Rev. Mat. Iberoamericana **19** (2003), 367–384

Computation of Centralizers in Braid groups and Garside groups

Nuno Franco and Juan González-Meneses

Abstract

We give a new method to compute the centralizer of an element in Artin braid groups and, more generally, in Garside groups. This method, together with the solution of the conjugacy problem given by the authors in [9], are two main steps for solving conjugacy systems, thus breaking recently discovered cryptosystems based in braid groups [2]. We also present the result of our computations, where we notice that our algorithm yields surprisingly small generating sets for the centralizers.

Introduction

Given a group G, the *centralizer* of an element $a \in G$, denoted $Z(a)$, is the subgroup of G consisting of all elements which commute with a. Our goal in this paper is to give a good algorithm to compute a generating set for the centralizer of an element in a *Garside group.*

Garside groups were introduced by Dehornoy and Paris [7] (their original name was *small Gaussian groups*, but there has been a convention to call them Garside groups). We will consider Artin braid groups [1] as the main examples of Garside groups.

Given an integer $n \geq 2$, the *braid group on n strands*, B_n, is defined by the following presentation:

$$(1)\quad B_n = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j = \sigma_j\sigma_i & (|i-j| \geq 2) \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} & (i = 1, \ldots, n-2) \end{array} \right\rangle .$$

2000 Mathematics Subject Classification: Primary 20F36; Secondary 20F10, 94A60.
Keywords: Braid group, Garside group, centralizer, cryptography.

Braid groups are of interest not only in Combinatorial Group Theory, but also in Low Dimensional Topology and, more recently, in Cryptography. Other examples of Garside groups are spherical (finite type) Artin groups [5] and torus knot groups, among others.

Computing centralizers in a Garside group is of interest in itself, but can also be applied to solve other questions. For instance, consider two elements a, b in a Garside group G. Suppose that we know an element $c \in G$ that conjugates a to b, that is, $c^{-1}ac = b$. Consider then the set $Z_{a,b} = c\,Z(b) = \{c\alpha : \alpha \in Z(b)\} \subset G$. Then $Z_{a,b}$ is the set of all elements in G that conjugate a to b: Indeed, an element $d \in G$ conjugates a to b if and only if $d^{-1}ad = b$, then $b = d^{-1}(cc^{-1})a(cc^{-1})d = (d^{-1}c)b(c^{-1}d)$, so $c^{-1}d \in Z(b)$; hence $d \in Z_{a,b}$.

This property may be used for solving *conjugacy systems* in Garside groups: Given $a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_k \in G$, find an element $c \in G$ such that $c^{-1}a_ic = b_i$, for $i = 1, \ldots, k$. The solutions of such a system are the elements in $Z_{a_1,b_1} \cap \cdots \cap Z_{a_k,b_k}$. These kind of problems play a central role in some new public-key cryptosystems (see [2] and [12]), based on braid groups. To break such cryptosystems, one must solve a conjugacy system such as the previous one.

The conjugacy problem in braid groups has been solved by Garside [10], and his algorithm has been improved in [8] and generalized to all Garside groups in [16]. In [9], the authors gave a more efficient algorithm than all the above, to solve the conjugacy problem in all Garside groups. So, given two conjugated elements $a, b \in B_n$, we know how to find an element $c \in G$ such that $c^{-1}ac = b$. Using the algorithm that we shall explain in this paper, we can compute a generating set for $Z(b)$, hence we know how to generate elements in $Z_{a,b}$. We still do not know how to compute an element in $Z_{a_1,b_1} \cap \cdots \cap Z_{a_k,b_k}$ even if we know how to generate elements in each Z_{a_i,b_i}. We believe that a deeper study of the structure of centralizers in Garside groups will provide a solution to this problem. Anyway, we think that the algorithm we give to compute centralizers in Garside groups is a good step towards the solution of these systems.

There exists another algorithm to compute the centralizer of an element in braid groups, which was given by Makanin [14]. It can be easily generalized to all Garside groups, but it is a fairly theoretical algorithm, which has a huge complexity and gives a large amount of redundant generators. One could also make use of the bi-automatic structure of Garside groups [6] to find the centralizer of an element. But this also seems quite inefficient. The new method that we introduce is quite simple and surprisingly efficient. Actually, the generating sets obtained in our computations with braid groups are so small, that they led us to conjecture that the centralizer of any braid in B_n can be generated by no more than $n - 1$ elements.

After writing an early version of this paper, we were told by M. Korkmaz of a family of counterexamples to this conjecture, due to N. V. Ivanov (the smallest counterexample belongs to B_9, while our computations were up to B_8). Nevertheless, it has been recently proven by the second author and Bert Wiest [11] that, for $a \in B_n$, $Z(a)$ can be generated by less than $\frac{k(k+1)}{2}$ elements if $n = 2k$ and $\frac{k(k+3)}{2}$ elements if $n = 2k+1$.

The algorithm in this paper works as follows: given an element a in a Garside group G, it constructs a graph Γ associated to a, such that the fundamental group of Γ maps onto $Z(a)$. Then it computes a generating set for the fundamental group of Γ, which maps to a generating set for $Z(a)$.

This paper is structured in the following way: In Section 1 we give the basic definitions and results concerning Garside groups. In Section 2, we introduce a special kind of elements, the *minimal simple elements*, which are used to construct the graph Γ. This graph is studied in Section 3. We explain our algorithm in detail in Section 4, then we study its complexity in Section 5 and, finally, in Section 6 we present the results obtained by implementing the algorithm.

1. Garside groups and simple elements

In this section we will give the definitions of Garside monoids and groups, and the basic results which we shall need to present our algorithm. To find the proofs of the results, and more details, see [10], [8], [17], [3], [7], [6] and [15].

Consider a cancellative monoid M, with no invertible elements. We can define a partial order on its elements, called the *prefix order*, as follows: For $a, b \in M$, we say that $a \prec b$ if b can be written in such a way that a is a prefix of b, that is, if there exists $c \in M$ such that $ac = b$. In this case, we say that a is a left divisor of b. There also exists the *suffix* order, but we will not use it in this paper, so in the above situation we will just say that a divides b, or that b is a multiple of a.

Given $a, b \in M$, we can naturally define their (left) *least common multiple*, $a \vee b$, and their (left) *greatest common divisor*, $a \wedge b$, if they exist. That is, $a \vee b$ is the minimal element (with respect to $\prec$) such that $a \prec a \vee b$ and $b \prec a \vee b$. In the same way, $a \wedge b$ is the maximal element (with respect to $\prec$) such that $a \wedge b \prec a$ and $a \wedge b \prec b$.

Definition 1 *Let M be a monoid. We say that $x \in M$ is an* atom *if $x \neq 1$ and if $x = yz$ implies $y = 1$ or $z = 1$. M is said to be an* atomic monoid *if it is generated by its atoms and, moreover, for every $a \in M$, there exists an integer $N_a > 0$ such that a cannot be written as a product of more than N_a atoms.*

Definition 2 *We say that a monoid M is a* Gaussian monoid *if it is atomic, (left and right) cancellative, and if every pair of elements in M admits a (left and right) l.c.m. and a (left and right) g.c.d.*

Definition 3 *A* Garside monoid *is a Gaussian monoid which has a* Garside element. *A* Garside element *is an element $\Delta \in M$ whose left divisors coincide with their right divisors, they form a finite set, and they generate M.*

Definition 4 *The left (and right) divisors of Δ in a Garside monoid M are called simple elements. We denote by S the (finite) set of simple elements.*

It is known that every Garside monoid admits a group of fractions, and we have:

Definition 5 *A group G is called a* Garside group *if it is the group of fractions of a Garside monoid.*

The main example of a Garside monoid, as with groups, is the Artin braid monoid on n strands, B_n^+. It is defined by Presentation (1), considered as a presentation for a monoid. Its group of fractions is the braid group B_n, and Garside [10] showed that $B_n^+ \subset B_n$. Actually, every Garside monoid embeds into its corresponding Garside group [7].

Braids in B_n are usually represented as n disjoint strands in $\mathbb{R}^3$, whose endpoints are fixed, where every horizontal plane between the top and the bottom level intersects each strand in exactly one point, as in Figure 1. Simple elements in B_n^+ are easy to recognize: they are those braids in which any two strands cross at most once. The Garside element of B_n^+ is $\Delta = (\sigma_1\sigma_2 \cdots \sigma_{n-1})\,(\sigma_1\sigma_2 \cdots \sigma_{n-2}) \cdots (\sigma_1\sigma_2)\,\sigma_1$, and is represented in Figure 1 for $n = 4$ (where, as usual, σ_i represents a crossing of the strands in positions i and $i+1$).

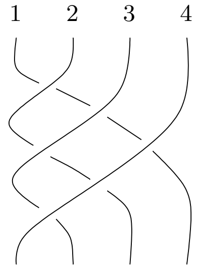

Figure 1: The Garside element $\Delta \in B_4^+$.

There is another important example of Garside monoid, the Birman-Ko-Lee monoid [3], which has the following presentation:

$$BKL_n^+ = \left\langle a_{ts}\,(n \geq t > s \geq 1) \;\middle|\; \begin{array}{l} a_{ts}a_{rq} = a_{rq}a_{ts} \text{ if } (t-r)(t-q)(s-r)(s-q) > 0 \\ a_{ts}a_{sr} = a_{tr}a_{ts} = a_{sr}a_{tr}, \text{ where } n \geq t > s > r \geq 1 \end{array} \right\rangle$$

Its group of fractions is again the braid group B_n. The Garside element in BKL_n^+ is $\delta = a_{n,n-1}a_{n-1,n-2}\cdots a_{2,1}$. In this monoid we can perform some computations concerning braid groups faster than using Artin monoids. Anyway, using the algorithm in [9], the conjugacy problem has virtually the same complexity in both monoids.

From now on, M will denote a Garside monoid, G its group of fractions and Δ the corresponding Garside element. Since $M \subset G$, we will refer to the elements in M as the *positive* elements of G.

From the existence of l.c.m.'s and g.c.d.'s, it follows that $(M, \prec)$ has a lattice structure, and S becomes a finite sublattice with minimum 1 and maximum Δ. In Figure 2 we can see the Hasse diagram of the lattice of simple elements in B_4^+, where the lines represent left divisibility (from bottom to top).

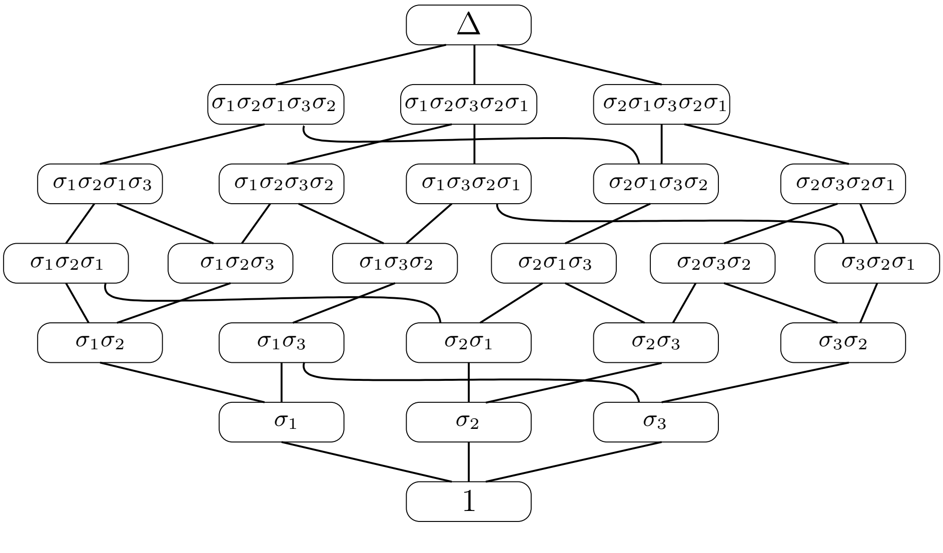
Δ
$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2$
$\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1$
$\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1$
$\sigma_1\sigma_2\sigma_1\sigma_3$
$\sigma_1\sigma_2\sigma_3\sigma_2$
$\sigma_1\sigma_3\sigma_2\sigma_1$
$\sigma_2\sigma_1\sigma_3\sigma_2$
$\sigma_2\sigma_3\sigma_2\sigma_1$
$\sigma_1\sigma_2\sigma_1$
$\sigma_1\sigma_2\sigma_3$
$\sigma_1\sigma_3\sigma_2$
$\sigma_2\sigma_1\sigma_3$
$\sigma_2\sigma_3\sigma_2$
$\sigma_3\sigma_2\sigma_1$
$\sigma_1\sigma_2$
$\sigma_1\sigma_3$
$\sigma_2\sigma_1$
$\sigma_2\sigma_3$
$\sigma_3\sigma_2$
σ_1
σ_2
σ_3
1

Figure 2: The lattice of simple elements in B_4^+.

We end this section with an important result concerning Garside groups.

Theorem 6 [7] *For every element a in a Garside group G, there exists a unique word in the atoms of G (and their inverses) representing a, called the* normal form *of a, and there exists an algorithm that, given a word w in the atoms and their inverses, computes the normal form of the element represented by w.*

2. Minimal simple elements

Simple elements represent a key concept in almost every algorithm concerning Garside groups (or braid groups): they have been used to compute bi-automatic normal forms in [17] and [6], to solve the conjugacy problem in [16], [8] and [3], and to compute centralizers in [14]. In some cases, the complexity of these algorithms is too big due to the size of the set S. For instance, in B_n^+, the cardinal of S is $n!$, and this makes the algorithm in [8] work too slowly. This problem was avoided in [9], by considering *minimal simple elements*. We will also use minimal simple elements in this paper, so this section is devoted to them.

Given an element a in a Garside group G, there exists a subset $C^{sum}(a)$ of the conjugacy class of a, called *Summit Class of* a, satisfying some suitable properties. In [8], when talking about braids, this subset is called *Super Summit Set*, but when we talk about Garside groups we prefer to use the terminology in [16]. Roughly speaking, $C^{sum}(a)$ is the set of conjugates of a having the 'simplest' normal form, in a certain sense. Hence, $C^{sum}(a)$ is an invariant of the conjugacy class of a (it does not depend on a, but on its conjugacy class).

There exists a procedure, called 'cycling and decycling', to obtain an element $a' \in C^{sum}(a)$ and an element x such that $x^{-1}ax = a'$ (see [8]). The centralizers of a and a' are then related as follows: $Z(a) = xZ(a')x^{-1}$. Hence, if we know a generating set for $Z(a')$, we obtain immediately a generating set for $Z(a)$, with the same number of elements: it suffices to conjugate every generator by x. Therefore, we will just study the elements in the Summit Class of a.

Consider an element $v \in C^{sum}(a)$. If we conjugate v by a nontrivial simple element, we obtain an element in G, that may or may not be in $C^{sum}(a)$. We will consider just the elements in $S\backslash\{1\}$ that conjugate v to an element in $C^{sum}(a)$. Among these simple elements, we take those which are minimal with respect to $\prec$, and we call this set S_v^{sum}. In other words, we define S_v^{sum} as the set of minimal elements (with respect to $\prec$) in $\{s \in S\backslash\{1\} : s^{-1}vs \in C^{sum}(a)\}$.

There are two important results concerning these minimal simple elements:

Proposition 7 [9] *Let M be a Garside monoid with t atoms, G its corresponding Garside group, and $a \in G$. For every $v \in C^{sum}(a)$, the cardinal of S_v^{sum} is no bigger than t.*

Proposition 8 [9] *Let u, v be two conjugate elements in $C^{sum}(a)$. Then there exists a sequence $u = u_1, u_2, \dots, u_k = v$ of elements in $C^{sum}(a)$ such that, for $i = 1, \dots, k-1$, there exists $s_i \in S_{u_i}^{sum}$ verifying $u_is_i = s_iu_{i+1}$.*

We will represent the above property as follows:

$$u = u_1 \xrightarrow{s_1} u_2 \xrightarrow{s_2} u_3 \rightarrow \cdots \rightarrow u_{k-1} \xrightarrow{s_{k-1}} u_k = v,$$

where $s_i \in S^{sum}_{u_i}$ for every i, and the arrow means conjugation by the corresponding s_i. We call such a sequence a *minimal chain* from u to v.

Example 1 *Consider the braid monoid B_4^+. As we saw in the previous section, the set of simple elements in B_4^+ has* 24 *elements (see Figure 2). Consider $\sigma_1 \in C^{sum}(\sigma_1) \subset B_4$. Then $S^{sum}_{\sigma_1} = \{\sigma_1,\ \sigma_2\sigma_1,\ \sigma_3\}$. The conjugates of σ_1 by these three elements are, respectively, σ_1, σ_2 and σ_1. All of them lie in $C^{sum}(\sigma_1)$. The conjugating elements are clearly minimal: σ_1 and σ_3 do not have nontrivial divisors, and the only nontrivial divisor of $\sigma_2\sigma_1$ is σ_2, which does not conjugate σ_1 to a positive element (hence to an element in $C^{sum}(\sigma_1)$).*

Remark 1 *It is shown in* [9] *that for every $v \in C^{sum}(a)$ and every atom x, there exists at most one element $s \in S^{sum}_v$ which is a multiple of x. This is why the cardinal of S^{sum}_v is bounded by the number of atoms. In B_n^+, the atoms are $\sigma_1, \dots, \sigma_{n-1}$, and in the above example we can clearly see which element in $S^{sum}_{\sigma_1}$ corresponds to each atom.*

In general, for a given $v \in C^{sum}(a)$, there are strictly fewer minimal simple elements than atoms, as we can see in the following example:

Example 2 *Let $v = \sigma_1\sigma_2 \in C^{sum}(\sigma_1\sigma_2) \subset B_4^+$. Then $S^{sum}_v = \{\sigma_1, \sigma_3\sigma_2\sigma_1\}$. Indeed, conjugating we obtain $\sigma_1^{-1}(\sigma_1\sigma_2)\sigma_1 = \sigma_2\sigma_1$, and*

$$(\sigma_3\sigma_2\sigma_1)^{-1}\sigma_1\sigma_2(\sigma_3\sigma_2\sigma_1) = \sigma_2\sigma_3.$$

But the minimal multiple of σ_2 which conjugates v to a positive element is $\sigma_2\sigma_1\sigma_2$, which is also a multiple of σ_1, so it is not in S^{sum}_v (since it is not minimal).

In [9] the authors give an algorithm to compute S^{sum}_v, given $v \in C^{sum}(a)$, and use it to compute the whole Summit Class of any element. Sometimes, a problem can be solved using either simple elements, or minimal simple elements. The latter possibility is usually much faster. For instance, in the braid monoid B_n^+, computing the set S^{sum}_v takes time $O(l^2n^4)$, where l is the word-length of v. After performing this fast computation, we can work with a set of less than $n-1$ elements (S^{sum}_v), instead of a set with $n!$ elements (S).

In order to compute centralizers in Garside groups, Makanin [14] used simple elements, but we are going to see in the next section how the use of minimal simple elements, and a new approach to the problem, can make the computations much faster.

3. Minimal summit graph

We shall explain in this section a new approach to our problem, which involves the fundamental group of a certain graph. Consider an element a in a Garside group G. We want to find a generating set for the centralizer of a. As we said in Section 1, we will study the elements in its Summit Class $C^{sum}(a)$.

Let us construct a directed graph Γ, that we call *minimal summit graph* of a. The vertices of Γ are the elements in $C^{sum}(a)$. The arrows of Γ are labelled by simple elements, in the following way: For every two vertices v and w, an arrow labelled by s goes from v to w if and only if $s \in S_v^{sum}$ and $s^{-1}vs = w$. In other words, s is a minimal simple element that conjugates v to an element in $C^{sum}(a)$, and w is the result of that conjugation. Therefore, every path in Γ going from a vertex u to another vertex v, and moving always in the sense of the arrows, is a minimal chain from u to v (see Proposition 8).

The minimal summit graph of $\sigma_1 \in B_4^+$ is represented in Figure 3, and that of $\sigma_1\sigma_2$ in Figure 4.

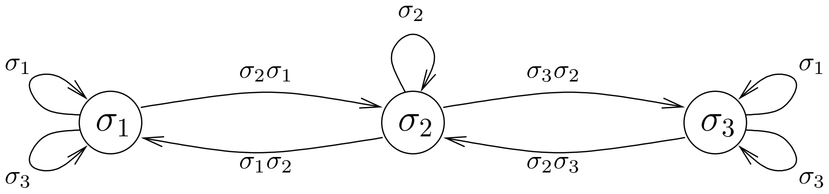

FIGURE 3: Minimal summit graph of $\sigma_1 \in B_4^+$.

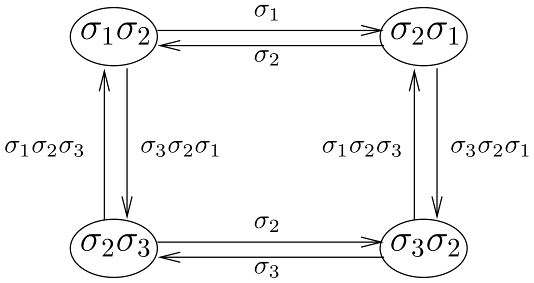

FIGURE 4: Minimal summit graph of $\sigma_1\sigma_2 \in B_4^+$.

The main idea in our algorithm is the following: Given $a' \in C^{sum}(a)$, every element in $Z(a')$ can be seen as a loop in Γ, based at a'. So every generating set for the fundamental group of Γ corresponds to a generating set for $Z(a')$ (recall that if we know a generating set for $Z(a')$, we also know a generating set for $Z(a)$).

We devote the rest of this section to proving this. We shall need the following results:

Lemma 9 *For every $a \in G$, the centralizer of a can be generated by elements in M.*

Proof. Let $c \in Z(a)$. We will try to write c as a product of positive elements in $Z(a)$ (and their inverses). We know by [7] that there is an integer k such that Δ^k is in the center of G (thus in $Z(a)$), and another integer r, big enough, such that $\Delta^{kr}c \in M$. Hence, $c = (\Delta^{kr})^{-1}(\Delta^{kr}c)$, where Δ^{kr} and $\Delta^{kr}c$ belong to $M \cap Z(a)$. This implies the result. ■

Theorem 10 [16] *Let $u, v \in C^{sum}(a)$ and $x \in M$ such that $x^{-1}ux = v$. Let $s \in S$ be the maximal simple prefix of x, that is, s is maximal (with respect to $\prec$) among the simple elements dividing x. Then $s^{-1}us \in C^{sum}(a)$.*

Corollary 11 *Let $u, v \in C^{sum}(a)$, and $x \in M$ as above. Then there exists a decomposition $x = s_1 s_2 \cdots s_{k-1}$, and k elements $u = u_1, u_2, \ldots, u_k = v \in C^{sum}(a)$, such that*

$$u = u_1 \xrightarrow{s_1} u_2 \xrightarrow{s_2} u_3 \rightarrow \cdots \rightarrow u_{k-1} \xrightarrow{s_{k-1}} u_k = v,$$

is a minimal chain from u to v.

Proof. First, let us decompose $x = t_1 t_2 \cdots t_{p-1}$, where for every i, t_i is the maximal simple prefix of $t_i t_{i+1} \cdots t_{p-1}$ (this is the *left greedy normal form* of x, in the sense of [17]). By Theorem 10, we obtain a chain

$$u = w_1 \xrightarrow{t_1} w_2 \xrightarrow{t_2} w_3 \rightarrow \cdots \rightarrow w_{p-1} \xrightarrow{t_{p-1}} w_p = v,$$

where $w_i \in C^{sum}(a)$ for $i = 1, \ldots, p$. But this chain is not necessarily minimal. Now, for every t_i, we proceed as follows: if it is minimal (among the simple elements that conjugate w_i to an element in $C^{sum}(a)$), we do not touch it. Otherwise, there exists an element $r \in S_{w_i}^{sum}$ dividing t_i. So we can decompose the arrow $w_i \xrightarrow{t_i} w_{i+1}$ as $w_i \xrightarrow{r} w' \xrightarrow{r'} w_{i+1}$, where $t_i = r\,r'$ and $w' \in C^{sum}(a)$. If r' is not minimal, we decompose it in the same way. If we continue this process we obtain, at each step, a decomposition $t_i = r_1 \cdots r_m$, where every r_j is a simple element. Hence we have a chain $r_1 \prec r_1 r_2 \prec r_1 r_2 r_3 \prec \cdots \prec (r_1 \cdots r_m)$ of simple elements. But the length of such a chain is bounded above, since there is a finite number of simple elements. Therefore, we cannot decompose t_i indefinitely, and this process must stop.

At the end, we will have decomposed every t_i as a product of minimal simple elements, so the result follows. ■

We can finally prove the main result of this section. Consider the natural group homomorphism $p : \pi_1(\Gamma, a') \longrightarrow G$, which sends every loop in Γ based at a' to the element in G obtained by reading the labels in the path, with the corresponding signs. One has the following:

Theorem 12 *The homomorphism p maps $\pi_1(\Gamma, a')$ onto $Z(a')$.*

Proof. Since every loop $\gamma \in \pi_i(\Gamma, a')$ starts and ends at a', then $p(\gamma)$ conjugates a' to itself, so $p(\gamma) \in Z(a')$. Hence, we get $p : \pi_1(\Gamma, a') \longrightarrow Z(a')$.

By Lemma 9, we know that $Z(a')$ is generated by positive elements. Every positive element $y \in Z(a')$ verifies $y^{-1}a'y = a'$, so by Corollary 11, y can be decomposed into minimal simple elements, yielding a minimal chain from a' to itself. This minimal chain is actually an element in $p^{-1}(y)$. Hence, there exist preimages by p for all positive elements in $Z(a')$. Since the positive elements generate $Z(a')$, we get that p is a surjection, and we are done. ■

By the above result, in order to compute a generating set for $Z(a')$ we just need to compute a generating set for $\pi_1(\Gamma, a')$. It is well known how to do this (see, for instance, [13]): Choose a maximal tree T in Γ. For every vertex v in Γ, call γ_v the only simple path in T going from a' to v. Let A be the set of arrows in $\Gamma\backslash T$ and, for every $\alpha \in A$, denote $s(\alpha)$ and $t(\alpha)$ the *starting vertex* and the *target* of α, respectively. Then there is a generating set F for $\pi_1(\Gamma, a')$, which is in one-to-one correspondance with A. It is the following: $F = \{\gamma_{s(\alpha)}\, \alpha\, \gamma_{t(\alpha)}^{-1};\ \alpha \in A\}$. So $p(F)$ is the generating set for $Z(a')$ that our algorithm will compute.

Remark 2 *In a previous version of this paper, we considered the whole conjugacy class of a (in M) instead of its Summit Class, hence we computed the* minimal conjugacy graph *instead of the minimal summit graph. Although there are no known bounds for the sizes of these sets, the Summit Class is in general much smaller than the whole conjugacy class, so this new approach is more efficient. We thank A. Kalka for his observation on this matter.*

4. The algorithm

We shall now explain our algorithm in detail. Let a be an element of a Garside group G, and let $a' \in C^{sum}(a)$. Let Γ be the minimal summit graph of a'. We will start by computing Γ and, for every vertex $v \in \Gamma$, a path γ_v going from a' to v in a maximal tree T in Γ.

In the following routine, v denotes the current vertex of Γ under study, U is the set of known vertices of Γ (i.e. the known elements in $C^{sum}(a)$), and V is the set of vertices which have already been studied by the routine.

Routine 1: Computation of Γ and T.

Input: $a' \in C^{sum}(a)$.

1. Set $v = a'$, $U = \{a'\}$, $V = \phi$, $\Gamma = \phi$, $T = \phi$ and $\gamma'_a = 1$.
2. Compute S_v^{sum}.
3. For every $s \in S_v^{sum}$ do the following:
    - (a) Set $w = s^{-1}vs \in C^{sum}(a)$, written in normal form. Set
    $$\Gamma = \Gamma \cup \{(v, s, w)\}.$$
    - (b) If $w \notin U$, set $U = U \cup \{w\}$, $T = T \cup \{(v, s, w)\}$ and $\gamma_w = \gamma_v s$.
4. Set $V = V \cup \{v\}$. If $U \neq V$, take an element $x \in U \backslash V$. Set $v = x$ and go to Step 2.
5. Stop.

From the results in the previous sections we can see that, at the end of this routine, we will obtain the following data:

- A set $U = V = C^{sum}(a)$, which is the set of vertices of Γ.
- A set Γ which corresponds to the graph Γ: it contains an element (v, s, w) for every arrow of the graph Γ labelled by s, and going from v to w.
- A set T which corresponds to a subgraph of Γ.
- For every $v \in V$, a path γ_v in the subgraph T, going from a' to v.

Proposition 13 *T is a maximal tree in Γ.*

Proof. The graph T is computed by Routine 1 as follows: Let s be an arrow such that $t(s) = w \neq a'$. Then s is added to T (in Step 3(b)) if and only if it is the first arrow considered by Routine 1 whose target is w. Hence, for every $w \in V$, $w \neq a'$, there is exactly one arrow in T ending at w. And there is no arrow in T ending at a'.

Therefore, if we start at a vertex v, and we try to construct a path in T, as long as possible, moving always in the sense opposite to the arrows, we have a unique choice. This path would always end at a', and it is actually the inverse of the path γ_v computed by Routine 1: Just notice that the path γ_v goes from a' to v always in the sense of the arrows.

Let us then show that T is a tree. Suppose that there exists a nontrivial simple loop α in T. Since there is no pair of arrows of T with the same target, we can assume that α moves always in the sense of the arrows. Since a' is not the target of any arrow in T, then a' does not belong to the set of vertices in α. But, if we start at a vertex v in α, and we try to follow γ_v^{-1} as above, we would go along α^{-1} an infinite number of times, never reaching a'. This contradiction shows that there are no loops in T, so it is a tree.

Finally, T is maximal since it is connected (every vertex is connected to a'), and it contains all the vertices in Γ. ■

Therefore, we can use the data given by Routine 1 to compute a generating system for $Z(a)$, by the procedure explained in the previous section:

Routine 2: Computation of a generating set for $\mathbf{Z(a)}$.

```
Input: $a \in G$.

1. Using ‘cyclings and decyclings’, compute $a' \in C^{sum}(a)$, and $x \in G$
   such that $x^{-1}ax = a'$.
2. Apply Routine 1 to $a'$, obtaining $\Gamma$, $T$ and the paths $\gamma_v$.
3. Set $N = \phi$.
4. For every $(v, s, w) \in \Gamma \backslash T$ do the following:
   (a) Compute the normal form $\alpha$ of $x(\gamma_v s \gamma_w^{-1})x^{-1}$ (given as an element
       of $G$).
   (b) If $\alpha \notin N$, set $N = N \cup \{\alpha\}$.
5. Return $N$. Stop.
```

5. Complexity

In order to study the complexity of our algorithm, we should know some data concerning the Garside monoid M, and the element $a \in G$ under study:

t: The number of atoms in M.

m: The maximal length of a simple element in M.

k: The number of elements in $C^{sum}(a)$ (i.e. the number of vertices in Γ).

l: The maximal word length of an element in $C^{sum}(a)$.

D: The complexity of computing $a' \in C^{sum}(a)$ and x, as in Routine 2.

C: The complexity of an algorithm to compute S_v^{sum} for an element v of word length l.

N_i: The complexity of computing the normal form of a word of length i.

If we know all the previous data, we can compute the complexity of our algorithm by the following result:

Proposition 14 *Given an element a in a Garside group G, we can compute a generating set for $Z(a)$ in time $O(D + kC + ktN_{2km})$.*

Proof. We start by computing a' and x, taking time $O(D)$. Then we run Routine 1, which does the following: For every vertex v in T, it computes S_v^{sum} and then, for every $s \in S_v^{sum}$, it computes the normal form of $s^{-1}vs$. The other steps in Routine 1 are negligible. Moreover, the algorithm used in [9] to compute S_v^{sum} also gives the normal forms of $s^{-1}vs$, for $s \in S_v^{sum}$. Hence, Routine 1 has complexity $O(kC)$.

Routine 2 continues by computing the normal form of $x(\gamma_v s \gamma_w^{-1})x^{-1}$, for every arrow (v, s, w) in $\Gamma\backslash T$. We know that the number of arrows in Γ is bounded by kt, since there are at most t arrows for each vertex, and there are k vertices. On the other hand, there is exactly one arrow in T whose target is v, for every vertex in Γ different from a'. Hence, T has $k-1$ arrows, so $\Gamma\backslash T$ has at most $kt - k - 1$ arrows. Now $x(\gamma_v s \gamma_w^{-1})x^{-1}$ is a product of at most $2(k + |x|) - 1$ simple elements. Hence, written as a word in the atoms and their inverses, its length is bounded by $2(k + |x|)m$. Since the length of x is negligible compared to k, $N_{2(k+|x|)m}$ is equivalent to N_{2km}. Therefore, the complexity of this loop is $O(ktN_{2km})$, and the result follows. ■

For some particular Garside monoids and groups, one would like to know the complexity in more detail, just depending on the word length of a, and on some integer related to the monoid. This can be done more easily for Garside monoids in which every relation is homogeneous, for in this case, all the elements in the conjugacy class of a have the same word length. This is the case for B_n^+, BKL_n^+ and Artin monoids. The authors have studied in [9] the complexity C of computing S_v^{sum} for a braid v of length l (either in B_n^+ or in BKL_n^+). In [17] and in [3] we can find the complexity N_i, for elements in B_n^+ and in BKL_n^+ respectively, and in [4] the complexity D is given. Hence, we obtain the following results:

Corollary 15 *Given $a \in B_n$ of word length l in the Artin generators, the complexity of computing $Z(a)$ (using the Garside structure given by B_n^+) is $O(k^3 l^2 n^6 \log n)$.*

Proof. In B_n^+, one has $t = n - 1$, $m = \frac{n(n-1)}{2}$, $C = O(l^2 n^4)$ (see [9]), $N_i = O(i^2 n \log n)$ (see [17]) and $D = O(l^2 n^3)$ (see [4]). Hence, the complexity of our algorithm to compute $Z(a)$ becomes

$$\begin{aligned} O\left(l^2n^3 + kl^2n^4 + k(n-1)(kn(n-1))^2 n \log n\right) &= O(kl^2n^4 + k^3n^6 \log n) \\ &= O(k^3l^2n^6 \log n), \end{aligned}$$

so the result is true. ■

Corollary 16 *Given $a \in B_n$ of word length l in the Birman-Ko-Lee generators, the complexity of computing $Z(a)$ (using the Garside structure given by BKL_n^+) is $O(k^3l^2n^5)$.*

Proof. This time, in BKL_n^+, one has $t = n(n-1)/2$, $m = n-1$, $C = O(l^2n^5)$ (see [9]), $N_i = O(i^2n)$ (see [3]) and $D = O(l^2n^2)$ (see [4]). Therefore, our algorithm for computing $Z(a)$ has complexity

$$O\left(l^2n^2 + kl^2n^5 + k\frac{n(n-1)}{2}(2k(n-1))^2n\right) = O(kl^2n^5+k^3n^5) = O(k^3l^2n^5).$$

■

It would remain to know, in both cases, a bound for k in terms of l and n. This is still not known, but Thurston, in [17], conjectures that k is bounded by a polynomial in l (although it seems to be exponential in n).

6. Effective computations

In this section we show the results we have obtained by implementing our algorithm. We have computed generating sets for the centralizers of many positive elements in the braid monoids B_n^+, for $n = 3, \dots, 8$. We have been exhaustive, computing centralizers of all braids of a given length, in order to conjecture an upper bound for the number of generators.

We proceeded as follows: first, by using the algorithm in [9], we computed the conjugacy classes in B_n^+ of all braids of the considered length. Notice that two elements in the same conjugacy class have conjugated centralizers: if $c^{-1}ac = b$ and $x \in Z(a)$, then $c^{-1}xc \in Z(b)$; hence, the number of generators in the centralizer of a and b are the same. Therefore, we just had to compute the centralizer of one representative for each conjugacy class in B_n^+.

The results of these computations were surprising, since the number of generators were quite small. In the following table we can see the braids that we tested, and the maximal size of a generating set for the centralizer, in each case:

n	Length of braids	Number of Conj. Classes	Max. number of generators
3	$4 \leq l \leq 20$	1634	4
4	$4 \leq l \leq 15$	4225	16
5	$4 \leq l \leq 12$	2314	17
6	$4 \leq l \leq 10$	1152	12
7	$4 \leq l \leq 10$	1753	17
8	$4 \leq l \leq 8$	521	22

Actually, we found out that the generators obtained by the algorithm were not always independent, so we were able to eliminate some of them. For instance, if we compute $Z(a)$ for $a = \sigma_1 \in B_4$, the algorithm will give the following generating set:

$$\{\sigma_1,\ \sigma_2\sigma_1\sigma_1\sigma_2,\ \sigma_3,\ \sigma_2\sigma_1(\sigma_3\sigma_2\sigma_2\sigma_3)\sigma_1^{-1}\sigma_2^{-1}\}.$$

But the fourth element can also be written as $(\sigma_3)^{-1}(\sigma_2\sigma_1\sigma_1\sigma_2)(\sigma_3)$, so it can be eliminated from the generating set, yielding:

$$Z(\sigma_1) = \langle \sigma_1,\ \sigma_2\sigma_1\sigma_1\sigma_2,\ \sigma_3 \rangle \subset B_4.$$

In the case of B_3, we were able to obtain the following: for every positive braid $a \in B_3^+$ of length $l \leq 20$, there is a generating set for $Z(a)$ with at most two elements.

We cannot show here all the results but we can see, as an example, the following table: it contains a representative for each conjugacy class of elements in B_3^+ of length 11, and a generating set for their centralizers.

Centralizers of braids in B_3^+ of length 11		
a	Generators for $Z(a)$	
σ_1^{11}	σ_1	$\sigma_2\sigma_1^2\sigma_2$
$\sigma_1^{10}\sigma_2$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^2\sigma_2^2\sigma_1^2\sigma_2\sigma_1^{-6}$
$\sigma_1^9\sigma_2^2$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^6\sigma_2^{-1}\sigma_1^{-2}\sigma_2^{-3}\sigma_1^{-1}$
$\sigma_1^8\sigma_2^3$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^6\sigma_2^{-1}\sigma_1^{-3}\sigma_2^{-2}\sigma_1^{-1}$
$\sigma_1^2\sigma_2^6\sigma_1^2\sigma_2$	$\sigma_1^2\sigma_2\sigma_1^{-2}$	$\sigma_1^3\sigma_2^2\sigma_1^{-1}$
$\sigma_1^7\sigma_2^4$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^6\sigma_2^{-2}\sigma_1^{-2}\sigma_2^{-2}\sigma_1^{-1}$
$\sigma_1^6\sigma_2^2\sigma_1^2\sigma_2$	$\sigma_1^4\sigma_2\sigma_1^{-4}$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$
$\sigma_1^6\sigma_2^5$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^6\sigma_2\sigma_1^{-1}\sigma_2^{-2}\sigma_1^{-2}\sigma_2^{-2}\sigma_1^{-1}$
$\sigma_1^5\sigma_2^3\sigma_1^2\sigma_2$	$\sigma_1^3\sigma_2^2\sigma_1^{-4}$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$
$\sigma_1^5\sigma_2^2\sigma_1^3\sigma_2$	$\sigma_1^2\sigma_2^3\sigma_1^{-4}$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$
$\sigma_1^5\sigma_2^2\sigma_1^2\sigma_2^2$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1\sigma_2^4\sigma_1^{-4}$
$\sigma_1^4\sigma_2^2\sigma_1^4\sigma_2$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^2\sigma_2^2\sigma_1^2\sigma_2^{-1}\sigma_1^{-4}$
$\sigma_1^4\sigma_2^2\sigma_1^3\sigma_2^2$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1\sigma_2^3\sigma_1^2\sigma_2^{-1}\sigma_1^{-4}$
$\sigma_1^4\sigma_2^2\sigma_1^2\sigma_2^3$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1\sigma_2^2\sigma_1^3\sigma_2^{-1}\sigma_1^{-4}$
$\sigma_1^4\sigma_2^3\sigma_1^2\sigma_2^2$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1\sigma_2^4\sigma_1\sigma_2^{-1}\sigma_1^{-4}$
$\sigma_1^3\sigma_2^2\sigma_1^3\sigma_2^3$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^3\sigma_2^2\sigma_1\sigma_2^{-1}\sigma_1^{-3}\sigma_2^{-2}\sigma_1^{-1}$

When n becomes bigger, it is more difficult to eliminate generators by hand. Nevertheless, we can show as an example the following table, where we can see a representative for every conjugacy class of elements of length 6 in B_4^+. We were able to reduce the number of generators to be less than or equal to 3 in every case:

Centralizers of braids in B_4^+ of length 6

a	Generators for $Z(a)$		
σ_1^6	σ_1	σ_3	$\sigma_2\sigma_1^2\sigma_2$
$\sigma_1^5\sigma_2$	$\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3$	$\sigma_1^2\sigma_2\sigma_1^{-3}$	$\sigma_1^5\sigma_2$
$\sigma_1^5\sigma_3$	σ_1	σ_3	$\sigma_2\sigma_1\sigma_3\sigma_2^2\sigma_1\sigma_3\sigma_2$
$\sigma_1^4\sigma_2^2$	$\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	$\sigma_1^4\sigma_2^2$
$\sigma_1^4\sigma_2\sigma_3$	$\sigma_1^2\sigma_2\sigma_1\sigma_3\sigma_2^{-2}\sigma_1^{-3}$	$\sigma_1^4\sigma_2\sigma_3$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1^{-2}$
$\sigma_1^3\sigma_3\sigma_1\sigma_3$	σ_1	σ_3	$\sigma_2\sigma_1\sigma_3\sigma_2^2\sigma_1\sigma_3\sigma_2$
$\sigma_1^3\sigma_2^3$	$\sigma_1\sigma_2\sigma_1^{-2}$	$\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3$	$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$
$\sigma_1^3\sigma_2^2\sigma_3$	$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1^{-2}$	$\sigma_1^3\sigma_2^2\sigma_3$	
$\sigma_1^3\sigma_2\sigma_3\sigma_2$	$\sigma_1^2\sigma_3\sigma_2\sigma_3^{-1}\sigma_2^{-2}\sigma_1^{-1}$	$\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1^{-1}$	$\sigma_1^3\sigma_2\sigma_3\sigma_2$
$\sigma_1\sigma_3\sigma_1\sigma_3\sigma_1\sigma_3$	σ_1	$\sigma_2\sigma_1\sigma_3\sigma_2$	σ_3
$\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_1$	σ_1	σ_2	$\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3$
$\sigma_1^2\sigma_2\sigma_1\sigma_3\sigma_2$	$\sigma_1\sigma_2\sigma_1^{-1}$	$\sigma_1\sigma_3$	$\sigma_2\sigma_3\sigma_2^{-1}$
$\sigma_1^2\sigma_2^3\sigma_3$	$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_2^{-1}\sigma_1^{-2}$	$\sigma_1^2\sigma_2^3\sigma_3$	
$\sigma_1^2\sigma_2^2\sigma_3^2$	$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2^2\sigma_3\sigma_2\sigma_1\sigma_2^{-1}\sigma_1^{-2}$	$\sigma_1^3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1^{-1}$	$\sigma_1^2\sigma_2^2\sigma_3^2$
$\sigma_1^2\sigma_2\sigma_3^2\sigma_2$	σ_3	$\sigma_1\sigma_2\sigma_1^{-1}$	$\sigma_1^2\sigma_2\sigma_3^2\sigma_2$
$\sigma_1\sigma_2^4\sigma_3$	$\sigma_1\sigma_2^3\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}$	$\sigma_1^2\sigma_2\sigma_1\sigma_3\sigma_2$	

Actually, every time that we tried to reduce the number of generators associated to a conjugacy class, we were able to keep just $n-1$. Remark also that there are 1634 different conjugacy classes of elements of length l $(4 \leq l \leq 20)$ in B_3^+, all of them with no more than two generators. So all these evidences led us to think that the centralizer of every braid in B_n could be generated by at most $n-1$ elements.

As we said, this conjecture turned out to be false, since a family of counterexamples due to N. V. Ivanov gives a lower bound for the number of generators which is a quadratic in n. Precisely, there has been recently shown [11] that the centralizer of every element in B_n can be generated by less than $\frac{k(k+1)}{2}$ elements if $n=2k$ and $\frac{k(k+3)}{2}$ if $n=2k+1$.

In any case, the above results are valid just for braids, so we still would like to have an upper bound for the minimal number of generators of $Z(a)$, in the general case of Garside groups.

Acknowledgements: The authors want to thank the Laboratorie de Topologie de l'Université de Bourgogne, where we started to work in this subject, and to Luis Paris, Alain Jacquemard, José María Tornero, Carmen León, Mustafa Korkmaz, Arkadius Kalka and Bert Wiest for their valuable help.

References

[1] Artin, E.: Theory of braids. *Annals of Math.* **48** (1946), 101–126.

[2] Anshel, I., Anshel, M. and Goldfeld, D.: An algebraic method for public-key cryptography. *Math. Res. Lett.* **6** (1999), no. 3-4, 287–291.

[3] Birman, J., Ko, K. H. and Lee, S. J.: A new approach to the word and conjugacy problems in the braid groups. *Adv. Math.* **139** (1998), no. 2, 322–353.

[4] Birman, J., Ko, K. H. and Lee, S. J.: The infimum, supremum and geodesic length of a braid conjugacy class. *Adv. Math.* **164** (2001), 41–56.

[5] Brieskorn, E. and Saito, K.: Artin-Gruppen und Coxeter-Gruppen. *Invent. Math.* **17** (1972), 245–271.

[6] Dehornoy, P.: Groupes de Garside. *Ann. Sci. École Norm. Sup. (4)* **35** (2002), 267–306.

[7] Dehornoy, P. and Paris, L.: Gaussian groups and Garside groups, two generalizations of Artin groups. *Proc. London Math. Soc.* **79** (1999), no. 3, 569–604.

[8] Elrifai, E. A. and Morton, H. R.: Algorithms for positive braids. *Quart. J. Math. Oxford* **45** (1994), 479–497.

[9] Franco, N. and González-Meneses, J.: Conjugacy problem for braid groups and Garside groups, to appear in Journal of Algebra. Available at `http://arxiv.org/math.GT/0112310`

[10] Garside, F. A.: The braid group and other groups. *Quart. J. Math. Oxford* **20** (1969), 235–154.

[11] González-Meneses, J. and Wiest, B.: On the structure of the centralizer of a braid. In preparation.

[12] Ko, K. H., Lee, S. J., Cheon, J. H., Han, J. W., Kang, J. and Park, C.: New public-key cryptosystem using braid groups. In *Advances in cryptology–CRYPTO 2000 (Santa Barbara, CA)*, 166–183. Lecture Notes in Comput. Sci. **1880**, Springer, Berlin, 2000.

[13] Lyndon, R. C. and Schupp, P. E.: *Combinatorial group theory.* Reprint of the 1977 edition. Classics in Mathematics, Springer-Verlag, Berlin, 2001.

[14] Makanin, G. S.: The normalizers in the braid group. *Mat. Sb. (N. S.)* **86 (128)** (1971), 171–179.

[15] Picantin, M.: *Petits groupes gaussiens*. Ph. D. Thesis, Université de Caen, 2000.

[16] Picantin, M.: The conjugacy problem in small Gaussian groups. *Comm. Algebra* **29** (2001), no. 3, 1021–1039.

[17] Thurston, W. P.: Braid Groups, Chapter 9 of *Word processing in groups*, D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson and W. P. Thurston. Jones and Bartlett Publishers, Boston, MA, 1992.

Recibido: 8 de abril de 2002
Revisado: 10 de diciembre de 2002

Nuno Franco
Departamento de Matemática
CIMA-UE, Universidade de Évora
7000-Évora, Portugal
nmf@uevora.pt

Juan González-Meneses
Departamento de Matemática Aplicada I
ETS Arquitectura, Universidad de Sevilla
Avda. Reina Mercedes 2, 41012-Sevilla, Spain
meneses@us.es

This paper is dedicated to José Luis Vicente Córdoba, on his 60th birthday. Both authors partially supported by the European Network TMR Sing. Eq. Diff. et Feuill. N. Franco partially supported by SFRH/BD/2852/2000. J. González-Meneses partially supported by MCYT, BFM2001-3207 and FEDER.