AAECC (2004)
Digital Object Identifier (DOI) 10.1007/s00200-004-0160-x

AAECC
Applicable Algebra in Engineering, Communication and Computing

Computation of the Decomposition Group of a Triangular Ideal

I. Abdeljaouad-Tej[1], S. Orange[2], G. Renault[2], A. Valibouze[2]

[1] École Supérieure de la Statistique et de l'Analyse de l'Information à l'INSAT, Boulevard de la Terre, Zone Urbaine Nord Tunis, BP 675, 1080 Tunis, Tunisie
(e-mail: Ines.Abdeljaoued@insat.rnu.tn)
[2] LIP6 - Université Pierre et Marie Curie, 4, place Jussieu, 75005 Paris, France
(e-mail: {orange, renault, valibouze}@calfor.lip6.fr)

Received: July 1, 2003; revised version: June 10, 2004
Published online: October 15, 2004 –

Abstract. This article describes two algorithms in order to search decomposition groups of ideals of polynomials with coefficients in a perfect field when those ideals are generated by a triangular system of generators.

Keywords: Triangular Ideal, Strong Generating Set

Introduction

Let $K[X_1, \dots, X_n]$ be a multivariate polynomial ring over a perfect field K and I a triangular ideal of this ring. We have a canonical action of S_n over $K[X_1, \dots, X_n]$. We are interested in computation of $Dec(I)$ the set of permutations which leaves I globally invariant. Actually, this set is a group called the *decomposition group* of I consistently with the classical definition on prime ideals (see [4, Définition 2 page 36]).

In the specific case where I is a relations ideal of f an irreducible polynomial f of degree n, Anai, Noro and Yokoyama give in [1] an algorithm for $Dec(I)$ computation which is, up to an isomorphism, the Galois group of f. They bound by $O(n^4)$ the number of normal forms computations needed by their algorithm.

Our algorithm includes the backtracking technique (see [5]) in order to compute a strong generating set of $Dec(I)$ (see Section 3). This algorithm uses the naive algorithm of Section 2 as a subroutine. Recall that a strong generating set E of a group $G \subset S_n$ is a set of permutations verifying: for all $i \in [\![1, n]\!]$, $Fix_G(\{a_1, \dots, a_i\}) \cap E$ generates $Fix_G(\{a_1, \dots, a_i\})$ (see [5]).

In Section 4, we generalize Theorem 5 of [1] to the case of Galois ideal and we prove, in Section 5.1, that the number of normal forms computations needed by our algorithm is bounded by $O(n^3)$. In Section 5.2, we give an heuristic comparison of our algorithms and the one given in [1], in the case of Galois ideals (see Section 5.2).

Notations

In this paper, the following notations are used:

- $K[X_{1,} \dots , X_n]$ is the polynomial ring over a perfect field K in n algebraic independent variables $X_1, \dots, X_n$;
- I is an ideal of $K[X_{1,} \dots , X_n]$ generated by a triangular set of n polynomials in $K[X_{1,} \dots , X_n]$:

$$S = \{f_1(X_1), f_2(X_1, X_2), \dots, f_n(X_1, X_2, \dots, X_n)\};$$

- for all subsets $\mathcal{A}$ of the set $\{1, \dots, n\}$ and for all subgroups G of S_n, $Fix_G(\mathcal{A})$ is the pointwise stabilizer of G in $\mathcal{A}$, which is the subgroup composed by all permutations σ in G verifying $\forall i \in \mathcal{A},\ \sigma(i) = i$;
- for any non empty set $\mathcal{E}$ of S_n, $\langle\mathcal{E}\rangle$ is the subgroup generated by $\mathcal{E}$.

1. Decomposition Group and Ideal Membership Test

The symmetric group S_n acts naturally on the ring $K[X_1, \dots, X_n]$: for all $P \in K[X_1, \dots, X_n]$ and for all $\sigma \in S_n$, we define $\sigma.P$ by

$$\sigma.P(X_1, \dots, X_n) = P(X_{\sigma(1)}, \dots, X_{\sigma(n)}).$$

For this group action, the stabilizer $Dec(I)$ of the ideal I is called its *decomposition group*:

$$Dec(I) = \{\sigma \in S_n \mid \forall P \in I, \sigma.P \in I\}.$$

In order to test whether P, a polynomial, belongs to I, the triangular ideal, we will use the following classical result (see [7]):

Let $P \in K[X_{1,} \dots , X_n]$ and $(r_i)_{i\in[\![1,n]\!]}$ the sequence of $K[X_{1,} \dots , X_n]$ inductively defined by: $r_n = P$ and, for all $i \in [\![2, n]\!]$, r_{i-1} is the rest of the euclidean division of r_i by f_i relatively to the variable X_i. The following equivalence holds:

$$(P \in I) \text{ if and only if } (r_1 = 0).$$

The polynomial r_1 is usually called the *normal form of P with respect to S*, the triangular basis chosen for I.

2. Computing all the Permutations of the Group *Dec(I)*

Since $\{f_1, f_2, \ldots, f_n\}$ is a generating system of the ideal I, the decomposition group can be written:

$$Dec(I) = \{\sigma \in S_n \mid \forall i \in \{1, \ldots, n\},\ \sigma.f_i \in I\}.$$

Thus:

$$\sigma \in Dec(I) \quad \text{iff} \quad \begin{cases} f_1(X_{\sigma(1)}) \in I \\ f_2(X_{\sigma(1)}, X_{\sigma(2)}) \in I \\ \quad \vdots \\ f_n(X_{\sigma(1)}, \ldots, X_{\sigma(n)}) \in I \end{cases} \qquad \{*\}$$

Let the condition set $\mathcal{P} = \{P_1, \ldots, P_n\}$ be: for all $r \in [\![1, n]\!]$:

$$P_r(a_1, \ldots, a_r) \text{ is true if } f_r(X_{a_1}, X_{a_2}, \ldots, X_{a_r}) \in I.$$

Our first algorithm, named `DecompositionGroup`, uses equivalence $\{*\}$ and can be described in the following way:

- The first step of the algorithm computes all possible values $a_1 \in \{1, \ldots, n\}$ verifying $P_1(a_1)$ and the second step is applied to each of these values.
- At the r^{th} step ($2 \leq r \leq n$), the algorithm has found $r-1$ distinct values $a_1, \ldots, a_{r-1}$ such that $\forall i \in [\![1, r-1]\!]$, $P_i(a_1, a_2, \ldots, a_i)$ is true. The algorithm computes all possible values a_r taken among the set $\{1, \ldots, n\} \setminus \{a_1, \ldots, a_{r-1}\}$ and verifying $P_r(a_1, \ldots, a_r)$. The next step is applied to each of the sequences $a_1, \ldots, a_r$.
- When $r = n + 1$, the algorithm has found $\sigma = \left(\begin{smallmatrix} 1 & \cdots & n \\ a_1 & \cdots & a_n \end{smallmatrix}\right)$, a permutation belonging to the group $Dec(I)$.

Algorithm 2.1

Function `DecompositionGroup`($\mathcal{P}$)

/*
Input : The condition set $\mathcal{P}$ defined above.

Output : The decomposition group of the ideal I.
*/

Return `ConstructionOfPermutations` $(1, [\ ], \{Id\}, \mathcal{P})$;

End Function

Function `ConstructionOfPermutations` $(r, [a_1, \ldots, a_{r-1}], G, \mathcal{P})$

/*
Input : . r an integer of $[\![1, n+1]\!]$.
. $[a_1, \ldots, a_{r-1}]$ a list of distinct integers from $\{1, \ldots, n\}$, for which are searched the suffix lists $[a_r, \ldots, a_n]$ such that $\left(\begin{smallmatrix} 1 & \cdots & n \\ a_1 & \cdots & a_n \end{smallmatrix}\right) \in Dec(I)$.
. G a set containing the already found permutations belonging to $Dec(I)$.
. The condition set $\mathcal{P}$.

Output : . The group $Dec(I)$.
*/

If $r = n + 1$ **Then**
. $G := G \cup \{\begin{pmatrix} 1 & \cdots & n \\ a_1 & \cdots & a_n \end{pmatrix}\}$; /* $\begin{pmatrix} 1 & \cdots & n \\ a_1 & \cdots & a_n \end{pmatrix}$ belongs to $Dec(I)$ */
Else
. **For All** $a \in \{1, \ldots, n\} \backslash \{a_1, \ldots, a_{r-1}\}$ **Do**
. . /* The possible images a of r are then searched */
. . **If** $P_r(a_1, \ldots, a_{r-1}, a)$ **Then**
. . $G :=$ `ConstructionOfPermutations` $(r + 1, [a_1, \ldots, a_{r-1}, a],$
. . $G, \mathcal{P})$;
. . **End If**;
. **End For**;
End If;
Return G;
End Function

At the r^{th} step ($r \in [\![1, n]\!]$), the function `ConstructionOfPermutations` realizes at least $n + 1 - r$ recursive calls; this insures the algorithm ending. A permutation σ will be added to G if it verifies the n conditions $\{*\}$; thus the set returned by the Function `DecompositionGroup` is the group $Dec(I)$.

Example 2.2 Let I be a Galois ideal of $\mathbb{Q}[x_1, \ldots, x_6]$ (see Definition 4.3) generated by the polynomials:

$$\begin{aligned}
f_1(x_1) &= x_1^6 - x_1^5 - 10x_1^4 + x_1^3 + 12x_1^2 - 3x_1 - 1, \\
f_2(x_1, x_2) &= 17x_2 - 5x_1^5 + 4x_1^4 + 44x_1^3 + 14x_1^2 + 4x_1 - 8, \\
f_3(x_1, x_2, x_3) &= 17x_3^2 - 8x_3x_1^5 + 3x_3x_1^4 + 84x_3x_1^3 + 36x_3x_1^2 - 65x_3x_1 \\
&\quad -6x_3 - 29x_1^5 + 13x_1^4 + 296x_1^3 + 139x_1^2 - 276x_1 - 77, \\
f_4(x_1, \ldots, x_4) &= 17x_4 + 17x_3 - 8x_1^5 + 3x_1^4 + 84x_1^3 + 36x_1^2 - 65x_1 - 6, \\
f_5(x_1, \ldots, x_5) &= 17x_5^2 + 13x_5x_1^5 - 7x_5x_1^4 - 128x_5x_1^3 - 50x_5x_1^2 + 78x_5x_1 \\
&\quad -3x_5 + 11x_1^5 - 19x_1^4 - 107x_1^3 + 95x_1^2 + 168x_1 - 115, \\
f_6(x_1, \ldots, x_6) &= 17x_6 + 17x_5 + 13x_1^5 - 7x_1^4 - 128x_1^3 - 50x_1^2 + 78x_1 - 3\ .
\end{aligned}$$

Algorithm 2.1 runs through the full tree of Figure 1, in which, for each node, it tests whether a permuted polynomial belongs to I:

In this example, many computations are useless. For instance, $(3; 4)(5; 6)$ belongs to the group $\langle(3; 4), (5; 6)\rangle$; in the same way $(1; 2)(3; 5; 4; 6)$, $(1; 2)(3; 6; 4; 5)$ and $(1; 2)(3; 6)(4; 5)$ belong to the group $\langle(3; 4), (5; 6), (1; 2)(3; 5)(4; 6)\rangle$. To look for an other algorithm computing only a set of generators for $Dec(I)$ appears natural. We remark that Algorithm 2.1 determines successively the increasing sequence of pointwise stabilizer:

$$Fix_{Dec(I)}(\{1, \ldots, 6\}) < \cdots < Fix_{Dec(I)}(\{1, 2\}) < Fix_{Dec(I)}(\{1\}).$$

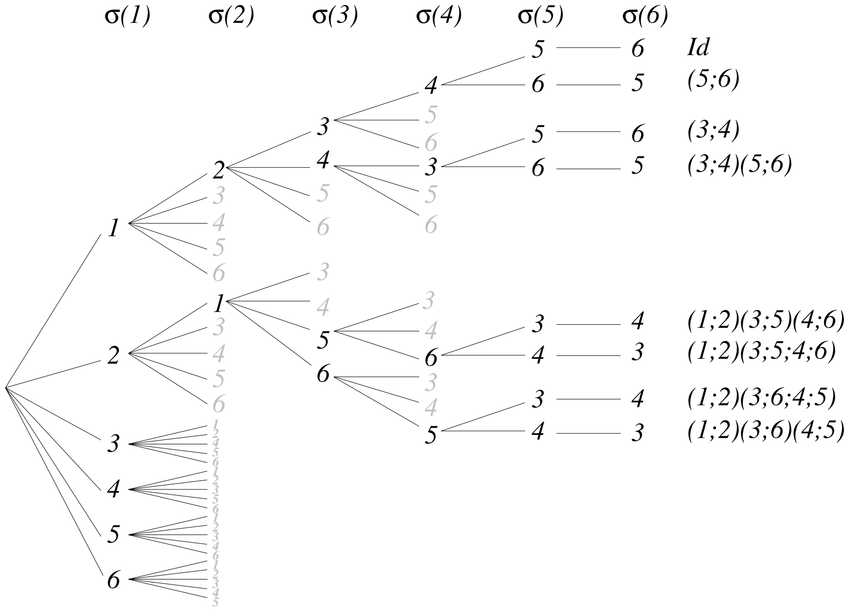

Fig. 1. Full tree

3. Determination of a Generating set for *Dec(I)*

In the case $Dec(I) = S_n$, Algorithm 2.1 presents the double disadvantage to realize $n!$ membership tests to the ideal I and to stock $n!$ permutations, which can compromise the computation.

In this section, we focus on the determination of a generating set of $Dec(I)$. Thus, when $Dec(I) = S_n$, the algorithm will compute only $n(n+1)/2 - 1$ membership tests to I in order to determinate a generating set composed by $n - 1$ transpositions.

3.1. Notation

For now on, k is an integer of $[\![1, n]\!]$.

Let G_0 be the group $Dec(I)$ and G_k be the group $Fix_{G_0}(\{1, \dots, k\})$. For all subgroup L of S_n, $Orb_L(k)$ is the L-orbit of k.

Algorithm 2.1 determines successively all the increasing sequence terms:

$$\{Id\} = G_n < G_{n-1} < \cdots < G_2 < G_1 < G_0.$$

This algorithm constructs the group G_{k-1}, for all $k \in [\![1, n]\!]$, by adding to G_k the elements of $G_{k-1} \backslash G_k$. In order to avoid the computation of all the permutations of $G_{k-1} \backslash G_k$, the propositions of Section 3.2 permit the construction of a G_{k-1} generating set from any G_k generating set. And so on, up to obtaining a G_0 strong generating set.

3.2. Construction of a Generating set

Given a generating set of L a group such that $G_k \subset L \subset G_{k-1}$ and a permutation of G_{k-1} (determined Algorithm 2.1), we construct a generating set of L' a group strictly containing L (Proposition 3.1). Iterating this process, we determine an increasing chain of groups between G_k and G_{k-1} (Algorithm 3.7). Each group will be represented by a generating set.

Proposition 3.1 *Let L be a subgroup of S_n such as $G_k \subseteq L \subseteq G_{k-1}$. Let $\mathcal{G}$ be a generating set of L and O an L-orbit of $\{1, \ldots, n\}$ included in $\{k+1, \ldots, n\}$.*

Let $\mathcal{E}$ be the set $\{\sigma \in G_{k-1} \mid \sigma(k) \in O\}$ and $L' = \langle L \cup \mathcal{E} \rangle$. If $\mathcal{E}$ is not empty then the group L' strictly contains L and is generated by $\mathcal{G} \cup \{\sigma\}$ for any σ in $\mathcal{E}$.

Proof. Let $\sigma \in \mathcal{E}$. Since $\mathcal{G} \cup \{\sigma\}$ generates $\langle L \cup \{\sigma\} \rangle$ and $\langle L \cup \mathcal{E} \rangle$ generates L', it is sufficient to prove that any permutation of the set $L \cup \mathcal{E}$ belongs to the group $\langle L \cup \{\sigma\} \rangle$. Let σ' be a permutation of $L \cup \mathcal{E}$.

If $\sigma' \in L$, the result is immediate.

If $\sigma' \in \mathcal{E}$ then the integers $\sigma'(k)$ and $\sigma(k)$ belong to the orbit O. Then, there exists $\tau \in L$ such that $\tau(\sigma(k)) = \sigma'(k)$, thus $\sigma^{-1}(\tau^{-1}(\sigma'(k))) = k$. The permutation $\rho = \sigma^{-1}\tau^{-1}\sigma'$ belongs to G_{k-1} (like σ, τ and σ') and fixes k. Then $\rho \in L_k \subset L$ because

$$G_k = Fix_{G_{k-1}}(k).$$

Thus, $\sigma' = \tau\sigma\rho$ and σ' is a product of σ and of two elements of L. □

Remark 3.2 We take the notations of Proposition 3.1. Let a be $Min(O)$, the minimal integer of O. It can be easily proved that, if $\mathcal{E}$ is not empty, there exists $\sigma \in \mathcal{E}$ such that $\sigma(k) = a$. Therefore, searching a permutation of $\mathcal{E}$ can be restricted to searching the one sending k to a.

The following lemma is a consequence of Lagrange's Theorem:

Lemma 3.3 *Let L be a subgroup of S_n verifying $G_k \subseteq L \subseteq G_{k-1}$. Then:*

$$Card(L) = Card(G_k)\,.\,Card(Orb_L(k))\ .$$

Proposition 3.4 gives conditions to test the equality $L = G_{k-1}$ (see Algorithm 3.8). Its proof and also the proof of theoreme 3.4 are two simple consequences of Lemma 3.3.

Proposition 3.4 *Let L be a subgroup of S_n verifying $G_k \subseteq L \subseteq G_{k-1}$. Then*

1. *either, no L-orbit of $\{1, \ldots, n\}$ is included in $\{k+1, \ldots, n\}$ and, in this case, $L = G_{k-1}$.*

2. *or, let* $\mathcal{O} = \{O_1, \ldots, O_r\}$ *be the non empty set of the* L*-orbits of* $\{1, \ldots, n\}$ *included in* $\{k+1, \ldots, n\}$*; if, for all* $i \in \{1, \ldots, r\}$*, there is no* $\sigma \in G_{k-1}$ *such that* $\sigma(k) \in O_i$ *then* $L = G_{k-1}$ *else* $L \neq G_{k-1}$.

To construct a group chain between G_k and G_{k-1}, we need to determine the orbits of $\{1, \ldots, n\}$ under the action of the subgroup $L' = \langle\{\sigma\} \cup L\rangle$ mentioned in Propositions 3.1. Proposition 3.5 allows this determination from the L-orbits.

Proposition 3.5 *Let* $\mathcal{O}$ *be the set of the orbits of* $\{1, \ldots, n\}$ *under the action of the subgroup* L *of* S_n *and* $O \in \mathcal{O}$*. Let* σ *be a permutation of* S_n *and denote by* L' *the subgroup generated by* $\{\sigma\} \cup L$.

Let $(E_r)_{r\in\mathbb{N}}$ *and* $(P_r)_{r\in\mathbb{N}}$ *be the sequences recursively defined by:*

- $E_1 = O$ *and* $P_1 = (\sigma.E_1) \cup E_1$*;*
- *For all* $k \in \mathbb{N}^*$,

$$E_{k+1} = \cup_{\{O' \in \mathcal{O} \mid O' \cap P_k \neq \emptyset\}} O' \text{ and } P_{k+1} = (\sigma.E_{k+1}) \cup E_{k+1} .$$

Then, the sequence $(E_k)_{k\in\mathbb{N}^*}$ *is stationary from an index* k_0 *on and the set* E_{k_0} *is the orbit of* $\{1, \ldots, n\}$ *under the action of* L' *such that* $O \subset E_{k_0}$.

Proof. Since the sub-sequences $(E_k)_{k\in\mathbb{N}^*}$ and $(P_k)_{k\in\mathbb{N}^*}$ of $\{1, \ldots, n\}$ are increasing for inclusion, they are stationary from an index k_0 on. It can be easily proved that E_{k_0} is stable under both σ and L actions. Then, E_{k_0} can be written as a union of L'-orbits. By trivial recurrence on k, each E_k is included in the L'-orbit containing O. Consequently, E_{k_0} is the L'-orbit containing O. □

3.3. Algorithm for Computing a Generating set of G_0

We define a function `NewOrbits` witch determines the orbits of $\{1, \ldots, n\}$ under the action of $L' = \langle L \cup \{\sigma\}\rangle$. They are computed by generating the sequences $(E_k)_{k\in\mathbb{N}^*}$ and $(P_k)_{k\in\mathbb{N}^*}$ mentioned in Proposition 3.5 .

Algorithm 3.6 (Synopsis)

Function `NewOrbits` (*orbits*,σ)

/*

Input : . *orbits*, the set of orbits of $\{1, \ldots, n\}$ under the action of L, a subgroup of S_n.
. σ a permutation of S_n.

Output : . the set of orbits of $\{1, \ldots, n\}$ under the action of $\langle L \cup \{\sigma\}\rangle$.

*/

Consider `FindAPermutation`, the function based on the same algorithm than the function `ConstructionOfPermutations` (see Algorithm 2.1), returning a permutation of the group G_0, when it exists, and the identity otherwise. Note that, the formal parameter G appearing in `ConstructionOfPermutations` which represents a set of permutations is replaced in `FindAPermutation` by a parameter representing a permutation.

The next function `From_Gk_to_G(k-1)` constructs inductively from G_k a finite and increasing sequence of groups:

$$G_k = L_0 < L_1 < \cdots < L_m = G_{k-1}$$

each group is represented by a generating set of permutations.

Let L be one of the groups L_i, where $i \in [\![0, m]\!]$, represented by $\mathcal{G}$, a generating set. This function determines, when it exists, a permutation of $G_{k-1} \backslash L$ which, together the elements of $\mathcal{G}$, make up a generating set of a new group $L_{i+1} = L'$. And so on, until no new permutation can be found and, in this case, $L = G_{k-1}$.

The explicit method to compute L' from L is described below.

Let $(O_1, \ldots, O_r)$ be the L-orbits of $\{1, \ldots, n\}$. Then:

Case 1. None of the orbits is included in $\{k+1, \ldots, n\}$; then $L = G_{k-1}$ (case 1. Proposition 3.4).

Case 2. Let $O'_1, \ldots, O'_s$ be the L-orbits included in $\{k+1, \ldots, n\}$; the function `From_Gk_to_G(k-1)` tries to find an integer i in $[\![1, s]\!]$ and a permutation $\sigma \in G_{k-1} \backslash G_k$ verifying $\sigma(k) = Min(O'_i)$. The determination of such a permutation is done by the call:

$$\texttt{FindAPermutation}(k+1, [1, \ldots, k, Min(O'_i)], Id, \mathcal{P}).$$

For $i \in [\![1, s]\!]$, we set $\mathcal{E}_i = \{\sigma \in G_{k-1} \backslash G_k \mid \sigma(k) \in O'_i\}$. There are two sub-cases:

Case 2.1. For all $i \in [\![1, s]\!]$, there is no permutation $\sigma \in G_{k-1} \backslash G_k$ verifying $\sigma(k) = Min(O'_i)$; from Remark 3.2, this is equivalent to

$$\forall i \in [\![1, s]\!],\ \mathcal{E}_i = \emptyset.$$

Then $L = G_{k-1}$ (case 2. Proposition 3.4).

Case 2.2. There exists $i_0 \in [\![1, s]\!]$ such that $\sigma(k) = Min(O'_{i_0})$. In this case, the group $L' = \langle L \cup \mathcal{E}_{i_0} \rangle$ is generated by the set of permutations $\mathcal{G}' = \mathcal{G} \cup \{\sigma\}$ (see Proposition 3.1).

If $L = G_{k-1}$, the process is finished. Otherwise, Function `From_Gk_to_G(k-1)` is recursively called with, as new arguments, the set of permutations $\mathcal{G}'$ and the L'-orbits of $\{1, \ldots, n\}$ determined by using the function `NewOrbits`.

Algorithm 3.7

Function `From_Gk_to_G(k-1)`$(k, \mathcal{G}, orbits, \mathcal{P})$
/*
Input : . k, the index of the group G_k.
. $\mathcal{G}$, the list used to stock the elements of a generating set of G_{k-1} and equals to a generating set of G_k at the first call.
. $orbits$, the set of the orbits of $\{1, \ldots, n\}$ under the action of G_k.
. The condition set $\mathcal{P} = (P_1, \ldots, P_n)$ described in Section 2.

Output : . $\mathcal{G}$, a generating set of G_{k-1}.
. $orbits$, the set of the orbits of $\{1, \ldots, n\}$ under the action of G_{k-1}.
*/
$elts := \{\mathrm{Min}(O) \mid O \in orbits$ and $O \subset \{k+1, \ldots, n\}\}$;
While $elts \neq \emptyset$ **Do**
. $a :=$ Min $(elts)$;
. $elts := elts \setminus \{a\}$;
. **If** $P_k(1, 2, \ldots, k-1, a)$ **Then** $(Condition\ C)$
. . $\sigma :=$ `FindAPermutation`$(k+1, [1, 2, \ldots, k-1, a], Id, \mathcal{P})$;
. . **If** $\sigma \neq Id$ **Then**
. . . $\mathcal{G} := \mathcal{G} \cup \{\sigma\}$;
. . . $orbits :=$ `NewOrbits`$(orbits, \sigma)$;
. . . $elts := \{\mathrm{Min}(O) \mid O \in orbits$ and $O \subset \{k+1, \ldots, n\}\}$;
. . **End If**;
. **End If**;
End While;
Return $\mathcal{G}, orbits$;
End Function;

The following function constructs the increasing and finite sequence of groups

$$\{Id\} = G_n < G_{n-1} < \cdots < G_2 < G_1 < G_0.$$

Each generating set of these groups is computed by `From_Gk_to_G(k-1)`. In addition, the decomposition group order is computed ; this will be used in Sections 4 and 5.

Algorithm 3.8

Function `Generators`$(\mathcal{P})$
/*
Input : . The condition set $\mathcal{P}$ described in Section 2.

Output : . The integer, $Cardinal$, which is the order of $Dec(I)$;
. A list $\mathcal{G}$ of generators of the group $Dec(I)$.
*/
$\mathcal{G} := \{Id_{S_n}\}$;
$orbits := \{\{1\}, \ldots, \{n\}\}$;
$k := n - 1$;
$Cardinal := 1$;
While $k \neq 0$ **Do**
. $\mathcal{G}, orbits :=$ `From_Gk_to_G(k-1)`$(k, \mathcal{G}, orbits, \mathcal{P})$;

. $Cardinal := Card(Orb_{G_k}(k+1)) * Cardinal$;
. $k := k-1$;
End While ;
Return $Cardinal, \mathcal{G}$;
End Function;

Remark 3.9 The computation of $Card(G_0)$ uses the equality which is the direct consequence of the Lemma 3.3:

$$Card(G_0) = \prod_{i=0}^{n-1} Card(Orb_{G_i}(i+1)) \ .$$

Remark 3.10 A straightforward recurrence shows that, for each step,

$$Card(\mathcal{G}) + Card(orbits) = n + 1.$$

Hence, the cardinal of the generating set returned by this algorithm is at most n.

Example 3.11 Let I_{6T3} be the ideal of Example 2.2.

The recursive algorithm 3.8 returns the list $[Id, (5,6), (3,4), (1,2)(3,5)(4,6)]$ by running through the partial tree of Figure 2. It does 32 membership tests to the ideal I_{6T3}; in Algorithm 2.1, 64 were necessary.

4. Decomposition Group and Galois Ideals

Definition 4.1 *An ideal is said to be triangular if it is radical and generated by a triangular set of generators.*

Let $\hat{K}$ be an algebraic closure of K. Let from now on I be triangular. Let $V(I)$, its variety (i.e. the set of its zeros in $\hat{K}^n$). Then its cardinal $\Pi(I)$ is:

$$\Pi(I) = \prod_{i=1}^{n} deg_{X_i}(f_i),$$

(see [2] for example). Proposition 4.2 is a direct consequence of this equality.

Proposition 4.2 *Let I be a triangular ideal. The decomposition group $Dec(I)$ acts faithfully on the variety $V(I)$ and:*

$$Card(Dec(I)) \leq \Pi(I)\,. \tag{4.1}$$

When equality holds, the variety $V(I)$ is uniquely determined by $\underline{\alpha} \in \hat{K}^n$ any of its elements and by $Dec(I)$ because:

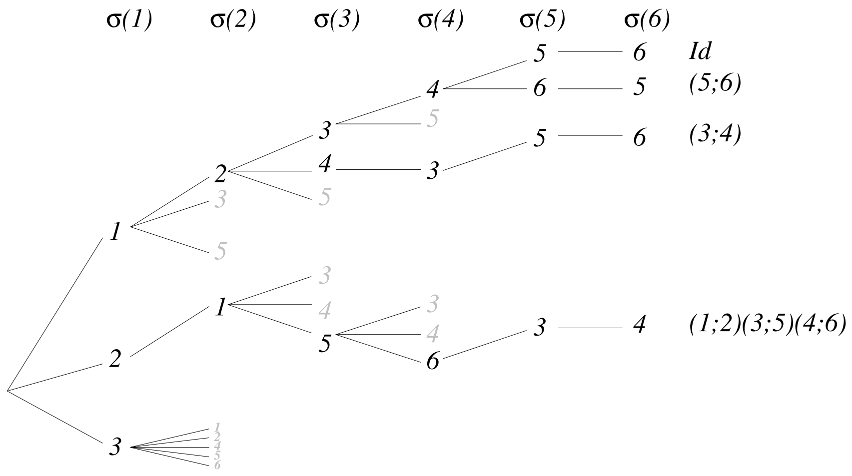

Fig. 2. Partial tree

$$V(I) = Dec(I).\underline{\alpha} \quad .$$

In this case, we say that I is a *pure Galois ideal*.

We are interested in testing whether I is a pure Galois ideal and then to compute $Dec(I)$. Algorithm 3.8 can be used to do this task but so much useless computation is done. Hereafter, we show how to specialize this algorithm for this particular problem.

A first observation is that I is a pure Galois ideal if it is at least a Galois ideal:

Definition 4.3 *Let I be an ideal of $K[X_1, \ldots, X_n]$ and $\underline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ in $V(I)$. The ideal I is said to be an $\underline{\alpha}$-Galois ideal if the two following conditions hold:*

(1) if $i \neq j$ then $\alpha_i \neq \alpha_j$;
(2) there exists L, a subset of S_n such that

$$I = \{f \in K[X_1, \ldots, X_n] \mid f(\sigma.\underline{\alpha}) = 0 \ \forall \sigma \in L\}.$$

*Such an ideal is denoted by $I_{\underline{\alpha}}^L$ and the set L is called its $\underline{\alpha}$-*injector *if L is the maximal set satisfying condition (2).*

Definition 4.4 *Let σ be a permutation of S_n and $t \in [\![1, n]\!]$. The first t-part of σ is the sequence $(\sigma(1), \ldots, \sigma(t))$.*

Theorem 4.5 *Let I be a Galois ideal generated by the triangular set:*

$$\mathcal{T} = \{f_1, f_2, \ldots, f_n\} \quad .$$

Let $\underline{\alpha}$ be a zero of I and L its $\underline{\alpha}$-injector. Let $t \in [\![1, n-1]\!]$ and $\{c_1, \ldots, c_t\}$ be a t-subset of $\{1, \ldots, n\}$. Let D be the product $deg_{x_{t+1}}(f_{t+1}) \cdots deg_{x_n}(f_n)$.
For all $i \in [\![1, t]\!]$, $f_i(\alpha_{c_1}, \ldots, \alpha_{c_i}) = 0$ if and only if there exists an element $\sigma \in L$ with $(c_1, \ldots, c_t)$ as first t-part.
More precisely, there are exactly D many of these permutations in L.

Proof. Let t be an element of $[\![1, n-1]\!]$. Since the variety V of I is equiprojectable (see [2]), each element $\underline{\beta}$ in the variety of the ideal $\langle f_1, f_2, \ldots, f_t\rangle$ is the projection on the first t coordinates of D elements in V.
The one-to-one map between V and L gives the result. □

Remark 4.6 In the particular case where I is a maximal Galois ideal, we obtain, as a corollary, Theorem 5 of [1].

The following proposition is the key for the improvement of Algorithm 3.8 in order to test if I is a pure Galois ideal and then to compute its decomposition group. We say that Algorithm 3.7 made a *backtrack* when a permutation σ verifying the (Condition C) cannot be continued, i.e. `FindAPermutation` returns Id.

Proposition 4.7 *Let I be a triangular ideal generated by S, a triangular set. If the function* `Generators` *produces a backtrack in one of* `From_Gk_to_G(k-1)` *calls, then I is no pure Galois.*

Proof. In Algorithm 3.7 a backtrack appears when a first t-part $(c_1, \ldots, c_t)$, verifying $\forall i \in [\![1, t]\!]$ $f_i(\alpha_{c_1}, \ldots, \alpha_{c_i}) = 0$, can not be completed in $(c_1, \ldots, c_{t+1})$, a first $(t+1)$-part such that $f_{t+1}(\alpha_{c_1}, \ldots, \alpha_{c_{t+1}}) = 0$. In other words, the algorithm has found a permutation $\sigma \notin Dec(I)$ such that its first t-part verifies the above condition. By Theorem 4.5, it is possible only if I is not a Galois ideal or I is a Galois ideal such that $Dec(I)$ is not an injector of I. The result follows. □

Let I be a triangular ideal. During $Dec(I)$ computation with Algorithm 3.8, two cases happen:

(1) a backtrack appears and I is no pure Galois,
(2) otherwise, I is a pure Galois if and only if $Card(Dec(I)) = \Pi(I)$.

In order to test whether a triangular ideal I is a pure Galois ideal, an algorithm called `IsPureGaloisIdeal` can be derived from Algorithm 3.8 by testing the backtracking condition during computation.

Application in Galois theory

In general case, an injector of I can be computed only when a maximal ideal $\mathcal{M}$ containing I is known. But, when I is a pure Galois ideal, this injector is unique and equals its decomposition group (computed by Algorithm `IsPureGaloisIdeal`). Injectors are needed in Algorithm `GaloisIdeal` of [9] which computes $\mathcal{M}$. The ideal $\mathcal{M}$ is needed because the splitting field of polynomial $\prod_{i=1}^{n}(x-\alpha_i)$, $\underline{\alpha} \in V(\mathcal{M})$, is isomorphic to $k[x_1, x_2, \ldots, x_n]/\mathcal{M}$. When a Galois ideal is not pure this information can be used in $\mathcal{M}$ computation (see [8]).

5. Comparisons of Algorithms

5.1. Complexity

In this section, we study efficiency of Algorithm `IsPureGaloisIdeal`. As its total cost is dominated by the cost of normal forms computation, we evaluate this efficiency by the following bound:

Proposition 5.1 *Let $\langle S \rangle$ be a triangular ideal of $K[X_1, \ldots, X_n]$. The number of normal forms computations in Function* `IsPureGaloisIdeal(`S`)` *is bounded by $O(n^3)$.*

Proof. We can bound the number of normal forms computations by the one needed to compute $Dec(I)$ in the worst case, i.e. when the hypothesis:

$\mathcal{H}$: *no backtrack is realized during the calculation*

is verified. In Algorithm `IsPureGaloisIdeal`, all normal forms are computed in the different calls of function `From_Gk_to_G(k-1)`.
To begin with, we study the former function complexity.
To one call of Function `From_Gk_to_G(k-1)`, one normal form is computed to test Condition C of Algorithm 3.7. Next, there are two cases:

- C is false and no other computation is done. Then, only one normal form is computed. Moreover, to one call of Function `From_Gk_to_G(k-1)`, this case appears at most n times ;
- C is true. Then, Function `FindAPermutation` is called and the hypothesis $\mathcal{H}$ insures that a new element is returned and added to Parameter $\mathcal{G}$ of `From_Gk_to_G(k-1)`. Theorem 4.5 allows a straightforward complexity analysis of this function: the number of normal forms needed for computing such a generator is bounded by $O(n^2)$.

Function `IsPureGaloisIdeal` calls $n-1$ times Function `From_Gk_to_G(k-1)`. Thus, the normal forms computations number corresponding to "(Condition C) is false" is bounded by $O(n^2)$.

The cardinal of Parameter $\mathcal{G}$ is bounded by n (see Remark 3.10); thus Condition C is true at most n times. Hence, during the execution of Algorithm

IsPureGaloisIdeal, the normal forms computations number corresponding to "(Condition C) is true" is bounded by $n\ O(n^2)$.

The Algorithm IsPureGaloisIdeal complexity evaluated in term of normal forms number is bounded by $O(n^2) + n\ O(n^2) = O(n^3)$. □

5.2. *Heuristic Comparisons*

In this section, we adopt Butler and McKay notation nT_i for transitive subgroups of S_n (see [6]).

All algorithms have been implemented using the MAGMA software (see [3]). We denote by $f_{n,i}$ the polynomial which nT_i Galois group presents in galpol package. We use some triangular Galois ideal $I_{n,i}$ of the polynomial $f_{n,i}$ computed by using the technique described in [8].

To establish the following table, we compare the number of membership tests to each $I_{n,i}$ realized by Algorithms 2.1 and 3.8 for $Dec(I_{n,i})$ determination.

The symbol $*$ means that I is a pure Galois ideal. In this case, the decomposition group of $I_{n,i}$ is also its injector (see Section 4).

Table 2 compares the tests number done by Algorithm 2.1, Algorithm 3.8 and the one called STRONG_GENERATORS by Anai, Noro and Yokoyama (see [1]). Since this last algorithm can only be applied to maximal Galois ideals, comparisons concern this very case. In Table 2, we write nT_i instead of $I_{n,i}$ because the $I_{n,i}$ decomposition group equals, up to an isomorphism, the Galois group of $f_{n,i}$.

We can see that Algorithm STRONG_GENERATORS computes around 5 times more membership tests than Algorithm 3.8.

It is possible to improve the algorithms of this paper by using some modular method for membership tests. In most of the preceding examples, those modular pre-tests reduce the time of computation by a factor 20.

Table 1. Comparisons between Algorithms 2.1 and 3.8

Ideal	$Card(Dec(I))$	Algorithm 2.1	Algorithm 3.8
$I_{7,1}*$	7	154	31
$I_{7,2}$	8	115	49
$I_{6,4}*$	24	144	28
$I_{9,4}$	24	258	52
$I_{7,3}$	36	193	38
$I_{6,10}*$	72	264	30
$I_{6,13}*$	72	264	30
$I_{9,28}*$	648	2637	64
$I_{6,12}*$	720	1956	20
$I_{9,20}$	2160	5970	42
$I_{7,4}*$	5040	13699	27
$I_{7,5}*$	5040	13699	27
$I_{7,6}*$	5040	13699	27

Table 2. Comparisons in the case of relations ideals

nT_i	$Card(nT_i)$	Algorithm 2.1	Algorithm 3.8	STRONG_GENERATORS
8T1	8	232	65	224
8T2	8	232	83	224
8T3	8	232	83	224
8T6	16	352	62	299
8T7	16	280	68	291
8T8	16	352	64	298
8T9	16	280	69	292
8T10	16	280	69	291
8T11	16	280	57	292
8T12	24	472	63	328
8T13	24	472	63	329
8T14	24	472	59	329
8T16	32	448	71	324
8T19	32	448	76	325
8T22	32	448	66	324
8T24	48	712	59	354
8T27	64	480	47	327
8T29	64	640	64	377
8T31	64	480	63	331
8T47	1152	3520	34	387
8T50	40320	40320	35	463

6. Conclusion

We present a method to compute the decomposition group of a triangular ideal. This new method can be applied, not only to a maximal Galois ideal as in Algorithm STRONG_GENERATORS of [1], but also to Galois ideals. Theorem 4.5 generalizes Theorem 5 of [1] to Galois ideals. We show that the complexity of this new method in case of pure Galois ideals have a better bound than the one given in [1] for the specific case of maximal Galois ideals. The heuristic comparison of Algorithms in Table 2 shows that Algorithm 3.8 computes less normal forms for the decomposition group determination than the two others algorithms.

Acknowledgments. The authors would like to thank the anonymous referees for extensive comments on clarifying and improving the exposition of this paper and Laurence Bessis for her helpful suggestions and comments.

References

1. Anai, H., Noro, M., Yokoyama, K.: Computation of the splitting fields and the Galois groups of polynomials. In: Algorithms in algebraic geometry and applications (Santander, 1994), volume 143 of Progr. Math. Birkhäuser, Basel, 1996, pp. 29–50

2. Aubry, P., Valibouze, A.: Using Galois ideals for computing relative resolvents. J. Symbolic Comput. **30**(6), 635–651 (2000). Algorithmic methods in Galois theory
3. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24**(3–4), 235–265 (1997). Computational algebra and number theory (London, 1993)
4. Bourbaki, N.: Algèbre Commutative. Chapitres 5 à 7. Éléments de mathématiques. Masson, 1985
5. Butler, G.: Fundamental algorithms for permutation groups, volume 559 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1991
6. Butler, G., McKay, J.: The transitive groups of degree up to eleven. Comm. Algebra **11**(8), 863–911 (1983)
7. Cox, D., Little, J., O'Shea, D.: Ideals, varieties, and algorithms. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1997. An introduction to computational algebraic geometry and commutative algebra
8. Orange, S., Renault, G., Valibouze, A.: Calcul efficace d'un corps de décomposition. LIP6 Research Report 2003.005, LIP6, Université Pierre et Marie Curie, France, 2003
9. Valibouze, A.: Étude des relations algébriques entre les racines d'un polynôme d'une variable. Bull. Belg. Math. Soc. Simon Stevin **6**(4), 507–535 (1999)