An Efficient Implementation for Computing Gröbner Bases over Algebraic Number Fields

Masayuki Noro

Kobe University, Rokko, Kobe 657-8501, Japan
noro@math.kobe-u.ac.jp

Abstract. In this paper we discuss Gröbner basis computation over algebraic number fields. Buchberger algorithm can be executed over any computable field, but the computation is often inefficient if the field operations for algebraic numbers are directly used. Instead we can execute the algorithm over the rationals by adding the defining polynomials to the input ideal and by setting an elimination order. In this paper we propose another method, which is a combination of the two methods above. We implement it in a computer algebra system Risa/Asir and examine its efficiency.

1 Introduction

From the theoretical point of view, Gröbner basis computation can be done over any field by using Buchberger algorithm. In practice, however, the computational efficiency depends on the ground field. The difficulty of Gröbner basis computation over a finite field mainly comes from its combinatorial property because no coefficient swell occurs. But over the rationals, we often suffer from coefficient swell and various methods to avoid it have been investigated. The trace algorithm [4] and F_4 algorithm with modular computation [6] are successful ones. In this article we consider Gröbner basis computation over algebraic number fields. If the operations over an algebraic number field are provided, we can apply usual Buchberger algorithm over the field. Instead, an algebraic number field K can be represented as a residue class ring $\mathbf{Q}[x_1, \ldots, x_l]/J$, where J is a zero-dimensional maximal ideal and a Gröbner basis computation over K can be reduced to that over $\mathbf{Q}$ by joining J to the ideal to be considered. However these method are not satisfactory in view of efficiency. Here we give a simple but efficient method which is a combination of the two methods above.

Notation 1

$\mathrm{HT}(f)$	*: the highest term of f with respect to a term order*
$\mathrm{HC}(f)$	*: the coefficient of* $\mathrm{HT}(f)$
$\mathrm{GF}(p)$	*: the finite prime field of order p*
$\mathrm{NF}_G(f)$	*: a remainder of f with respect to a polynomial set G By fixing the method for choosing a reducer, the remainder is uniquely determined even if G is not a* Gröbner *basis.*
$\mathrm{S}(f,g)$	*: the S-polynomial of a pair $\{f,g\}$.*

$\mathbf{Z}_{\langle p \rangle}$: $\{a/b \mid a \in \mathbf{Z}; b \in \mathbf{Z} \setminus p\mathbf{Z}\} \subset \mathbf{Q}$
ϕ_p : *the canonical projection from* $\mathbf{Z}_{\langle p \rangle}[X]$ *to* $\mathrm{GF}(p)[X]$

2 The Algorithm

Suppose that an algebraic number field $K = K_l$ is represented as a tower of simple extensions:

$$K_0 = \mathbf{Q}, K_i = K_{i-1}(\alpha_i) \quad (i = 1, 2, \ldots, l),$$

where α_i is a root of a monic irreducible polynomial over K_{i-1}:

$$m_i(\alpha_1, \ldots, \alpha_{i-1}, t_i) \in K_{i-1}[t_i] \quad (m_i(t_1, \ldots, t_i) \in R_i = \mathbf{Q}[t_1, \ldots, t_i])$$

and

$$K_i = \mathbf{Q}[t_1, \ldots, t_i]/J_i, \quad J_i = \langle m_1, \ldots, m_i \rangle.$$

Each J_i is a zero-dimensional maximal ideal of R_i. Set $S = K[x_1, \ldots, x_n]$, $T = \mathbf{Q}[x_1, \ldots, x_n, t_1, \ldots, t_l]$, $D = \{m_1, \ldots, m_l\}$, $J = \langle D \rangle$, $x = (x_1, \ldots, x_n)$, $t = (t_1, \ldots, t_l)$ and $\alpha = (\alpha_1, \ldots, \alpha_l)$. Let $\prec$ be a term order in S and $\prec_K$ a product term order of $\prec$ and the lexicographic order in R_l such that $t_1 \prec t_2 \prec \cdots \prec t_l$ and $t^a \prec_K x^b$.

Definition 1. $\mathrm{HC}_R(f) \in R$ *denotes the head coefficient of f as an element of $R[x_1, \ldots, x_n]$ with respect to $\prec$. We call f monic if* $\mathrm{HC}_R(f) = 1$.

Let $\tilde{B} = \{g_1(x,t), \ldots, g_d(x,t)\}$ be a subset of T and $I \subset S$ an ideal generated by $B = \{g_1(x,\alpha), \ldots, g_d(x,\alpha)\}$. The following theorem is well known.

Theorem 1. *Let $\tilde{G}$ be the reduced* Gröbner *basis of $\tilde{I} = \langle \tilde{B} \cup D \rangle$ with respect to $\prec_K$. Then $(\tilde{G} \setminus D)|_{t=\alpha}$ is the reduced* Gröbner *basis of I with respect to $\prec$.*

By this theorem, we can apply Buchberger algorithm over $\mathbf{Q}$ to compute the Gröbner basis of I over K. However, if we observe the execution of Buchberger algorithm, we notice that many intermediate basis elements of the form $t^b x^a + \mathit{lower}$ are generated before a monic element $x^a + \mathit{lower}$ is generated. Once we have such a monic element, the intermediate basis elements become all redundant. This phenomenon is explained as follows. Suppose that $h(x,t) = t^b x^a + \mathit{lower} \in \tilde{I}$ is reduced with respect to D. We set $h_a(t) = \mathrm{HC}_R(h)$. Then $h_a(\alpha)$ is non-zero because $h_a(t)$ is reduced with respect to D. Then $h_a(\alpha)$ is invertible in K, which means that there exists $u(t) \in R$ such that $u(t)h_a(t) \equiv 1 \bmod J$. Then we have a monic element $u(t)h(x,t) = x^a + \mathit{lower} \in \tilde{I}$. That is, the intermediate elements are those which have intermediate polynomials generated during the computation of the inverse of $h_a(\alpha)$ as their head terms. This affects the process of computation in various ways. For example, each generated basis element produces new S-pairs. It is well known that the efficiency of Buchberger algorithm is sensitive to the order of S-pairs to be processed, and the new S-pairs may make the subsequent computation inefficient. Or, the inverse computation implicitly appeared in our case is nothing but Euclid algorithm, and it is well known that computing the inverse of an algebraic number by Euclid algorithm tends to cause coefficient swell. Our remedy for this difficulty is very simple. We modify Buchberger algorithm as follows. We omit the selection strategy and the criteria for useless pair detection in the algorithm description.

Algorithm 1 (Buchberger algorithm over an algebraic number field)

```
$L \leftarrow \{\{f, g\} \mid f, g \in \tilde{B}, f \neq g\}$
$G \leftarrow \tilde{B}$
while $L \neq \emptyset$ do
    $\{f, g\} \leftarrow$ an element of $L$; $L \leftarrow L \setminus \{\{f, g\}\}$
    $\tilde{r}(x, t) \leftarrow \mathrm{NF}_{D \cup G}(\mathrm{S}(f, g))$
    if $\tilde{r} \neq 0$ then
        $u(t) \leftarrow$ the inverse of $\mathrm{HC}_R(\tilde{r}) \bmod J$
        $r \leftarrow \mathrm{NF}_D(u\tilde{r})$
        $L \leftarrow L \cup \{\{f, r\} \mid f \in G\}$
        $G \leftarrow G \cup \{r\}$
    end if
end while
return $G$
```

In Algorithm 1, G consists of monic polynomials because each normal form is made monic before added to G, therefore $\tilde{r}(x, \alpha)$ is a normal form with respect to $G|_{x=\alpha}$. So Algorithm 1 executes Buchberger algorithm over K and it returns a Gröbner basis of $\langle B \rangle$. Algorithm 1 avoids generating redundant basis elements. Furthermore the trace algorithm[4] and the efficient content reduction [5] can be applied because the computation itself is done over $\mathbf{Q}$.

3 The Implementation

We implemented Algorithm 1 in Risa/Asir [1]. We already have an implementation of Buchberger algorithm over $\mathbf{Q}$ with various optimization (the trace algorithm, the homogenized trace algorithm and the efficient content reduction) and the only function to be newly implemented is an efficient inverse computation of algebraic numbers. In addition, the normal form computation with respect to D, the set of defining polynomials greatly affects the whole efficiency.

3.1 Simplification of Algebraic Numbers

The normal form computation with respect to D is equivalent to the simplification of algebraic numbers by the defining polynomials. As D is the reduced Gröbner basis with respect to the lexicographic order, the result of simplification does not depend on the order of monomial reductions. However its cost depends on the order.

Example 1. Set

$$m_1(t_1) = t_1^{20} + t_1^{19} + 2,$$
$$m_2(t_1, t_2) = t_2^{30} + (t_1^{19} + t_1^{18} + 1)t_2^{29} + 1.$$

$m_1(t_1)$ is irreducible over $\mathbf{Q}$. Let α_1 be a root of $m_1(t)$. Then $m_2(\alpha_1, t)$ is irreducible over $\mathbf{Q}(\alpha_1)$ and let α_2 be a root of $m_2(\alpha_1, t)$. Let us consider the simplification of α_2^{58}. If we always choose the reducer m_k with the smallest possible k during a simplification, it takes only 0.02 sec. But if we always choose m_k with the largest possible k, it takes 2 sec. In the latter strategy, the cost for simplification by m_2 is dominant. After simplifying all the occurrences of α_2^n for $n \geq 20$, the intermediate remainder contains 4674 monomials and its α_1 degree is $551 = 19 \times 29$, which is a kind of intermediate expression swell.

In our current implementation, the simplification is done by the following algorithm. Let $f \in R$ be a polynomial to be simplified.

Algorithm 2

```
$r \leftarrow 0$
*while* $f \neq 0$ *do*
    *if* $\mathrm{HT}(f)$ *is reduced with respect to* $D$ *then*
      $r \leftarrow r + \mathrm{HT}(f)$
      $f \leftarrow f - \mathrm{HT}(f)$
  *else*
      $k \leftarrow$ *the smallest* $k$ *such that* $\mathrm{HT}(m_k) | \mathrm{HT}(f)$
$(*)$    $f \leftarrow f - \mathrm{HC}(f) \cdot \frac{\mathrm{HT}(f)}{\mathrm{HT}(m_k)} m_k$
    *endif*
*end while*
*return* $r$
```

This is based on the following proposition.

Proposition 1. *Let* f_1 *be the right hand side of* $(*)$ *in Algorithm 2. Then we have*

$$\deg_{t_i}(f_1) \leq \begin{cases} \mathrm{MAX}(2(\deg_{t_i}(m_i) - 1), \deg_{t_i}(f)) & (i \leq k-1) \\ \deg_{t_i}(f) & (i \geq k) \end{cases}$$

Proof. By the property of k, $\deg_{t_i}(\mathrm{HT}(f)) \leq \deg_{t_i}(m_i) - 1$ for $i = 1, \ldots, k-1$ holds and we have

$$\deg_{t_i}(\frac{\mathrm{HT}(f)}{\mathrm{HT}(m_k)}(m_k)) \leq \begin{cases} 2(\deg_{t_i}(m_i) - 1) & (i \leq k-1) \\ \deg_{t_i}(\mathrm{HT}(f)) & (i \geq k) \end{cases}$$

because m_k is reduced with respect to m_i for $i \leq k-1$ and m_k does not contain t_i for $i \geq k+1$. This proves the assertion. □

In particular, if $f(t)$ and $g(t)$ is reduced with respect to D, then each t_i-degree of the intermediate reminders does not exceed $2(\deg_{t_i}(m_i)-1)$ during the execution of Algorithm 2 and the remainder computation is expected to be efficient.

3.2 Computation of the Inverse of an Algebraic Number

Even if the ground field K is a simple extension, the computation of the inverse of an algebraic number is not an easy task. Non-modular extended Euclid algorithms often cause coefficient swell and it seems better to apply modular methods. We can apply Hensel lifting or Chinese remainder theorem (CRT) to compute the inverse of $f(\alpha)$ for $f(t) \in R$. Let $M = \{s_1, \ldots, s_d\}$ be the set of monomials which spans $R/\langle D\rangle$ over $\mathbf{Q}$, where $d = \dim_{\mathbf{Q}} R/\langle D\rangle$. Then we can compute $g(t) \in R$ such that $g(\alpha)f(\alpha) = 1$ as follows.

Algorithm 3

```
Convert $\sum_i c_i \mathrm{NF}_G(fs_i) = 1$ into a system of linear equations $Ac = b$
with respect to $c = (c_1, \ldots, c_d)^T$.
$a = (a_1, \ldots, a_d)^T \leftarrow$ the solution of $Ac = b$
                              (Apply a modular method such as Hensel lifting or CRT.)
return $\sum_i a_i s_i$
```

A more detailed explanation can be found in [7].

3.3 The Homogenized Trace Algorithm

Homogenization is very useful for avoiding intermediate coefficient swell over $\mathbf{Q}$, but it also increases the number of S-polynomials reduced to zero. By combining homogenization and the trace algorithm, we can cut the additional cost introduced by homogenization. Let B be a set of polynomials and p a prime. Algorithm 4 is a general algorithm to compute a Gröbner basis candidate.

Algorithm 4 (`Candidate`(B, p))

```
$L \leftarrow \{\{f, g\} \mid f, g \in B, f \neq g\}$
$G \leftarrow B$; $G_p \leftarrow \phi_p(B)$
while $L \neq \emptyset$ do
    $\{f, g\} \leftarrow$ an element of $L$
    $L \leftarrow L \setminus \{\{f, g\}\}$
    if $\mathrm{NF}_{G_p}(\phi_p(f), \phi_p(g)) \neq 0$ then
      $r \leftarrow \mathrm{NF}_G(\mathrm{S}(f, g))$
      if $\phi_p(\mathrm{HC}(r)) = 0$ then return failure
      $L \leftarrow L \cup \{\{f, r\} \mid f \in G\}$
      $G \leftarrow G \cup \{r\}$; $G_p \leftarrow G_p \cup \{\phi_p(r)\}$
    end if
end while
return $G$
```

Let S_d be the set of S-polynomials whose total degrees are equal to d in an execution of Buchberger algorithm. For a homogeneous input, we process S_d in the increasing order with respect to d. Then, after processing all S_i $(i < d)$, S_d produces all the Gröbner basis elements of the total degree d. We observe that the total efficiency is improved if we execute an inter-reduction after processing

S_d. Furthermore, we also observe that inter-reductions during processing S_d often improves the efficiency. Note that such inter-reductions are allowed because what we are doing on S_d is nothing but the computation of a linear basis of the homogeneous component I_d of the total degree d of the input ideal I. In the current implementation, an inter-reduction is executed every after k new basis elements are generated, where k can be set by users and the default value is 6.

Algorithm 5 (Homogenized trace algorithm)

```
$B_h \leftarrow$ the homogenization of $B$
loop
    $p \leftarrow$ a new prime
    $G_h \leftarrow \mathtt{Candidate}(B_h, p)$
    if $G_h \neq$ failure then
        $G \leftarrow$ the dehomogenization of $G_h$
        $G \leftarrow \{g \in G \mid \mathrm{HT}(h) \not| \mathrm{HT}(g)$ for all $h \in G \setminus \{g\}\}$
        if $G$ is a Gröbner basis of $\langle G \rangle$ and $\mathrm{NF}_G(f) = 0$ for all $f \in B$ then return $G$
    endif
end loop
```

In Algorithm 5, a candidate of a Gröbner basis of $\langle B_h \rangle$ is computed, but the final check is done for the dehomogenized candidate. If B is not homogeneous, we usually have many redundant elements by dehomogenizing G_h. Except for finitely many p, G_h is a Gröbner basis of $\langle B_h \rangle$, in that case G is a Gröbner basis of $\langle B \rangle$. Then G is still a Gröbner basis after removing the redundant elements and the final checks are expected to be easy by the removal of redundancy [1].

Now we go back to our Gröbner basis computation over algebraic number fields. The normal forms in Algorithm 1 are computed over $\mathbf{Q}$, so we can apply Algorithm 5 by modifying Algorithm 4 as in Algorithm 1. We mention here the homogenization of elements in $\mathbf{Q}[x, t]$. We use the same notation as in Theorem 1. Algorithm 1 is essentially an algorithm over an algebraic number field K and it is natural to regard t-variables as parts of the coefficient field. Therefore, when we homogenize $\tilde{B}$ before entering Algorithm 4, the weights of t-variables, which are used to compute sugar in Algorithm 4, are set to 0. This setting seems natural, but it is not always optimal from the viewpoint of practical efficiency. This will be discussed later.

4 Experiments

4.1 Related Functions

We briefly explain Risa/Asir functions for Gröbner basis computation over algebraic number fields.

[1] In some cases, the check is hard because of the large coefficients of the final basis elements.

– `newalg`($DefPoly$) generates a root of $DefPoly$, where $DefPoly$ is a monic univariate polynomial whose coefficients are polynomials of already defined roots. That is, Risa/Asir can deal with multiple extension fields. Note that the system does not check whether $DefPoly$ is irreducible over the field generated over $\mathbf{Q}$ by the roots contained in the polynomial. The irreducibility can be checked by `af`.
– `af`($Poly, AlgList$) factorizes a univariate polynomial $Poly$ over an algebraic number field generated by roots listed in $AlgList$.
– `nd_gr_trace`($PolyList, VarList, Homo, Trace, Order$) computes the reduced Gröbner basis of $\langle PolyList\rangle \subset K[VarList]$ with respect to a term order specified by $Order$, where K is an algebraic number field generated over $\mathbf{Q}$ by all the roots appeared in $PolyList$. $Order = 0$ and $Order = 2$ mean the graded reverse lexicographic order (grlex) and the lexicographic order (lex) respectively. If $Trace = Homo = 1$, Algorithm 5 is executed.This function implements various improvements such as vectorized length exponent vector [9] and geobucket addition [8].

Remark 1. For comparison we also implemented a function which computes Gröbner bases over algebraic number fields by using addition, subtraction and multiplication over algebraic number fields. In this implementation, each normal form is appended to the intermediate basis without being made monic. Then we have to multiply the polynomial to be reduced by an algebraic number in each reduction step, which makes the coefficients larger. Furthermore, the trace algorithms cannot be applied because the ground field is an algebraic number field. Thus all the computations have to be done over the algebraic number field. We applied this function to several examples which are easily computed by `nd_gr_trace` and we found this function is useless for our purpose.

Example 2. Univariate polynomial GCD can be computed by Gröbner basis computation. In the following Risa/Asir session, `A1` is a root of `M1`. Then `M2` is defined over $\mathbf{Q}$(`A1`) and `af` tells that it is irreducible over $\mathbf{Q}$(`A1`). `A2` is a root of `M2`. `F1` and `F2` are polynomials over $\mathbf{Q}$(`A1`, `A2`) and GCD(`F1`, `F2`) is computed by `nd_gr_trace`.

```
[0] load("sp")$
[101] M1=t1^6+6*t1^4+2*t1^3+9*t1^2+6*t1-4$
[102] A1=newalg(M1);
(#0)
[103] M2=t^3+3*t+A1^3+3*A1+2$
[104] af(M2,[A1]);
[[t^3+3*t+(#0^3+3*#0+2),1]]
[105] A2=newalg(M2);
(#1)
[106] F1 = x^6+(-6*A2+3*A1)*x^5+(15*A2^2-15*A1*A2+6*A1^2+27)*x^4
+(-20*A2^3+30*A1*A2^2+(-24*A1^2-108)*A2+7*A1^3+54*A1)*x^3+(15*A2^4
-30*A1*A2^3+(36*A1^2+162)*A2^2+(-21*A1^3-162*A1)*A2-27*A1^5+6*A1^4
-135*A1^3-216*A1)*x^2+(-6*A2^5+15*A1*A2^4+(-24*A1^2-108)*A2^3
+(21*A1^3+162*A1)*A2^2+(54*A1^5-12*A1^4+270*A1^3+432*A1)*A2+3*A1^5
+27*A1^4+27*A1^3+27*A1^2+162*A1-108)*x+(A2^6-3*A1*A2^5+(6*A1^2+27)
*A2^4+(-7*A1^3-54*A1)*A2^3+(-27*A1^5+6*A1^4-135*A1^3-216*A1)*A2^2
+(-3*A1^5-27*A1^4-27*A1^3-27*A1^2-162*A1+108)*A2-54*A1^5-6*A1^4
-218*A1^3-90*A1^2-276*A1+220)$
[107] F2 = x^4+(2*A2+2*A1)*x^3+(3*A2^2+3*A1*A2+3*A1^2+12)*x^2
+(3*A1*A2^2+(3*A1^2+6)*A2+6*A1-4)*x+(-2*A2-2*A1)$
[108] G=nd_gr_trace([F1,F2],[x],1,1,0);
[20*x^2+(20*#1+20*#0)*x+((-6*#0^5+3*#0^4-30*#0^3+3*#0^2-48*#0+8)
*#1^2+(3*#0^5+6*#0^4+15*#0^3+36*#0^2+34*#0+36)*#1-12*#0^5+6*#0^4
-60*#0^3+26*#0^2-96*#0+96)]
```

4.2 Timings

In the following examples, Algorithm 5 is applied over an algebraic number field or **Q**. Timings were measured on a PC with Intel Xeon 3.4GHz. In the tables, “total”, “#basis”, “check”, “monic” mean the total time, the number of the intermediate basis elements, the time for checking the Gröbner basis candidate and the time for making the normal forms monic. The last two are included in the total time. Timings are shown in seconds.

Example 3. Let C_7 be Cyclic-7 system with variables $c_1, \dots, c_7$. We compute the reduced Gröbner basis of $C_{7,\omega} = C_7|_{c_7=\omega}$ with respect to grlex order for $c_1 \succ \dots \succ c_6$, where ω is a root of an irreducible factor $m(c_7)$ of the minimal polynomial of c_7 in $\mathbf{Q}[c_1, \dots, c_7]/\langle C_7 \rangle$.

1. $m(c_7) = c_7^6 + c_7^5 + c_7^4 + c_7^3 + c_7^2 + c_7 + 1$
2. $m(c_7) = c_7^2 + 5c_7 + 1$
   The Gröbner basis is very simple:
   $$\langle c_2 - 1, c_3 - 1, c_4 - 1, c_5 - 1, c_1 + c_6 + \omega + 4, c_6^2 + (\omega + 4)c_6 - \omega - 5 \rangle,$$
   but the computation is hard.
3. $m(c_7) = c_7^{12} - 5c_7^{11} + 24c_7^{10} - 115c_7^9 + 551c_7^8 - 2640c_7^7 + 12649c_7^6 - 2640c_7^5 + 551c_7^4 - 115c_7^3 + 24c_7^2 - 5c_7 + 1$

Table 1. Gröbner computations of $C_{7,\omega}$

	total	#basis	check	monic
1 over $\mathbf{Q}(\omega)$	9.3	268	0.2	1.4
1 over $\mathbf{Q}$	74	588	–	–
2 over $\mathbf{Q}(\omega)$	198	306	0	83
2 over $\mathbf{Q}$	119	675	–	–
3 over $\mathbf{Q}(\omega)$	256	306	0	128
3 over $\mathbf{Q}$	840	857	–	–

Example 4.

$$m_1(t_1) = t_1^7 - 7t_1 + 3,$$
$$m_2(t_1,t_2) = t_2^6 + t_1t_2^5 + t_1^2t_2^4 + t_1^3t_2^3 + t_1^4t_2^2 + t_1^5t_2 + t_1^6 - 7.$$

$m_1(t_1)$ is irreducible over $\mathbf{Q}$. Let α_1 be a root of $m_1(t)$. $m_2(\alpha_1, t)$ is an irreducible factor of $m_1(t)$ over $\mathbf{Q}(\alpha_1)$ and let α_2 be a root of $m_2(\alpha_1, t)$.

$Cap = \{f_1, f_2, f_3, f_4\}$

$f_1 = (2ty - 2)x - (\alpha_1 + \alpha_2)zy^2 - z$

$f_2 = 2\alpha_2\alpha_1^4zx^3 + (4ty + \alpha_2)x^2 + (4zy^2 + 4z)x + 2ty^3 - 10y^2 - 10ty + 2\alpha_1^2 + \alpha_2^2$

$f_3 = (t^2 - 1)x + (\alpha_2\alpha_1^4 + \alpha_2^3\alpha_1^3)tzy - 2z$

$f_4 = (-z^2 + 4t^2 + \alpha_2\alpha_1 + 2\alpha_2^3)zx + (4tz^2 + 2t^3 - 10t)y + 4z^2 - 10t^2 + \alpha_2\alpha_1^3$

Cap is constructed from *Caprasse* [2] by replacing its several coefficients by algebraic numbers in $\mathbf{Q}(\alpha_1, \alpha_2)$. In the computation over $\mathbf{Q}(\alpha_1, \alpha_2)$, almost all the

Table 2. Gröbner computations of *Cap*

	total	#basis	check	monic
over $\mathbf{Q}(\alpha_1, \alpha_2)$	306	45	242	20
over $\mathbf{Q}$	$>$ 1hour	—	–	–

time is spent on the check of the Gröbner basis candidate because the result contains many algebraic numbers with large integer coefficients. The computation over $\mathbf{Q}$ does not terminate within one hour.

4.3 Discussion and Future Works

The results of our experiments show not only an advantage of our new method, but also show that further improvements are required. In Example 3-2, the computation via Theorem 1 is more efficient than the new method. We apply homogenization in both computations, but the results of homogenization differ according to the different settings of the weights, which affects the selection strategy and eventually the behaviors of the computations. The inefficiency of the new method in this example comes from large intermediate integer coefficients. They are still large even if we execute inter-reductions frequently, but they become small by the last inter-reduction executed after all S-pairs having the same total degree have been processed. Therefore it is possible to improve the new method if we have an efficient F_4-like implementation.

The current implementation requires that the each root is defined as a root of an irreducible polynomial and it is often hard to check the irreducibility. If we apply Dynamic Evaluation [3] we can weaken the requirement: the ideal generated by the defining polynomials is required to be only zero-dimensional and radical. Then the leading coefficient $\mathrm{HC}_R(\tilde{r})$ in Algorithm 1 is not necessarily invertible. If it is not invertible, we can split the ground ring by using the non-invertible element and the execution of Buchberger algorithm itself is split. Suppose that the ground ring R is represented as $\mathbf{Q}[t_1, \ldots, t_l]/J$ with a zero-dimensional radical ideal J. The following algorithm computes a set of pairs (J_i, r_i) such that $R = \oplus_i R_i$, $R_i = \mathbf{Q}[t_1, \ldots, t_l]/J_i$, $r_i \in R_i[x_1, \ldots, x_n]$ and $r_i = c_i\tilde{r}$ where c_i is the inverse of $\mathrm{HC}_{R_i}(\tilde{r})$ in R_i, thus r_i is monic.

Algorithm 6 (SPLIT_GROUND_RING($J, \tilde{r}$))

```
loop
    $h \leftarrow \mathrm{HC}_R(\tilde{r})$
    if $h \bmod J$ is invertible then
        $u(t) \leftarrow$ the inverse of $h \bmod J$
        $r \leftarrow \mathrm{NF}_D(u\tilde{r})$
        return $\{(J, r)\}$
    else
        $J_1 \leftarrow J : h$
        $J' \leftarrow J + \langle h \rangle$
        $u_1(t) \leftarrow$ the inverse of $h \bmod J_1$ ($h \bmod J_1$ is invertible in $J_1$)
        $r_1 \leftarrow \mathrm{NF}_D(u_1\tilde{r})$
        $r' \leftarrow \tilde{r} \bmod J'$
        $S' \leftarrow$SPLIT_GROUND_RING($J', r'$)
        return $\{(J_1, r_1)\} \cup S'$
    end if
end loop
```

By applying a new modular method proposed in [7], we can quickly check the invertibility of $h \bmod J$ and compute $J : h$ and $J + \langle h \rangle$ if $h \bmod J$ is not invertible. We plan to implement this method in near future.

References

1. Risa/Asir : A computer algebra system. `http://www.math.kobe-u.ac.jp/Asir/asir.html`.
2. SymbolicData. `http://www.SymbolicData.org`.
3. J. Della Dora, C. Discrescenso, D. Duval. About a new method for computing in algebraic number fields, In Proc. Eurocal'85 (LNCS 204), Springer-Verlag, 289-290, 1985.
4. C. Traverso. Gröbner trace algorithms. In Proc. ISSAC'88, LNCS **358**, Springer-Verlag 125-138, 1988.
5. M. Noro and J. McKay. Computation of replicable functions on Risa/Asir. In Proceedings of the Second International Symposium on Symbolic Computation PASCO'97, ACM Press, 130-138, 1997.
6. J. C. Faugère. A new efficient algorithm for computing Groebner bases (F_4). Journal of Pure and Applied Algebra (139) 1-3, 61-88, 1999.

7. M. Noro. Modular Dynamic Evaluation. To appear in Proc. ISSAC 2006, 2006.
8. T. Yan. The Geobucket Data Structure for Polynomials. J. Symb. Comp. 25,3, 285-294, 1998.
9. H. Schönemann. SINGULAR in a Framework for Polynomial Computations. M. Joswig and N. Takayama (eds.), Algebra, Geometry, and Software Systems, Springer, 163-176, 2003.