An algorithm for the word problem in braid groups

Bert Wiest

UFR de Mathématiques, Université de Rennes 1, Campus de Beaulieu,
35042 Rennes cedex, France; bertw@math.univ-rennes1.fr

Abstract We suggest a new algorithm for finding a canonical representative of a given braid, and also for the harder problem of finding a σ_1-consistent representative. We conjecture that the algorithm is quadratic-time. We present numerical evidence for this conjecture, and prove two results: (1) The algorithm terminates in finite time. (2) The conjecture holds in the special case of 3-string braids - in fact, we prove that the algorithm finds a minimal-lenght representative for any 3-string braid.

AMS Classification 20F36; 20F60, 20F65

Keywords braid, word problem, quasigeodesic

1 Introduction

In this paper we propose an algorithm for finding a unique short representative for any given element of the Artin braid group

$$B_n \cong \langle \sigma_1, \ldots, \sigma_{n-1} \mid [\sigma_i, \sigma_j] = 0 \text{ if } |i-j| \geqslant 2,\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \rangle.$$

Several algorithms with the same aim are already known, for instance Artin's combing of pure braids [1], as well as Garside's [7, 4, 6] and Dehornoy's [2] algorithms. Still, we believe that the new algorithm, which we call the *relaxation algorithm,* is of theoretical interest: if our conjecture that the algorithm is efficient – a conjecture which is supported by strong numerical evidence – is correct, then we would be dealing with a new type of convexity property of mapping class groups. A particularly surprising aspect is that the idea of the algorithm appears to have been well-known to the experts for decades (we just have to be specific about some details), only its efficiency has apparently been overlooked.

The algorithm is very geometric in nature, exploiting the natural identification of B_n with $\mathcal{MCG}(D_n)$, the mapping class group of the n times punctured disk. The idea is very naive, and not new – c.f. [8, 5]. One thinks of the disk D_n as being made of an elastic material; if a given braid is represented by a homeomorphism $\varphi\colon D_n \to D_n$, then one obtains a representative of the inverse of the braid by authorizing the puncture points to move, and letting the map φ relax into the identity map. However, this relaxation process is decomposed into a sequence of applications of generators of the braid group, where in each step one chooses the generator which reduces the tension of the elastic D_n by as much as possible. Of course, both the notion of "tension/relaxation", as well as the choice of generating system of the braid group, must be specified carefully. Here are, in more detail, the properties of the algorithm.

(a) The algorithm appears to be quadratic-time in the length of the input braid, and the length of the output braid appears to be bounded linearly by the length of the input braid. Unfortunately, we are not currently able to prove this. However, we shall prove some partial results, and report on some strong numerical evidence supporting the conjecture that the algorithm is efficient in the above sense.

(b) The algorithm can be finetuned to output only σ_1-consistent braids (in the sense of Dehornoy [3]: the output braid words may contain the letter σ_1 but not σ_1^{-1}, or vice versa). The bounds from (a) on running time and length of the output word still seem to hold for this modified algorithm. This is remarkable, because it is not currently known whether every braid of lenght l admits a σ_1-consistent representative of length $c(n) \cdot l$, where $c(n) > 1$ is a constant depending only on the number of strings n.

Characteristics (a) and (b) are almost shared by Dehornoy's handle reduction algorithm [2]: it is believed to be of cubic complexity and to yield σ_1-consistent braids of linearly bounded length, but this is currently only a conjecture.

(c) The algorithm generalizes to other surface braid groups. We conjecture that in this setting as well it is efficient in the sense of (a) above (note that we have no reasonable notion of σ_1-consistency here).

(d) The main interest of the algorithm, however, is theoretical: any step towards finding a polynomial bound on its algorithmic complexity would likely give new insights into the interactions between the geometry of the Cayley graph of B_n and the dynamical properties of braids. Indeed, we conjecture that the output braid words are quasi-geodesics in the Cayley graph. Moreover, the problems raised here may be linked to the question whether train track splitting sequences are quasi-geodesics, as well as to the efficiency of the Dehornoy algorithm [2].

The plan of this paper is as follows: in section 2 we describe the algorithm and its possible modifications. In section 3 we present numerical evidence that the algorithm is efficient. In section 4 we prove that it terminates in finite time.

In section 5 we prove that in the special case of 3-string braids the algorithm is indeed quadratic time, and finds a shortest possible representative for any braid.

2 The algorithm

We start by setting up some notation. We denote by D_n ($n \in \mathbb{N}$) the closed disk in the complex plane which intersects $\mathbb{R}$ in the interval $[0, n+1]$, but with the points $1, \ldots, n \in \mathbb{C}$ removed. We recall that the braid group B_n is naturally isomorphic to the mapping class group $\mathcal{MCG}(D_n)$. We denote by E the diagram in D_n consisting of $n-1$ properly embedded line segments intersecting the real axis halfway between the punctures, as indicated in figure 1. We shall also consider the diagram E' in D_n, consisting of $n+1$ horizontal open line segments. The arcs of both diagrams are labelled as indicated.

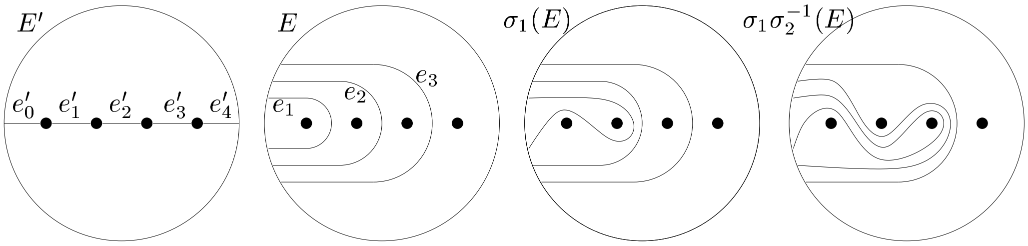

Figure 1: The diagram E', and reduced curve diagrams for id, σ_1 and $\sigma_1\sigma_2^{-1} \in B_4$

Given any homeomorphism φ: $D_n \to D_n$ with $\varphi|_{\partial D^2} = id$, we write $[\varphi]$ for the element of $\mathcal{MCG}(D_n)$ represented by φ. A *curve diagram* for $[\varphi] \in \mathcal{MCG}(D_n)$ is the image $\varphi(E)$ of E under any homeomorphism φ: $D_n \to D_n$ representing $[\varphi]$. We say a curve diagram is *reduced* if it intersects the diagram E' in finitely many points, transversely, in such a way that $\varphi(E)$ and E' together do not enclose any bigons. This is equivalent to requiring that the number of intersection points $\varphi(E) \cap E'$ be minimal among all homeomorphisms representing $[\varphi]$ (see e.g. [5]). Reduced curve diagrams are essentially unique in the sense that any two reduced curve diagrams of $[\varphi] \in \mathcal{MCG}(D_n)$ can be deformed into each other by an isotopy of D_n which fixes E' setwise.

We shall call the diagram E the "trivial curve diagram", because it is a reduced curve diagram for the trivial element of $\mathcal{MCG}(D_n)$.

We define the *complexity* of an element $[\varphi]$ of $\mathcal{MCG}(D_n)$, and of any curve diagram of $[\varphi]$, to be the number of intersections points of a reduced curve diagram of $[\varphi]$ with E'. In other words, the complexity of $[\varphi]$ is $\min|\psi(E)\cap E'|$, where the minimum is taken over all homeomorphisms ψ isotopic to φ. For instance, the elements $[id], \sigma_1$ and $\sigma_1\sigma_2^{-1}$ of $\mathcal{MCG}(D_4)$ have complexity $3, 5$ and 9, respectively. We remark that the complexity can grow exponentially with the number of crossings of the braid. (The reader may feel that our definition of complexity is quite arbitrary. In informal computer experiments we have tested some variations of this definition, and the results appear to be qualitatively unchanged.)

Next we define the set of generators of the braid group $\mathcal{MCG}(D_n)$ which will be useful for our purposes: we define a *semicircular move* to be any element of $\mathcal{MCG}(D_n)$ either of the form $\sigma_i^\epsilon\sigma_{i+1}^\epsilon\ldots\sigma_{j-1}^\epsilon\sigma_j^\epsilon$ with $i<j$, or of the form $\sigma_i^\epsilon\sigma_{i-1}^\epsilon\ldots\sigma_{j+1}^\epsilon\sigma_j^\epsilon$ with $i>j$; in either case we have $\epsilon=\pm1$ and $i,j\in\{1,\ldots,n-1\}$. In order to explain the name, we remark that semicircular braids can be realised by a semicircular movement of one puncture of D_n in the upper or lower half of D_n back into $D_n\cap\mathbb{R}$. (The movement is in the upper half if $j>i$ and $\epsilon=1$, or if $j<i$ and $\epsilon=-1$, it is in the lower half in the reverse case, and when $i=j$ then it can be regarded as lying in either half.) There are $2(n-1)^2$ different semicircular moves in $\mathcal{MCG}(D_n)$.

We say a semicircular movement is *disjoint from* a curve diagram D if it is given by a movement of a puncture along a semicircular arc in the upper or lower half of D_n which is disjoint from the diagram D.

The "standard" relaxation algorithm

We are now ready to describe the algorithm. One inputs a braid $[\varphi]$, (e.g. as a word in the generators σ_i), and it outputs an expression of $[\varphi]^{-1}$ as a product of semicircular moves (and thus of standard generators $\sigma_i^{\pm1}$):

Step 1 Construct the curve diagram D of $[\varphi]$.

Step 2 For each semicircular movement γ which exists in B_n and which is disjoint from D, calculate the complexity of $\gamma(D)$.

Step 3 Among the possible moves in step 2, choose the one (γ' say) which yields the minimal complexity. (If several moves yield the same minimal complexity, choose one of them arbitrarily or e.g. the lexicographically smallest one.) Write down the move γ', and define $D:=\gamma'(D)$.

Step 4 If D is the trivial diagram, terminate. If not, go to Step 2.

This could be summarized in one sentence: iteratively untangle the curve diagram of $[\varphi]$, where each iteration consists of the one semicircular move which simplifies the curve diagram as much as possible. We shall prove in section 4 that the algorithm terminates, i.e. that every nontrivial curve diagram admits a move which decreases the complexity. Note that there is no obvious reason to expect this algorithm to be efficient: one might expect that one has to perform some seemingly inefficient steps first which then allow a very rapid untangling later on. Yet, as we shall see, it appears that such phenomena do not occur.

The σ_1-consistent version of the relaxation algorithm

Next we indicate how the algorithm can be improved in order to output only σ_1-consistent braid words. We recall that a braid word is σ_1*-consistent* if it contains only the letter σ_1 but not σ_1^{-1} ("σ_1-positive"), or if it contains σ_1^{-1} but not σ_1 ("σ_1-negative"), or indeed if it contains no letter $\sigma_1^{\pm 1}$ at all ("σ_1-neutral"). It is a theorem of Dehornoy [3] that every braid has a σ_1-consistent representative. More precisely, it was shown in [5] that a braid $[\varphi]$ is σ_1-positive if and only if in a reduced curve diagram $\varphi(E)$, the "first" intersection of $\varphi(e_1)$ with E' lies in e'_0, where e_1 is oriented from bottom to top. By contrast, the braid is σ_1-negative if and only if in the opposite orientation of e_1 (from top to bottom) the "first" intersection with E' lies in e'_0. Finally, a braid is σ_1-neutral if and only if $\varphi(e_1) \sim e_1$. By abuse of notation we shall also speak of a reduced curve diagram as being σ_1-positive, negative, or neutral. Roughly speaking, a diagram is σ_1-positive if $\varphi(e_1)$ starts by going up, and σ_1-negative if it starts by going down. We remark that these geometric conditions are easy to check once the curve diagram of a braid has been calculated: using the notation of the appendix, a curve diagram is σ_1-positive if $d^0_{/} > 0$ holds, it is σ_1-negative if $d^0_{\backslash} > 0$ holds, and it is σ_1-neutral if $d^0_{/} = d^0_{\backslash} = 0$. For instance, one sees in figure 1 that the curve diagrams for both $[\sigma_1]$ and $[\sigma_1\sigma_2^{-1}]$ are σ_1-positive.

Here are the changes that need to be made to the above algorithm: after Step 1 is completed, one checks whether the braid $[\varphi]$ is σ_1-positive, negative, or neutral. For definiteness let us say it is σ_1-positive (the σ_1-negative case being symmetric). Thus during the untangling process we want to avoid using semicircular moves which involve the letter σ_1, only σ_1^{-1}s are allowed. Thus in Step 3 we cannot choose among *all* semicircular moves which are disjoint from D, but we restrict our choice to those moves γ which satisfy

(a) the move γ does not involve the letter σ_1,

(b) the braid $[\varphi]\gamma^{-1}$ is not σ_1-negative; in other words, the curve diagram $\gamma(D)$ is still σ_1-positive or σ_1-neutral, but not σ_1-negative.

Note that condition (b) is really necessary: if applying γ turned our σ_1-positive curve diagram D into a σ_1-negative one, then we would have no chance of completing the untangling process without using the letter σ_1 later on.

Remark 2.1 The restriction to semicircular moves *which are disjoint from the curve diagram* is superfluous – all results in this paper are true without it, and indeed for the standard algorithm the semicircular movement which reduces complexity by as much as possible is automatically disjoint from the diagram. We only insist on this restriction because it simplifies the proof in section 5

Example 2.2 The curve diagram of the braid $\sigma_1^{-1}\sigma_2\sigma_1$ is shown in figure 2(a). The move $\sigma_1^{-1}\sigma_2^{-1}$ reduces the complexity of the diagram by 4, whereas σ_2 (the only other semicircular move disjoint from the diagram) reduces the complexity only by 2. However, the move $\sigma_1^{-1}\sigma_2^{-1}$ is forbidden, since its action would turn the positive diagram into a negative one. Thus we start by acting by σ_2, and the resulting diagram is shown in figure 2(b). In this diagram, the action of $\sigma_1^{-1}\sigma_2^{-1}$ is legal, and relaxes the diagram into the trivial one. In total, the curve diagram of $\sigma_1^{-1}\sigma_2\sigma_1$ was untangled by $\sigma_2\ \sigma_1^{-1}\sigma_2^{-1}$.

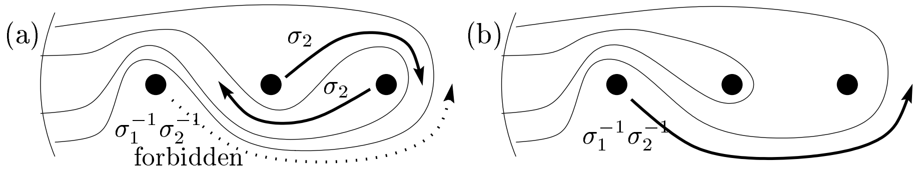

Figure 2: Untangling the curve diagram of the braid $\sigma_1^{-1}\sigma_2\sigma_1$ yields $\sigma_2 \cdot \sigma_1^{-1}\sigma_2^{-1}$.

3 Numerical evidence

The above algorithms were implemented by the author using the programming language C [10]. The aim of this section is to report on the results of systematic experimentation with this program. A brief description how curve diagrams were coded and manipulated can be found in Appendix 1. Since the algorithm involves calculations with integer numbers whose size grows exponentially with the length of the input braid, and no software capable of large integer arithmetic was used, the search was restricted to braids with up to 50 crossings.

The following notation will be used: for any braid word w in the letters $\sigma_i^{\pm 1}$ we denote by $l(w)$ the number of letters (i.e. the number of crossings), and by

$l_{\text{out}}(w)$ the number of letters of the output of our algorithm. The results of the experiments have two surprising aspects: firstly, they strongly support the following

Conjecture 3.1 (Main conjecture) *For every $n > 2$ there exists a constant $c(n) \geqslant 1$ such that for all braid words w we have $l_{\text{out}}(w) < c(n)l(w)$.*

Secondly, they show that the σ_1-consistent version of the algorithm very often yields *shorter* output-braids than the standard version.

The first table gives the average length of the output braid among 10.000 randomly generated braid words of length 10, 20, 30, 40 and 50, with 4, 5, and 6 strings. Our random braid words did not contain any subwords of the form $\sigma_i^{\epsilon}\sigma_i^{-\epsilon}$, i.e. they did not contain any obvious simplifications. Each entry in the table is of the form ***/***, where the first number refers to the standard algorithm, and the second one to the σ_1-consistent one.

	length 10	length 20	length 30	length 40	length 50
N=4	8.5/8.4	16.0/15.6	23.4/22.8	30.9/29.9	38.3/36.8
N=5	8.7/8.7	17.2/17.0	25.9/25.3	34.5/33.8	43.2/42.2
N=6	8.7/8.6	17.3/17.2	26.2/26.2	35.6/35.7	45.0/45.0

The next table shows the worst cases (that is, the longest output words) that occurred among $\geqslant$ 40.000.000 randomly generated braids. (Thus every entry in this table required 40.000.000, and in some cases much more, complete runs of the algorithm.)

	length 10	length 20	length 30	length 40	length 50
N=4	24 / 20	52 / 38	70 / 54	86 / 66	106 / 80
N=5	26 / 26	58 / 54	76 / 66	116 / 82	130 / 104
N=6	30 / 38	78 / 68	102 / 100	140 / 120	178 / 132

While these values appear to support conjecture 3.1, one should keep in mind that the worst examples may be exceedingly rare, and thus may have been overlooked by the random search.

If conjecture 3.1 holds, then we would have that the running time of the algorithm (on a Turing machine) is bounded by $c'(n)(l(w))^2$, where c' is some other constant depending only on n. This is because there is an exponential bound on the numbers $d^i_{\supset}, d^i_{\subset}, d^i_{\overline{\cdot\cdot}}, d^i_{\underline{\cdot\cdot}}, d^i_{/}$, and $d^i_{\backslash}$ describing the curve diagram (see appendix) in terms of the length of the braid word. Performing an elementary operation (addition, comparison etc.) on numbers bounded by $\exp(l(w))$ takes time at most $O(\log(\exp(l(w)))) = O(l(w))$. Since the number of operations performed in each step of the algorithm is constant, and the number of steps which need to be performed is (we conjecture) bounded linearly by $l(w)$, we get a total running time bounded by $c'(n)(l(w))^2$.

The algorithm admits variations. For instance, one can use a different diagram on D_n as a trivial curve diagram (instead of E), and one can define the complexity of a curve diagram in a different way (e.g. not counting the intersection points of $\varphi(E)$ but those of $\varphi^{-1}(E)$ with the horizontal axis). Informal experiments indicate that our results seem to be "stable" under such modifications of the algorithm: computation time and length of output braids still appear to be quadratic, respectively linear.

It should also be mentioned that the requirement that the complexity of the curve diagrams be reduced *as much as possible* in each step is essential: if one only asks for the complexity to be reduced (by any amount), then $l_{\text{out}}(w)$ can depend exponentially on $l(w)$.

The reader should be warned of two attacks on conjecture 3.1 which appear not to work. Firstly, running the algorithm on pairs of words $(w, w\sigma_i)$ or $(w, \sigma_i w)$, where σ_i is a generator, often yields pairs of output braid words with dramatically different lengths. Thus it appears unlikely that any fellow-traveller type property between output braid words can serve as an explanation for the algorithm's efficiency. Secondly, there are instances of braids where the number of semicircular factors in the output braid word is larger than the number of Artin factors in the input braid word.

One may hope, however, that Mosher's automatic structure on B_n [9] can help to prove the main conjecture 3.1, because the conjecture would follow from a positive answer to the following

Question Suppose that the algorithm produces a sequence $D^{(1)}, \ldots, D^{(N)}$ of curve diagrams of decreasing complexity and with $D^{(N)} = E$, representing braids $[\varphi_1], \ldots, [\varphi_N]$ in $\mathcal{MCG}(D_n)$. (In particular, $[\varphi_N] = 1$). Is it true that the lengths of the Mosher normal forms (the "flipping sequences") of these braids are strictly decreasing?

4 The algorithm terminates

While we have no explanation for the extraordinarily good performance of the algorithm, we can at least prove that it terminates in finite time (and one could easily deduce from our proof an exponential bound on the running time). In fact, in order to ascertain the termination of the algorithm, even of its σ_1-consistent version, it suffices to prove

Proposition 4.1 *For any reduced, σ_1-positive curve diagram D in D_n there exists a semicircular braid γ in the upper half of D_n such that the diagram $\gamma(D)$ is still σ_1-positive or σ_1-neutral and such that the complexity of $\gamma(D)$ is strictly lower than the complexity of D.*

We recall that by a semicircular braid "in the upper half of D_n" we mean one which can be realized by a simicircular movement of one of the punctures in the part of D_n belonging to the upper half plane.

Let us first see why this proposition implies that the algorithm terminates. The curve diagram of the input braid $[\varphi]$ has finite complexity (and in fact if $[\varphi]$ is given as a braid word of length l in the letters σ_i^ϵ then one can easily obtain an exponential bound on the complexity in terms of l and n: the complexity is bounded by $(n-1)l^3$). If $[\varphi]$ is σ_1-positive, then the proposition implies that the curve diagram can be turned into a σ_1-neutral diagram by applying a sequence of semicircular braids which contain no letter σ_1. By a symmetry argument we can prove an analogue statement for σ_1-negative curve diagrams: they can be untangled by applying a σ_1-positive braid. Once we have arrived at a σ_1-neutral diagram, we can simply cut D_n along a vertical arc through the leftmost puncture, concentrate on the part of our cut D_n which contains punctures number $2, \ldots, n$, and proceed by induction to make the diagram σ_2-neutral, then σ_3-neutral etc., until it coincides with E.

Proof of proposition 4.1 Our argument was partly inspired by the unpublished thesis of Larue [8]. We denote the lower half of D_n by $D_n^\vee$ and the upper half by $D_n^\wedge$. We say a that the ith puncture of D_n is *being pushed up* by the diagram D if one of the path components of $D \cap D_n^\vee$ consists of an arc with one endpoint in e'_{i-1} and the other in e'_i.

We first claim that some puncture other than the first (leftmost) one is being pushed up by D.

To see this, let us denote by v the vertical arc in D_n through the first puncture - it is cut into two halves by the first puncture. Let us write $D_n^>$ for the closure of the component of $D_n \backslash v$ which contains punctures number $2, \ldots, n$ (that is, $D_n^>$ equals D_n with everything left of v removed). Since D is σ_1-positive, we have that the upper half of v has two more intersection points with D than the lower half. Therefore at least one of the components of $D_n^> \cap D$ is an arc both of whose endpoints lie in the upper half of v. Since this arc has to intersect the diagram E', we deduce that some component of $D \cap D_n^\vee$ has both endpoints in intervals e'_i, e'_j with $i, j \neq 0$. Now that we have found at least one "U-shaped" arc in $D_n^\vee \cap D_n^>$, we can consider an innermost one. Since all punctures of D_n must lie in different path components of $D_n \backslash D$, all innermost U-shaped arcs must have endpoints in a pair of *adjacent* arcs e'_{i-1}, e'_i of E' with $i-1 \neq 0$. This completes the proof of the first claim.

It is this ith puncture that is going to perform the semicircular move: our second claim is that there exists an oriented arc g in $D_n \backslash D$ which starts on the ith puncture, lies entirely in the upper half of D_n, and terminates on one of the horizontal arcs e'_j with $j \neq i, i-1$.

To convince ourselves of the second claim, let us look at the two arcs of $D \cap D_n^\wedge$ which have boundary points in common with our innermost U-shaped arc in $D \cap D_n^\vee$. Not both of these arcs can end on ∂D_n, so at least one of them, which we shall call g', ends on E'. Then we can define g to be the arc in $D_n \backslash D$ which starts on the ith puncture, and stays parallel and close to g', until it terminates on the same arc of E' as g'. This construction proves the second claim.

We now define our semicircular braid γ to be the slide of the ith puncture along the arc g.

Next, we prove that this move decreases the complexity of the curve diagram. Let D' be the diagram obtained from the diagram D by first applying the homeomrphism γ and then reducing the resulting diagram with respect to the horizontal line E'. In comparison to D, in this new diagram D' the arc e'_j has been divided into two arcs by the formerly ith puncture, whereas the arcs e'_{i-1} and e'_i have merged. Moreover, if the diagram D contained k U-shaped arcs in $D_n^\vee$ with one endpoint in e'_{i-1} and the other in e'_i, then the complexity of D' is $2k$ lower than the complexity of D: our semicircular move γ eliminated these $2k$ intersections of D with E', and did not create or eliminate any others.

The only thing left to be seen is that the diagram D' is still σ_1-positive or σ_1-neutral, but not σ_1-negative. But a σ_1-negative diagram could only be obtained if the arc g ended in the interval e'_0 (i.e. if the last letter of γ was σ_1^{-1}). The proof is now exactly analogue to the proof of Proposition 4.3, Claim 2 in [5] which states that curve diagram obtained by "sliding a puncture along a useful arc" is not σ_1-negative. $\square$

5 The special case of 3-string braids

The reader may have wondered why no experimental results for 3-string braids were presented in section 3. The reason is that we can *prove* that the best possible results hold in this case:

Theorem 5.1 *For three-string braids, the σ_1-consistent version of our algorithm outputs only braid words of minimal length.*

Thus for $n = 3$, conjecture 3.1 holds with $c(n) = 1$. In other words, if we input any braid word w' in the letters σ_i^ϵ with $i \in \{1, 2\}$ and $\epsilon = \pm 1$, then the output word w satisfies $l(w) \leqslant l(w')$. Unfortunately, the proof below is very complicated, and it would be nice to have a more elegant proof. We remark that the analogue of theorem 5.1 for the standard algorithm also holds. We shall not prove this here, because the argument becomes even more complicated.

In what follows, we shall always denote by w a braid word output by the algorithm. The word w is given as a product of semicircular braids, and we shall indicate this decomposition, when necessary, by separating the semicircular factors by a dot " $\cdot$ ". Thus $\sigma_2\sigma_1 \cdot \sigma_2^{-1}$ denotes the product of the two semicircular braids $\sigma_2\sigma_1$ and σ_2^{-1}, and the word $\sigma_2\sigma_1^{-1} \cdot \sigma_2$ (without a dot between σ_2 and σ_1^{-1}) is meaningless. Finally, we shall denote by an asterisk $*$ any symbol among $\{1, -1, 2, -2, \cdot\}$, or the absence of symbols.

Lemma 5.2 *Let w be the output braid word of the σ_1-consistent algorithm. Then*

(i) w does not contain any subword of the form $* \sigma_i \cdot \sigma_i^{-1} *$ with $i \in \{1, 2\}$.

(ii) The only place in w where the subword $\cdot \sigma_2^{-1} \cdot \sigma_1\sigma_2$ can occur is near the end of w, in the context of a terminal subword $\cdot \sigma_2^{-1} \cdot \sigma_1\sigma_2 (\cdot \sigma_2)^k$ with $k \in \mathbb{N} \cup \{0\}$.

(iii) w does not contain any subword of the form $\cdot \sigma_1 \cdot \sigma_2 \cdot$ or $\cdot \sigma_2 \cdot \sigma_1 \cdot$.

Items (i),(ii), and (iii) are also true if every single letter in their statement is replaced by its inverse.

(iv) Subwords of the form $\cdot \sigma_2\sigma_1 \cdot \sigma_2\sigma_1 \cdot \sigma_2^{-1} *$ cannot occur in w.

(v) Subwords of the form $\cdot \sigma_2\sigma_1 \cdot \sigma_2\sigma_1 \cdot \sigma_1 *$ cannot occur in w.

(vi) Subwords of the form $\cdot \sigma_1 \cdot \sigma_2\sigma_1 \cdot$ and $\cdot \sigma_2\sigma_1 \cdot \sigma_2 \cdot$ can occur in w, but only in the context of a terminal *subword which is positive (i.e. has not a single letter σ_1^{-1} or σ_2^{-1}.) Moreover, they cannot be immediately preceded by the subword $\cdot \sigma_2^{-1}$.*

Items (iv)–(vi) are also true if every single letter in their statement is replaced by its inverse or if the rôles of σ_1 and σ_2 are interchanged.

Items (iii)–(vi) state essentially that the subwords $*\sigma_1 \cdot \sigma_2 *$ and $*\sigma_2 \cdot \sigma_1 *$ cannot occur, except in a few very special cases near the end of the word.

Proof of lemma 5.2 For the proof of (i) we suppose, for a contradiction, that w contains a subword consisting of two semicircular braids: the first subword consists of or terminates in σ_i^{ϵ}, the second one consists of or begins with $\sigma_i^{-\epsilon}$. For definiteness we suppose that we have the subword $*\sigma_2 \cdot \sigma_2^{-1} *$, the other cases being exactly analogue. The first factor $\sigma_1\sigma_2$ or σ_2 can be realized by a semicircular movement in the upper half of D_n. We consider the curve diagram D that is obtained after the action of the first factor. In this diagram, the third puncture is either being pulled down, or it may be completely to the right of D (in this case, D can be untangled by a braid which contains only the letters $\sigma_1^{\pm 1}$), but it is certainly not being pushed up. But in this situation, acting on D by σ_2^{-1} or $\sigma_2^{-1}\sigma_1^{-1}$ cannot reduce the complexity. (It is worth remarking that the statement (i) is wrong for braids with more than three strings – there the first factor might be followed by a semicircular movement of the second puncture in the lower half of D_n to a point further to the right.)

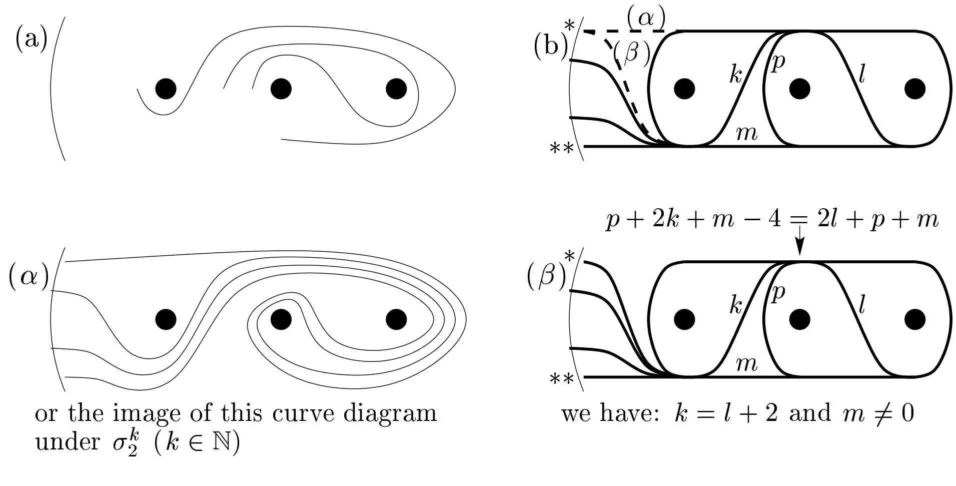

Figure 3: Which curve diagrams might give rise to the subword $\sigma_2^{-1} \cdot \sigma_1\sigma_2$?

The proof of (ii) is more complicated. Suppose that D is a curve diagram which the σ_1-consistent algorithm wants to untangle by first performing a σ_2^{-1}, and then a $\sigma_1\sigma_2$. Then the curve diagram D must be negative (because it can be untangled by a σ_1-positive braid word), and it must contain arcs as indicated in figure 3(a) (note that there may be a number of parallel copies of these arcs). We deduce that D is carried by one of two possible train tracks, which are both indicated in figure 3(b) – the two possibilities are labelled (α) and (β). The labels k, l, m, p are, of course, integer variables which indicate how often the corresponding piece of track is traversed by the curve diagram.

Case (α) is quite easy to eliminate: the essential observation is that in case (α) we must have $m = 0$. To see this, consider the arc of the curve diagram whose extremities are the points labelled $*$ and $**$. This arc, together with the segment of ∂D_n shown, must bound a disk which contains exactly two of the punctures. If we had $m \neq 0$, however, then the arc from $*$ to $**$ would intersect E' only once, namely in e_3', and the disk would contain all three punctures, which is absurd. Thus, in case (α) we must have $m = 0$.

It is now an exercice to show that the only curve diagrams carried by the train track of figure 3(b)(α) with $m = 0$ and containing arcs as indicated in figure 3(a) belong to the family of curve diagrams given in figure 3(α). However, the algorithm chooses to untangle these diagrams with the movement $(\sigma_2^{-1} \cdot)^{2+k} \sigma_1$, not with $\sigma_2^{-1} \cdot \sigma_1 \sigma_2 ...$, as hypothesised. Thus the curve diagram D cannot be carried by the train track from figure 3(b)(α).

For case (β) we consider again the arc of the curve diagram with endpoints $*$ and $**$. This arc, together with the segment of ∂D_n shown, must bound a disk which contains two of the punctures. For this to happen, however, we need that starting from the point $**$ and following the arc of the diagram we traverse the track labelled m, not the track labelled k. Therefore in case (β) we must have $m \neq 0$.

Moreover, the calculation shown in the figure proves that the curve diagram D traverses the track labelled k twice more than the track labelled l. This implies that the third puncture is being pushed up by l U-shaped arcs below it (and similarly there are l caps above the second puncture), but there are at least $l+1$ U-shaped arcs below the first puncture.

Thus the standard algorithm would choose to perform the movement $\sigma_1 \sigma_2$, and the σ_1-consistent algorithm only avoids doing so because the movement $\sigma_1 \sigma_2$ is illegal (it would render the diagram positive). However, we know by hypothesis that after applying σ_2^{-1}, the move $\sigma_1 \sigma_2$ *is* legal.

This means that the curve diagram is of one of two shapes shown in figure 4. We observe that the diagram in figure 4(b) cannot be extended to a reduced curve diagram – indeed, the reader will find that any attempt to extend the arcs labelled by a question-mark will yield arcs of infinite length. Therefore we

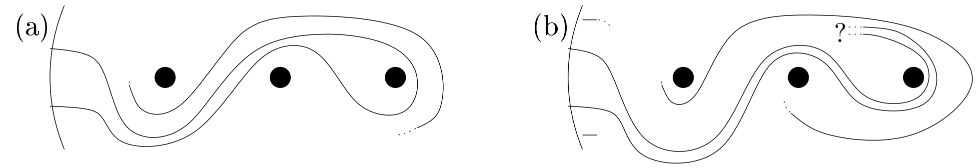

Figure 4: Applying σ_2^{-1} to these diagrams legalizes the move $\sigma_1\sigma_2$

must be in case (a). It follows that after applying the braid $\sigma_2^{-1}\cdot\sigma_1\sigma_2$ to the diagram D we are left with a curve diagram that can be untangled by σ_2^k for some $k\in\mathbb{N}\cup\{0\}$. This completes the proof of (ii).

For the proof of (iii) we remark that the algorithm prefers the word $\cdot\sigma_1^\epsilon\sigma_2^\epsilon\cdot$ to the word $\cdot\sigma_1^\epsilon\cdot\sigma_2^\epsilon\cdot$, and similarly the word $\cdot\sigma_2^\epsilon\sigma_1^\epsilon\cdot$ to the word $\cdot\sigma_2^\epsilon\cdot\sigma_1^\epsilon\cdot$ for $\epsilon=\pm 1$.

For the proof of (iv) and (v) we consider a diagram D such that the untangling of D begins, according to our algorithm, with the movements $\sigma_2\sigma_1\cdot\sigma_2\sigma_1$. We observe that the diagram D must contain curve segements as shown in figure 5(a) (solid lines). The action of $\sigma_2\sigma_1\cdot\sigma_2\sigma_1$ on this diagram is indicated by the arrows.

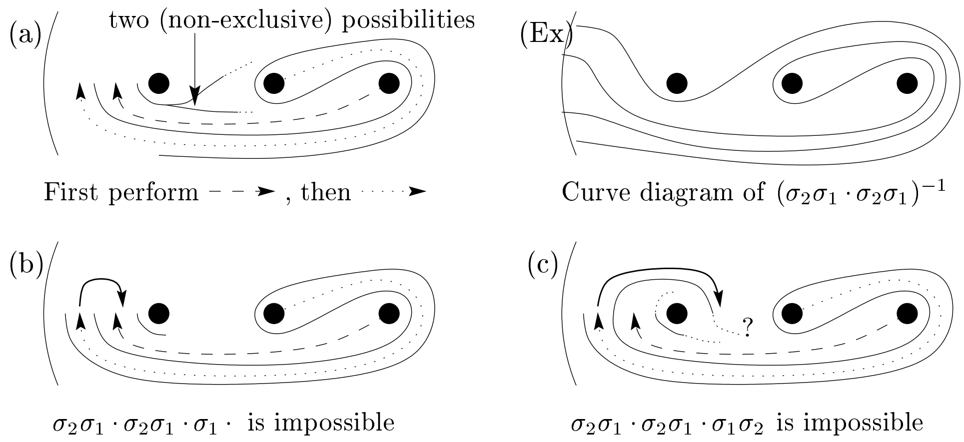

Figure 5: Which curve diagrams are relaxed by $\sigma_2\sigma_1\cdot\sigma_2\sigma_1\cdot *$?

Let us first prove (iv), i.e. that the movement $\sigma_2\sigma_1\cdot\sigma_2\sigma_1\cdot\sigma_2^{-1}*$ cannot occur. After the action of $\sigma_2\sigma_1\cdot\sigma_2\sigma_1$ on D, the puncture which started out in the leftmost position is now the rightmost one. In order to get the move σ_2^{-1} or $\sigma_2^{-1}\sigma_1^{-1}$ next, we would need that in the diagram $\sigma_2\sigma_1\cdot\sigma_2\sigma_1(D)$ the rightmost puncture is being pushed up by a U-shaped arc below it. However, one can see that applying $\sigma_2\sigma_1 \cdot \sigma_2\sigma_1$ to the diagram in figure 5(a) can never yield such a diagram.

For the proof of (v), we distinguish two cases. First, the sequence of moves $\sigma_2\sigma_1 \cdot \sigma_2\sigma_1 \cdot \sigma_1 \cdot$ is impossible, because the fat arrow in figure 5(b) must intersect the curve diagram D. Regarding the second case, if we suppose that the curve diagram D gives rise to the moves $\sigma_2\sigma_1 \cdot \sigma_2\sigma_1 \cdot \sigma_1\sigma_2$, then we can add information to figure 5(a): in this case, the diagram D must have contained the arcs shown in figure 5(c). However, this diagram cannot be extended to a reduced curve diagram which is disjoint from the dashed arrow: indeed, any attempt to extend the arc labelled by a question-mark yields an arc of infinite length. This completes the proof of (v).

Finally, we turn to the proof of (vi). Again, we suppose that the algorithm chooses to untangle the curve diagram D by a sequence of semicircular moves that starts with $\cdot\sigma_1 \cdot \sigma_2\sigma_1 \cdot$ or with $\cdot\sigma_2\sigma_1 \cdot \sigma_2 \cdot$. We observe that in both cases the curve diagram must be carried by one of the two train tracks (α or β) both shown in figure 6(a), where the tracks labelled k and l must each be traversed by at least one curve segment of D. However, as explained in the figure, case (α) is completely impossible, and case (β) is possible only if each of the two pieces of track labelled k and l is traversed by exactly one segment of D.

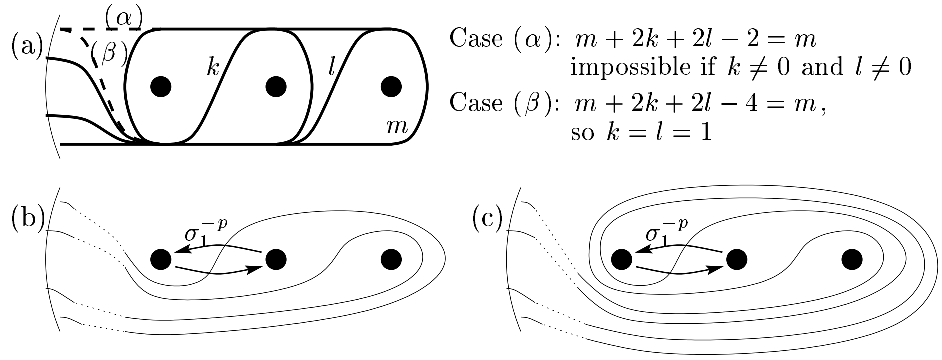

Figure 6: Case (vi) of Lemma 5.2

We leave it as an exercice to show that the only curve diagrams fitting these restrictions are the two shown in figure 6(b) and (c), and all their images under the action of $\sigma_1^{-p}\Delta^{-2q}$, where $p, q \in \mathbb{N} \cup \{0\}$. In other words, our curve diagram must in fact be the curve diagram of a braid $\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_1^{-p}\Delta^{-2q}$ or $\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_1^{-p}\Delta^{-2q}$, where $p, q \in \mathbb{N} \cup \{0\}$. Now we observe that any untangling of any of these curve diagrams that our algorithm may find has only positive letters. Moreover, if we act by σ_2 on any of these curve diagrams, then the relaxation (according to our algorithm) of the resulting diagram starts with $\sigma_1\sigma_2$, *not* with σ_2^{-1}. This completes the proof of (vi). $\square$

Proof of theorem 5.1 Let us denote by $\widetilde{w}$ the braid word obtained from an output braid word w by removing all the dot-symbols. Then lemma 5.2 implies that $\widetilde{w}$ cannot contain any subword of the form $\sigma_2(\sigma_1\sigma_2\sigma_2\sigma_1)^k\sigma_2^{-1}$, or $\sigma_2^{-1}(\sigma_1\sigma_2\sigma_2\sigma_1)^k\sigma_2$, or $\sigma_1(\sigma_2\sigma_1\sigma_1\sigma_2)^k\sigma_2\sigma_1\sigma_2^{-1}$, or $\sigma_2^{-1}(\sigma_1\sigma_2\sigma_2\sigma_1)^k\sigma_1\sigma_2\sigma_1$, $(k \in \mathbb{N}\cup\{0\})$, or any of the images of such braids under one of the automorphisms of B_3 $(\sigma_1 \to \sigma_2, \sigma_2 \to \sigma_1)$, or $(\sigma_1 \to \sigma_1^{-1}, \sigma_2 \to \sigma_2^{-1})$, or $(\sigma_1 \to \sigma_2^{-1}, \sigma_2 \to \sigma_1^{-1})$.

That is, the word $\widetilde{w}$ admits no obvious simplifications. It is well-known that 3-string braids without this type of obvious shortenings are in fact of minimal length. This completes the proof of theorem 5.1. $\square$

Appendix: Coding of curve diagrams

If we cut the disk D_n along n vertical lines through the puncture points, we obtain $n+1$ vertical "bands". The connected components of the intersection of a curve diagram with every such band come in six different types, which are in an obvious way described by the symbols $\supset, \subset, \overline{\cdot\cdot}, \underline{\cdot\cdot}, /$ and $\backslash$. Thus curve diagrams in D_n are coded by $6(n+1)$ integer numbers $d^i_{\supset}, d^i_{\subset}, d^i_{\overline{\cdot\cdot}}, d^i_{\underline{\cdot\cdot}}, d^i_{/}$, and $d^i_{\backslash}$ (with $i = 0, \dots n$). For instance, in figure 2(b), we have $d^0_{/} = 2, d^0_{\overline{\cdot\cdot}} = 2, d^1_{\backslash} = 2, d^1_{\overline{\cdot\cdot}} = 2, d^2_{\overline{\cdot\cdot}} = 1, d^2_{\supset} = 1, d^2_{\underline{\cdot\cdot}} = 1, d^3_{\supset} = 1$, and all other coeffiecients equal zero..

Using this coding, reduced curve diagrams of arbitrary braids can be calculated by the rule that applying σ_i to a curve diagram D changes the diagram as follows. Firstly, all $d^j_{\times}$ (where $\times$ can be any symbol) with $j \notin \{i-1, i, i+1\}$ are unchanged. Apart from that, the following rules apply simultaneously:

$$d^{i-1}_{\supset} \leftarrow d^{i-1}_{\supset}$$

$$d^{i-1}_{\subset} \leftarrow d^i_{\subset} + d^i_{\backslash} + \max(0, d^i_{\underline{\cdot\cdot}} - d^{i-1}_{\underline{\cdot\cdot}} - d^{i-1}_{\backslash})$$

$$d^{i-1}_{\overline{\cdot\cdot}} \leftarrow d^{i-1}_{\overline{\cdot\cdot}} + d^{i-1}_{\backslash} - \min(d^{i-1}_{\subset}, \max(0, d^i_{\underline{\cdot\cdot}} - d^{i-1}_{\underline{\cdot\cdot}}))$$

$$d^{i-1}_{\underline{\cdot\cdot}} \leftarrow \min(d^{i-1}_{\underline{\cdot\cdot}}, d^i_{\underline{\cdot\cdot}})$$

$$d^{i-1}_{/} \leftarrow d^{i-1}_{/} + d^{i-1}_{\underline{\cdot\cdot}} - \min(d^{i-1}_{\underline{\cdot\cdot}}, d^i_{\underline{\cdot\cdot}})$$

$$d^{i-1}_{\backslash} \leftarrow \min(d^{i-1}_{\subset}, \max(0, d^i_{\underline{\cdot\cdot}} - d^{i-1}_{\underline{\cdot\cdot}}))$$

$$d^i_{\supset} \leftarrow \min(\min(d^i_{/}, d^{i+1}_{\supset}), \max(0, d^{i-1}_{\subset} + d^{i-1}_{\underline{\cdot\cdot}} - d^i_{\underline{\cdot\cdot}}))$$

$$d^i_{\subset} \leftarrow \min(\min(d^{i-1}_{\subset}, d^i_{/}), \max(0, d^{i+1}_{\overline{\cdot\cdot}} + d^{i+1}_{\backslash} - d^i_{\overline{\cdot\cdot}}))$$

$$d^i_{\overline{\cdot\cdot}} \leftarrow d^i_{\supset} d^i_{\backslash} + d^i_{\overline{\cdot\cdot}} - \min(d^{i-1}_{\subset}, d^i_{/}, d^i_{\underline{\cdot\cdot}})$$

$$d^i_{\underline{\cdot\cdot}} \leftarrow d^i_{\subset} + d^i_{\backslash} + d^i_{\underline{\cdot\cdot}} - \min(d^{i+1}_{\supset}, d^i_{/}, d^i_{\underline{\cdot\cdot}})$$

$$d^i_{/} \leftarrow \max(0, \min(d^i_{/}, d^{i-1}_{\subset}) + \min(d^i_{/}, d^{i+1}_{\supset}) - d^i_{/})$$

$$d^i_{\backslash} \leftarrow d^i_{\supset} + d^i_{\subset} + d^i_{\backslash} + \max(0, d^i_{/} - d^{i-1}_{\subset} - d^{i+1}_{\supset})$$

$$d^{i+1} \leftarrow d^i_{\supset} + d^i_{\backslash} + \max(0, d^i_{\overline{\cdot\cdot}} - d^{i+1}_{\overline{\cdot\cdot}} - d^{i+1}_{\backslash}$$

$$d^{i+1}_{\subset} \leftarrow d^{i+1}_{\subset}$$

$$d^{i+1}_{\overline{\cdot\cdot}} \leftarrow \min(d^{i+1}_{\overline{\cdot\cdot}}, d^i_{\overline{\cdot\cdot}})$$

$$d^{i+1}_{\underline{\cdot\cdot}} \leftarrow d^{i+1}_{\underline{\cdot\cdot}} + d^{i+1}_{\backslash} - \min(d^{i+1}_{\backslash}, \max(0, d^i_{\overline{\cdot\cdot}} - d^{i+1}_{\overline{\cdot\cdot}}))$$

$$d^{i+1}_{/} \leftarrow d^{i+1}_{/} + d^{i+1}_{\overline{\cdot\cdot}} - \min(d^{i+1}_{\overline{\cdot\cdot}}, d^i_{\overline{\cdot\cdot}})$$

$$d^{i+1}_{\backslash} \leftarrow \min(d^{i+1}_{\backslash}, \max(0, d^i_{\overline{\cdot\cdot}} - d^{i+1}_{\overline{\cdot\cdot}}))$$

Acknowledgement I thank Lee Mosher and Juan Gonzalez-Meneses for helpful discussions, and Heather Jenkins and Sandy Rutherford of the Pacific institute for the mathematical sciences (PIMS) at UBC Vancouver for letting me use their most powerful computers.

References

[1] E. Artin, Theory of braids, Ann. of Math. (2) 48 (1947), 101–126

[2] P. Dehornoy, A fast method for comparing braids. Adv. Math. 125 (1997), no. 2, 200–235

[3] P. Dehornoy, Braids and self-distributivity. Progress in Mathematics, 192. Birkhäuser Verlag, Basel, 2000. xx+623 pp. ISBN: 3-7643-6343-6

[4] D. B. A. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, W. Thurston, Word processing in groups, Jones and Bartlett Publishers, Boston, MA, 1992

[5] R. Fenn, M. T. Greene, D. Rolfsen, C. Rourke, B. Wiest, Ordering the braid groups, Pacific J. Math. 191 (1999), no. 1, 49–74.

[6] N. Franco, J. Gonzalez-Meneses, Conjugacy problem for braid groups and Garside groups, arXiv preprint math.GT/0112310

[7] F. A. Garside, The braid group and other groups, Quart. J. Math. Oxford Ser. (2) 20 1969 235–254

[8] D. Larue, PhD thesis, Boulder, 1994, unpublished

[9] L. Mosher, Mapping class groups are automatic, Ann. of Math. (2) 142 (1995), no. 2, 303–384.

[10] B. Wiest, computer programs used for the computations, and further data, available at `http://www.maths.univ-rennes1.fr/~bertw`