A program to search for homotopy 3–spheres

Michael Greene
Colin Rourke

Mathematics Institute
University of Warwick
Coventry CV4 7AL, UK

Email: `mtg@maths.warwick.ac.uk` and `cpr@maths.warwick.ac.uk`

Abstract The mathematical background and some details are presented for an implementation in C of the Rêgo–Rourke algorithm [6].

AMS Classification 57M40; 57-04, 57N40

Keywords Rego-Rourke algorithm, C-program, homotopy 3-sphere

Dedicated to Rob Kirby on the occasion of his 60th birthday

The Rêgo–Rourke algorithm [6] generates a complete list of homotopy 3–spheres (with redundancy).

This note gives the mathematical background and some of the programming details for an implementation of this algorithm as a C program. The C program has been through three major revisions. Each revision has been concerned with sharpening the search to make it more likely that interesting examples are found. No interesting examples have been found so far and it remains unclear whether any can be found within the limits of current computer technology. For details of the results of the search to date, see theorem 5.1 and for full details of the present version of the program see [3].

The note is arranged as follows. Section 1 gives a brief summary of the Rêgo–Rourke algorithm and serves to fix the basic notation for the standard handlebody. Section 2 discusses some general preliminary points about the algorithm and sections 3 and 4 discuss properties of curves on the surface of a standard handlebody. Finally section 5 describes the programs and discusses results that may be expected using them.

History. The conception of the first version of the program (described in section 5) is joint work of Michael Greene and Bert Wiest during 1995. Michael Greene is responsible for the second version of the program and the result stated in theorem 5.1 is his (August 1996). The current version and the complete mathematical background presented here are joint work of Greene and Rourke.

1 Summary of the Rêgo–Rourke algorithm

The main purpose of this section is to fix notation. For full details on the material presented here see [6].

Let T be the standard unknotted solid handlebody of genus $g+h$ embedded in $\mathbb{R}^3$. We shall picture T (as illustrated in figure 1) as a ball with handles attached with rotational symmetry and we shall use the following notation for standard curves on ∂T. Curves $\{a_1, a_2, \ldots, a_{g+h}\}$ are longitudinal curves for the handles and bound discs in $\mathbb{R}^3 - T$. Curves $\{b_1, b_2, \ldots, b_{g+h}\}$ are meridional curves for the handles and bound discs in T. Curve c_i for $i = 1, \ldots, g+h$ is the band sum of b_i with b_{i+1} (indeces taken mod $g+h$), and again bounds bounds a disc in T.

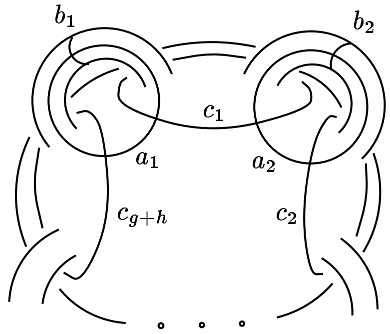

Figure 1: The standard solid handlebody

An *RR–system of genus g and co-genus h* comprises a complete system of curves $\{x_1, x_2, \ldots, x_g, y_1, y_2, \ldots, y_h\}$ on ∂T such that:

1) $\{x_1, x_2, \ldots, x_g\}$ bound a set of disjoint surfaces in $\mathbb{R}^3 - T$

2) y_i bounds a disc in T for $i = 1, \ldots, h$ (and hence $\{y_1, y_2, \ldots, y_h\}$ bound a set of disjoint discs in T).

If $\{x_1, x_2, \ldots, x_g, y_1, y_2, \ldots, y_h\}$ is an RR–system then define T' to be the handlebody of genus g obtained by slicing T along the discs bounded by the y–curves. Then the x–curves form a complete system on $\partial T'$ and we can form a

3–manifold M^3 by attaching 2-handles to T' along the x–curves and a 3–handle to the resulting S^2 boundary. There is a degree 1 map from $S^3 = \mathbb{R}^3 \cup \infty$ to M^3 which is the identity on T', maps the surfaces which the x–curves bound to the cores of the attached 2–handles and maps the rest to the 3–handle. Thus M^3 is a homotopy 3–sphere.

Theorem 1.1 (Rêgo–Rourke [6]) *Every homotopy 3–sphere arises in this way.*

Now it is easy to test whether a given complete system $\{x_1, x_2, \ldots, x_g, y_1, y_2, \ldots, y_h\}$ is an RR–system:

Test 1.2 To decide if the x–curves bound a set of disjoint surfaces in $\mathbb{R}^3 - T$, read the word w_i in the symbols $\{x_1, x_1^{-1}, x_2, x_2^{-1}, \ldots, x_g, x_g^{-1}\}$ given by the intersections of a_i with the x–curves. Then the x–curves bound disjoint surfaces if and only if the word w_i cancels to the trivial word for each $i = 1, \ldots, g+h$.

Test 1.3 To decide if y_i bounds a disc in T read the word z_i in the symbols $\{b_1, b_1^{-1}, b_2,\ b_2^{-1}, \ldots, b_{g+h}, b_{g+h}^{-1}\}$ given by the intersections of y_i with the b–curves. Then y_i bounds a disc if and only if the word z_i cancels to the trivial word.

Thus to list RR–systems of genus g and co-genus h we list all complete systems on ∂T and test each in turn. Theorem 1.1 implies that by doing this (in some order of complication) for increasing (g, h) we list all homotopy 3-spheres.

2 Some general points

Since the Poincaré conjecture is known to be true for 3–manifolds of genus 2 [8], we must have $g \geq 3$ to find a counterexample.

If there are no y–curves or, if the result of slicing T along all the y–curves is an unknotted handlebody, then the degree 1 map constructed in section 1 can be replaced by a homeomorphism (since a homeomorphism from the boundary of a handlebody to the boundary of another handlebody which extends to a degree 1 map between the handlebodies also extends to a homeomorphism):

Lemma 2.1 *Let $f\colon T \to T$ be a degree 1 map which restricts to a homeomorphism $f'\colon \partial T \to \partial T$ then f' extends to homeomorphism $T \to T$.*

Proof Let B_i be a complete system of meridional discs for T then for each i, $f(\partial B_i)$ is a curve in ∂T which is null-homotopic in T and hence bounds a disc D_i say by Dehn's lemma. Further a simple innermost curve argument makes the discs D_i disjoint. Slicing along the D_i or the B_i produces a 3–ball and the required homeomorphism is now constructed by mapping D_i to B_i for each i and extending to the 3–balls. $\square$

Further if the result of slicing along any particular y–curve is unknotted then the co-genus can be reduced. If an x–curve bounds a disc and attaching the (thick) disc to T fails to knot the outside of T then the genus can be reduced. Finally if an x–curve α is *transversally trivial* (ie if there is a curve β which meets α transversally in just one point such that β bounds a disc inside T) then the resulting homotopy sphere has a cancelling pair of handles:

Lemma 2.2 *Suppose that an x–curve in an RR–system is transversally trivial, then the resulting homotopy 3–sphere M has a pair of cancelling handles.*

Proof Let the x–curve α meet the curve β on ∂T transversally in one point where β bounds a disc D inside T. By a simple outermost arc argument we can assume that β misses the y–curves and by an innermost curve argument that D misses the discs bounded by the y–curves. Then D can be taken to be the co-core of a 1–handle of M which cancels the 2–handle attached along β. $\square$

Thus if we are working at the minimal genus or co-genus, or if we do not wish to construct lower genus/co-genus systems then we can discard x or y–curves which bound discs and fail to knot, and x–curves which are transversally trivial. If an x–curve bounds a disc then the disc fails to knot the outside if and only if the curve is transversally trivial (see lemmas 3.1, 3.2 and 3.3) thus transverse triviality only matters for curves which bound surfaces but not discs.

Further is all the x–curves bound discs then T' must be unknotted and again the degree 1 map constructed in section 1 is a homeomorphism. Thus at least one of the x–curves must bound a surface and not a disc.

Note that all the curves we use must be non-separating curves on ∂T and a non-separating curve cannot bound surfaces both inside and outside T.

To summarise the above discussion we have the following classification of (non-separating) curves:

(XU) bounds a disc outside T which fails to knot

(XK) bounds a disc outside T which knots

(XT) bounds a surface outside T, but not a disc, and is transversally trivial

(XS) bounds a surface outside T, but not a disc, and is not transversally trivial

(YU) bounds a disc inside T which fails to knot

(YK) bounds a disc inside T which knots

(Z) none of the above.

And we seek a complete system $\{x_1, x_2, \ldots, x_g, y_1, y_2, \ \ldots, y_h\}$ such that the x–curves are of class XK or XS (with at least one in class XS) and the y–curves are of class YK.

Further the x–curves must bound *disjoint* surfaces outside T (which is not a consequence of bounding surfaces individually).

Define an RR–system $\{x_1, x_2, \ldots, x_g, y_1, y_2, \ldots, y_h\}$ to be *interesting* if the x–curves are of class XK or XS (with at least one in class XS) and the y–curves are of class YK. *We seek interesting RR–systems.*

3 Classifying curves

A curve on ∂T determines an element of $\pi_1(\partial T)$ up to conjugation. Now $\pi_1(\partial T)$ is a one-relator group with $2n$ generators which are based versions of the curves $\{a_i, b_i \mid 1 \leq i \leq n\}$ (see figure 7) and for which we use the same letters. Thus a curve determines an *AB string*, ie a word in the letters $\{a_i, b_i, a_i^{-1}, b_i^{-1} \mid 1 \leq i \leq n\}$, up to conjugation and the equivalence given by the relator.

Given a curve in the form of an AB string, it is simple to describe the effect of a particular Dehn twist (for detail see section 5) and thus we can readily convert a curve, described as a Dehn string applied to a standard curve, to an AB string,

Once a curve α is given as an AB string Q it can be quickly classified as in one of the classes above. Use the notation Q_A, Q_B for the A string (respectively B string) obtained from Q by deleting all the b's (respectively a's). Write $V(Q)$ (the *AB vector*) for the abelianised AB string and similarly $V(Q_A)$ and $V(Q_B)$.

The first check is whether α bounds a surface either inside or outside. This is just a homology check: α bounds a surface inside if and only if $V(Q_A)$ is the zero vector (since the a's generate $H_1(T)$ freely). Similarly α bounds a surface outside if and only if $V(Q_B)$ is the zero vector. If it fails this test it is in class Z. If α bounds a surface inside then it bounds a disc inside if and only if Q_B cancels to the trivial word, since the a's generate $\pi_1(T)$ freely. If it fails these tests it is in class Z otherwise it is in class YU or YK. Similarly we can distingush classes XT,XS from classes XU,XK.

Next we explain how to decide between classes XU and XK (and similarly between YU and YK); this is done as follows.

Notice a solid handlebody is unknotted in S^3 iff the fundamental group of the complement is free:

Lemma 3.1 *Suppose that $h\colon T \to S^3$ is an embedding and that $\pi_1(S^3 - h(T))$ is free then $h(T)$ is unknotted, ie there is a homeomorphism of S^3 carrying T to $h(T)$.*

Proof Note that $\pi_1(\partial T)$ is a surface group and not free. It follows from the Nielsen–Schreier theorem that $\pi_1(h(\partial T))$ does not inject into $\pi_1(S^3 - h(T))$ and hence by the loop theorem there is a properly embedded disc D in $S^3 - h(T)$ with ∂D essential in $h(\partial T)$. Now let $Q = T \cup N(D)$ then ∂Q is again a surface, which is either S^2 or, again by the Nielsen–Schreier theorem, there is a loop in ∂Q which bounds a singular disc D' in $S^3 - h(T)$. By a simple outermost arc argument we can assume that $\partial D'$ misses ∂D and by an innermost curve argument that D' misses D; then by the proof of the loop theorem we can assume that D' is embedded and disjoint from D. Continuing in this way we construct a system of curves on $h(\partial T)$ bounding disjoint discs in $S^3 - h(T)$ such that the result of surgering $h(\partial T)$ along all these discs is a collection of S^2's. The S^2's bound 3-balls outside $h(T)$ by the Schönflies theorem and it can now be seen that $S^3 - h(T)$ is a handlebody. The existence of the required homeomorphism follows from Waldhausen [9]. □

Next notice that the fundamental group of $T \cup D$, where D is a disc outside T with boundary α, is a one-relator group with relation Q_A. But a one-relator group is free if and only if the relation is either trivial or primitive and there is a very simple algorithm to decide if a word is primitive given by considering Whitehead automorphisms:

Let F be a free group on generators $\{b_1, b_2, \ldots, b_k\}$. Define a *Whitehead automorphism* of F to be given by choosing a fixed $i \leq k$ and $\varepsilon = \pm 1$, and for each $j \neq i, j \leq k$ making one of the following choices

$$\begin{aligned} b_j &\longmapsto b_j \\ b_j &\longmapsto b_j b_i^{\varepsilon} \\ b_j &\longmapsto b_i^{-\varepsilon} b_j \\ b_j &\longmapsto b_i^{-\varepsilon} b_j b_i^{\varepsilon}. \end{aligned}$$

Now let $w \in F$ (ie w is a word in $\{b_1, b_1^{-1}, b_2, b_2^{-1}, \ldots, b_k, b_k^{-1}\}$) and let $G = F/N$ where N is the normal closure of w in F. Thus G is the *one-relator group* with generators $\{b_1, b_2, \ldots, b_k\}$ and relation w.

We say that w is *primitive* if it can be included in a basis for F, ie if there is an automorphism of F which carries w to b_1,

Lemma 3.2 *Suppose that $w \neq 1$ then the following are equivalent:*

1) *the one-relator group G is free,*

2) *w is primitive,*

3) *there is a sequence of Whitehead automorphisms of F which take w to b_i or b_i^{-1} for some i and such that the lengths of successive images of w decrease.*

Proof The equivalence of (1) and (2) is proved in Lyndon and Schupp [5, II proposition 5.10] and the equivalence of (2) and (3) follows from the theorem of Whitehead and Rappaport that, if there is an automorphism of F which decreases the total length of a set of words, then there is a Whitehead automorphism which also decreses total length (see [5, page 31]). $\square$

Concluding this discussion we can deduce that a curve α which bounds a disc outside T (ie α is in class XU or XK) is in class XU iff the corresponding A string Q_A (ie the element of $\pi_1(T)$ determined by α) is primitive. Similarly a curve which bounds a disc inside T (ie class YU or YK) is in class YU iff Q_B is primitive.

Finally we have to distingush between classes XT and XS. This turns out to be exactly the same as distingushing between XU and XK. A curve α is transversally trivial if and only if Q_A is primitive:

Lemma 3.3 *A curve α on ∂T is transversally trivial iff Q_A is primitive, where Q is the corresponding AB string.*

Proof Attach a thick disc to T along α to form T' say. Then $\pi_1(T')$ is a one-relator group with relation Q_A.

If α is transversally trivial then T' has cancelling handles and is therefore a handlebody with free fundamental group and Q_A is primitive by lemma 3.2. Conversely if Q_A is primitive then by Zieschang [10] there is a homeomorphism of T carrying α to a standard longitudinal curve and therefore there is an automorpfism of F carrying Q_A to a generator, ie Q_A is primitive. □

4 Disjunction of curves and surfaces

The second version of the program needs some supporting lemmas on making curves and surfaces disjoint. These are collected here.

Disjunction of curves

The following material is in some sense "well-known" and proved by pulling curves tight with respect to each other. We shall follow the formulation given in [1]. Let α and β be two simple closed curves on a surface S. Define a D–disc (or bigon) to be a disc in S meeting $\alpha \cup \beta$ in its boundary and with boundary comprising an arc of α and an arc of β. Say that α and β are *reduced* with respect to each other if there are no D–discs. It is easy to see that we can isotope α say (keeping β fixed) by pushing across D–discs to make the two curves reduced with respect to each other. As each D–disc is removed, the number of intersections of α and β drops by two.

Apart from a couple of exceptional cases, see figures 4 and 5 below, curves which are reduced enjoy a unique reduced position:

Lemma 4.1 *Suppose that α is isotopic to α' which are both reduced with respect to β. Then, either both α and α' are disjoint from β, or there is an isotopy of S fixing β setwise and carrying α to α'.*

Proof We use an argument similar to the proof of [2, proposition 3.2]. The isotopy of α to α' can be realised by an isotopy of S fixing β setwise, apart from a number of critical stages which correspond to the introduction or deletion of D–discs. Write $\alpha \nearrow \alpha'$ or $\alpha' \searrow \alpha$ if α' is obtained from α by inserting a D–disc. We claim:

Claim (Diamond property) *Suppose that $\alpha \nearrow \alpha' \searrow \alpha''$ then either α and α' are both disjoint from β or $\alpha \searrow \alpha''' \nearrow \alpha''$ or there is an isotopy of S fixing β setwise and carrying α to α''.*

The lemma follows from the diamond property by a simple induction argument similar to the proof [2, corollary 3.3]: we consider the sequence of introductions and deletions of D–discs. The diamond property allows a local reduction in the height of the sequence at a local maximum. Eventually the sequence is flat and the lemma follows.

To prove the diamond property we observe that the two D–discs can meet in various ways as illustrated in figures 2 to 5. In figure 2 we have $\alpha \searrow \alpha''' \nearrow \alpha''$ where α''' is obtained by removing both discs. In figure 3 α is isotopic to α'' by an isotopy fixing β setwise and in figures 4 and 5 α and α' are disjoint from β. □

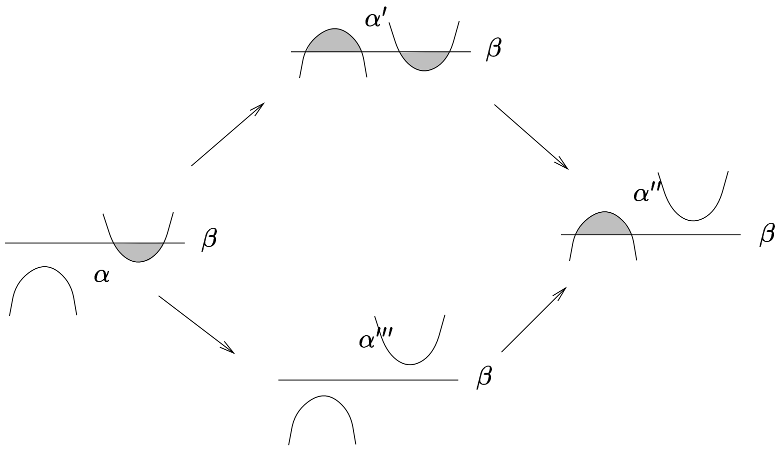

Figure 2: D–discs disjoint

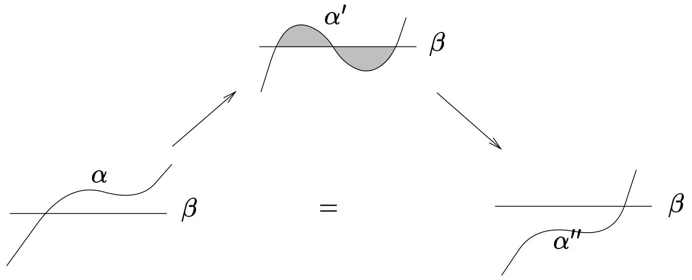

Figure 3: D–discs meet in one point

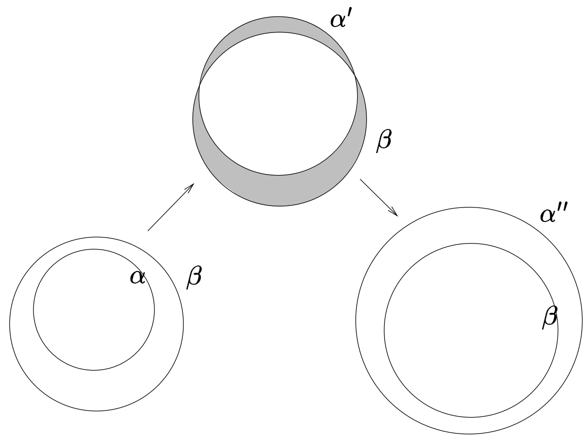

Figure 4: D–discs meet in two points

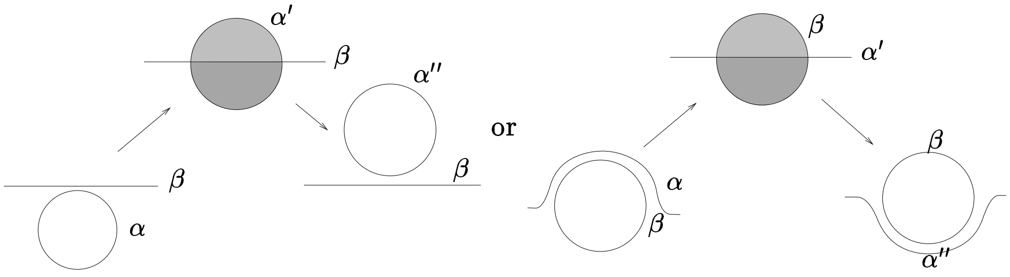

Figure 5: D–discs meet in a interval

We say that curves α and β are *disjunctive* if there is an isotopy of α making it disjoint from β. The lemma implies that non-disjoint disjunctive curves are not reduced. Thus we have:

Corollary 4.2 *Reduced disjunctive curves are disjoint.* □

We now prove that any set of curves which are pairwise disjunctive can be made simultaneously disjoint:

Lemma 4.3 *Suppose that curves $\{\alpha_i \mid 1 \leq i \leq t\}$ on a surface S are pairwise disjunctive. Then there are disjoint curves $\{\alpha'_i \mid 1 \leq i \leq t\}$ on S with α_i isotopic to α'_i for each i.*

Proof The proof is by induction t. For $t = 2$ the result is obvious. Suppose it is true for $t - 1$, then we may suppose that $\{\alpha_i \mid 1 \leq i \leq t-1\}$ are already disjoint. Consider the D–discs formed by α_t with respect to the other curves. Since the other curves are disjoint these must nest as illustrated in figure 6. Eliminate innermost first. By corollary 4.2 this process ends with α_t disjoint from $\{\alpha_i \mid 1 \leq i \leq t-1\}$. □

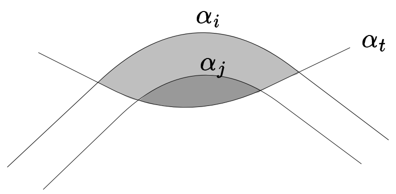

Figure 6: Nested D–discs

Disjunction of surfaces

We need to consider the analogue for surfaces of lemma 4.3. Unfortunately the direct analogue is false. Inspecting test 1.2 it is not hard to find examples of the following behaviour. Curves $\alpha, \alpha', \alpha''$ disjoint on ∂T bounding surfaces S, S', S'' in T which are pairwise disjunctive but such that there do not exist disjoint surfaces bounding $\alpha, \alpha', \alpha''$.

The best we can do is observe that if some of the surfaces are discs then they can be made disjoint from everything else:

Lemma 4.4 *Suppose that M is a 3–manifold with boundary and that there are disjoint curves $\{\alpha_i \mid 1 \leq i \leq t\}$ on ∂M. Suppose that $\{\alpha_i \mid 1 \leq i \leq t'\}$, where $t' < t$ bound disjoint surfaces in M and that α_i bounds a disc in M for each $t' < i \leq t$. Then there is a disjoint collection of surfaces and discs bounded by all the curves $\{\alpha_i \mid 1 \leq i \leq t\}$, which are discs for $t' < i \leq t$.*

Proof Consider the disc D bounded by $\alpha_{t'+1}$. Make D transverse to the surfaces bounding $\{\alpha_i \mid 1 \leq i \leq t'\}$. Choose an innermost curve of intersection with surface S say. Then by surgering S and gluing in copies of the little disc in D we remove one circle of intersection. Continue until D is disjoint from all the surfaces. Now treat $\alpha_{t'+2}$ in a similar way observing that surgery replaces D by a disc. Continue in this way. □

5 The programs

We start by describing common features. All versions of the program use Dehn twists as the main engine. By Lickorish [4] any homeomorphism of ∂T is a product of Dehn twists on the standard curves $\{a_i, b_i, c_i \mid 1 \le i \le n = g + h\}$ (in fact only $n-1$ of the c_i are needed, but it is convenient to retain symmetry). Thus we can generate all non-separating curves on ∂T by listing all *Dehn strings* (sequences of Dehn twists) and applying these sequences to one particular non-separating curve b_1. Similarly we can generate all complete systems by applying Dehn strings to one particular complete system, the $\{b_i\}$.

To avoid overuse of the letters a and b we shall use the notation L_i, M_i, N_i for the Dehn twist on the curves a_i, b_i, c_i respectively. Dehn twists commute if the curves do not intersect and this simplifies the list of Dehn strings that need to be considered.

The Dehn string generator

Dehn strings are generated in rough lexicographic (lex) order. Twists are ordered in the natural way: $\{L_1, M_1, N_1, L_2, M_2, N_2, \ldots\}$ and observe that twists at least three apart commute and that in some cases those one or two apart commute. Thus if we generate strings in rough lex order then we never need to add a twist with index much less than the last one added. There is thus a simple inductive procedure to generate all distinct Dehn strings. In practice the generator is set to stop at a given Dehn string length (say 10 twists).

The Dehn string to AB converter

This is the heart of the program. There are simple rules for applying a Dehn twist to an AB string which corresponds to applying the twist to the curve. Inspecting figure 7 the following rules can be readily checked.

(1) The effect of L_1 is to replace all occurences of b_1 in the string with $a_1 b_1$.

(2) M_1 : replace occurences of a_1 in the string with $a_1 b_1$.

(3) N_1 : replace b_1 by $b_1 q$, replace b_2 by $q b_2$ and replace a_2 by $q a_2 q^{-1}$ where $q = b_1^{-1} a_1^{-1} b_1 a_2$.

The rules for other twists are obtained by cyclic permutation. To convert a Dehn string to an AB string we now start with a particular standard curve (say b_1) and apply the Dehn twists in order. To find the abelianised AB string (AB vector) is easier. The analogous rules are:

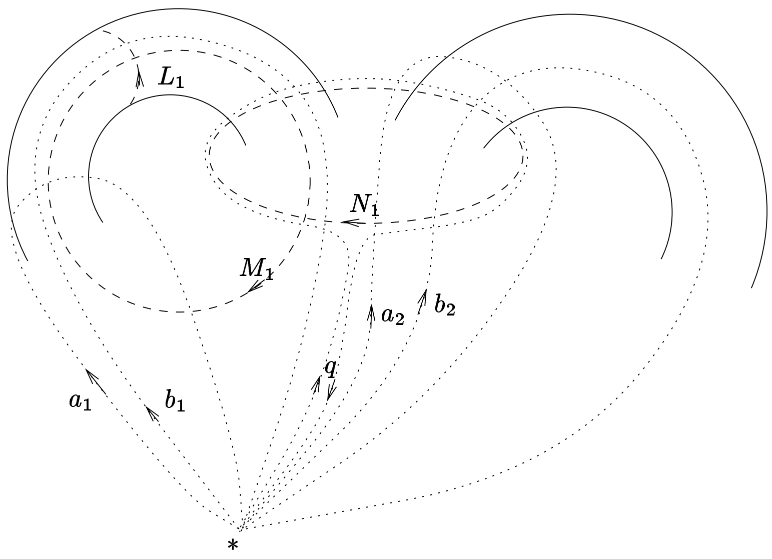

Figure 7: The effect of the Dehn twists

(1) L_1 : add the coefficient of a_1 to the coefficient of b_1, ie replace $v(b_1)$ by $v(b_1) + v(a_1)$, where $v(\cdot)$ means coefficient of.

(2) M_1 : replace $v(a_1)$ by $v(b_1) + v(a_1)$

(3) N_1 : replace $v(b_1)$ by $v(b_1) - v(a_1) + v(a_2)$ and $v(b_2)$ by $v(b_2) - v(a_1) + v(a_2)$.

This is far quicker than finding the AB string and then abelianising it.

The curve classifier

A curve is defined as a Dehn string applied to a standard curve. It is first converted to an AB vector and then the corresponding A and B vectors are read off. If both are non-zero then the curve is in class (Z) (see section 2). If one is non-zero then the curve is converted to full AB string, which now becomes its definition. Then further tests are applied as detailed in section 2 in order to classify the curve into classes (X $*$) or (Y $*$).

Version 1

This is now very easy to describe. Complete systems are generated by applying Dehn strings to the standard complete system $\{b_i\}$. Each resulting curve is classified as above. We check for complete systems where the x–curves are of class XK or XS and the y–curves are of class YK (see section 2) and finally apply the test for bounding disjoint surfaces (test 1.2) to the x–curves.

This is an idealised version of the program. In reality only very crude versions of all the tests were applied before the program was abandoned for reasons which we now outline.

In order to be successful, this program needs to find a single homeomorphism of the surface which turns each of the standard curves $\{b_i\}$ into a curve with special properties. For one curve this is unlikely, and, as the length of the Dehn string grows, becomes increasingly unlikely. Here is a quick calculation. The chance that a random curve of AB–length 50 (the rough length after about 10 Dehn twists) bounds a surface on a given side of T is about 1 in 25. The chance that each of the curves bounds a surface is thus far lower. We also need that at least one curve bounds a disc, which has a far smaller likelyhood.

What happens in practice with this program is that the same curves keep being retested. Each curve occurs many times using different Dehn strings from different b_i. But the work done in finding one suitable x or y–curve is lost if the whole system is unsuitable. Thus the program keeps finding suitable single curves and then discarding them; there is no way to keep hold of successful parts of the search and build on them. Thus although simple to describe and understand, this program is highly impractical.

Version 2

In the second version, the process of constructing complete systems is broken into two steps. In step 1 curves are generated individually and classified and then suitable curves are stored; in step 2 complete systems are constructed from the list of stored curves.

Although this sounds a lot more complicated than the first version of the program, the work involved in building complete systems from the stored curves is of the same order of magnitude as the work involved in classifying and storing the curves. But the final result is a far deeper search. In the first program we are seeking complete systems all of whose members are obtained from one set of standard curves by applying the *same* Dehn string. In the second program we seek complete systems whose members are obtained from a standard curve by applying *different* Dehn strings. This is a far larger set of possible systems.

Step 1 : Generating and classifying curves

The same Dehn string generator as in the first version is used, but is applied only to b_1. There is now a further simplification in the list of Dehn strings. Inspecting figure 1 it can be seen that the only Dehn twist which moves b_1 is L_1, so the Dehn string must start L_1. The next twist must be L_1, M_1, N_1 or N_{g+h}. In general we can define the *width* of the curve after the first so many Dehn twists which measures (roughly) the part of ∂T which the curve occupies. If a Dehn twist does not intersect this width, then it does not move the curve. Other than this simplification, the Dehn string generator as the same as in the first version. The curve classifier is also the same. Curves which are classified in classes (XK), (XS) or (YK) are stored, together with a Dehn string representation and their classification. They are stored in lex order using the AB string representation. This ensures that a particular suitable curve, which may be found several times by the Dehn string generator, is stored only once. At this point in the program it is necessary to know the genus of T. The AB string is not unique and we may be able to simplify a curve (as AB string) by using the (unique) relation in $\pi_1(\partial T)$. Even in simplest form, a curve may have more than one representative as an AB string. It is however easy to check if two minimal AB strings differ by application of the relation and the storage routine takes this into account. A further simplification is made by exploiting the circular symmetry of the situation. Each curve is lex minimised using circular permutation of the variables. This ensures that we do not store what is essentially the same surve n times.

Step 2 : Making complete systems of curves

A set of curves on ∂T is complete iff the curves are all disjoint and the set does not separate ∂T. By lemma 4.3, to check if the curves can be taken to be disjoint, we merely have to check that each pair is disjunctive. For this there is a simple routine. Before describing this routine, note that the other condition is trivial to check from the AB vectors corresponding to the curves. For homology reasons, a set of disjoint curves separates ∂T iff the AB vectors are linearly dependent. Now to check if two curves are disjunctive, where each is represented both as a Dehn string applied to b_1 and as an AB string, we apply the inverse of the first Dehn string to the second curve (as an AB string) and we then have a curve which we wish to test for disjuntion from b_1; this is trivial. The curve can be made disjoint from b_1 iff the AB string does not contain a_1. There is a small technicality here: there may be a way to use the relation to get rid of occurrences of a_1. But this is easy to test. If the (cyclic) string contains the substring $a_1b_1^{-1}a_1^{-1}$ or $a_1b_1a_1^{-1}$ then these two occurrences of a_1 can be removed by applying the relation. Otherwise no simplification is possible.

We now search for compatible n–tuples of curves of the required kind. For example if the search is taking place on the surface of genus 5 and we are searching for RR–systems of genus 3 and co-genus 2, then we search for three x and two y–curves. The search proceeds by starting with a particular curve, testing the cyclic permutes of other curves (further down the list) for disjucntion as detailed above and when two disjunctive curves are found, testing succeeding curves against both and so on.

This version of the program was run for several weeks on some of the fastest computers at Warwick. The full curve classifier was not implemented. Cruder tests sufficed for the curves found during this search. Further the test for bounding disjoint surfaces (test 1.2) was not implemented (or needed). The results were negative. An exhaustive search was made through Dehn strings of length ≤ 10 on a surface of genus 4 for RR–systems of genus 3 and co-genus 1. No interesting ones were found. As a result of this search we have the following theorem:

Theorem 5.1 (Greene) *There is no homotopy 3–sphere (other than S^3) which corresponds to an RR–system of genus 3 and co-genus 1, such that each curve is obtained by applying at most 10 Dehn twists to a standard curve.* □

The current version (version 3)

The current version now under test is an improved and sharpened version of version 2. The main changes are the following.

Complete curve classifier

The curve classifier is now in the form described in this note. The main new program fragment is the primitivity tester (of which crude versions appeared in version 2). This is available as a stand-alone C program [7]. It is about 100 times faster than than the group-theory package Magnus for the job that it does (checking one word in a free group for primitivity). This is unfair comparison, Magnus is a comprehensive package which does a whole library of tests!

Exploiting inside-outside symmetry

Version 2 of the program exploits the circular symmetry in ∂T. In version 3 the inside-outside symmetry of ∂T is exploited. Notice that if an AB string bounds a disc or surface outside T then the same string with a_i replaced by b_i and vice-versa bounds inside T.

We now store an AB string in form which bounds *inside*. Note that the classification is already largely symmetric. Thus we now have just two useful classes. K : bounds disc inside and knots outside, and S : bounds a surface inside and is not transversely-trivial. The test for both these is the same, namely, not primitive in the fundamental group of the outside.

Supercharging

Observe that the Dehn twists L_i and N_i extend to homeomorphisms of T (twists across their bounding discs). Thus if a curve α bounds a disc (respectively a surface) inside T, then we can find more curves which bound discs (respectively surfaces) inside by applying L_i or N_i to α for varying i. This may convert a curve of class U or T into one of class K or S (or vice-versa). In any case it makes sense to investigate these twists, because the resulting curves have a higher than average chance of being interesting. So whenever a curve is found which bounds a surface inside or outside T is found, then it is first converted (by interchanging a's and b's if necessary) to one which bounds inside and then the Dehn string generator starts running through additional strings of just L_i and N_i.

We call this process *supercharging*. It greatly increases the probability of finding interesting curves in given computer time.

The test for disjoint surfaces

This part of the program is not (at the time of writing) implemented. To test a set of curves for bounding disjoint surfaces outside we first discard those that bound discs (following lemma 4.4) and then order the intersections of the curves with each b_i by comparing the curves from that point on. Once these intersections are ordered, the words in the curves obtained by reading round each b_i can be read and these are required to cancel to the trivial word.

The future

Computers are significantly faster and larger in terms of usable memory than in 1996 when theorem 5.1 was verified. It is now practical to search through Dehn strings up to length 10 with supercharging (as explained above) of length 4. Further, once version 3 is complete, we intend to open the search to other researchers. The search for curves can be carried out in sections on several machines. The process of assembling curves into systems can also be broken into stages. What matters is the disjunctive pairs. We propose to extend the storage to carry the information of which other curves are disjucntive (up to cyclic and inside/outside symmetry). This will make the final search for systems very quick.

Hopefully interesting RR–systems will be found and then it may be worth investigating implementations of the Rubinstein–Thompson algorithm (there is at least one being tested, written in Java by Letscher).

References

[1] **Roger Fenn**, **Michael Greene**, **Dale Rolfsen**, **Colin Rourke**, **Bert Wiest**, *Ordering the braid groups*, Pacific J. (to appear)

[2] **Roger Fenn**, **Ebru Keyman**, **Colin Rourke**, *The Singular Braid Monoid Embeds in a Group*, J. Knot Theory and its Ramifications (to appear)

[3] **Michael Greene**, **Colin Rourke**, *A C program to implement the Rêgo–Rourke algorithm*, in preparation

[4] **W B R Lickorish**, *A finite set of generators for the mapping class group*

[5] **R C Lyndon**, **P E Schupp**, *Combinatorial Group Theory*, Springer–Verlag, Berlin (1977)

[6] **Eduardo Rêgo**, **Colin Rourke**, *Heegaard diagrams and homotopy* 3*–spheres*, Topology 27 (1988) 137–143

[7] **Colin Rourke**, *A primitivity tester:* `primtest.c`, available from: `http://www/maths/warwick.ac.uk/ cpr`

[8] Reference for PC for genus 2 3–manifolds

[9] **F Waldhausen**, *Heegaard-Zerlegungen der* 3*-Sphäre*, Topology 7 (1968) 195–203

[10] **H Zieschang**,*On simple systems of paths on complete pretzels*, AMS Translations, (2) 92 (1970) 127–137