Conjugacy problem for braid groups and Garside groups[1]

Nuno Franco[2]

Dep. de Matemática, CIMA-UE
Universidade de Évora
7000-Évora (Portugal)
E-mail: nmf@uevora.pt

Université de Bourgogne
Laboratoire de Topologie
UMR 5584 du CNRS
B.P. 47870
21078 - Dijon Cedex (France)
E-mail: nmf@u-bourgogne.fr

and

Juan González-Meneses[3]
Dept. de Matemática Aplicada I
ETS Arquitectura
Universidad de Sevilla
Avda. Reina Mercedes, 2
41012-Sevilla (Spain)
E-mail: meneses@us.es

January, 2002

Key Words: Braid groups; Artin groups; Garside groups; Small Gaussian groups; Conjugacy problem.

Subject Classification: Primary: 20F36. Secondary: 20F10.

We present a new algorithm to solve the conjugacy problem in Artin braid groups, which is faster than the one presented by Birman, Ko and Lee [3]. This algorithm can be applied not only to braid groups, but to all *Garside groups* (which include finite type Artin groups and torus knot groups among others).

1. INTRODUCTION

Given a group G, the conjugacy problem in G consists on finding an algorithm which, given $a, b \in G$, determines if there exists $c \in G$ such that $a = c^{-1}bc$. Sometimes one also needs to compute c, for instance, when one tries to attack cryptosystems based on conjugacy in G ([2], [12]).

We are mainly interested in Artin braid groups, which are defined, for $n \geq 2$, by the following presentation:

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

[1]Both authors partially supported by the European network TMR Sing. Eq. Diff. et Feuill.
[2]Partially supported by SFRH/BD/2852/2000.
[3]Partially supported by BFM-3207.

The first conjugacy algorithm for braid groups was given by Garside [11]. It was improved by Elrifai and Morton [10] and, more recently, by Birman, Ko and Lee ([3] and [4]).

In all these algorithms, one of the key points is the existence of a finite set $S \subset B_n$, whose elements are called *simple elements*, verifying some suitable properties (we will be more precise later). One of the main disadvantages is the size of S, which is always greater than 3^n.

In this paper we will show how one can avoid this problem by defining some small subsets of S, whose size is smaller than $n-1$. Their elements will be called *minimal simple elements.* Unlike S, these sets of minimal simple elements are not unique for every group: The suitable set of minimal simple elements must be recomputed many times in our algorithm. Nevertheless, we will see that it is much faster to compute and use these very small subsets, than to use the whole S all the time.

For instance, the known upper bound for the complexity of the Birman-Ko-Lee algorithm, to decide wether two braids a and b are conjugated in B_n, is $O(kl^2n3^n)$ (where k is a number that will be explained later, and l is the maximum of the word lengths of a and b). An upper bound for the complexity of our algorithm for B_n is $O(kl^2n^4)$.

Let us mention that our algorithm, as well as the previous ones, also computes the element $c \in B_n$ such that $a = c^{-1}bc$. Moreover, since our construction relies on the existence of simple elements and their basic properties, we can extend our results to a much larger class of groups, called *Garside groups.* They were introduced by Dehornoy and Paris [9]. At the origin, these groups were called *small Gaussian groups*, but there has been a convention to call them *Garside groups.* They include, besides Artin braid groups, spherical (finite type) Artin groups, torus knot groups and others.

One final remark: one important property of Garside groups is the existence of embedable monoids (for instance the monoid of positive braids, B_n^+, which embeds in B_n). The conjugacy class of an element a in such a monoid is known to be a finite set, $C^+(a)$. We will also show how to compute $C^+(a)$, using the techniques mentioned above.

This paper is structured as follows: In Section 2, we give a brief introduction to Garside monoids and groups; In Section 3, the known algorithms mentioned in this introduction are detailed; We introduce the minimal simple elements in Section 4, and in Section 5 we present our algorithms in detail; Complexity issues are treated in Section 6 and, finally, some effective computations are described in Section 7.

2. GARSIDE MONOIDS AND GROUPS

The results contained in this section are well known, and can be found in [11], [10], [16], [3], [9], [8] and [14]. We will define the *Garside monoids* and *Garside groups*, and explain some basic properties.

Given a cancellative monoid M, with no invertible elements, we can define two different partial orders on its elements, $\prec$ and $\succ$. Given $a, b \in M$, we say that $a \prec b$ ($b \succ a$) if there exists $c \in M$ such that $ac = b$ ($b = ca$), and we say that a is a left (right) divisor of b.

In this situation, we can naturally define the (left or right) *least common multiple* and *greatest common divisor* of two elements. Given $a, b \in M$, we denote by $a \vee b$ the left lcm of a and b, if it exists. That is, a minimal element (with respect to $\prec$) such that $a \prec a \vee b$ and $b \prec a \vee b$. We denote by $a \wedge b$ the left gcd of a and b, if it exists. That is, a maximal element (with respect to $\prec$), such that $a \wedge b \prec a$ and $a \wedge b \prec b$.

Definition 2.1. *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 $x \in M$, there exists an integer $N_x > 0$ such that x cannot be written as a product of more than N_x atoms.*

Definition 2.2. *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) lcm and a (left and right) gcd*

Definition 2.3. *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 2.4. *The left (and right) divisors of Δ in a Garside monoid M are called simple elements. The (finite) set of simple elements is denoted by S.*

It is known that every Garside monoid admits a group of fractions. So we have:

Definition 2.5. *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 (actually the monoid studied by Garside) 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 [11] showed that $B_n^+ \subset B_n$. Actually, every Garside monoid embeds into its corresponding Garside group [9].

The classical choice of a Garside element for B_n^+ is the following: $\Delta = (\sigma_1\sigma_2\cdots\sigma_{n-1})(\sigma_1\sigma_2\cdots\sigma_{n-2})\cdots(\sigma_1\sigma_2)\,\sigma_1$. It can be defined as the positive braid (braid in B_n^+) in which any two strands cross *exactly* once (where, as usual, σ_i represents a crossing of the strands in positions i and $i+1$). It is represented in Figure 1 for $n = 4$. The simple elements in this case are the positive braids in which any two strands cross *at most* once. Then one has $\#(S) = n!$

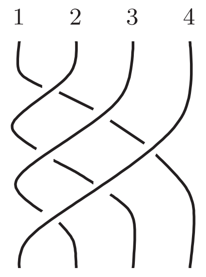

FIG. 1 The Garside element $\Delta \in B_4^+$.

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

$$BKL_n^+ = \left\langle a_{ts}(n \geq t > s \geq 1) \;\middle|\; \begin{array}{ll} a_{ts}a_{rq} = a_{rq}a_{ts} & \text{if } (t-r)(t-q)(s-r)(s-q) > 0 \\ a_{ts}a_{sr} = a_{tr}a_{ts} = a_{sr}a_{tr} & \text{where } n \geq t > s > r \geq 1 \end{array} \right\rangle \qquad (2)$$

Its group of fractions is again the braid group B_n. The usual Garside element in BKL_n^+ is $\delta = a_{n,n-1}a_{n-1,n-2}\cdots a_{2,1}$. The advantage of this monoid with respect to B_n^+ is that $\#(S) = \mathcal{C}_n$, where $\mathcal{C}_n = \frac{(2n)!}{n!(n+1)!} < 4^n$ is the n^{th} Catalan number. Hence, the number of simple elements is much smaller in this case, but it is still quite big, since $\mathcal{C}_n > 3^n$. Notice also that $|\delta| = n-1$, while in B_n^+, $|\Delta| = \frac{n(n-1)}{2}$.

As we mentioned before, there are other examples of Garside groups, such as finite type Artin groups, or torus knot groups (see [14] to find more examples of Garside groups).

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 left lcm's and gcd's, it follows that $(M, \prec)$ has a lattice structure, and S becomes a finite sublattice with minimum 1 and maximum Δ. See in Figure 2 the Hasse diagram of the lattice of S in B_4^+, where the lines represent left divisibility (from bottom to top). The analogous properties are also verified by $\succ$.

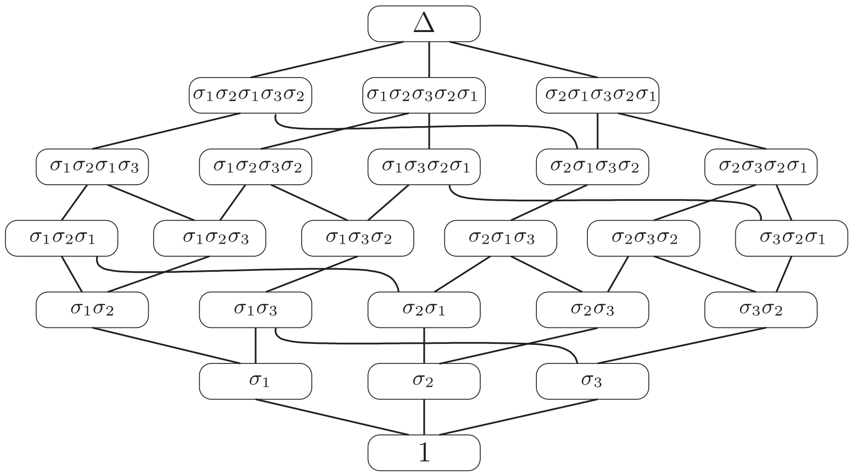
Δ
$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2$
$\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1$
$\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1$
$\sigma_1\sigma_2\sigma_1\sigma_3$
$\sigma_1\sigma_2\sigma_3\sigma_2$
$\sigma_1\sigma_3\sigma_2\sigma_1$
$\sigma_2\sigma_1\sigma_3\sigma_2$
$\sigma_2\sigma_3\sigma_2\sigma_1$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

FIG. 2 The lattice of simple elements in B_4^+.

DEFINITION 2.6. *For $a \in M$ we define $LM(a) \in S$ as the maximal simple left divisor of a, that is, $LM(a) = \Delta \wedge a$. We also define $RM(a)$ as the maximal simple right divisor of a.*

PROPOSITION 2.7 ([11]). *For $a \in G$, there exists a unique decomposition $a = \Delta^p a_1 \cdots a_l$, called* left normal form *of a, where:*

1. *$p = \max\{r \in \mathbb{Z} : \Delta^{-r} a \in M\}$ (hence $a_1 \cdots a_l \in M$).*
2. *$a_i = LM(a_i \cdots a_l) \in S \backslash \{\Delta, 1\}$, for all $i = 1, ..., l$.*

Symmetrically, one defines the right normal form *of $a \in G$, using RM.*

Sometimes, if we are dealing with elements in M and it does not lead to confusion, we will say that an element $w = w_1 \cdots w_t \in M$ is in left normal form to express that $w_i \in S \backslash \{1\}$ for all i and, for some $p \geq 0$, the normal form of w is $\Delta^p w_{p+1} \cdots w_t$.

Later we will use these technical results:

LEMMA 2.8 ([13], Prop. 2.1). *Let $w_1 \cdots w_t \in M$ be in left normal form, and $x_1 \cdots x_t \in M$ in right normal form. For every $v \in M$, one has $LM(vw_1 \cdots w_t) = LM(vw_1)$ and $RM(x_1 \cdots x_t v) = RM(x_t v)$.*

LEMMA 2.9 ([13], Prop. 5.3). *Let $w = w_1 \cdots w_t \in M$ be written in right normal form. If we write w in any other way as a product of t simple elements, $w = u_1 \cdots u_t$, then $w_1 \prec u_1$.*

Lemma 2.10 ([7], 3.1). *Let $w = w_1 \cdots w_t \in M$ be written in right normal form, and let $s \in S$. Then we can decompose $w_i = w_i' w_i''$, for all i, in such a way that the right normal form of ws is $(w_1')(w_1'' w_2') \cdots (w_{t-1}'' w_t')(w_t'' s)$ if it has $t+1$ factors, or $(w_1 w_2') \cdots (w_{t-1}'' w_t')(w_t'' s)$ if it has t factors.*

Corollary 2.11. *Let $w = w_1 \cdots w_t \in M$ be written in right normal form. Let $s \in S$ and suppose that we can write ws as a product of t simple elements, that is, $w_1 \cdots w_t s = u_1 \cdots u_t$. Then $w_1 \prec u_1$.*

Proof. Since ws can be written as a product of t simple elements, then its right normal form has t factors, say $v_1 \cdots v_t$. By Lemma 2.10, $w_1 \prec v_1$, and by Lemma 2.9 $v_1 \prec u_1$, so the result follows. ∎

We end this section with a last property of Garside groups: There is a power of their Garside element which belongs to the center. For instance, in B_n the element $\Delta^2 = \delta^n$ generates the center of B_n.

3. KNOWN ALGORITHMS FOR THE CONJUGACY PROBLEM.

We present here the Elrifai-Morton algorithm for the conjugacy problem in braid groups [10], which is also valid for Garside groups, as can be seen in [15].

It goes as follows: for every element $a \in G$, it computes a finite subset $C^{sum}(a)$ of the conjugacy class of a. This set is shown to be independent of a, so it is an invariant of its conjugacy class. Therefore, two elements a and b are conjugated if and only if $C^{sum}(a) = C^{sum}(b)$.

Let us explain the algorithm in more detail.

3.1. Definition of $\mathbf{C^{\geq m}(a)}$ and $\mathbf{C^{sum}(a)}$

Proposition 3.1. [10, 15] *Let $a = \Delta^p a_1 \cdots a_l \in G$ be in left normal form. Then the right normal form of a is as follows: $a = x_1 \cdots x_l \Delta^p$, where l and p are the same as above.*

Definition 3.2. *Let $a = \Delta^p a_1 \cdots a_l \in G$ be in left normal form. We define the* infimum, supremum *and* canonical length *of a, respectively, by* $\inf(a) = p$, $\sup(a) = p + l$, *and* $\|a\| = l$.

Definition 3.3. *Let $a \in G$ and denote by $C(a)$ the conjugacy class of a. We define the* summit infimum, *the* summit supremum *and the* summit length *of a as, respectively,* $\max\{\inf(x) : x \in C(a)\}$, $\min\{\sup(x) : x \in C(a)\}$ *and* $\min\{\|x\| : x \in C(a)\}$.

Definition 3.4. *Let $a \in G$.*

1. *For every integer m, we define $C^{\geq m}(a) = \{v \in C(a) : \inf(v) \geq m\}$.*
2. *We define the* summit class *of a, $C^{sum}(a)$, as the subset of $C(a)$ containing all elements of minimal canonical length.*

Remarks:
1. One has $C^{\geq 0}(a) = C(a) \cap M = C^+(a)$.
2. In [10], $C^{sum}(a)$ is called the *Super Summit Set*.

Proposition 3.5. [10, 15] *For every $b \in C^{sum}(a)$, the infimum, supremum and canonical length of b are equal, respectively, to the summit infimum, the summit supremum and the summit length of a.*

It is known that $C^{\geq m}(a)$ and $C^{sum}(a)$ are finite sets. Moreover, by Proposition 3.5, if $C^{\geq m}(a) \neq \phi$, then $C^{sum}(a) \subset C^{\geq m}(a)$.

3.2. Cycling and decycling

Let $\tau : G \to G$ be the automorphism defined by $\tau(a) = \Delta^{-1}a\Delta$. The restriction of τ to S is a bijection $\tau : S \to S$.

Definition 3.6. *Let $a = \Delta^p a_1 \cdots a_l \in G$ be written in left normal form. The functions* cycling *and* decycling *are the maps* $\mathbf{c}$ *and* $\mathbf{d}$, *from G to itself, defined by:*

$$\begin{aligned} \mathbf{c}(a) &= \Delta^p a_2 \cdots a_l \tau^{-p}(a_1); \\ \mathbf{d}(a) &= \Delta^p \tau^p(a_l) a_1 \cdots a_{l-1}. \end{aligned}$$

Notice that $\mathbf{c}(a)$ and $\mathbf{d}(a)$ are conjugates of a. Furthermore, for every $a \in G$, $\inf(a) \leq \inf(\mathbf{c}(a))$ and $\sup(a) \geq \sup(\mathbf{d}(a))$.

Suppose that we have an element $a \in G$, such that $\inf(a)$ is not equal to the summit infimum of a. Then we can try to increase the infimum by repeated cycling. By [10] (and [15]), this always works: there exists a positive integer k such that $\inf(\mathbf{c}^k(a)) > \inf(a)$. We know a bound for this integer k only for some special Garside monoids and groups: If M is *homogeneous*, i.e. it has only homogeneous relations (for instance, if M is B_n^+ or BKL_n^+), then every two words representing an element $a \in M$ have the same length, denoted $|a|$. It is shown in [4] that, in this case, $k < |\Delta|$.

Therefore, by repeated cycling, we can conjugate a to another element $\widehat{a}$ of maximal infimum. Even if M is not homogeneous, we know that we reached the summit infimum when we enter into a loop: at some point $\mathbf{c}^k(v) = v$ for some v conjugated to a. This always happens since the set $C^{\geq m}(a)$ is finite for every m, in particular for the summit infimum.

Once $\widehat{a}$ is obtained, we can try to decrease its supremum by repeated decycling. By [10] (and [15]), this also works: either we enter into a loop, and then the supremum is minimal, or there exists an integer k such that $\sup(\mathbf{d}^k(\widehat{a})) < \sup(\widehat{a})$. Again by [4], $k < |\Delta|$ in homogeneous monoids.

Therefore, using repeated cycling and decycling a finite number of times, one obtains an element $\widetilde{a} \in C^{sum}(a)$. And, if M is homogeneous, this can be done in polynomial time in $|a|$.

3.3. The Elrifai-Morton algorithm

Once that we obtained an element $\widetilde{a} \in C^{sum}(a)$, we can construct the whole $C^{sum}(a)$, by using the next result:

Proposition 3.7. [10, 15] *For u, v conjugate elements in $C^{sum}(a)$ (resp. $C^{\geq m}(a)$), there exists a sequence $u = u_1, u_2, ..., u_k = v$ of elements in $C^{sum}(a)$ (resp. $C^{\geq m}(a)$) such that, for $i = 1, \dots, k-1$, u_i and u_{i+1} are conjugated by an element in S.*

The Elrifai-Morton algorithm does the following: Given $a, b \in G$ it computes, using cyclings and decyclings, $\widetilde{a} \in C^{sum}(a)$ and $\widetilde{b} \in C^{sum}(b)$. Then it defines $V_1 = \{\widetilde{a}\}$ and it computes, by recurrence,

$$V_i = \{s^{-1}vs;\ s \in S, v \in V_{i-1}\} \cap C^{sum}(a).$$

Since $1 \in S$, this creates an ascending chain of subsets of $C^{sum}(a)$. By the above proposition, one has $V_k = V_{k+1}$ for some k, and then $V_k = C^{sum}(a)$. Hence, when the chain stabilises, the whole $C^{sum}(a)$ has been computed. Then a and b are conjugated if and only if $\widetilde{b} \in C^{sum}(a)$.

Remark 3.8. *This algorithm can be modified to compute $C^{\geq m}(a)$ for $a \in M$ and $m \in \mathbb{Z}$. We just need to replace $C^{sum}(a)$ by $C^{\geq m}(a)$ in the above discussion.*

Notice that $C^{sum}(a)$ (resp. $C^{\geq m}(a)$) is computed at the cost of conjugating every element in $C^{sum}(a)$ (resp. $C^{\geq m}(a)$) by every element in S. All these sets are quite big, and this makes the algorithm to be slow. In what follows, we will get rid of the problem caused by the size of S, using the *minimal simple elements*.

4. MINIMAL SIMPLE ELEMENTS

In this section we shall define some very small subsets of S, which will enable us to compute $C^{\geq m}(a)$ and $C^{sum}(a)$, for $a \in G$, much faster than the previous algorithms.

Recall the definition of the partial order $\prec$ in M.

DEFINITION 4.1. *Let $\mathcal{P}$ be a property for simple elements. We denote by $S_{\mathcal{P}}$ the set of simple elements satisfying $\mathcal{P}$. The* set of minimal simple elements *for $\mathcal{P}$, $\min(S_{\mathcal{P}})$, is the set of minimal elements (with respect to $\prec$) in $S_{\mathcal{P}}$.*

We shall enforce $\mathcal{P}$ to be *closed under g.c.d*, that is, if $s_1, s_2 \in S_{\mathcal{P}}$ then $s_1 \wedge s_2 \in S_{\mathcal{P}}$. Let us see that, under this assumption, the set $\min(S_{\mathcal{P}})$ turns to be very small. For every atom $x \in M$, let $mult(x) = \{s \in S;\ x \prec s\}$.

LEMMA 4.2. *Suppose that $\mathcal{P}$ is closed under gcd, and let x be an atom of M. If the set $S_{\mathcal{P}} \cap mult(x)$ is non-empty, then it has a unique minimal element, that we denote ρ_x.*

Proof. Suppose that there are two distinct minimal elements $s_1, s_2 \in S_{\mathcal{P}} \cap mult(x)$. Since $s_1, s_2 \in S_{\mathcal{P}}$, then $s_1 \wedge s_2 \in S_{\mathcal{P}}$. Moreover, since x divides s_1 and s_2, it also divides $s_1 \wedge s_2$. Therefore $s_1 \wedge s_2 \in S_{\mathcal{P}} \cap mult(x)$, so s_1 and s_2 cannot be both minimal. ∎

COROLLARY 4.3. *Suppose that M has m atoms. If $\mathcal{P}$ is closed under gcd, then $\#(\min(S_{\mathcal{P}})) \leq m$.*

Proof. Notice that every element in M must be divisible by an atom. Take $s \in \min(S_{\mathcal{P}})$ and consider an atom $x \prec s$. Since s is minimal in $S_{\mathcal{P}}$, it is also minimal in $S_{\mathcal{P}} \cap mult(x)$. Hence $s = \rho_x$. Therefore

$$\min(S_{\mathcal{P}}) \subset \{\rho_x :\ x \text{ is an atom}\}$$

and the result follows. ∎

EXAMPLE 4.4. *In B_n^+ there are $n-1$ atoms, namely $\sigma_1, \dots, \sigma_{n-1}$. Therefore, if $\mathcal{P}$ is a property closed under gcd, then $\min(S_{\mathcal{P}})$ has at most $n-1$ elements, while $\#(S) = n!$*

EXAMPLE 4.5. *In BKL_n^+ there are $\frac{n(n-1)}{2}$ atoms (the generators in Presentation 2). Hence, if $\mathcal{P}$ is a property closed under gcd, then $\#(\min(S_{\mathcal{P}})) \leq \frac{n(n-1)}{2}$, while $\#(S) = \mathcal{C}_n > 3^n$.*

We must now define some suitable properties, closed under gcd, that will allow us to compute $C^{\geq m}(a)$ and $C^{sum}(a)$, for $a \in G$. These properties will depend on some given elements in M, so we will have an infinite number of properties, each one corresponding to a set of minimal simple elements.

4.1. Minimal simple elements to compute $\mathbf{C^{\geq m}(a)}$

DEFINITION 4.6. *Let $a \in G$ and $v \in C^{\geq m}(a)$, for some $m \in \mathbb{Z}$. We will say that a simple element s satisfies the property $\mathcal{P}_v^{\geq m}$ if it conjugates v to an element in $C^{\geq m}(a)$, that is, $s^{-1}vs \in C^{\geq m}(a)$.*

Proposition 4.7. *(Caracterization of elements satisfying $\mathcal{P}_v^{\geq m}$). If $v \in C^{\geq m}(a)$, one can write $v = \Delta^m w$, where $w \in M$. Then a simple element s satisfies the property $\mathcal{P}_v^{\geq m}$ if and only if $\tau^m(s) \prec ws$.*

Proof. The first assertion comes from the definition of infimum. Let then $v = \Delta^m w$, where $w \in M$, and let $s \in S$. One has $s^{-1}vs = s^{-1}\Delta^m ws = \Delta^m \tau^m(s^{-1})ws = \Delta^m(\tau^m(s))^{-1}ws$. Hence, s satisfies $\mathcal{P}_v^{\geq m}$ if and only if $(\tau^m(s))^{-1}ws \in M$, that is, $\tau^m(s) \prec ws$. ∎

Proposition 4.8. *For every $v \in M$ and every $m \in \mathbb{Z}$, the property $\mathcal{P}_v^{\geq m}$ is closed under gcd.*

Proof. Suppose that s_1 and s_2 satisfy $\mathcal{P}_v^{\geq m}$, and let $s = s_1 \wedge s_2$. Notice that τ preserves gcd's, since it preserves left divisibility. Hence $\tau(s) = \tau(s_1) \wedge \tau(s_2)$, and thus $\tau^m(s) = \tau^m(s_1) \wedge \tau^m(s_2)$.

One has $\tau^m(s) \prec \tau^m(s_1) \prec vs_1$ and $\tau^m(s) \prec \tau^m(s_2) \prec vs_2$. But it is easy to show that, for every $v \in M$, $vs_1 \wedge vs_2 = vs$. Hence, since $\tau^m(s)$ divides vs_1 and vs_2 then it divides its gcd, i.e. $\tau^m(s) \prec vs$. Therefore, s satisfies $\mathcal{P}_v^{\geq m}$, and the result follows. ∎

Definition 4.9. *For every $v \in C^{\geq m}(a)$, we define $S_v^{\geq m} = min(S_{\mathcal{P}_v^{\geq m}})$. That is, $S_v^{\geq m}$ is the set of minimal simple elements (with respect to $\prec$) among those who conjugate v to an element in $C^{\geq m}(a)$.*

Notice that, by Corollary 4.3 and Proposition 4.8, the cardinal of $S_v^{\geq m}$ for every $v \in C^{\geq m}(a)$ is no bigger than the number of atoms in M. Moreover, we have the following result, analogous to Proposition 3.7.

Proposition 4.10. *Given $u, v \in C^{\geq m}(a)$ for some $a \in G$, there exists a sequence $u = u_1, u_2, ..., u_k = v$ of elements in $C^{\geq m}(a)$ such that, for $i = 1, ..., k-1$, the elements u_i and u_{i+1} are conjugated by an element in $S_{u_i}^{\geq m}$.*

Proof. Just notice that any left or right divisor of a simple element is also a simple element, and then decompose every simple element in the sequence given by Proposition 3.7 into a product of minimal ones. ∎

This result implies that, in order to compute $C^{\geq m}(a)$ for $a \in M$, it suffices to conjugate every $v \in C^{\geq m}(a)$ by the elements in the small set $S_v^{\geq m}$.

4.2. Minimal simple elements to compute $\mathbf{C^{sum}(a)}$

Definition 4.11. *Let $a \in G$, and let $v \in C^{sum}(a)$. We will say that a simple element s satisfies the property $\mathcal{P}_v^{sum}$ if it conjugates v to an element in $C^{sum}(a)$. In other words, if the canonical length of $s^{-1}vs$ is equal to the canonical length of v (which is the summit length of a).*

Proposition 4.12. *For every $v \in C^{sum}(a)$, the property $\mathcal{P}_v^{sum}$ is closed under gcd.*

Proof. Let s_1 and s_2 be two simple elements satisfying $\mathcal{P}_v^{sum}$, and denote $s = s_1 \wedge s_2$. Write $s_i = sr_i$ for $i = 1, 2$, thus $r_1 \wedge r_2 = 1$.

Suppose that $\inf(v) = p$ and $\|v\| = t$. Then $v = \Delta^p v'$, where $v' \in M$ and we can write v' as a product of t simple elements (but not less). Since s_1 satisfies $\mathcal{P}_v^{sum}$, one has $s_1^{-1}vs_1 = s_1^{-1}\Delta^p\, v's_1 = \Delta^p\, \tau^p(s_1^{-1})\, v's_1 = \Delta^p\, (\tau^p(s_1))^{-1}v's_1$, where $(\tau^p(s_1))^{-1}v's_1 \in M$ and we can write it as a product of t simple elements, say $x_1 \cdots x_t$. The same happens for $(\tau^p(s_2))^{-1}v's_2 \in M$.

Now consider $s^{-1}vs$. By Proposition 4.8 it belongs to $C^{\geq p}(a)$, that is, $(\tau^p(s))^{-1}v's \in M$. We must show that we can write this element as a product of t simple elements. Suppose this is not true, and write $(\tau^p(s))^{-1}v's = z_1 \cdots z_{t+1}$ in right normal form (it has no more than $t+1$ factors since it is a right divisor of $v's$ which has $t+1$ factors). One has $x_1 \cdots x_t = (\tau^p(s_1))^{-1}v's_1 = (\tau^p(r_1))^{-1}(\tau^p(s))^{-1}v'sr_1 = (\tau^p(r_1))^{-1}z_1 \cdots z_{t+1}r_1$. Hence, $z_1 \cdots z_{t+1}r_1 = \tau^p(r_1)x_1 \cdots x_t$, and $z_1 \cdots z_{t+1}$ is in right normal form. Then by Corollary 2.11, $z_1 \prec \tau^p(r_1)$. In the same way, $z_1 \prec \tau^p(r_2)$. Therefore $z_1 \prec \tau^p(r_1) \wedge \tau^p(r_2) = \tau^p(r_1 \wedge r_2) = \tau^p(1) = 1$. A contradiction. ∎

Definition 4.13. *For every $v \in C^{sum}(a)$, we define $S_v^{sum} = min(S_{\mathcal{P}_v^{sum}})$. That is, S_v^{sum} is the set of minimal simple elements (with respect to $\prec$) among those who conjugate v to an element in $C^{sum}(a)$.*

As before, by Corollary 4.3 and Proposition 4.12, the cardinal of S_v^{sum} for every $v \in M$ is no bigger than the number of atoms in M. Furthermore, we can adjust the algorithm by Elrifai-Morton to these new sets, since we have the following result, analogous to Propositions 3.7 and 4.10.

Proposition 4.14. *For u, v conjugate elements in $C^{sum}(a)$, there exists a sequence $u = u_1, ..., u_k = v$ of elements in $C^{sum}(a)$ such that, for $i = 1, ..., k-1$, the elements u_i and u_{i+1} are conjugated by an element in $S_{u_i}^{sum}$.*

The proof of this result parallels that of Proposition 4.10. It implies that, in order to compute $C^{sum}(a)$ for $a \in G$, it suffices to conjugate every $v \in C^{sum}(a)$ by the elements in S_v^{sum}.

We have then described small subsets of S which suffice to compute $C^{\geq m}(a)$ and $C^{sum}(a)$. But we still need to show how to compute these subsets. This is what we do in the next section.

5. ALGORITHMS FOR THE CONJUGACY PROBLEM

We shall explain in this section our algorithms to compute $C^{\geq m}(a)$ and $C^{sum}(a)$, given $a \in G$.

Let us first explain a technical algorithm, which we did not find in the literature. Let $s \in S$ and $v \in M$. We will show how to compute their lcm $s \vee v$. More precisely, our algorithm will compute a simple element s' such that $s \vee v = vs'$. We must indicate that it is well known how to compute the lcm and the gcd of two simple elements, as well as the normal forms of any element in G.

Algorithm 1 (for computing s' such that $s \vee v = vs'$).

```
1. Compute the normal form of $v = v_1 \cdots v_t$.
2. $s_0 = s$.
3. For every $i = 1, \ldots, t$, compute $s_{i-1} \vee v_i$, and write it $v_i s_i$.
4. Return $s_t$.
```

Proposition 5.1. *Let $s \in S$ and $v \in M$. Let s_t be the simple element computed by Algorithm 1. Then $s \vee v = vs_t$.*

Proof. We proceed by induction on $t = \sup(v)$. If $t = 1$ the result is trivial, so suppose that $t > 1$ and the result is true for $t-1$. Denote $v' = v_1 \cdots v_{t-1}$. We have $s \vee v' = v's_{t-1}$, that is, s_{t-1} is the smallest element such that $v's_{t-1}$ is divisible by s. Therefore, an element $r \in M$ satisfies $s \prec vr = v'(v_t r)$ if and only if $s_{t-1} \prec v_t r$, and this is equivalent to $s_{t-1} \vee v_t \prec v_t r_t$, that is $v_t s_t \prec v_t r$ hence $s_t \prec r$. Therefore, s_t is the smallest element satisfying $s \prec vs_t$, as we wanted to show. ∎

5.1. Computation of $C^{\geq m}(a)$

Let $a \in G$ and $m \in \mathbb{Z}$. As we saw in Section 4, the main problem to compute $C^{\geq m}(a)$ is to compute $S_v^{\geq m}$, for every $v \in C^{\geq m}(a)$.

Let then $v \in C^{\geq m}(a)$ and let x be an atom of M. Consider the set $S_{\mathcal{P}_v^{\geq m}} \cap mult(x)$. It is always nonempty, since Δ satisfies $\mathcal{P}_v^{\geq m}$ (for every v) and is divisible by every atom (by definition of the Garside element). Then, by Lemma 4.2, this set has a unique minimal element, which we denote now r_x.

Recall that $S_v^{\geq m} \subset \{r_x : \ x \text{ is an atom}\}$, so our first step consists of computing r_x, for every atom $x \in M$.

Let $\inf(v) = p \geq m$, so $v = \Delta^m w$ where $w \in M$. Recall that, by Proposition 4.7, a simple element s satisfies $\mathcal{P}_v^{\geq m}$ if and only if $\tau^m(s) \prec ws$.

<u>**Algorithm 2**</u> (for computing r_x, minimal element in $S_{\mathcal{P}_v^{\geq m}} \cap mult(x)$).

```
1. Compute the left normal form of $v = \Delta^p w_1 \cdots w_t$.
2. If $p > m$ then return $x$ and stop.
3. $w = w_1 \cdots w_t$;  $s = x$.
4. Compute $\tau^m(s)$.
5. Use Algorithm 1 to compute $s'$ such that $\tau^m(s) \vee ws = wss'$.
6. If $s' = 1$ then return $s$ and stop.
7. $s = ss'$; go to Step 4.
```

Proposition 5.2. *Algorithm 2 gives an output, and it is r_x.*

Proof. First notice that, if we conjugate any $g \in G$ by a simple element, we can decrease the infimum of g by at most one. Hence, if $p > m$, $\inf(x^{-1}vx) \geq m$, so $r_x = x$ and the algorithm gives the correct output.

Now suppose $p = m$, we have computed $w \in M$ such that $v = \Delta^m w$, and we need to find the smallest r_x, such that $x \prec r_x$ and $\tau^m(r_x) \prec wr_x$. This is done as follows: we take a simple element s such that $x \prec s \prec r_x$ (at the first step $s = x$). Then we use Algorithm 1 to compute s' such that $\tau^m(s) \vee ws = wss'$. If $s' = 1$ then $\tau^m(s) \prec ws$, so $s = r_x$ and we obtain the correct output. Otherwise, notice that $\tau^m(s) \prec \tau^m(r_x) \prec wr_x$, so wr_x is divisible by $\tau^m(s)$ and by ws. Hence $wss' \prec wr_x$, so $ss' \prec r_x$.

Therefore, if s is not equal to r_x, the algorithm gives an element $s' \neq 1$ such that $s \prec ss' \prec r_x$, and it starts again checking if $ss' = r_x$. This process must stop, since the number of left divisors of r_x is finite, so the algorithm finds r_x in finite time. (moreover if M has homogeneus relations, like B_n^+ or BKL_n^+, we find r_x in at most $|\Delta|$ steps). ∎

<u>**Algorithm 3**</u> (for computing $S_v^{\geq m}$).

```
1. List the atoms of $M$, say $x_1, \ldots, x_\nu$. Set $R = \phi$.
2. For $i = 1, \ldots, \nu$, do the following:
    2a. Compute $r_{x_i}$, using Algorithm 2.
    2b. Compute $J_i = \{j : j \in R \text{ and } x_j \prec r_{x_i}\}$ and
                $K_i = \{j : j > i \text{ and } x_j \prec r_{x_i}\}$.
    2c. If $J_i = K_i = \phi$, then set $R = R \cup \{i\}$.
3. Return $\{r_{x_i} : i \in R\}$.
```

PROPOSITION 5.3. *Algorithm 3 computes $S_v^{\geq m}$.*

Proof. We know by Corollary 4.3 that $S_v^{\geq m}$ is the set of minimal elements in $\{r_{x_i} : i = 1, \dots, \nu\}$. We want to find a set $R \subset \{1, \dots, \nu\}$ such that $S_v^{\geq m} = \{r_{x_i} : i \in R\}$. Since we could have $r_{x_i} = r_{x_j}$ for some $i \neq j$, and we want R to be as small as possible, we define it in the following way: $i \in R$ if and only if r_{x_i} is minimal and there is no $j > i$ such that $r_{x_i} = r_{x_j}$.

Suppose that, for some i, we already computed the elements in $\{1, \dots, i-1\} \cap R$ (for $i = 1$, this is the empty set). Then we compute r_{x_i} and the sets J_i and K_i defined in the algorithm. If $i \notin R$, we have two possibilities: either there is some $j < i$ such that r_{x_j} is a proper divisor of r_{x_i}, or there is some $j > i$ such that $r_{x_j} \prec r_{x_i}$. In the first case $J_i \neq \phi$, and in the latter $K_i \neq \phi$. Therefore, if both sets are empty, $i \in R$.

Using this procedure for $i = 1, \dots, \nu$, the algorithm computes R, thus $S_v^{\geq m}$. ∎

REMARK 5.4. *Since $S_v^{\geq m}$ is just the subset of $\{r_{x_i} : i = 1, \dots, \nu\}$ formed by its minimal elements, we could have computed $S_v^{\geq m}$ just by comparing the r_{x_i}'s and keeping the minimal ones. We prefer to use Algorithm 3 since it is much faster to see if an atom divides an element, than to compare two elements, even if these two elements are simple ones.*

REMARK 5.5. *Algorithm 3 can still be improved in two different ways. First, we do not need to compute all r_{x_i}: if during the computation of r_{x_i} (using Algorithm 2), we see that $x_j \prec s$, for some $j \in R$ or some $j > i$, we can stop Algorithm 2 and increase the index i in Algorithm 3. Also, we do not need to compute the whole sets J_i and K_i: if we find some element belonging to one of them, we can directly increase the index i. We presented Algorithm 3 as above for the clarity of the exposition, and because these two improvements do not really change the complexity.*

Finally, let $a \in G$ and $m \in \mathbb{Z}$. The following algorithm works after Proposition 4.10.

Algorithm 4 (for computing $C^{\geq m}(a)$)

```
1. Compute the left normal form of $a$.
2. Apply repeated cycling to $a$, to obtain $\widehat{a} \in C^{\geq m}(a)$ (if it exists).
3. If $\widehat{a}$ is not obtained, return $\phi$ and stop.
4. Set $v = \widehat{a}$, $V = \{\widehat{a}\}$ and $W = \phi$.
5. Compute $S_v^{\geq m}$, using Algorithm 2.
6. For every $r \in S_v^{\geq m}$, do the following:
    6a. Set $w = r^{-1}vr \in C^{\geq m}(a)$.
    6b. Compute the left normal form of $w$.
    6c. If $w \notin V$, set $V = V \cup \{w\}$.
```

```
7. $W = W \cup \{v\}$.
8. If $V = W$ then return $V$ and stop.
9. Take a new $v \in V \backslash W$; go to Step 5.
```

REMARK 5.6. *If we take $m = 0$, then $C^{\geq 0}(a) = C^{+}(a) = C(a) \cap M$, so this algorithm can be used to compute all positive elements conjugated to a.*

5.2. Computation of $C^{sum}(a)$

Let now $a \in M$ and $v \in C^{sum}(a)$. Let $\inf(v) = p$ and $\|v\| = t$ (so $\sup(v) = p+t$), and consider an atom x of M. Similarly to the previous case, the key point to compute $C^{sum}(a)$ consists on finding the minimal element $\rho_x \in S_{\mathcal{P}_v^{sum}} \cap mult(x)$ (it exists since Δ belongs to this set, and it is unique by Lemma 4.2).

Algorithm 5 (for computing ρ_x, minimal in $S_{\mathcal{P}_v^{sum}} \cap mult(x)$)

```
1. Using Algorithm 2 compute $r_x$, minimal in $S_{\mathcal{P}_v^{\geq p}} \cap mult(x)$.
2. $s = r_x$.
3. Compute the right normal form of $s^{-1}vs$, say $w_1 \cdots w_k \Delta^p$.
4. If $k = t$, return $s$ and stop.
5. $s = sw_1$; go to step 3.
```

REMARK 5.7. *Algorithm 5 can be explained in a very natural way: first compute r_x and the right normal form $r_x^{-1}vr_x = w_1 \cdots w_k \Delta^p$. If k is not minimal, that is, if $k = t+1$, start* **decycling** *this right normal form, that is, compute the right normal form of $w_2 \cdots w_k \Delta^p w_1$ (and multiply r_x by w_1). If this new word has not minimal canonical length, decycle again, and so on. The proposition below shows that this works.*

PROPOSITION 5.8. *Algorithm 5 computes ρ_x, the minimal element in $S_{\mathcal{P}_v^{sum}} \cap mult(x)$.*

Proof. We can assume, by Proposition 5.2, that we already know r_x, the minimal element in $S_{\mathcal{P}_v^{\geq p}} \cap mult(x)$. We also know that $r_x \prec \rho_x$ by minimality of r_x, since $\rho_x^{-1} v \rho_x \in C^{sum}(a) \subset C^{\geq p}(a)$. So we have an element $s \in S$ such that $r_x \prec s \prec \rho_x$, $s^{-1}bs \in C^{\geq p}(a)$ and $\|s^{-1}vs\| \leq t+1$ (at the first step, $s = r_x$). We compute the right normal form of $s^{-1}vs$, say $w_1 \cdots w_k \Delta^p$.

There are two possible cases: either $k = t$ or $k = t+1$. If $k = t$, then $\|s^{-1}vs\| = t$, so $s = \rho_x$ and Step 4 gives the correct output. Otherwise, $s \neq \rho_x$ and there exists a non trivial element $s' \in S$ such that $ss' = \rho_x$. In this case

$$\rho_x^{-1} v \rho_x = (s')^{-1} s^{-1} v s s' = (s')^{-1} w_1 \cdots w_{t+1} \Delta^p s' = u_1 \cdots u_t \Delta^p,$$

where $u_1 \cdots u_t \Delta^p$ is in right normal form. Hence $w_1 \cdots w_{t+1} \Delta^p s' = s' u_1 \cdots u_t \Delta^p$, so $w_1 \cdots w_{t+1} \tau^{-p}(s') = s' u_1 \cdots u_t$. But $w_1 \cdots w_{t+1}$ is in right normal form, so by Corollary 2.11, $w_1 \prec s'$. Therefore $r_x \prec sw_1 \prec \rho_x$, and s is a proper divisor of sw_1. Also, $(sw_1)^{-1} v (sw_1) = w_2 \cdots w_{t+1} \Delta^q w_1 \in C^{\geq p}(a)$ and $\|(sw_1)^{-1} v (sw_1)\| \leq t+1$. So we can set $s = sw_1$, and start again.

This procedure must stop, finding $s = \rho_x$, since the number of left divisors of ρ_x is finite. (In homogeneous monoids, the number of steps is bounded by $|\Delta|$). ∎

Algorithm 6 (for computing S_v^{sum}).

```
1. List the atoms of $M$, say $x_1, \dots, x_\nu$. Set $R = \phi$.
2. For $i = 1, \dots, \nu$, do the following:
    2a. Compute $\rho_{x_i}$, using Algorithm 5.
    2b. Compute $J_i = \{j :\ j \in R$ and $x_j \prec \rho_{x_i}\}$ and
                $K_i = \{j :\ j > i$ and $x_j \prec \rho_{x_i}\}$.
    2c. If $J_i = K_i = \phi$ then $R = R \cup \{i\}$.
3. Return $\{\rho_{x_i} :\ i \in R\}$.
```

PROPOSITION 5.9. *Algorithm 6 computes S_v^{sum}.*

Proof. This algorithm parallels Algorithm 3, and it works in the same way, since S_v^{sum} is the set of minimal elements in $\{\rho_{x_i} :\ i = 1, \dots, n\}$. ∎

Finally, the following algorithm is analogous to Algorithm 4. Using the previous algorithms, it will then compute $C^{sum}(a)$ for any given $a \in G$.

Algorithm 7 (for computing $C^{sum}(a)$)

```
1. Compute the left normal form of $a$.
2. Using cyclings and decyclings, compute $\widetilde{a} \in C^{sum}(a)$.
3. $v = \widetilde{a}$;  $V = \{\widetilde{a}\}$;  $W = \phi$.
4. Compute $S_v^{sum}$, using Algorithm 6.
5. For every $r \in S_v^{sum}$, do the following:
    5a. $w = r^{-1} v r \in M$.
    5b. Compute the left normal form of $w$.
    5c. If $w \notin V$ then $V = V \cup \{w\}$.
6. $W = W \cup \{v\}$.
7. If $V = W$ then return $V$ and stop.
8. Take a new $v \in V \backslash W$; go to Step 4.
```

6. COMPLEXITY

In this section we shall study the complexity of our algorithms, applied to several examples of Garside monoids, such as B_n^+, BKL_n^+ and Artin monoids. We do not discuss here the general case, since the complexity strongly depends on the way of computing normal forms, LM(a), $a \vee b$, $\widetilde{a}$, etc. in each particular case.

We shall give theoretical upper bounds for this complexity. In the next section we will also compare, with many examples in B_n^+, the running time of our algorithm with the one by Elrifai and Morton, to show that our improvement is significant in practice.

We know that the theoretical results in this section can be improved: we are just interested on showing that our algorithms are much faster than the preceding ones, while sharper bounds for the complexity would require a deeper study.

6.1. The Artin braid monoid $\mathbf{B_n^+}$.

Complexity of computing $\mathbf{C^{\geq m}(a)}$

Recall the definition of B_n^+, and notice that the relations are homogeneous, so every two conjugate elements in B_n^+ have the same word length. Let then $v \in B_n^+$ be of word length l. Its left normal form $v_1 \cdots v_t$ verifies $t \leq l$.

Algorithm 1. It computes the normal form of v, which takes time $O(l^2 n \log n)$ (see[16]). Then it computes t words, $s_1, \dots, s_t$. The computation of each one takes time $O(n \log n)$ ([16]), so all of them are computed in time $O(ln \log n)$, and the whole algorithm has complexity $O(l^2 n \log n)$.

Now suppose that $v \in B_n$, and its left normal form is $\Delta^p w_1 \cdots w_t$. One has $t \leq l$. In B_n, the homomorphism τ can be defined as follows: $\tau(\sigma_i) = \sigma_{n-i}$. Hence $\tau^2 = \mathrm{id}$, so for every word w, $\tau^m(w)$ can be computed in time $O(|w|)$.

Algorithm 2. First it computes the left normal form of v in time $O(l^2 n \log n)$. If $p = m$, it runs the loop consisting of Steps 4-7. The two important steps are the following:

- **Step 4:** Notice that, every time the loop is repeated, we already know $\tau^m(s)$ for the old value of s. So in order to compute $\tau^m(ss') = \tau^m(s)\tau^m(s')$, we just need to compute $\tau^m(s')$, which is $O(|s'|)$. Since the product of all possible s' is still a simple element, r_x, all repetitions of this step can be made in time $O(n(n-1)/2)$, which is the length of Δ.

- **Step 5:** Here, when we apply Algorithm 1, we compute the normal form of ws, and the elements $s_1, \dots, s_t$, where s runs over an ascending chain of divisors of Δ. As above, all these computations together require the same number of operations as just applying Algorithm 1 to wr_x (see [16]). Moreover, we have computed the normal form of w at the beginning of Algorithm 2 so, again by [16], the normal form of wr_x can be computed in time $O(ln \log n)$. Hence all repetitions of this step can be made in time $O(ln \log n)$.

Therefore, the theoretical complexity of Algorithm 2 is $O(l^2 n^2)$.

Algorithm 3. The only non-negligible step is the loop of Step 2, which is repeated $n-1$ times (the number of atoms in B_n^+), and does the following: it computes r_{x_i} (which takes time $O(l^2 n^2)$) and it verifies at most $n-2$ times if an atom divides r_{x_i} (this takes time $O(n)$ by [16]). Hence, the complexity of Algorithm 3 is $O(l^2 n^3)$.

Finally, let $a \in B_n$ be given as a word of length l in the generators σ_i.

Algorithm 4. It starts by computing the normal form of a (time $O(l^2 n \log n)$). Then it finds $\widehat{a}$, which takes $O(l^2 n^3)$ by [4]. Next it starts a loop, which is repeated k times (the number of elements in $C^{\geq m}(a)$), and does the following: First, it computes $S_v^{\geq m}$, taking time $O(l^2 n^3)$. Then it runs another loop, repeated at most $n-1$ times, which works at follows:

- It computes the normal form of a word of length l, thus taking time $O(l^2 n \log n)$.

- It verifies if an element is in a list V, taking a negligible time compared to the previous step.

Finally, it verifies if $V = W$, but since $W \subset V$ we just have to compare the lengths. The time to do this is negligible. Therefore, the complexity of Algorithm 4 (i.e. the complexity of computing $C^{\geq m}(a)$), is $O\left(l^2 n \log n + l^2 n^3 + k(l^2 n^3 + (n-1) l^2 n \log n)\right)$, which yields the following.

Proposition 6.1. *Given $a \in B_n$ as a word of length l, the complexity of computing $C^{\geq m}(a)$ (for the Artin presentation) is $O\left(kl^2 n^3\right)$, where k is the number of elements in $C^{\geq m}(a)$.*

Remark that if we try to compute $C^{\geq m}(a)$ using the techniques of Elrifai and Morton, we should use Algorithm 4, but replacing $S_v^{\geq m}$ by the whole S, which has cardinality $n!$. The time would be in this case $O\left(kl^2(n!)n\log n\right)$.

Complexity of computing $\mathbf{C^{sum}(a)}$

The study of the complexity of Algorithms 5, 6 and 7, is very similar to that of Algorithms 2, 3 and 4.

Algorithm 5. It starts by computing r_x, taking $O(l^2n^2)$. Next it computes a right normal form ($O(l^2n\log n)$), and then it does a number of decyclings, which is bounded by $\frac{n(n-1)}{2}$. By [4], each decycling takes time $O(ln)$, so the whole complexity of Algorithm 5 is $O(l^2n^3)$.

Algorithm 6. It does the same as Algorithm 3, but it computes ρ_{x_i} instead of r_{x_i} in Step 2a. Hence its complexity is $O(l^2n^4)$.

Algorithm 7. It has two main differences with respect to Algorithm 4. It computes $\widetilde{a}$ instead of $\widehat{a}$ (but this can be made in $O(l^2n^3)$ by [4]), and it computes S_v^{sum} instead of $S_v^{\geq m}$. Therefore one has the following.

Proposition 6.2. *Given $a \in B_n$ as a word of length l, the complexity of computing $C^{sum}(a)$ (for the Artin presentation) is $O(kl^2n^4)$, where k is the number of elements in $C^{sum}(a)$.*

Remark that, if we compute the complexity of the algorithm by Elrifai and Morton, using the above methods, we obtain $O\left(kl^2(n!)n\log n\right)$.

6.2. The Birman-Ko-Lee monoid $\mathbf{BKL_n^+}$.

Complexity of computing $\mathbf{C^+(a)}$

We just need to follow here the same reasoning that in the previous subsection, taking into account the differences between BKL_n^+ and B_n^+. The complexities of the basic computations in BKL_n^+ can be found in [3]. For instance, in BKL_n^+, the computation of each s_i in Algorithm 1 takes time $O(n)$, the length of the Garside element is $n-1$, and the normal form of a word w is computed in time $O(|w|^2n)$. This implies that Algorithm 1 and Algorithm 2 both have complexity $O(l^2n)$.

In order to study Algorithm 3, we must know that the number of atoms in BKL_n^+ is $\frac{n(n-1)}{2}$, and to check if an atom divides a simple element takes time $O(n)$, so Algorithm 3 has complexity $O(l^2n^5)$.

Finally, the computation of $\widehat{a}$ takes time $O(l^2n^2)$ (see [4]), so the above method to compute the complexity of Algorithm 4 yields the following:

Proposition 6.3. *Given $a \in BKL_n$ as a word of length l, the complexity of computing $C^{\geq m}(a)$ (for the Birman-Ko-Lee presentation) is $O\left(kl^2n^5\right)$, where k is the number of elements in $C^{\geq m}(a)$.*

As above, the complexity of the previously known algorithm is much worse: $O(kl^2\mathcal{C}_n n)$, where $3^n < \mathcal{C}_n < 4^n$.

Complexity of computing $\mathbf{C^{sum}(a)}$

We just need to know that the computation of $\widetilde{a}$ takes time $O(l^2n^2)$ (see [4]). Hence the complexities of Algorithms 5, 6 and 7 are respectively $O(l^2n^2)$, $O(l^2n^5)$ and $O(kl^2n^5)$. Therefore, one has:

Proposition 6.4. *Given $a \in BKL_n$ as a word of length l, the complexity of computing $C^{sum}(a)$ (for the Birman-Ko-Lee presentation) is $O(kl^2n^5)$, where k is the number of elements in $C^{sum}(a)$.*

Notice that the complexity of the known algorithm was $O(kl^2\mathcal{C}_n n)$, so our algorithm improves it considerably.

One interesting remark is that our algorithm works faster, a priori, for the monoid B_n^+ than for BKL_n^+. This is due to a simple fact: in our algorithm the number of atoms is more relevant than the number of simple elements. In BKL_n^+, the number of simple elements is much smaller than in B_n^+, but the number of atoms is $\frac{n(n-1)}{2}$, while in B_n^+ is $n-1$.

6.3. Artin monoids

As we mentioned in the introduction, the Artin groups of finite type are Garside groups, so we can apply our algorithms to the corresponding Artin monoids (see [5] for an introduction to Artin monoids and groups). In [6] we can find algorithms to deal with Artin monoids: computation of normal forms, greatest common divisors, division algorithms, etc. Although these algorithms seem to be exponential in the length of the words involved, in [7] it is shown that finite type Artin groups are biautomatic, so there are quadratic algorithms to compute all of the above.

Nevertheless, since we are mainly interested in comparing our algorithms with the previous ones, we just need to know the length of the Garside element Δ, and the number of simple elements in any given Artin group. Let then G be an Artin group of rank n, that is, A_n, B_n, D_n, E_n (if $n = 6, 7, 8$), F_n (if $n = 4$), H_n ($n = 3, 4$) or $I_2(p)$ (if $n = 2$), and let h be its Coxeter number. It is known that $|\Delta| = \frac{nh}{2}$, where $h = O(n)$, and that $\#(S) \geq n!$.

Hence, if the complexity of the conjugacy algorithm by Elrifai and Morton is $O(xn!)$ for some x depending on n and l, our algorithm will have complexity $O(xn^3)$. This is shown by using the same arguments as in the previous subsections.

7. EFFECTIVE COMPUTATIONS

7.1. Comparison with the Elrifai-Morton algorthim

In the previous section, we found theoretical upper bounds for the complexity of our algorithms. We showed that our algorithm is, in theory, much better than the Elrifai-Morton one (for $n > 5$). In this section we effectively compare the two algorithms, in the following way: For given n and l, ($3 \leq n \leq 5$ and $10 \leq l \leq 20$) we took 5000 random pairs of positive braids in B_n of length l (using Artin presentation), we tested conjugacy using both algorithms, and we compared the Average Running Time (ART) and the Maximum Running Time (MRT). We did the same for $n = 6$ and $l = 10$, for 1144 pairs.

We can conclude that our algorithm is faster for $n \geq 4$, and much faster for $n \geq 5$ (We were not able to compute the cases $n = 5$ and $l = 19, 20$ using the Elrifai-Morton algorithm since the computations were too long).

In the tables below one can see the results: We wrote F-GM for our algorithm and E-M for the Elrifai-Morton one. The time is given in seconds.

$n = 3$

l	10	11	12	13	14	15
ART F-GM	0.1526	0.2011	0.2361	0.3038	0.3386	0.3951
ART E-M	0.1144	0.1460	0.1692	0.2133	0.2367	0.2723
MRT F-GM	2.429	3.599	4.680	6.080	7.450	6.960
MRT E-M	1.659	2.539	3.220	4.089	5.029	4.599

l	16	17	18	19	20
ART F-GM	0.3896	0.5021	0.5473	0.6494	0.7292
ART E-M	0.2710	0.3392	0.3710	0.4329	0.4841
MRT F-GM	11.299	10.530	12.469	15.090	16.539
MRT E-M	7.219	6.729	7.970	9.950	11.039

$n = 4$

l	10	11	12	13	14	15
ART F-GM	0.3559	0.4796	0.6772	0.7870	1.0264	1.2599
ART E-M	0.6118	0.7233	1.0127	1.2086	1.5909	1.9538
MRT F-GM	8.680	11.390	16.519	23.949	33.969	42.029
MRT E-M	16.319	22.329	28.440	41.579	61.790	74.999

l	16	17	18	19	20
ART F-GM	1.4548	1.7436	2.2029	2.6616	2.9942
ART E-M	2.3106	2.7995	3.5548	4.3280	4.7226
MRT F-GM	41.910	62.940	72.940	103.470	148.989
MRT E-M	70.039	107.969	137.720	173.060	245.740

$n = 5$

l	10	11	12	13	14	15
ART F-GM	1.0997	1.8463	2.7657	3.7962	3.8195	4.4797
ART E-M	7.8690	11.1207	17.1455	23.2491	26.2595	29.7934
MRT F-GM	21.239	46.070	65.530	88.940	139.180	155.260
MRT E-M	177.489	322.039	456.669	611.609	1068.970	1178.790

l	16	17	18	19	20
ART F-GM	5.6410	7.1540	8.8198	9.4597	10.6614
ART E-M	38.7974	51.0028	62.0018		
MRT F-GM	254.770	411.320	401.409	516.119	532.469
MRT E-M	2116.239	3221.880	3218.93		

$n = 6$

l	10
ART F-GM	2.2450
ART E-M	506.224
MRT F-GM	43.935
MRT E-M	7495.288

7.2. Exhaustive computation of conjugacy classes and summit classes

In the previous section, we saw that the complexity of all our algorithms depends on the size of the sets $C^{\geq m}(a)$ or $C^{sum}(a)$, for $a \in M$. In the cases of B_n^+ or BKL_n^+, the only upper bounds known for these sets are exponential in n and in $l = |a|$. Nevertheless, we have the following (recall that, in this case, $C^+(a) = C^{\geq 0}(a) = C(a) \cap B_n^+$):

Conjecture: (Thurston, [16]) Let n be a fixed integer and let $a \in B_n^+$, having word length l. There is an upper bound for $C^+(a)$ which is a polynomial in l.

The existence of this upper bound for $C^+(a)$, or even for $C^{sum}(a)$, would imply the following:

Conjecture: (Birman, Ko and Lee, [4]) For every fixed integer n, there exists a solution for the conjugacy problem in B_n, which is polynomial in the word length of the elements involved.

In order to have some numerical evidence to support these conjectures, we have computed, for $n = 3, \ldots, 8$ and several values of l, all the conjugacy classes of words of length l in B_n^+, as well as the corresponding summit classes. In the tables below we present the following data, for the set W_l of elements in B_n^+ having word length l:

- $\mathbf{CC^+}$: The number of Conjugacy Classes in $W_l \subset B_n^+$.
- **max** $\mathbf{C^+}$: The size of the biggest one. That is, the number of elements in the biggest $C^+(a)$, for $a \in W_l$.
- **max** $\mathbf{C^{sum}}$: The size of the biggest summit class.
- **v**: A representative from one of those biggest summit class. That is, an element $v \in C^{sum}(a)$, where $C^{sum}(a)$ has maximal size.

n=3				
l	CC^+	$\max C^+$	$\max C^{sum}$	v
4	3	6	2	$\sigma_1^3\sigma_2$
5	3	10	6	$\sigma_1^3\sigma_2^2$
6	5	12	8	$\sigma_1^4\sigma_2^2$
7	5	16	10	$\sigma_1^5\sigma_2^2$
8	8	20	12	$\sigma_1^6\sigma_2^2$
9	9	29	14	$\sigma_1^7\sigma_2^2$
10	13	30	16	$\sigma_1^8\sigma_2^2$
11	16	40	18	$\sigma_1^9\sigma_2^2$
12	27	48	20	$\sigma_1^{10}\sigma_2^2$
13	33	64	22	$\sigma_1^{11}\sigma_2^2$
14	50	80	24	$\sigma_1^{12}\sigma_2^2$
15	70	125	26	$\sigma_1^{13}\sigma_2^2$
16	107	126	28	$\sigma_1^{14}\sigma_2^2$
17	153	160	30	$\sigma_1^{15}\sigma_2^2$
18	241	192	32	$\sigma_1^{16}\sigma_2^2$
19	349	256	34	$\sigma_1^{17}\sigma_2^2$
20	542	320	36	$\sigma_1^{18}\sigma_2^2$

n=4				
l	CC^+	$\max C^+$	$\max C^{sum}$	v
4	7	12	4	$\sigma_1^3\sigma_2$
5	9	20	12	$\sigma_1^3\sigma_2^2$
6	16	40	16	$\sigma_1^4\sigma_2^2$
7	21	54	22	$\sigma_1^5\sigma_2\sigma_3$
8	36	72	32	$\sigma_1^6\sigma_2\sigma_3$
9	54	94	50	$\sigma_1^4\sigma_2^2\sigma_3^2\sigma_2$
10	96	156	60	$\sigma_1^5\sigma_2^2\sigma_3^2\sigma_2$
11	160	252	70	$\sigma_1^6\sigma_2^2\sigma_3^2\sigma_2$
12	304	344	88	$\sigma_1^5\sigma_2^2\sigma_1\sigma_3\sigma_1\sigma_2\sigma_3$
13	538	582	114	$\sigma_1^6\sigma_2^2\sigma_1\sigma_3\sigma_1\sigma_2\sigma_3$
14	1030	752	140	$\sigma_1^7\sigma_2^2\sigma_1\sigma_3\sigma_1\sigma_2\sigma_3$
15	1954	1114	166	$\sigma_1^8\sigma_2^2\sigma_1\sigma_3\sigma_1\sigma_2\sigma_3$

n=5				
l	CC^+	$\max C^+$	$\max C^{sum}$	v
4	10	24	8	$\sigma_1^2\sigma_2\sigma_3$
5	15	36	18	$\sigma_1^3\sigma_2^2$
6	28	80	24	$\sigma_1^4\sigma_2^2$
7	44	136	44	$\sigma_1^5\sigma_2\sigma_3$
8	81	188	64	$\sigma_1^6\sigma_2\sigma_3$
9	141	288	104	$\sigma_1^5\sigma_2\sigma_3\sigma_2\sigma_4$
10	281	516	156	$\sigma_1^6\sigma_2\sigma_3\sigma_2\sigma_4$
11	520	702	208	$\sigma_1^7\sigma_2\sigma_3\sigma_4^2$
12	1194	1018	260	$\sigma_1^8\sigma_2\sigma_3\sigma_4^2$

n=6				
l	CC^+	$\max C^+$	$\max C^{sum}$	v
4	13	36	16	$\sigma_1\sigma_2\sigma_3\sigma_4$
5	22	56	30	$\sigma_1\sigma_2\sigma_1\sigma_4^2$
6	44	120	36	$\sigma_1^4\sigma_2\sigma_3$
7	76	272	72	$\sigma_1^4\sigma_2\sigma_3\sigma_4$
8	148	412	124	$\sigma_1^5\sigma_2\sigma_3\sigma_4$
9	276	576	208	$\sigma_1^5\sigma_2\sigma_3\sigma_4^2$
10	573	1032	372	$\sigma_1^5\sigma_2\sigma_3\sigma_4\sigma_5^2$

n=7				
l	CC^+	$\max C^+$	$\max C^{sum}$	v
4	14	60	24	$\sigma_1\sigma_2\sigma_4^2$
5	26	84	60	$\sigma_1\sigma_2\sigma_1\sigma_4^2$
6	56	160	72	$\sigma_1\sigma_2\sigma_4\sigma_2\sigma_1^2$
7	104	408	108	$\sigma_1^4\sigma_2\sigma_3\sigma_4$
8	215	824	192	$\sigma_1^4\sigma_2\sigma_3\sigma_4\sigma_5$
9	424	1160	416	$\sigma_1^4\sigma_2\sigma_3\sigma_4\sigma_5^2$
10	914	1992	744	$\sigma_1^5\sigma_2\sigma_3\sigma_4\sigma_5^2$

n=8				
l	CC^+	$\max C^+$	$\max C^{sum}$	v
4	15	100	48	$\sigma_1\sigma_2\sigma_3\sigma_5$
5	29	144	100	$\sigma_1\sigma_2\sigma_1\sigma_4^2$
6	66	216	144	$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_5^2$
7	130	544	168	$\sigma_1\sigma_2\sigma_1\sigma_3\sigma_4\sigma_6^2$
8	281	1236	360	$\sigma_1\sigma_2\sigma_1\sigma_4^3\sigma_5^2$

ACKNOWLEDGMENTS

The main ideas in this work were developed during a stay of both authors at the *Laboratoire de Topologie de l'Université de Bourgogne* at Dijon (France). We are very grateful to all members of the Laboratoire, and in particular to Luis Paris for his many useful suggestions. We are also grateful to Jean Michel for his precise and helpful comments on an earlier version of this paper.

REFERENCES

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

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

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

[4] J. Birman, K. H. Ko and S. J. Lee, The infimum, supremum and geodesic length of a braid conjugacy class, Preprint (2000).

[5] N. Bourbaki, "Groupes et algebres de Lie", Chaps. IV-VI, Hermann, Paris, 1968.

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

[7] R. Charney, Artin groups of finite type are biautomatic, *Math. Ann.* **292**, No. 4 (1992), 671-683.

[8] P. Dehornoy, Groupes de Garside, *Ann. Scient. Éc. Norm. Sup.*, 4^e série, t. 35, 2002, 267-306.

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

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

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

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

[13] J. Michel, A note on words in braid monoids, *J. of Algebra* **215** (1999) 366-377.

[14] M. Picantin, Petits groupes gaussiens, Ph. D. Thesis, Université de Caen (2000).

[15] M. Picantin, The conjugacy problem in small Gaussian groups, *Comm. Algebra* **29**, No. 3 (2001), 1021-1039.

[16] W. P. Thurston, 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.