Multi-modular Algorithm for Computing the Splitting Field of a Polynomial

Guénaël Renault, Kazuhiro Yokoyama

Multi-modular Algorithm for Computing the Splitting Field of a Polynomial

Guénaël Renault
INRIA, Paris-Rocquencourt, SALSA Project
UPMC, Univ. Paris 06, LIP6
CNRS, UMR 7606, LIP6
UFR Ingéniérie 919, LIP6 Passy-Kennedy,
Case 169, 4, Place Jussieu, F-75252 Paris
guenael.renault@lip6.fr

Kazuhiro Yokoyama
Rikkyo University
3-34-1 Nishi Ikebukuro, Toshima-ku
Tokyo 171-8501, Japan
yokoyama@rkmath.rikkyo.ac.jp

ABSTRACT

Let f be a univariate monic integral polynomial of degree n and let $(\alpha_1, \ldots, \alpha_n)$ be an n-tuple of its roots in an algebraic closure $\bar{\mathbb{Q}}$ of $\mathbb{Q}$. Obtaining an algebraic representation of the splitting field $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ of f is a question of first importance in effective Galois theory. For instance, it allows us to manipulate symbolically the roots of f. In this paper, we propose a new method based on multi-modular strategy. Actually, we provide algorithms for this task which return a triangular set encoding the *splitting ideal* of f. We examine the ability/practicality of the method by experiments on a real computer and study its complexity.

Categories and Subject Descriptors

I.1 [**Computing Methodologies**]: Symbolic and algebraic manipulations

General Terms

Algorithms, Theory

Keywords

Galois theory, splitting field

1. INTRODUCTION

In [18] the authors proposed an approach for computing the splitting field of a monic integral polynomial f. This approach is based on *indeterminate coefficients* strategy and Hensel lifting. It takes as input the action of the Galois group of f over approximation of roots of f in a p-adic number field $\mathbb{Q}_p$ (or one of its extensions). To compute the Galois group G_f of f over $\mathbb{Q}$, the approach of p-adic approximation is very practical and efficient (see [21, 10, 9]).

In the approach of [18], the authors did not use all the data obtained during Galois group computation. Also, on some particular examples, the method was not so efficient. By experimentations, we discovered that using these data is a very low time consuming. Thus, in order to use a large part of these data and obtain a compromise for this approach we propose here a multi-modular approach for computing the splitting field of f. Moreover, the recent version 2.13 of the *computer algebra system* MAGMA [5] provides a new implementation (by Fieker and Klüners) of the Galois group computation based on p-adic approximations where it becomes easy to access the data computed during this procedure. Also, since the new computer architectures are now based on multi-core processors it is important to study new algorithms which can benefit from these new features. For all these reasons, a multi-modular strategy has to be studied.

The key of the multi-modular strategy proposed here comes from the following assertion: From data obtained during the computation of the permutation action of G_f over approximate roots of f modulo a prime p_1, we can easily obtain the action of the same permutation representation over approximate roots of f modulo another prime p_2 (see Section 3).

From this action over approximate roots of f modulo different primes we compute approximations, modulo the same primes, of the Gröbner basis $\mathcal{G}$ of the splitting ideal $\mathcal{M}$, that is the ideal of all the algebraic relations of the roots of f. Then we reconstruct it by *Chinese Remainder Theorem* (See Section 4). Thus, the splitting field of f is given by $\mathbb{Q}[x_1, ..., x_n]/\mathcal{M}$; let us remark that it is easy to perform arithmetic operations in this algebra. Moreover, in general, expressions by primitive elements tend to suffer "expression swell", that is, huge coefficients appear and those harm the efficiency. So, for our purpose, simple extension does not seem suited.

In order to compute the approximate projections of $\mathcal{M}$ in different p-adic fields, we use the knowledge of certain algebraic structures, the action of G_f over the p-adic approximation of roots of f and a theoretical form of $\mathcal{G}$ given by the corresponding *computation scheme*, a very useful object introduced in [18] (see also [17]). The computation scheme gives sparse forms with indeterminate coefficients for the polynomials in $\mathcal{G}$ and techniques to avoid some computations (see Section 3.3). In Section 3 we show how to interpolate these sparse form by adaptation of the formulae given in [6] and modular computations. From these theoretical forms we deduce, in the same section, the best bounds

in our knowledge for the coefficients of a basis of $\mathcal{M}$.

In Section 4 we give basic discussion on efficiency of multi-modular strategy in a general form, and present concrete algorithms for our subject corresponding to this multi-modular strategy with effective tests for *correctness*. Then, based on the algorithms, we give certain results about the multi-modular strategy's theoretical efficiency. Section 5 is devoted to the experiments, by which the practicality of the multi-modular strategy is examined.

2. PRELIMINARIES

We provide necessary notions and summarize some results of [21] and [18].

2.1 Splitting Field and Galois Group over $\mathbb{Q}$

Let $f(x)$ be a monic square-free integral polynomial of degree n and $\underline{\alpha}$ the set of all its roots in an algebraic closure $\bar{\mathbb{Q}}$ of $\mathbb{Q}$. The splitting field K_f of f is the extension field $\mathbb{Q}(\underline{\alpha})$ obtained by adjoining $\underline{\alpha}$ to $\mathbb{Q}$. The group G_f of $\mathbb{Q}$-automorphisms of K_f acts faithfully on $\underline{\alpha}$, thus one can consider the permutation representation G_f of this group. Fixing a numbering of the roots $\underline{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$ of f, G_f is viewed as a subgroup of S_n. The group G_f is called the Galois group of f.

To express K_f symbolically, we consider the epimorphism $\phi : \mathbb{Q}[x_1, \ldots, x_n] \ni x_i \longmapsto \alpha_i \in K_f$ of $\mathbb{Q}$-algebras. For simplicity, we write $X = \{x_1, \ldots, x_n\}$. Then K_f is represented by the residue class ring $\mathcal{A}$ of the polynomial ring $\mathbb{Q}[X]$ factored by the kernel $\mathcal{M}$ of ϕ. We call $\mathcal{M}$ *the splitting ideal of f associated with the assignment of the roots $\alpha_1, \ldots, \alpha_n$*. In this setting, computing K_f means to compute a *Gröbner basis* $\mathcal{G}$ of $\mathcal{M}$ (see [4]). Now we fix the lexicographic order $\prec$ on terms with $x_1 \prec \cdots \prec x_n$, then the reduced Gröbner basis of $\mathcal{M}$ coincides with the generating set $\{g_1, g_2, \ldots, g_n\}$ obtained by *successive extensions*, that is, for each i,

1. g_i is a polynomial in $x_1, \ldots, x_i$ and monic with respect to x_i, and
2. $\mathbb{Q}(\alpha_1, \ldots, \alpha_i) \cong \mathbb{Q}[x_1, \ldots, x_i]/\langle g_1, \ldots, g_i \rangle$,

where $\langle F \rangle$ denotes the ideal generated by an element or a set F. This implies that g_i is an irreducible factor of $f(x_i)$ over $\mathbb{Q}[x_1, \ldots, x_{i-1}]/\langle g_1, \ldots, g_{i-1} \rangle$ such that $g_i(\alpha_1, \ldots, \alpha_i) = 0$. Thus this reduced Gröbner basis can be obtained by *"algebraic factoring methods"* (see [3]) and is said to be a *triangular basis* (see [12, 6]). For a Gröbner basis $\mathcal{G} \subset \mathbb{Q}[X]$ and a polynomial P, let $\mathrm{NF}(P, \mathcal{G})$ denote the normal form of P in $\mathbb{Q}[X]$ with respect to $\mathcal{G}$ (see [4]).

The group S_n acts naturally on $\mathbb{Q}[X]$ with $x_i^\sigma = x_{i^\sigma}$ for $1 \leqslant i \leqslant n$ and $\sigma \in S_n$. Thus G_f is the $\mathbb{Q}$-automorphisms group of $\mathcal{A}$ denoted by $\mathrm{Aut}_{\mathbb{Q}}(\mathcal{A})$ (see [3, 1]). We use the following notation for groups: For a group G acting on a set $\mathcal{S}$, the stabilizer in G of an element or a subset A of $\mathcal{S}$ is denoted by $\mathrm{Stab}_G(A)$, i.e. $\mathrm{Stab}_G(A) = \{\sigma \in G \ : \ A^\sigma = A\}$. If G is the full symmetric group on $\mathcal{S}$, we simply write $\mathrm{Stab}(A)$ for $\mathrm{Stab}_G(A)$. We denote by $\mathrm{Stab}_G([a_1, \ldots, a_k])$ the point-wise stabilizer of a subset $A = \{a_1, \ldots, a_k\}$ of $\mathcal{S}$, i.e. $\mathrm{Stab}_G([a_1, \ldots, a_k]) = \{\sigma \in G \mid a_i^\sigma = a_i, \ \forall i \in \{1, \ldots, k\}\}$. The set of right cosets of H in G is denoted by $H\backslash G$ and the set of all representatives of $H\backslash G$ by $H\backslash\backslash G$.

Definition 1. We call the ideal generated by the polynomials $t_1 + a_1, \ldots, t_n + (-1)^{n-1} a_n$, where t_i is the i-th elementary symmetric function on X and $f(x) = x^n + a_1 x^{n-1} + \cdots + a_n$, the *universal splitting ideal* of f and denote it by $\mathcal{M}_0$. We call the residue class ring $\mathbb{Q}[X]/\mathcal{M}_0$ *the universal splitting ring of f over* $\mathbb{Q}$ and denote it by $\mathcal{A}_0$.

With respect to the fixed order $\prec$, the reduced Gröbner basis of $\mathcal{M}_0$ is composed of the n *Cauchy's modules* of f (see [19]) and it is called the *standard generating set*. Since S_n stabilizes $\mathcal{M}_0$, S_n also acts faithfully on $\mathcal{A}_0$, i.e. $S_n \subset \mathrm{Aut}_{\mathbb{Q}}(\mathcal{A}_0)$. We have the following theorem (see [16, 2, 21] for details and other references).

THEOREM 2.1. *There is a one-to-one correspondence between the set of all primitive idempotents of $\mathcal{A}_0$ and the set of all prime divisors of $\mathcal{M}_0$. Let m be the primitive idempotent corresponding to the fixed prime divisor $\mathcal{M}$. Then, $G_f = Stab(\mathcal{M}) = Stab(m)$ and $\mathcal{M}^\sigma = \{g \in \mathbb{Q}[X] \mid gm^\sigma = 0 \text{ in } \mathcal{A}_0\}$. Moreover, we have $\mathcal{M}_0 = \cap_{\sigma \in G_f \backslash\backslash S_n} \mathcal{M}^\sigma$ and $\mathcal{A}_0 = \oplus_{\sigma \in G_f \backslash\backslash S_n} m^\sigma \mathcal{A}_0 = \oplus_{\sigma \in G_f \backslash\backslash S_n} \mathbb{Q}[X]/\mathcal{M}^\sigma$.*

2.2 Splitting Field and Modular Computation

Now we consider the relation between the splitting ring over $\mathbb{Q}$ and that over a p-adic field $\mathbb{Q}_p$. The n-tuple $\underline{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$ and the splitting ideal $\mathcal{M}$ associated with the assignment x_i to α_i are fixed. The primitive idempotent of $\mathcal{A}_0$ corresponding to $\mathcal{M}$ is denoted by m. For a prime integer p, we denote by $\mathbf{Z}_p^0$ (resp. $\mathbf{Z}_p$) the localization of $\mathbf{Z}$ at p (resp. the completion of $\mathbf{Z}_p^0$). We denote by π_p the projection from $\mathbf{Z}_p[X]$ to $\mathbb{F}_p[X]$ (the natural extension of the projection from $\mathbf{Z}$ to $\mathbb{F}_p$). From now on, we will consider prime numbers p satisfying the following property:

$$\mathbb{P} : \quad \pi_p(f) \text{ is square-free.}$$

Let $\bar{\mathcal{M}}_0^p$ denote the ideal $\pi_p(\mathcal{M}_0 \cap \mathbf{Z}_p^0[X])$ in $\mathbb{F}_p[X]$ and $\mathcal{G}_0$ denote the standard generating set of $\mathcal{M}_0$. By construction, the Cauchy's modules of f are polynomials with integral coefficients and monic in their greatest variable. Thus, the set $\pi_p(\mathcal{G}_0)$ is a Gröbner basis of $\bar{\mathcal{M}}_0^p$. Moreover, $\mathcal{G}_0$ is a Gröbner basis of the universal splitting ideal $\mathbb{Q}_p \otimes_{\mathbb{Q}} \mathcal{M}_0$ of f as a polynomial with coefficients in $\mathbb{Q}_p$ and that of $\mathbf{Z}_p[X] \otimes_{\mathbf{Z}_p^0} (\mathcal{M}_0 \cap \mathbf{Z}_p^0[X])$ over $\mathbf{Z}_p$. The ideal $\mathbb{Q}_p \otimes_{\mathbb{Q}} \mathcal{M}_0$ is denoted by $\mathcal{M}_0^{(p,\infty)}$. We denote $\mathbb{F}_p[X]/\bar{\mathcal{M}}_0^p$ by $\bar{\mathcal{A}}_0^p$ and $\mathbb{Q}_p[X]/\mathcal{M}_0^{(p,\infty)}$ by $\mathcal{A}_0^{(p,\infty)}$. We have the following result (see [21, 18]).

THEOREM 2.2. *We have the following assertions:*
1. The projection π_p gives a one-to-one correspondence between the set of all primitive idempotents of $\mathcal{A}_0^{(p,\infty)}$ and that of $\bar{\mathcal{A}}_0^p$. Moreover, for each pair $(\bar{m}^{(p)}, m^{(p,\infty)})$ of corresponding primitive idempotents, $Stab(\bar{m}^{(p)}) = Stab(m^{(p,\infty)})$.
2. The idempotent m of $\mathcal{A}_0$ is also an idempotent of $\mathcal{A}_0^{(p,\infty)}$. Let $\bar{m}^{(p)}$ be a component of $\pi_p(m)$ and $m^{(p,\infty)}$ the primitive idempotent of $\mathcal{A}_0^{(p,\infty)}$ corresponding to $\bar{m}^{(p)}$. Then $Stab(m)$ contains $Stab(\bar{m}^{(p)})$ $(= Stab(m^{(p,\infty)}))$ and $Stab(\pi_p(m)) = Stab(m)$. Moreover, if we denote $Stab(\bar{m}^{(p)})\backslash\backslash Stab(m)$ by $\mathcal{S}$ then $\pi_p(m) = \sum_{\sigma \in \mathcal{S}} \bar{m}^\sigma$ and $m = \sum_{\sigma \in \mathcal{S}} m^{(p,\infty)^\sigma}$.

Now we fix a component $\bar{m}^{(p)}$ of $\pi_p(m)$ and its corresponding idempotent $m^{(p,\infty)}$ of $\mathcal{A}_0^{(p,\infty)}$. Let $\bar{\mathcal{M}}^p$ be the maximal ideal of $\mathbb{F}_p[X]$ corresponding to $\bar{m}^p$ and $\mathcal{M}^{(p,\infty)}$ the maximal ideal of $\mathbb{Q}_p[X]$ corresponding to $m^{(p,\infty)}$. Moreover, let $\mathcal{G}^{(p,\infty)}$ and $\bar{\mathcal{G}}^{(p)}$ be the reduced Gröbner basis of $\mathcal{M}^{(p,\infty)}$ and that of $\bar{\mathcal{M}}^p$ respectively.

Definition 2. Let $\mathcal{G}^{(p,\infty)} = \{g_1^{(p,\infty)}, \ldots, g_n^{(p,\infty)}\}$. For a positive integer k, we call the polynomials set $\{g_1^{(p,\infty)} \bmod p^{k+1}, \ldots, g_n^{(p,\infty)} \bmod p^{k+1}\}$ the *k-th approximation to* the basis $\mathcal{G}^{(p,\infty)}$ and denote it by $\mathcal{G}^{(p,k)}$. Note that $\mathcal{G}^{(p,0)} = \bar{\mathcal{G}}^{(p)}$.

Approximations of the roots of f

The Gröbner basis $\bar{\mathcal{G}}^{(p)}$ can be lifted to $\mathcal{G}^{(p,\infty)}$ by Hensel construction based on *quadratic iteration*.

THEOREM 2.3. *[21, 18] The reduced Gröbner basis $\mathcal{G}^{(p,\infty)}$ of the ideal $\mathcal{M}^{(p,\infty)}$ with respect to $\prec$ is contained in $\mathbf{Z}_p[X]$, and $\bar{\mathcal{G}}^{(p)}$ is lifted uniquely to $\mathcal{G}^{(p,\infty)}$ by Hensel construction.*

REMARK 3. *From $\bar{\mathcal{G}}^{(p)}$, we can construct the approximate Gröbner basis $\mathcal{G}^{(p,k)}$ for any integer k. As soon as we have $\mathcal{G}^{(p,k)}$, we can compute with the roots of f in $\mathbb{Z}/p^{k+1}\mathbb{Z}$ by computing normal forms modulo this basis. Thus, in the sequel, the expression* approximations of the roots of f *modulo p^{p+1} will mean that we have such a Gröbner basis $\mathcal{G}^{(p,k)}$*

REMARK 4. *As to the Gröbner basis $\mathcal{G}$, the denominators of its elements are related to the* discriminant $d(f)$ *of f. (See Section 3.4.) For each prime p satisfying the property $\mathbb{P}$, the square-freeness of $\pi_p(f)$ implies $\pi_p(d(f)) \neq 0$, and thus, it follows that $\pi_p(\mathcal{G})$ is well-defined, that is, p does not divide any of the denominators.*

Now we will study the construction of $\mathcal{G}$ by *Chinese Remainder Theorem*.

3. MODULAR CONSTRUCTION OF $\mathcal{G}$

In this section, we fix a *splitting ideal* $\mathcal{M}$ of f, the corresponding idempotent m and its stabilizer, the Galois group G_f represented as a sub-group of S_n. We denote by $Z_{\mathbb{K}}(I)$ the algebraic variety over a field $\mathbb{K}$ associated to an ideal I in a polynomial ring. Let $E = \{e_1 < \cdots < e_s\}$ be a subset of $\{1, \ldots, n\}$ and $\underline{\gamma} = (\gamma_1, \ldots, \gamma_n)$ an element of $Z_{\bar{\mathbb{Q}}}(\mathcal{M})$. We denote by $\underline{\gamma}(E)$ the projection of γ on the indexes given by E (i.e. $(\gamma_{e_1}, \ldots, \gamma_{e_s})$) and $Z_{\bar{\mathbb{Q}}}(\mathcal{M})(E) = \{\underline{\gamma}(E) : \underline{\gamma} \in Z_{\bar{\mathbb{Q}}}(\mathcal{M})\}$.

3.1 Approximation of $\mathcal{G}$

Let $\mathcal{G} = \{g_1, \ldots, g_n\}$ be the Gröbner basis of $\mathcal{M}$, we will describe here how to compute the coefficients of polynomials g_i's by indeterminate coefficient strategy and multi-modular computation.

More precisely, by the knowledge of a special subset E_i of $\{1, \ldots, n\}$, we can deduce an equation which defines the polynomial g_i (see Section 3.3):

$$g_i(\underline{\gamma}) = 0 \text{ for every } \underline{\gamma} \in Z_{\bar{\mathbb{Q}}}(\mathcal{M})(E_i). \tag{3.1}$$

We can replace the variety $Z_{\bar{\mathbb{Q}}}(\mathcal{M})(E_i)$ by $Z_{\bar{\mathbb{Q}}_p}(\mathbb{Q}_p \otimes_{\mathbb{Q}} \mathcal{M})(E_i)$, where p is a prime satisfying $\mathbb{P}$. Then, by using approximations of roots, we obtain the same equation but for the approximation of g_i. To do that, we need to recall some results (see [21]).

Let p be a prime integer satisfying $\mathbb{P}$, $\bar{m}^{(p)}$ a component of $\pi_p(m)$, $\bar{\mathcal{M}}^p$ its corresponding maximal ideal of $\mathbb{F}_p[X]$ and $\mathcal{M}^{(p,\infty)}$ its corresponding maximal ideal of $\mathbb{Q}_p[X]$ which is a divisor of $\mathbb{Q}_p[X] \otimes_{\mathbb{Q}} \mathcal{M}$.

PROPOSITION 3.1. *Let $\mathcal{S} = Stab(\bar{m}^{(p)}) \backslash\backslash Stab(m)$. Then*

$$\mathbb{Q}_p \otimes_{\mathbb{Q}} \mathcal{M} = \cap_{\sigma \in \mathcal{S}} (\mathcal{M}^{(p,\infty)})^\sigma, \text{ and } \pi_p(\mathcal{M} \cap \mathbf{Z}_p^0) = \cap_{\sigma \in \mathcal{S}} (\bar{\mathcal{M}})^\sigma.$$

By Proposition 3.1, we can reduce the equation (3.1) to the following.

$$\mathrm{NF}(g_i, (\mathcal{G}^{(p,\infty)})^\sigma) = 0 \text{ for every } \sigma \in G_{E_i} \backslash\backslash G_f, \tag{3.2}$$

where $G_f = \mathrm{Stab}(m)$ and G_{E_i} denotes $\mathrm{Stab}([\alpha_{e_{i,1}}, \ldots, \alpha_{e_{i,t}}])$ for $E_i = \{e_{i,1}, \ldots, e_{i,t}\}$. Because $\{(\mathcal{G}^{(p,\infty)})^\sigma : \sigma \in G_{E_i} \backslash\backslash G_f\}$ provides all elements of $Z_{\bar{\mathbb{Q}}_p}(\mathbb{Q}_p \otimes_{\mathbb{Q}} \mathcal{M})(E_i)$. Moreover, replacing $\mathcal{G}^{(p,\infty)}$ with $\mathcal{G}^{(p,k)}$ (see Remark 3), we have the following equation that approximation $g_i \bmod p^{k+1}$ must satisfy.

$$\mathrm{NF}(g_i, (\mathcal{G}^{(p,k)})^\sigma) \equiv 0 \pmod{p^{k+1}} \ \forall \sigma \in G_{E_i} \backslash\backslash G_f. \tag{3.3}$$

REMARK 5. *The components of $\pi_p(m)$ are conjugate to each other by the action of $G_f = \mathrm{Stab}(m)$. From this fact, it follows that any choice of $\bar{m}^{(p)}$ from the components of $\pi_p(m)$ still give the same Gröbner basis $\mathcal{G} = \{g_1, \ldots, g_n\}$.*

In [18] the authors presented a linear system resolution to compute $g_i \bmod p^{k+1}$. Here, we will present in a further section how to do this by interpolation. Before that, we present how to compute an approximation of g_i modulo an integer M by *Chinese Remainder Theorem*.

3.2 Chinese Remainder Construction

Recall that m is the idempotent of $\mathcal{A}_0$ corresponding to the fixed splitting ideal $\mathcal{M}$ corresponding to the specific roots ordering. We now consider primes $p_1, \ldots, p_i$ satisfying the property $\mathbb{P}$ and a component $\bar{m}^{(p_i)}$ of $\pi_{p_i}(m)$ for each p_i. Let $\mathcal{G}^{(p_1,k_1)}, \ldots, \mathcal{G}^{(p_i,k_i)}$ the approximate Gröbner bases corresponding to these components.

As seen in Section 3.1, we can approximate the polynomials g_i modulo each $p_i^{k_i}$. Thus, by *Chinese Remainder Theorem* we lift them in the ring $\mathbb{Z}/M\mathbb{Z}$ where $M = \prod_{j=1}^{i} p_j^{k_j}$. From this computation we obtain the projection of $\mathcal{G}$ modulo M, that is the set of polynomials $\{g_1 \bmod M, \ldots, g_n \bmod M\}$. But, in all this computation we assume that the idempotent m is fixed. In practice, we can not assume this hypothesis. Thus we need a general method to assure that each component $\bar{m}^{(p_j)}$ will correspond to the same idempotent m. (See Remark 5.)

To do that, we will use data produced during the computation of the Galois group G_f done modulo $p_1^{k_1}$. Then we reorder the roots modulo primes $p_2, \ldots, p_i$ by following criteria obtained from the data. From the computation of G_f [21, 10], we obtain a finite sequence $\{(I_i, A_i) : i = 1, \ldots, t\}$ of *invariants* and their *integer evaluation* modulo $p_1^{k_1}$ such that $NF(I_i - A_i, \mathcal{G}^{(p_1,k_1)}) \equiv 0 \pmod{p_1^{k_1+1}}$ for every i. Since $p_1^{k_1}$ exceeds the computed theoretical bound so that the corresponding relative Lagrange resolvent has A_i as its simple integral root. This implies $(I_i - A_i)m = 0$ for every i and $NF(g_i, \mathcal{G}^{(p_1,k_1)}) \equiv 0 \pmod{p_1^{k_1+1}}$ for every g_i. Conversely, tracing the determination process of G_f with different modulus q^k, if q^k exceeds the bound, we have the same result. As $p_1^{k_1}$ already exceeds the bound, we have

THEOREM 3.2. *Let q be a prime satisfying the property $\mathbb{P}$, $\mathcal{G}^{(q,k)}$ be the k-th approximation of a Gröbner basis of a maximal divisor of $\mathbb{Q}_q \otimes_{\mathbb{Q}} \mathcal{M}_0$ and $\bar{m}^{(q)}$ its corresponding primitive idempotent of $\bar{\mathcal{M}}_0^q$. If $q^k > p_1^{k_1}$ and $NF(I_i - A_i, \mathcal{G}^{(q,k)}) \equiv 0 \pmod{q^{k+1}}$ for every i, then $NF(g_i, \mathcal{G}^{(q,k)}) \equiv 0 \pmod{q^{k+1}}$ for every g_i, that is, $\bar{m}^{(q)}$ is a component of $\pi_q(m)$.*

This method can be seen as a modular Galois group computation guided by the knowledge of the exact branch taken during the descent from S_n to G_f in the permutations subgroups tree of degree n. Efficient implementations of Galois group computation use some techniques to cut this descending branch and allow to begin from a subgroup of S_n (see [10]). To be more efficient, we plan to adapt our method with these techniques.

3.3 Computation Scheme and i-relations

In this section, we recall the definition and give some new results about *computation scheme* and *i-relation* (see [18]).

In [18, Section 3] the authors present a framework for the computation of the Gröbner basis $\mathcal{G} = \{g_1, \ldots, g_n\}$ with indeterminate coefficients strategy. In this framework, we attache to a particular permutation representation G_f a set of *good* theoretical form for polynomials of $\mathcal{G}$ and *techniques* which allow us to avoid computations for some g_i. This is what we call *computation scheme* since this guides the algorithm for computing $\mathcal{G}$.

In particular, we associate to each polynomial g_i an integers set $E_i = \{e_1 < \cdots < e_s = i\}$ which describes a triangular set $T_i = \{g_1^*, \ldots, g_s^* = g_i\}$ where $g_k^* \in \mathbb{Q}[x_{e_1}, \ldots, x_{e_k}]$ and $g_k^*(x_{e_k}, \alpha_{e_{k-1}}, \ldots, \alpha_{e_1})$ is a minimal polynomial of the $\mathbb{K}$-extension $\mathbb{K}(\alpha_{e_k})$ where $\mathbb{K} = \mathbb{Q}(\alpha_{e_{k-1}}, \ldots, \alpha_{e_1})$; we will denote by $d(E_i)_k$ the degree of this extension. In [18] the theoretical form g_s^* is used to compute g_i by indeterminate coefficients strategy. The number of coefficients to compute is deduced from E_i (or equivalently by T_i) and is denoted by $d(E_i)$:

$$d(E_i) := \prod_{k=1}^{s} \deg_{x_{e_k}} g_k^*,$$

and this quantity is called *the degree of* E_i.

There may be a lot of different sets E_i which all correspond to the polynomial g_i but, the smaller $d(E_i)$ is, the more efficient our algorithm will be. For example, we can choose the trivial set $E_i = \{1, 2, \ldots, i-1, i\}$ which has maximal degree but in almost all cases we can find a better set E_i corresponding to g_i. This is why these sets E_i are important in our algorithm and we call them *i-relations*.

The *computation scheme* introduced in [18] provides also some techniques to avoid the computation of some g_i's. Thus, to this framework we attach the set $\mathcal{I}$ of integers corresponding to the polynomials we have to compute. The total number of coefficients to compute in $\mathcal{G}$ is the sum of the degrees of i-relation with i in $\mathcal{I}$ and we denote it $c(G_f)$. To compute the polynomials with index in $\mathcal{I}$ modulo a power of a prime, the strategy used was based on *indeterminate coefficients* followed by a *linear algebra* step. Here we want to replace the second step by an *interpolation* step. This is what we present in the next section.

3.4 Lagrange Formulae and i-relations

In [6], Lagrange formulae are presented for general triangular sets. These formulae can be used to compute the Gröbner basis $\mathcal{G}$, this is what is done in [13]. In this case, the total number of coefficients to compute will be of the order of the size of the Galois group G_f which may be very large. Thus, to overcome this problem we introduce Lagrange formulae for i-relations in order to use the *computation scheme*.

Let $E_i = \{e_1 < \cdots < e_s = i\}$ be an i-relation and T_i its associated triangular basis as defined in Section 3.3. The affine variety $Z_{\bar{\mathbb{Q}}}(T_i)$ is equiprojectable, thus one can apply the Lagrange formulae given in [6] but, since here we are in a very special context, we will restate the construction by using the permutation representation of the given Galois group G_f.

Let σ be a given permutation in S_n. (Here we write $\sigma(a)$ for a^σ for simplifying formulas.) We denote by $\mathcal{O}(j, \sigma)$ the orbit of $\sigma(e_j)$ under the action of the point-wise stabilizer $\mathrm{Stab}_{G_f}[\sigma(e_1), \sigma(e_2), \ldots, \sigma(e_{j-1})]$ defined by $\{\tau \in G_f \mid \tau(\sigma(e_i)) = \sigma(e_i),\ \forall i \in \{1, \ldots, j-1\}\}$. By using the map $i \mapsto \alpha_i$, the set $\mathcal{O}(j, \sigma)$ corresponds to the orbit of the element $\sigma(\alpha_{e_j})$ over the field $\mathbb{Q}(\alpha_{\sigma(e_1)}, \alpha_{\sigma(e_2)}, \ldots, \alpha_{\sigma(e_{j-1})})$.

From this orbit we can interpret the formula given in [6] in our specific case.

THEOREM 3.3. *Let $E_i = \{e_1 < \cdots < e_s = i\}$ be an i-relation. The corresponding polynomial g_i verify $g_i = \mathcal{L}_i$ with*

$$\mathcal{L}_i = \sum_{\sigma \in \mathrm{Trans}} \left(\left(\prod_{j=1}^{s-1} \prod_{\substack{e \in O(j,\sigma) \\ e \neq \sigma(e_j)}} \frac{x_{e_j} - \alpha_e}{\alpha_{\sigma(e_j)} - \alpha_e} \right) \prod_{e \in O(s,\sigma)} x_i - \alpha_e \right)$$

where Trans *is the transversal* $Stab_{G_f}([e_1, \ldots, e_s])\backslash\backslash G_f$

3.5 Bound for the Coefficients of g_s^*

In the formula $\mathcal{L}_i$ given in Theorem 3.3 the denominators can be canceled by multiplying with a sufficiently large power of $d(f)$ the discriminant of the polynomial f. The multiplication by $d(f)$ can cancel two denominators of the form $\prod_{\substack{e \in O(j,\sigma) \\ e \neq \sigma(e_j)}} \frac{1}{\alpha_{\sigma(e_j)} - \alpha_e}$. Thus, all the denominators can be canceled by multiplying $\mathcal{L}_i$ with $D_i = d(f)^{\lceil \frac{s}{2} \rceil}$.

The polynomial $D_i \mathcal{L}_i$ has integral coefficients, we will now investigate a bound over the coefficient c corresponding to the multi-degree $(k_1, \ldots, k_s)$. We denote by d_j the degree in x_j of $\mathcal{L}_i$. We note $d(E_i) = \prod_{k=1}^{s} d_k = |G_f : Stab_{G_f}([e_1, \ldots, e_s])|$. Let δ be a bound over the differences of roots $|\alpha_i - \alpha_j|$ and ν a bound over the absolute values of the roots $|\alpha_i|$. Here we will modify the proof given in [13] to deal with the case of an i-relation:

1. After cancellation of the denominators by multiplying with D_i, it remains, in the numerator, a product of $n(n-1)\lceil \frac{s}{2} \rceil - d_1 - \ldots - d_s + s$ elements of the form $(\alpha_j - \alpha_i)$. This product will be distributed on all the coefficients of g_i and is bounded by $\mathbb{B} = \delta^{n(n-1)\lceil \frac{s}{2} \rceil - d_1 - \ldots - d_s + s}$.
2. The indeterminate x_{e_i} of degree k_i in $\mathcal{L}_i$ comes from a product of $d_i - 1$ elements of the form $(x_{e_i} - \alpha_j)$. Thus, its absolute value can be bounded by the well known binomial quantity $\binom{d_i - 1}{k_i} \nu^{d_i - 1 - k_i}$.

Hence, by summing all these products over the transversal (see Theorem 3.3), we obtain the following bound for the absolute value of c:

$$d(E_i) \binom{d_1 - 1}{k_1} \nu^{d_1 - 1 - k_1} \cdots \binom{d_s}{k_s} \nu^{d_s - k_s} \mathbb{B}.$$

4. MULTI-MODULAR STRATEGY

In order to attain efficient computation of splitting fields, we can make good use of more sophisticated modular computation technique, "multi-modular" one. Here we show details on our technique and its variants for improvements.

4.1 Basic Discussion on Modular Techniques

There are several strategies on applying modular techniques for splitting field computation. Among those, a *multi-modular strategy* described below shall be effective and efficient under the following assumption which seems natural phenomena for our problem.

Assumption: The computed theoretical bound, say B_T, on the coefficients of the Gröbner basis $\mathcal{G}$ is much larger than the *real bound* B_R, that is, the maximal absolute value of numerators and denominators of coefficients of $\mathcal{G}$. Also B_T is much larger than the bound B_G used for the Galois group determination.

Under the assumption, it is quite natural to use some heuristic bound B_H much smaller than B_T. Our computation can be one instance of the following model:

Computational Model with Modular Computation
Here, the target which we want to compute is some mathematical object over the rational number field $\mathbb{Q}$.

Step 1. Candidate Computation: `Proc_CAND`
  Step 1-1. Modular Image Computation
    We set the modulus q, and then compute the modular image of the target modulo q.
  Step 1-2. Conversion
    By rational reconstruction, we have a candidate of the target.

Step 2. Correctness Check: `Proc_CHECK`
  We check whether a candidate is correct or not by some efficiently computable test. If the test is OK, we have the correct result.

In our case, `Proc_CHECK` can be executed by *ideal inclusion test*, which shall correspond to *trial division* for polynomial factorization, as pointed out in [18]. Also, we may use another modular technique for `Proc_CHECK`. Further discussion will be given later.

When we use $2B_T^2$ for the modulus, the computed candidate is always correct and `Proc_CHECK` is not necessary. (We note that for rational reconstruction, the modulus should be twice of square of the bound.) On the other hand, when we use some heuristic modulus q, we have to execute recursive computation to reach the correct answer. When `Proc_CHECK` fails, we can apply several strategies in Step 1:

S1: Replace a larger modulus q' and execute Step 1-1.

S2: Take another modulus q' prime to q, execute Step 1-1 with q' and combine the result with the old one by Chinese remainder theorem. Then the modulus in `Proc_CHECK` becomes $q \times q'$.

S3: Lift up the candidate to a larger modulus q' by Hensel construction.

Here, we call the strategy S2 *multi-modular* strategy, and the strategy S3 *p-adic* strategy. Apparently, the strategy S1 is not efficient compared with other two.

To find the most practical one among strategies in the above, we examine those total times of computation. Let $T_P(q)$ be the time of Step 1-1 of `Proc_CAND`, where q is the modulus used, $T_R(q)$ the time of Step 1-2 and T_C the time of `Proc_CHECK`. Here, we dare to omit the effect of $c(G_f)$ on those in order to make our argument clear. In Section 4.3, we will give further discussion taking $c(G_f)$ into account.

If we use $2B_T^2$ for the modulus, `Proc_CHECK` is not necessary, and thus the total time $T_0 = T_P(2B_T^2) + T_R(2B_T^2))$. If we use some heuristic modulus q, then we should repeat the computation till the modulus exceeds $2B_R^2$. Suppose that we reach to it by s times recursion, where $q_1, \ldots, q_s$ are moduli used. Then we have the following total times:

S2: $T_2 = \sum_{i=1}^{s}(T_P(q_i)+T_{CRT}(q_1 \cdots q_{i-1}, q_i)+T_R(\prod_{j=1}^{i} q_j)+ T_C)$, where $q_1 \cdots q_{s-1} < 2B_R^2 < q_1 \cdots q_s$ and $T_{CRT}(q, q')$ denotes the time for Chinese remainder theorem for two moduli q to q'.

S3: $T_3 = T_P(q_1) + \sum_{i=2}^{s}(T_H(q_i, q_{i-1}) + T_R(q_i) + T_C)$, where $q_{s-1} < 2B_R^2 < q_s$ and $T_H(q', q)$ denotes the time for Hensel lifting from the modulus q to q'.

Getting precise estimation of T_C is very difficult, when we apply *ideal inclusion test*. Thus, we may also apply additional modular technique to have efficient realization of `Proc_CHECK`. The basic procedure is the following:

Modular Check: `Proc_MODCHECK`
Once we have a candidate $\mathcal{C}$ constructed by using modulo q, we check if it is still valid modulo another q'. If so, we can show that $\mathcal{C}$ is still a candidate modulo $q \times q'$. Otherwise, we compute the modular image of a candidate modulo q' by Step 1-1 of `Proc_CAND` and apply Chinese remainder theorem to get the modular image of new candidate $\mathcal{C}'$ modulo $q \times q'$.

Now we denote by $T_{MC}(q)$ the time for modular check modulo q, not including any candidate construction. In our case, we suppose that $T_{MC}(q)$ is much smaller than $T_P(q)$ and $T_H(q', q'')$ with $q = q'/q''$. (See Section 4.3 for details.) Then, we have two types of usage of Modular Check:

U1: We can reduce the number of `Proc_CHECK` by repeating `Proc_MODCHECK` until we have a *stable* result. Then, it is highly supposed that the computed candidate is correct. With this practical assumption, the size of the total modulus is supposed the same order as that of $2B_R^2$, and so that of B_R. Then we have $T_2 = O(\sum_{i=1}^{s}(T_P(q_i) + T_{CRT}(q_1 \cdots q_{i-1}, q_i)+T_R(\prod_{j=1}^{i} q_j) + T_{MC}(q_i)) + T_C)$, where $\prod_{i=1}^{s} q_i = O(B_R)$, and $T_3 = O(T_P(q_1) + \sum_{i=2}^{s}(T_H(q_i, q_{i-1}) + T_R(q_i) + T_{MC}(q_i)) + T_C)$, where $q_s = O(B_R)$.

U2: We repeat `Proc_MODCHECK` till the total modulus reaches the theoretical bound. (So, the p-adic strategy is not suited for this approach.) This may sound somehow contradictory to our strategy. But, it is still able to give a practical solution as $T_{MC}(q)$ is much smaller than $T_P(q)$. In this case, we have $T_2 = O(\sum_{i=1}^{s}(T_P(q_i) + T_{CRT}(q_1 \cdots q_{i-1}, q_i) + T_R(\prod_{j=1}^{i} q_j)) + \sum_{i=1}^{t} T_{MC}(q_i))$, where $\prod_{i=1}^{s} q_i = O(B_R)$ and $\prod_{i=1}^{t} q_i = O(B_T)$.

Omitting the correctness check in U1, the total efficiency can be much improved. In this case, the result is not proven to be correct, but it will be with a high probability.

We will show certain practicality and theoretical efficiency of the multi-modular strategy, as it can use the both usages of *modular check*. In Section 4.2, we will give details of algorithms based on multi-modular strategy, and in Section 4.3, based on algorithms given in Section 4.2, we will discuss those efficiency, including estimation on T_C.

4.2 Algorithms

In the case of the computation of the splitting field of the polynomial f, the target will be the Gröbner basis $\mathcal{G}$ of the splitting ideal corresponding to the symmetric representation G of the Galois group of f. All our first inputs came from the computation of G_f by modular algorithm

(see [21],[9, 10]) modulo $p_1^{k_1}$ a power of a prime satisfying property $\mathbb{P}$. From this computation we obtain a finite sequence $\{(I_i, A_i) : i = 1, \ldots, t\}$ of *invariants* and their *integer evaluation* modulo $p_1^{k_1}$ as in Theorem 3.2. Because, we will use *multi-modular strategy* S2 for computing $\mathcal{G}$ we may need to reorder the roots of f modulo $p_2^{k_2}$, where p_2 is a prime satisfying $\mathbb{P}$ different from p_1. To do this we use the sequence $\{(I_i, A_i)\}$ as in Theorem 3.2, this is described in the following function:

Function: `GoodOrdering`$((p_2, k_2))$

We assume $p_2^{k_2} \geqslant p_1^{k_1}$ (else we exit with an error).
Let $\underline{\hat{\alpha}}$ be the roots of f modulo $p_2^{k_2}$.
Let t be the length of the sequence $\{(I_i, A_i)\}$.
for $i = 1$ to t **do**
  Let (G, H) be the groups corresponding to I_i.
  for $\sigma \in G \backslash\backslash H$ **do**
    if $I_i(\underline{\hat{\alpha}}^\sigma) = A_i$ **then**
      $\underline{\hat{\alpha}} := \underline{\hat{\alpha}}^\sigma$
      break
    end if
  end for
end for
return $\underline{\hat{\alpha}}$

As presented in Section 4.1 we now give an algorithm to compute a candidate $\mathcal{G}_{\text{cand}} = \{\tilde{g}_1, \ldots, \tilde{g}_n\}$ for the target $\mathcal{G}$ by using a heuristic bound B_T. This first step is described by the algorithm `Proc_CAND` that takes as input an approximation of the roots of $f \mod p_1^{k_1}$ as a Gröbner basis $\mathcal{G}^{(p_1,k_1)}$ of the splitting ideal of $f \mod p_1^{k_1}$, the Galois group G_f corresponding to the order of these roots and a set $\mathcal{P}$ of couples (p, k) where p's are different primes (different from p_1) satisfying property $\mathbb{P}$ and k's are integers such that $p_1^{k_1} \prod_{(p,k)\in\mathcal{P}} p^k \geq B_H$. To the group G_f is associated a *computation scheme* which is represented by the set $\mathcal{I}$ of indexes of the g_i to compute in $\mathcal{G}$ and the corresponding i-relation E_i. Before giving this first algorithm, we give a specific *Chinese Remaining Theorem* procedure which will be used in the sequel:

Procedure: `SpecCRT`$(\{g_i \mod M, \quad i \in \mathcal{I}\}, (p, k))$

Compute and reorder the roots of $f \mod p^k$.
Interpolate the g_i's mod p^k corresponding to E_i with $i \in \mathcal{I}$.
Let M be the modulus $M \times p^k$
By CRT compute all the $g_i \mod M$ $(i \in \mathcal{I})$.

Now, we can describe the first algorithm corresponding to the first step `Proc_CAND`:

Algorithm 1: `Proc_CAND`$(\mathcal{G}^{(p_1,k_1)}, G_f, \mathcal{P})$

Interpolate all the $g_i \mod p_1^{k_1}$ corresponding to E_i $(i \in \mathcal{I})$.
Let M be the modulus $p_1^{k_1}$
for $(p, k) \in \mathcal{P}$ **do**
  SpecCRT$(\{g_i \mod M, \quad i \in \mathcal{I}\}, (p, k))$.
  Let M be the modulus $M \times p^k$
end for
S1:
try to convert all g_i's mod M to rational polynomials h_i's.
if all the conversions above succeed **then**
  The polynomial $\tilde{g}_i$ is h_i.
else
  Find a new couple (p, k) not in $\mathcal{P}$.
  SpecCRT$(\{g_i \mod M, \quad i \in \mathcal{I}\}, (p, k))$.
  Let M be the modulus $M \times p^k$.
  Goto step **S1** and add (p, k) in $\mathcal{P}$.
end if
For each $\tilde{g}_j$ with $j \notin \mathcal{I}$, apply a technique over one $\tilde{g}_i$ with $j < i$ to obtain $\tilde{g}_j$.
Return $\mathcal{G}_{\text{cand}}$, G_f, $\mathcal{I}$, $\mathcal{P}$.

Since we concentrate here on multi-modular strategy for the computation of $\mathcal{G}$, we give an algorithm corresponding to `Proc_CHECK` based on the strategy S2. Recall that this algorithm is based on the *ideal inclusion test* given in [18]:

Algorithm 2: `Proc_CHECK_S2`$(\{\tilde{g}_1, \ldots, \tilde{g}_n\}, G_f, \mathcal{I}, \mathcal{P})$

S1:
if The equality $\text{NF}(\tilde{g}_i, c_i(f)) = 0$ is true for all $i \in \mathcal{I}$ **then**
  The Gröbner basis $\mathcal{G}$ is $\{\tilde{g}_1, \ldots, \tilde{g}_n\}$.
else
  Let M be the product $\prod_{(p,k)\in\mathcal{P}} p^k$.
  S2: Find a new couple (p, k) not in $\mathcal{P}$.
  SpecCRT$(\{g_i \mod M, \quad i \in \mathcal{I}\}, (p, k))$.
  Apply techniques to obtain the $\tilde{g}_j$'s with $j \notin \mathcal{I}$.
  Let M be the modulus $M \times p^k$.
  Convert all the $\tilde{g}_i$'s mod M to rational polynomials.
  If conversions above pass **then** goto **S1 else** goto **S2**.
end if
Return $\mathcal{G}$.

As we said in Section 4.1, we can take advantage of the multi-modular strategy in `Proc_CHECK` too. Here we give the implementation of `Proc_MODCHECK` with a generic stopping condition which can be stated in function of the chosen U1 or U2 version.

Algorithm 3: `Proc_MODCHECK`$(\{\tilde{g}_1, \ldots, \tilde{g}_n\}, G_f, \mathcal{I}, \mathcal{P})$

Let M be the product $\prod_{(p,k)\in\mathcal{P}} p^k$.
S1:
if The specified condition `COND` is not satisfied **then**
  Find a new couple (p, k) not in $\mathcal{P}$.
  Compute and reorder the roots $\underline{\hat{\alpha}}$ of $f \mod p^k$.
  if there exists $i \in \mathcal{I}$ such that $\tilde{g}_i(\underline{\hat{\alpha}}) \neq 0 \mod p^k$ **then**
    SpecCRT$(\{g_i \mod M, \quad i \in \mathcal{I}\}, (p, k))$.
    Apply techniques to obtain the $\tilde{g}_j$'s with $j \notin \mathcal{I}$.
  end if
  Let M be the modulus $M \times p^k$.
  Goto **S1**.
end if
Convert all the $\tilde{g}_i$'s mod M to rational polynomials g_i's.
S2: Return $\{g_1, \ldots, g_n\}$.

If condition `COND` is chosen to be the one corresponding to the version U1 then the output of `Proc_MODCHECK` is not proven but we can use it as an input of `Proc_CHECK`. Otherwise, the condition `COND` will correspond to version U2 and in this case the output will be proven.

4.3 Discussion on Efficiency

Here we discuss the efficiency of the proposed "multi-modular strategy" along by basic estimation on concrete procedures given in the previous subsection. We use the facts given in [18]. By actual experiments on real computers, the authors found that sometimes the correctness check dominated the total efficiency, even though the modulus used there was the same order of the real bound. To resolve this problem, we introduce the usage U2 and show that it can give good estimation on the total efficiency.

To estimate the total efficiency, we have to give concrete representation of the functions T_P, T_R, T_{CRT}, T_H, T_{MC} and T_C. To distinguish the time for Step 1-1 of `Proc_CAND` by linear system solving [18] and that by Lagrange interpolation in Section 3.4, we denote by T_{PL} the time by linear system solving and by T_{PI} that by interpolation. Also, we denote by $\mathbb{M}(q)$ the unit cost of integer arithmetics of size q. Then, $\mathbb{M}(q) = O(q^2)$ for usual multiplication technique, and $\mathbb{M}(q) = O(q^{1+\epsilon})$ for fast multiplication technique. Also by

[18], we have $T_C = O(\log(n)c(G_f)^2\mathbb{M}(\log(B'))$ and B' is the largest integer appeared in the normal form computation. (Finding a good estimation on B' is a difficult problem.)

For the strategy S2, it had better to use Lagrange interpolation for `Proc_CAND` and $T_{PI}(q) = O(c(G_f)^2\mathbb{M}(\log(q))) + T_{GO}$, where T_{GO} is the time for `GoodOrdering`. For the two quantities $T_{CRT}(A, B)$ and T_R, we can apply fast extended GCD computation technique for Chinese remainder Theorem and rational reconstruction (see [15]), and thus

$T_{CRT}(A, B) = O(c(G_f)\mathbb{M}(\log(C))\log\log(C))$, where $C = \max\{A, B\}$ and $T_R(q) = O(c(G_f)\mathbb{M}(\log(q))\log\log(q))$.
As to the modular check of correctness, we divide each candidate $\tilde{g}_i$ by $q = p^k$ and substitute the p-adic approximation of roots of f, and so $T_{MC}(q) = O(c(G_f)(\mathbb{M}(\log(q)) + \mathbb{M}(\log(D)))$, where D is the maximal absolute value of numerators and denominators of coefficients of $\tilde{g}_i$'s. As $D < B_R$, we have $T_{MC}(q) = O(c(G_f)(\mathbb{M}(\log(B_R)))$.

For the strategy S3, by the estimation in [18], we have $T_{PL}(q) = O((c(G_f)^\omega\mathbb{M}(\log(q)))$, and $\sum_{i=2}^{s} T_H(q_i, q_{i-1}) = O(n^2c(G_f)^2\mathbb{M}(\log(q_s)))$ where ω represents a feasible matrix multiplication exponent and $2 \le \omega \le 3$ (see [8]).

Thus, accumulating all representations, the total cost can be estimated in *slightly rough but simple form*, where we are assuming that $\log(B_T)$ is much larger than $\log(B_R)$ and the size of modulus q used is close to the size of the real bound B_R, that is, the number of correctness checks in U1 and that of CRT construction in U2 can be *bounded* (one or very small).

PROPOSITION 4.1. *Assume that* $\log(B_T)$ *is much larger than* $\log(B_R)$ *and the size of modulus* q *used and that of* B_R *are the same order.*
(1) When we use the modular check U1, we have

$$T_2 = O(c(G_f)^2\mathbb{M}(\log(B_R)) + c(G_f)\mathbb{M}(\log(B_R))\log\log(B_R) + T_{GO} + T_C),$$
$$T_3 = O(c(G_f)^\omega\mathbb{M}(\log(q_1)) + n^2c(G_f)^2\mathbb{M}(\log(B_R)) + T_C),$$

where $T_C = O(\log(n)c(G_f)^2\mathbb{M}(\log(B'))$ *and* B' *is the largest integer appeared in the normal form computation.*
(2) When we use the modular check U2, we have

$$T_2 = O(c(G_f)^2\mathbb{M}(\log(B_R)) + c(G_f)\mathbb{M}(\log(B_R))\log\log(B_R) + c(G_f)\mathbb{M}(\log(B_T)) + T_{GO}\frac{\log(B_T)}{\log(B_R)}).$$

Taking account of the estimation on B_T in Section 3.5, when `GoodOrdering` can be efficiently done like as T_{GO} is $O(c(G_f)\mathbb{M}(\log(B_R)))$, the strategy S2 with the modular check U2 can give the most efficient computation.

5. EXPERIMENTS AND REMARKS

We have implemented the algorithms of Section 4.2 with the *computer algebra system* MAGMA (version 2.13) in the case of an irreducible monic integral polynomial. This version of MAGMA provides a lot of functions for using the byproduct of the modular computation of the Galois group. All these functions are very efficient and easy to use but we need a little more, thus we have rewritten a large part of our sub-functions in order to be more efficient.

Reordering the roots: In the case of the normalizer of G_f in S_n is reduced to G_f, it is sometimes better to recompute the Galois group G_2 of f using another prime p_2 and we finally reorder the roots by applying the permutation which conjugates G_f and G_2, as we can use the efficient computation of the Galois group. This is specially effective when the descending process in the subgroups tree encounter two succeeding groups with large index. This method could be generalized by considering the normalizer of G_f and will study in a future implementation.

Choice of the primes p: Because of the limit imposed by the Tchebotarev's density theorem it may be hard to find primes which split completely in the stem field defined by the polynomial f. On the other hand, the costs of the p-adic arithmetic increase according to the order of the Galois group of f modulo p. Thus, we choose primes satisfying the property $\mathbb{P}$ such that the Galois group of f modulo theses primes has an order of at most 2.

Heuristic bound B_H: We choose, as an heuristic, to begin the computation `Proc_CAND` modulo the fifth power of the product of one, two or three minimal primes satisfying the condition state before.

Comments on the experiments: We try our implementation on several polynomials given by [11] from degree 6 to 9 and some polynomials of greater degree (not more than 13) corresponding to interesting computation schemes. By using this heuristic bound, `Proc_CAND` already computes in all the cases the final result, thus the remaining computation is the check procedure without any reconstruction. For groups G with small order or $c(G)$ (say under 500) our new implantation will not give a really better efficiency than the one presented in [18] which was already very efficient. The only gain is given by the use of the interpolation in place of the linear system resolution. Moreover, if the group is very small in the symmetric group of same degree, the reordering procedure may be time consuming. Thus, in this case it is preferable to use mono-modular algorithm and check the result by normal forms computations (`Proc_CHECK_S2`). In the case of groups G with high order and small $c(G)$, our experiments show that by using interpolation, the gain of efficiency of this new implementation is comparable to the gain of theoretical complexity. In these cases, the experimental cost of the reordering of the roots and the reconstruction of the polynomials by Chinese remainder theorem are very small in comparison to the other parts of the computation. Concerning the check procedure, in these cases, the modular strategy with version `U2` is always better than the procedure `Proc_CHECK_S2`. This experiments allow us to think that this new method should be use for the computation of splitting fields with high absolute degree, the remaining bottleneck is the efficient calculation of a computation scheme (we have tabulated them up to degree 10).

Remarks on parallel computations: The computer algebra system MAGMA do not integrate any possibility for parallel computation, thus we have implemented all our algorithms with a sequential style. But, during our experiments we reported that all the parts of the computation that can be shared on different cores have the same time consuming in general. Thus, implementing these algorithms in a parallel system will have a real impact on the efficiency. For example, in the case of `Proc_CAND` the computations modulo two different primes has almost the same time consuming in general (this is the case when the word sizes of these primes are the same and they have the same decomposition property modulo f).

We give here some specific examples and comments about them. We give, for each example, the name of the group G in Butler and McKay's nomenclature, the order of G and the integer $c(G)$ (as the sum of the i-relations degrees). The column `Primes` shows the timings of computing primes with good properties. The column `Lin.` shows timing `Proc_CAND` using linear systems solving as in [18], `Int.` those when we use interpolation, `S2` the timings for check procedure `Proc_CHECK_S2`, `U2` the timing of `Proc_MODCHECK` based on version `U2` and `Ord.` the timing of reordering the roots. The total timings can be obtained by summing the column corresponding to strategies used during the computation. The measurements were made on a personal computer with a 3.0GHz Intel Xeon 64bits (all the timings are given in seconds).

Group	$\|G\|$	$c(G)$	Primes	Lin.	Int.	S2	U2	Ord.
$10T_{39}$	3840	10	2.13	0.04	0.05	0.01	0.01	1.01
$10T_{36}$	1920	$960 + 10$	0.81	141.2	60.5	55.4	2.1	1.3
$9T_{32}$	1512	$1512 + 1512$	0.57	540.8	$65.1 + 67.3$	412.1	4.1	0.5
$9T_{29}$	648	$18 + 648$	0.67	40.1	9.3	1.22	1.5	1.0
$9T_{25}$	324	27	0.08	6.54	7.1	0.2	0.5	2.4
$8T_{48}$	1344	336	1.25	9.89	4.23	15.34	1.1	0.8
$7T_6$	2520	2520	1.5	785.3	$76 + 157.9$	8.2	2.1	2.5

The first line shows a very special case where the total timing is dominated by the research of primes with good property. Theses cases appear when the group is very large in comparison with its size $c(G)$ as here. On the last line, we show the two different timings for computing modulo two different primes. In this case, the first one corresponds to a prime which splits completely the polynomial f in contrary to the second one. This is a general fact, for a fixed word size, using a splitting prime will give better timings (in this case the first one is 10037 the second is 53). The remaining examples show that even the times between the different strategies are comparable, using multi-modular and interpolation approaches are in general better. We did not compare these timings with the ones given in [13] because of the difference of architecture, but if we project them on the same computer our implementation would be more efficient with a factor from 10 to 1000 (this big difference may come from the fact that *computation schemes* are not used in [13]). Also, we did not integrate in the last table the timings for computing $\mathcal{G}$ in MAGMA (version 2.14) with the new function `GaloisSplittingIdeal` (with parameter `Roots` set to false) since we can only obtain two of them by waiting not more than 300 seconds.

6. CONCLUSION

We have presented a new method based on a multi-modular strategy for the computation of the splitting field of a polynomial f. This new method is a good compromise for the one presented in [18] since it gives better results in the case where the later was inefficient. Also, the experiments show that this multi-modular method is a good candidate for a parallel implementation on a multi-core architecture.

The reordering roots function we give here is a general one. We plan to integrate efficient techniques of Galois groups computation (like the ones presented in [10]) in order to reduce the length of the descending branch in the subgroups tree.

We hope that such a multi-modular method could be generalized to the problem of computing algebraic or integral relations between the roots of a polynomial like in [7].

7. ACKNOWLEDGMENT

The authors would like to thank the referees for their valuable remarks.

8. REFERENCES

[1] I. Abdeljaouad, S. Orange, G. Renault, and A. Valibouze. Computation of the decomposition group of a triangular ideal. *AAECC Journ.*, 15(3-4):279–294, 2004.

[2] J.-M. Arnaudiès and A. Valibouze. Lagrange resolvents. *J. Pure Appl. Algebra*, 117/118:23–40, 1997. Algorithms for algebra (Eindhoven, 1996).

[3] Anai, H., Noro, M., and Yokoyama, K. Computation of the splitting fields and the Galois groups of polynomials. In *Algorithms in algebraic geometry and applications*, vol. 143 of *Progr. Math.* Birkhäuser, Basel, 1996, 29–50.

[4] Becker, T., and Weispfenning, V. *Gröbner bases*, vol. 141 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1993.

[5] Bosma, W., Cannon, J., and Playoust, C. The Magma algebra system. I. The user language. *J. Symbolic Comput. 24*, 3-4 (1997), 235–265.

[6] Dahan, X., and Schost, É. Sharp estimates for triangular sets. In *ISSAC '04: Proc. of the 2004 International Symposium on Symbolic and Algebraic Computation* (New York, 2004), ACM Press, pp. 103–110.

[7] Fieker, C. and Graaf, W. Integral linear dependencies of algebraic numbers and algebraic Lie algebras. *LMS J. Comput. Math.*, 10 (2007), 271–287

[8] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, second edition, 2003.

[9] Geissler, K. *Berechnung von Galoisgruppen über Zhal- und Funktionenkörpern*. PhD thesis, Univ. Berlin, 2003.

[10] Geissler K. and Klüners, J. Galois group computation for rational polynomials. *J. Symbolic Comput.*, 30(6):653–674, 2000.

[11] Klüners, J., and Malle, G. A database for field extensions of the rationals. *LMS J. Comput. Math. 4* (2001), 182–196 (electronic).

[12] Lazard, D. Solving zero-dimensional algebraic systems. *J. Symbolic Comput. 13*, 2 (1992), 117–131.

[13] Lederer, M., Explicit constructions in splitting fields of polynomials. *Riv. Mat. Univ. Parma (7)*, 3* (2004), 233–244.

[14] McKay, J., and Stauduhar, R. Finding relations among the roots of an irreducible polynomial. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation* (New York, 1997), ACM, pp. 75–77.

[15] Pan, V. and Wang, X. Acceleration of Euclidean Algorithm and Rational Number Reconstruction. *SIAM J. Comput.* Vol.32, No.2, pp.548-556, 2003.

[16] M. Pohst and H. Zassenhaus. *Algorithmic Algebraic Number Theory*. Cambridge Univ. Press, Cambridge, 1989.

[17] Renault, G. Computation of the splitting field of a dihedral polynomial. In *Proc. of the 2006 International Symposium on Symbolic and Algebraic Computation* (New York, 2006). ACM Press, pp. 290–297.

[18] Renault, G., and Yokoyama, K. A modular method for computing the splitting field of a polynomial. In *Proc. of the 7th Algorithmic Number Theory Symposium* ANTS-VII, Berlin, Germany, 2006, LNCS 4076, Springer, pp. 124–140.

[19] N. Rennert and A. Valibouze. Calcul de résolvantes avec les modules de Cauchy. *Exp. Math.*, 8(4):351–366, 1999.

[20] Tchebotarev, N. *Gründzüge des Galois'shen Theorie*. P. Noordhoff, 1950.

[21] Yokoyama, K. A modular method for computing the Galois groups of polynomials. *J. Pure Appl. Algebra 117/118* (1997), 617–636.