv \beta_k - \frac{m_\alpha(\beta_k)}{m'_\alpha(\beta_k)} \pmod{p^{2^{k+1}}\mathcal{O}}.$$

The double iteration has the advantage that we avoid the time consuming division.

2.4. **An upper bound for the number of iterations.** Let d be the largest positive integer whose square divides $\mathrm{disc}(m_\alpha(x))$. It is well known that $\mathcal{O} \subset (1/d)\mathbb{Z}[\alpha]$. We want to find a positive integer B such that all the conjugates of α can be expressed as $m(x)/d$, where the coefficients of $m(x) \in \mathbb{Z}[x]$ are bounded in absolute value by B. This gives us an upper bound for the number of iterations of our algorithm, for we can stop the lifting process as soon as $p^{2^k} > 2B^2$ (Lemma 4), that is after $\lceil \log\log_p 2B^2 \rceil$ iterations. Let $m_\alpha(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$, where by hypothesis $a_i \in \mathbb{Z}$ for $i = 0, \ldots, n$ and $a_n = 1$. Recall that $|m_\alpha(x)|$ is defined to be the Euclidean length of $m_\alpha(x)$, that is $(\sum_{i=0}^n a_i^2)^{1/2}$, and $|m_\alpha(x)|_{\max}$ is defined to be the height of $m_\alpha(x)$, that is $\max_i |a_i|$. For the proof of the following lemma we refer the reader to [9, Theorem 1.3].

Lemma 3. *Let $\alpha_1 = \alpha, \ldots, \alpha_n$ denote the conjugates of α.*
Let $\alpha_h = (1/d)\sum_{j=0}^{n-1} c_{hj}\alpha^j$, with $c_{hj} \in \mathbb{Z}$. Then $|c_{hj}| < B$ for $0 \le j < n$, $1 \le h \le n$, where

$$B = d(1 + |m_\alpha(x)|_{\max}) n(n-1)^{(n-1)/2} |m_\alpha(x)|^{n-1} |\, disc(m_\alpha(x))|^{-1/2} \tag{2}$$

Note that the theoretical bound B given by Lemma 3 is a bit too conservative, and in practice it is possible to use some heuristic bounds which are much smaller [16, p. 332]. We remark that we do not compute d. We only need it just to derive the theoretical bound B.

2.5. **Recovering σ.** So far we have shown how to compute for each $k = 0, 1, \ldots$ a polynomial $g_k(x) \in \mathbb{Z}[x]$ such that $\sigma(\alpha) \equiv g_k(\alpha) \pmod{p^{2^k}\mathcal{O}}$. Without loss of generality we can assume that $g_k(x)$ has positive coefficients smaller than p^{2^k}. The next lemma tells us how to recover a rational number C/D from a modulo M approximation. For its proof, and the related algorithm, we refer the reader to [4].

Lemma 4. *Let $U, M \in \mathbb{N}$ such that $(U, M) = 1$. Then there exists an efficient algorithm to compute a pair of integers (C, D) such that $C \equiv DU \pmod{M}$ with $D > 0$ and $|C|, D < \sqrt{M/2}$, if such a pair exists. Otherwise, an indication of failure is returned.*

Now we are able to compute the polynomial $g(x)$ corresponding to σ. We recover the polynomial g coefficientwise from g_k by Lemma 4. We remarked that the bound B grossly overestimates the size of the numerator of the coefficients of $g(x)$. One approach in practice is to compute $g(x)$ from $g_k(x)$ after each step and check if $m_\alpha(g(\alpha)) = 0$; however this is not a good idea since the test is very expensive if $g(\alpha)$ is not a zero of m_α. A better way is to determine $g(x)$ after each iteration and compare this $g(x)$ with the $g(x)$ computed at the previous iteration. We only need to check $m_\alpha(g(\alpha)) = 0$ if $g(x)$ remains invariant.

2.6. **Termination of the algorithm.** Since the Galois group G of L over $\mathbb{Q}$ acts regularly on the set of roots of $m_\alpha(x)$, it follows that, if we are given a root α_j of $m_\alpha(x)$, there is exactly one element τ of G such that $\tau(\alpha) = \alpha_j$. Hence, we are able to determine all the conjugates of α as soon as the Frobenius elements computed so far form a complete set of generators of G. Therefore, the average number of Frobenius elements that must be computed is equal to the average number of elements that must be selected from G in order to obtain a complete set of generators.

Even if the structure of G is not known, the estimates in [1] show that for an abelian group G the average number of elements that must be selected from G in order to generate G is very close to the minimal number $t(G)$ of generators of G. If $n = \prod_{i=1}^{k} p_i^{e_i}$, with p_i distinct primes, then clearly $t(G) \leq \sum_{i=1}^{k} e_i$.

Now, by the Chebotarev Density Theorem [11, Theorem 10, p. 169] the primes mapping to a fixed element of G have density $1/|G| = 1/n$.

Even if we do not know how to select at random a Frobenius automorphism σ, it is possible to show that there exists a positive number b, depending on the field L, such that each element of the Galois group of L is the Frobenius automorphism of some prime $p < b$. Effective bounds for b have been extensively studied in [8].

The best possible estimates for b are obtained by assuming the validity of a long standing conjecture in number theory, known as the Extended Riemann Hypothesis (ERH). If we assume the ERH, then the following lemma is an immediate consequence of the explicit formulas of Bach and Sorenson [2, Thm 5.1]:

Lemma 5 (ERH). *Each element of the Galois group of L is the Frobenius automorphism of some prime $p < (4\ \log|d_L| + 2.5n + 5)^2$.*

Clearly, larger values of b yield a better sampling of the elements of G, that is a distribution of the Frobenius elements which is closer to the uniform distribution.

Note that the knowledge of the structure of G could be used to handle some particular instances of the problem very efficiently. For example, if n is squarefree then G is cyclic, and there are exactly $\phi(n)$ generators of G. It can be shown [14, Exercise 1, p. 266] that for $n \geq 3$ we have $\phi(n)/n \geq c(\log\log n)^{-1}$ for some $c > 0$, and hence a generator of G can be found by random sampling in G in an expected number of $\mathcal{O}(\log\log n)$ trials. Therefore, when n is squarefree it is advisable to search first for a prime p whose degree of inertia is equal to n, and then compute its Frobenius automorphism, which clearly generates G.

3. Examples

Let L be the abelian number field generated by the polynomial $m_\alpha(x) = x^{16} - 112x^{14} + 4532x^{12} - 83472x^{10} + 730358x^8 - 2962896x^6 + 4936148x^4 - 2507824x^2 + 28561$. The Galois group is isomorphic to $C_2 \times C_2 \times C_2 \times C_2$. This is the hardest abelian group of order 16, since it requires at least four generators. In other words, we have to compute at least four Frobenius automorphisms. We computed all the automorphisms in 2.0 seconds on a HP 735 using KASH 1.8 under HP-UX 9.01. We print one non trivial automorphism to give an idea of the size of the coefficients. This automorphism sends α to

$$\frac{1}{1659051313 92}(-1460330046379\alpha + 3045184412874\alpha^3 - 1986742065521\alpha^5 + 506400762490\alpha^7 - 58437366385\alpha^9 + 3179647862\alpha^{11} - 78588459\alpha^{13} + 701446\alpha^{15})$$

We can compute the automorphism group of abelian fields of much higher degree as the following examples, which were given to us by [15], show. Let

$$f(t) := t^{p-1} - pa \text{ and } g(t) := t^p - pat$$

with $p \in \mathbb{P}, a \in \mathbb{Z}$ and $p \nmid a$. We define

$$f_n(t) := g(f^{n-1}(t))$$

where $f^n := f(f^{n-1})$ is defined recursively. What are the specializations for p and a such that the Galois group $\mathrm{Gal}(f_n)$ is abelian for all $n \in \mathbb{N}$? If we let $p = 2, a = -1$ we get the 2^n cyclotomic fields. We computed some examples for $p = 2,\ a = 1$ and $p = 3,\ a = 1$.

p	a	n	degree	time
2	1	2	2	0.2 s
2	1	3	4	0.4 s
2	1	4	8	0.3 s
2	1	5	16	0.8 s
2	1	6	32	12.8 s
2	1	7	64	103 s
2	1	8	128	1224 s
2	1	9	256	55685 s

p	a	n	degree	time
3	1	1	2	0.2 s
3	1	2	6	0.2 s
3	1	3	18	0.4 s
3	1	4	54	14.1 s
3	1	5	168	536 s

The computing times in the second table ($p = 3$) are slightly better since these fields are cyclic. In the first table ($p = 2$) the Galois group is isomorphic to $C_2 \times C_{2^{n-2}}$.

4. Complexity issues

In this section we will discuss the cost of computing the Frobenius automorphism associated to a prime p.

For the sake of efficiency it is advisable to carry out the computation of $g_k(x)$, at the k^{th} iteration of the lifting process, in the finite ring $(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]$, in order to avoid an unacceptable growth of the coefficients involved. This is allowed since if $g_k(x) = \sum_{i=0}^{n-1} b_i x^i \in \mathbb{Z}[x]$ satisfies $m_\alpha(g_k(x)) \equiv 0 \pmod{p^{2^k}\mathcal{O}}$, then any $l(x) = \sum_{i=0}^{n-1} c_i x^i \in \mathbb{Z}[x]$ such that $c_i \equiv b_i \pmod{p^{2^k}}$ for $i = 0, \dots, n-1$ satisfies $m_\alpha(l(x)) \equiv 0 \pmod{p^{2^k}\mathcal{O}}$, as well.

If $k \in \mathbb{N},\ k > 0$, let us assume that the multiplication of two elements of $\mathbb{Z}/p^{2^k}\mathbb{Z}$ requires $\mathcal{O}(\log^2 p^{2^k})$ bit operations, and that the multiplication of two elements of $(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]$ requires $\mathcal{O}(n^2 \log^2 p^{2^k})$ bit operations.

Let us consider first the cost of computing $\overline{\sigma}$. The computation of $\overline{\alpha_j}^p$ is carried out using the binary powering algorithm (see [3, p.8]); hence this step requires $\mathcal{O}(\log p)$ multiplications in $(\mathbb{Z}/p\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p\mathbb{Z})[x]$, and therefore $\mathcal{O}(n^2 \log^3 p)$ bit operations. The next step consists in the initialization of the automorphism $\overline{\sigma}$ (Lemma 2). This step requires $n - 2$ multiplications of two elements of $(\mathbb{Z}/p\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p\mathbb{Z})[x]$, and therefore $\mathcal{O}(n^3 \log^2 p)$ bit operations. In order to apply the automorphism $\overline{\sigma}$ we need n^2 multiplications of two elements of $\mathbb{Z}/p\mathbb{Z}$, and hence $\mathcal{O}(n^2 \log^2 p)$ bit operations. In the worst case the computation of $\overline{\sigma}^n$ requires $\mathcal{O}(n^3 \log^3 p)$ bit operations.

Let us consider now the computation of the inverse of $m'_\alpha(g_1(\alpha))$ modulo $p\mathcal{O}$. We compute it by applying the extended Euclidean gcd algorithm to two polynomials in $(\mathbb{Z}/p\mathbb{Z})[x]$ of degree n and $n-1$. Euclid's algorithm needs $\mathcal{O}(n^2)$ multiplications of two elements of $\mathbb{Z}/p\mathbb{Z}$, and therefore $\mathcal{O}(n^2 \log^2 p)$ bit operations.

Let us consider next the cost of the p-adic lifting. It is clear that the cost of the k^{th} iteration is dominated by the cost of computing $m_\alpha(g_{k-1}(\alpha))$. Using Horner's rule, this task requires $\mathcal{O}(n)$ operations in the ring $(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]$. The update of ω_k requires $\mathcal{O}(n)$ operations in the ring $(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]/m_\alpha(x)(\mathbb{Z}/p^{2^k}\mathbb{Z})[x]$, too. Hence $\mathcal{O}(n^3 2^{2k} \log^2 p)$ bit operations are required at the k^{th} iteration.

If s iterations are required, then the overall cost of the lifting is $\mathcal{O}(n^3 \log^2 p(4+4^2+\ldots+4^{s-1}+4^s))$ bit operations, that is $\mathcal{O}(n^3(\log^2 p)4^{s+1})$ bit operations. Now, if we let $s = \lceil \log(\log_p 2B^2) \rceil = \lceil \log\left((\log 2 + 2\log B)/\log p\right) \rceil$, we obtain $\mathcal{O}(n^3(\log^2 B))$ bit operations.

Finally, Mahler's bound on the discriminant of a polynomial [13, p. 261] shows that, if $m_\alpha(x) = \sum_{i=0}^n a_i x^i$ then $|\operatorname{disc}(m_\alpha(x))| < n^n(\sum_{i=0}^n |a_i|)^{2n-2}$ and hence $d < n^{n/2}(\sum_{i=0}^n |a_i|)^{(2n-2)/2}$. Therefore the cost of computing σ from $g_s(x)$ is dominated by the cost of computing $g_s(x)$, and we obtain an overall complexity of

$$\mathcal{O}(n^3((\log^2 B) + \log^3 p))$$

bit operations. By applying Lemma 3 we conclude that the algorithm runs in time polynomial in the size of p and of $m_\alpha(x)$, for a given p.

Acknowledgements

The first author wishes to thank Prof. V.L. Plantamura for his support.

References

[1] V. Acciaro, *The probability of generating some common families of finite groups,* Utilitas Mathematica 49 (1996), 243–254.
[2] E. Bach and J. Sorenson, *Explicit bounds for primes in residue classes,* Math. Comp. 65 (1996), 1717–1735.
[3] H. Cohen, *A Course in Computational Algebraic Number Theory,* Springer-Verlag, Berlin, 1993.
[4] G.E. Collins and M.J. Encarnación, *Efficient rational number reconstruction,* J. Symb. Comput. 20 (1995), 287–297.
[5] M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, M. Schörnig, K. Wildanger, *KANT V4,* J. Symb. Comput., Vol 24, No 3, 1997, 267–283.
[6] J.D. Dixon, *Computing subfields in algebraic number fields,* J. Austral. Math. Soc. 49 (1990), 434–448.
[7] J. Klüners and M. Pohst, *On computing subfields,* J. Symb. Comput., Vol 24, No 3, 1997, 385–397.
[8] J.C. Lagarias and A.M. Odlyzko, *Effective versions of the Chebotarev density theorem,* Algebraic number fields (A. Fröhlich, ed.), Academic Press, 1977, 409–464.
[9] S. Landau, *Factoring polynomials over algebraic number fields,* SIAM J. Comput. 14 (1985), 184–195.
[10] S. Lang, *Algebra*, Addison-Wesley, Reading, Massachusetts, 1974.
[11] S. Lang, *Algebraic Number Theory,* Addison-Wesley, Reading, Massachusetts, 1994.
[12] H.W. Lenstra, Jr., *Algorithms in algebraic number theory,* Bull. Amer. Math. Soc. 26 (1992), 211-244.
[13] K. Mahler, *An inequality for the discriminant of a polynomial,* Michigan Math. J. 11 (1964), 257-262.
[14] M. Mignotte, *Mathematics for Computer Algebra,* Springer-Verlag, New York, 1992.
[15] F. Nicolae, *private communication*, Berlin, 1997.
[16] P.S. Wang, *Factoring multivariate polynomials over algebraic number fields,* Math. Comp. 30 (1976), 324–336.

[17] M.E. Pohst and H. Zassenhaus, *Algorithmic Algebraic Number Theory,* Cambridge University Press, Cambridge, 1989.

Dipartimento di Informatica, Universita' degli Studi di Bari, via E. Orabona 4, Bari 70125, Italy.
E-mail address: `acciaro@seldi2.uniba.it`

Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany.
E-mail address: `klueners@iwr.uni-heidelberg.de`