Deciding Contractibility of a Non-Simple Curve on the Boundary of a 3-Manifold*

Éric Colin de Verdière† Salman Parsa‡

Abstract

We present an algorithm for the following problem. Given a triangulated 3-manifold M and a (possibly non-simple) closed curve on the boundary of M, decide whether this curve is contractible in M. Our algorithm is combinatorial and runs in exponential time. This is the first algorithm that is specifically designed for this problem; its running time considerably improves upon the existing bounds implicit in the literature for the more general problem of contractibility of closed curves in a 3-manifold. The proof of the correctness of the algorithm relies on methods of 3-manifold topology and in particular on those used in the proof of the Loop Theorem.

1 Introduction

Computational topology provides algorithms for topological problems. Among the most natural concepts in topology lies that of *homotopy*, which formalizes the notion of deformation. In particular, a closed curve in a topological space X is *contractible* if it can be continuously deformed in X so as to become a single point. More precisely, in a topological space X, a closed curve c is a continuous map from the unit circle into X; that curve is contractible if and only if there exists a continuous map from the unit disk into X whose restriction to the unit circle equals c. In general, two loops (with the same basepoint) are *homotopic* if there exists a continuous deformation between them fixing the basepoint; the set of homotopy classes is a group, called the *fundamental group* of X and denoted by $\pi_1(X)$. Deciding contractibility of a curve amounts to deciding whether that curve is in the trivial homotopy class.

1.1 Deciding Contractibility

Decision problems related to homotopy of curves have been considered from a combinatorial and algorithmic viewpoints for more than a century. Dehn [6] describes a combinatorial procedure to determine whether a closed curve on a surface (or 2-dimensional manifold) is contractible; his algorithm can be analyzed [7], and is a fundamental tool in geometric group theory [10]. Much more recently, Lazarus and Rivaud [18] and Erickson and Whittlesey [9] develop linear-time algorithms for the contractibility problem (and the more general free homotopy problem) on a surface.

Homotopy questions turn out to be much more complicated for more general topological spaces. Actually, the problem of contractibility in 2-dimensional simplicial complexes is undecidable; this follows from the facts that every finitely presented group is the fundamental group of a 2-complex, and that given a finitely presented group specified by generators and relations, there is no algorithm to decide whether a word in the generators and their inverses repre-

*This paper appeared in *Proc. ACM-SIAM Symposium on Discrete Algorithms* (SODA), 2017. Work supported by the French ANR Blanc project ANR-12-BS02-005 (RDAM).

†CNRS, LIGM, Université Paris-Est Marne-la-Vallée, France. Email: `eric.colindeverdiere@u-pem.fr`. Part of this work was done while this author was at CNRS, Département d'Informatique, École normale supérieure, Paris, France.

‡This work was done while this author was at Fondation Sciences Mathématiques de Paris and Département d'Informatique, École normale supérieure, Paris, France. Email: `parsa@alumni.duke.edu`

sents the identity in the group. This implies that the contractibility problem is also undecidable for 4-manifolds. See, e.g., Stillwell [26] for more details.

Between surfaces and 4-manifolds lies the case of 3-manifolds possibly with boundary (e.g., of nice subsets of $\mathbb{R}^3$). This is much more general than surfaces, and in general 3-dimensional topology is extremely more intricate than surface topology; for example, the classification of surfaces has been known for a century, while that of 3-manifolds requires, e.g., the proof of the Poincaré conjecture by Perelman [20, 21]. The general contractibility problem for 3-manifolds is known to be decidable, but the complexity of the problem has not been made explicit; see, e.g., Aschenbrenner et al. [3, Section 4.1], [2]. Most of the literature concentrates on different measures of complexity of the contractibility problem (e.g., automaticity and isoperimetric inequalities [8]).

1.2 Our Result

In this paper, we present an algorithm for deciding whether an input closed curve on the *boundary* ∂M of an input 3-manifold M is contractible in M or not. The 3-manifold is represented by a triangulation, an abstract set of tetrahedra whose faces are glued combinatorially in pairs. The input curve is represented by a piecewise linear (PL) closed curve on the boundary. We assume that the curve is in general position, by which we mean that its singularities consist of simple crossings of (images of) two edges. Our algorithm runs in time exponential in (the square of) the size of the input. More precisely:

Theorem 1.1 *Let T be a triangulation of a 3-manifold M and let $c : S^1 \to \partial M$ be an arbitrary general position PL curve on the boundary of M. Let t be the number of tetrahedra in T, m the number of self-intersection points of c, and n the number of edges of c. Then there is an algorithm that decides whether c is contractible in* $\exp\left(O(m+n+t)^2\right)$ *time.*

With our choice of representation of c, we may have $m = \Theta(n^2)$. If we require that no two edges of c cross except at a common endpoint, then $m = O(n)$. However, we want to make the dependency on m clear. For constant m, the algorithm runs in nondeterministic polynomial time, see the remark in Appendix B.

The same problem restricted to the case of *simple* closed curves, without self-intersections, has been (somewhat implicitly) studied in connection to the unknot problem [12, 1], leading to an algorithm with similar running time. It is also known that in this special case the problem is in NP. We do not know if this is the case for the curves with self-intersections. In addition to be explicit and implementable, the complexity of our algorithm is a considerable improvement over existing implicit bounds for the general contractibility problem[1].

We remark that a singular disk bounding the given curve c of the theorem can also be found in time $\exp(O((m+n+t)^2))$. However, to make the presentation simple, we will not give a formal proof of this statement. The singular disk then has size of the same order as the complexity of the algorithm.

1.3 Related Work in 3-Manifolds

The unknot problem. The *unknot problem* is to determine whether a simple closed polygonal curve in $\mathbb{R}^3$ is isotopic to a standard circle, or an unknot. This is one of the most central problems in 3-dimensional computational topology, shown to be decidable by Haken [11]. More recently, Hass et al. [12] prove that the problem is in NP. In the very recent years, Kuperberg has shown that it is in co-NP assuming the generalized Riemann hypothesis [16], and then Lackenby has announced that it is in co-NP unconditionally [17]. However, whether the problem is polynomial-time solvable remains a challenging open problem.

To provide a polynomial-size certificate for contractibility, Hass et al. [12] use normal surface theory developed by Haken [11] together with tools by

[1] This implicit bound for the complexity of the word problem in 3-manifolds seems to be triply exponential. This follows from the exponential lower bounds for the isoperimetric function of a presentation of the fundamental group of a 3-manifold, see [8, Theorem 8.1.3]. Then the the trivial algorithm for solving the word problem, see [8, Theorem 2.2.5, Lemma 2.2.4], is at least triply exponential in the worst case. See also [3].

Schubert [23] and Jaco and Tollefson [15], which provide a compact encoding of some "nice" embedded surfaces in triangulated 3-manifolds. This leads to an exponential-time algorithm for the contractibility problem for simple curves on the boundary of a 3-manifold. Our result essentially provides an algorithm with similar running time for the more general case of *possibly non-simple* closed curves on the boundary of a 3-manifold. (Incidentally, the above discussion indicates that studying contractibility of curves on the *boundary* of a 3-manifold is rather natural.)

Agol et al. [1] extend significantly the results of Hass et al. [12] to prove that for an arbitrary knot in a given orientable 3-manifold, deciding if the knot bounds an embedded orientable surface of genus less than a given integer g is NP-complete. The unknot problem is the case where $g = 0$.

Immersed normal surfaces. A natural idea for solving the contractibility problem of a non-simple curve c is to look for a singular (not necessarily embedded) disk bounded by c. However, by design, normal surface theory cannot handle non-embedded surfaces. Recently, Burton et al. [4] studied a natural generalization of normal surfaces that could in principle handle surfaces that are not necessarily embedded. However, they prove that it is NP-hard to decide whether such a relaxed normal surface corresponds to an *immersed* surface, or equivalently whether it corresponds to a surface without a branchpoint. In any case, there is no bound known on the complexity of a singular disk bounding a non-simple curve on the boundary, other than those given by the implicit bounds on the general word problem for 3-manifolds. New bounds follow from our algorithm.

1.4 Overview

The main inspiration for our algorithm comes from the celebrated Loop Theorem, proved by Papakyriakopoulos [19], a version of which can be stated as follows:

Theorem 1.2 *Let M be any 3-manifold. Suppose that there is a non-contractible closed curve c in ∂M that is contractible in M. Then there is a* simple *non-contractible closed curve in ∂M that bounds an embedded disk in M.*

Intuitively, if our input curve c is non-contractible in ∂M but contractible in M, this theorem tells us that there is a *simple* curve non-contractible in ∂M but contractible in M. The proof of the loop theorem is actually constructive (although the complexity in the number of steps cannot be bounded efficiently). At a very high level, revisiting the proof of the extensions of the loop theorem, namely those of Shapiro and Whitehead [24] and of Stallings [25], allows us to express the homotopy class of c on ∂M as a product of (exponentially many) homotopy classes of *simple* curves on ∂M, such that if all these simple curves are contractible in M, then so is the original curve c, and vice versa. In the course of our algorithm, we rely on two known algorithms: (1) we need to be able to decide whether a *simple* curve on ∂M is contractible in M; as mentioned above, algorithms for this purpose have been developed by Haken and Schubert as analyzed by Hass et al. [12] and Agol et al. [1]; (2) we need to check whether our input curve c is homotopic, in ∂M, to the concatenation of some other curves; algorithms for this purpose have been developed by Lazarus and Rivaud [18] and Erickson and Whittlesey [9].

As an application of the methods of this paper, we also state the following theorem, which follows from a version of the Loop Theorem that we state and prove. See Appendix D for the proof.

Theorem 1.3 *The following problem is in NP. Given a manifold M and an arbitrary PL curve c on a boundary component of M that is a torus, decide if c is contractible in M.*

The rest of the article is organized as follows. After the preliminaries (Section 2), we describe the algorithm in Section 3. That section is written with as little reference to topological arguments as possible, for the sake of clarity. Finally, Section 4 gives a proof of correctness with details in the Appendix. Although the algorithm as described in Section 3 may seem a bit artificial, it will be enlightened after reading the proof of correctness.

2 Preliminaries

In this section we bring necessary background material used in the algorithm and the proof of its correctness.

2.1 Basic Definitions

Conventions. Without loss of generality, we assume that the given curve c is a PL map in general position on the triangulated surface ∂M, by which we mean that it has finitely many self-intersection points, each of which is a double point where the intersection is transversal.

Notations. By D we mean a triangulated 2-disk. We denote the homotopy class of a curve c' in M by $[c']$ and the homotopy class of c' in ∂M by $[c']_\partial$. The basepoint for the fundamental groups is always a fixed point of ∂M. In general we use subscripts to record the space in which a homotopy class is computed. By a *subcurve* of c we mean any curve $c' : S^1 \to c(S^1) \subset \partial M$. Let I denote the unit interval. The number of elements of a set A is denoted by $\#A$. A map of a pair $f : (D, \partial D) \to (M, \partial M)$ is a map $f : D \to M$ such that $f(\partial D) \subset \partial M$. Moreover, we always will assume $f(D) \cap \partial M = f(\partial D)$.

Smoothings of a curve in a 2-manifold. Let $y \in c(S^1)$ be a double point of c on ∂M. There are two ways to modify the curve c in a small neighborhood of $c^{-1}(y)$, while staying on ∂M, such that the self-intersection y is "resolved". These are depicted in Figure 1. Observe that one of the possibilities results in two curves and the other in a single curve. These operations are called the *smoothings* of c at y. Now fix a choice of smoothing for a subset of self-intersection points of c and change the curve accordingly. The result is a collection of closed curves, that we say are obtained from c by a choice of smoothing for the self-intersection points in the subset.

Triangulations of 3-manifolds. A *3-manifold* (with boundary) is a topological space in which every point has a neighborhood homeomorphic to the

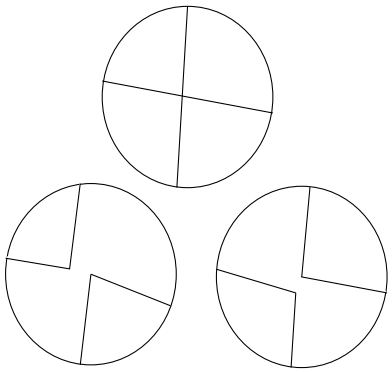

Figure 1: the two possible smoothings of a self-intersection

open unit ball in $\mathbb{R}^3$ or the *closed half-ball* $\{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 < 1, x \geqslant 0\}$. A *triangulation* T of a 3-manifold is the combinatorial data of a number t of tetrahedra, with the information of a gluing between some pairs of their faces (i.e., identification of triangles), such that the resulting topological space, after gluing, equals M. (In particular, without loss of generality, T could be chosen as a 3-dimensional simplicial complex.) Checking whether such a gluing of tetrahedra actually defines a 3-manifold with boundary is easy and standard, see, e.g., [1]; one needs to check that (i) each triangle is incident to at most two tetrahedra, (ii) no edge is identified with itself in reverse, (iii) the tetrahedra incident to a given vertex are "sphere-like" or "disk-like" (technically, the link of each vertex is a 2-dimensional sphere or a 2-dimensional disk).

Good singular disks. Let $f\colon (D, \partial D) \to (M, \partial M)$ be a *singular disk*. We also refer to the image of f as the singular disk. For the purposes of the algorithm, we need only deal with a very simple type of singular disks which we define in this paragraph. See Appendix A for the description of singularities of a singular disk in general position.

For any map f the *singular set* $S(f)$ of f is defined to be the closure of the set $\{x \in D, \#f^{-1}(f(x)) > 1\}$. Moreover, let $S_i(f) = \{x \in S(f), \#f^{-1}(f(x)) = i\}$. In this paper we almost exclusively work with singular disks f in general position with $S_1(f) = S_{\geqslant 3}(f) =$

$\varnothing$. We call such a singular disk a *good* singular disk. This assumption leaves us with only points in $S_2(f)$. It is now very easy to see that the set $f(S(f))$ of singular values of f forms a set of disjoint simple curves, each of which is either a *closed double curve*, a closed curve lying entirely in the interior of M or a *double arc*, a path intersecting ∂M exactly at its endpoints, which are points of self-intersections of the boundary curve $f|\partial D$. A *double curve* means either a closed double curve or a double arc. The preimage of (the image of) a double arc a under f consists of two disjoint chords of the disk D, and a maps each one homeomorphically onto its image. If a is a closed double curve then its preimage under f consists of two disjoint circles or one circle that covers the image twice, since a is locally a homeomorphism and hence a covering map. We note however that not all closed curves in ∂M that are contractible in M bound good singular disks; in particular, in order for this property to hold, the number of self-intersections of the curve has to be even.

2.2 Removing Singularities by Cut and Paste

We assume that $f : (D, \partial D) \to (M, \partial M)$ is a *good* singular disk. There is a process for changing the map f in a neighborhood of (the preimage of) double curves so as to change it into an embedding. We refer to this process as *cut-and-paste*. There are two main lemmas; see [14, Chapter 4] for proofs of the lemmas in this section.

Lemma 2.1 ([14, Lemma 4.6]) *Let M be a 3-manifold and $f : (D, \partial D) \to (M, \partial M)$ a general position map. Suppose f is a good map and consider a closed double curve α. Then given any neighborhood U of $\alpha(S^1)$ in M, there is a general position map $f_0 : (D, \partial D) \to (M, \partial M)$ satisfying : (i) f_0 has one less closed double curve, i.e., misses α, (ii) $f_0(D) \subset f(D) \cup U$, and (iii) f_0 agrees with f outside of a (preassigned) neighborhood of the 2-disk(s) in D bounded by $f^{-1}(\alpha(S^1))$.*

Observe that "resolving" the closed double curve α does not change the map f on the boundary ∂D. Hence, given a good singular disk bounding a curve, we can always obtain a good singular disk without closed double curves. The next lemma takes care of the double arcs which are more troublesome.

Lemma 2.2 ([14, Lemma 4.7]) *Let M be a 3-manifold and $f : (D, \partial D) \to (M, \partial M)$ be a general position map. Suppose f is a good map and consider a double arc α. Then given any neighborhood U of $\alpha(I)$, there are general position maps $f_0, f_1 : (D, \partial D) \to (M, \partial M)$ satisfying (for $i = 0, 1$): (i) f_i has at least one double arc less than f, namely α, (ii) $f_i(D) \subset f(D) \cup U$, and (iii) $[f|\partial D]_\partial$ lies in the smallest normal subgroup of $\pi_1(\partial M)$ containing $[f_0|\partial D]_\partial$ and $[f_1|\partial D]_\partial$.*

With our assumption on f, the double arcs of f can be removed one by one, until the map becomes an embedding. However, the result may be a disk whose boundary is not homotopic to $f|\partial D$ in ∂M. From part (iii) of the lemma it follows that if $[f|\partial D]_\partial$ is not trivial in ∂M, then not both $[f_0|\partial D]_\partial$ and $[f_1|\partial D]_\partial$ are trivial in ∂M. We note that this is the classical application of part (iii) of the lemma.

Assume f is a good general position map. Let $c = f|\partial D$ be the curve on the boundary. The double arcs of f define a pairing of the self-intersection points of $f|\partial D$. The above lemma can be explained as follows. A double arc α is removed by cutting the image of f open along the image of α and then gluing back the pieces in two possible ways. Figure 1 can be thought to be a cross section of a tube neighborhood of the singular arc for the two different possibilities. One of these cases results in a singular disk which is "smaller" than the original singular disk and removes a part of the original singular disk, we denote this by case 0 and the resulting map by f_0. The other map is not essentially "simpler" than the original one and we denote it by case 1 and the resulting map by f_1. See Figure 2 for an example. The curves $f_i|\partial D$ are closed curves obtained by smoothing the two self-intersections at the ends of the double arc. Recall that there exist four choices of smoothing for the two self-intersection points of the curve. Two of these choices result in two curves, one gives three curves and one keeps the curve connected. The cut-and-paste resulting in f_0 breaks the curve into three

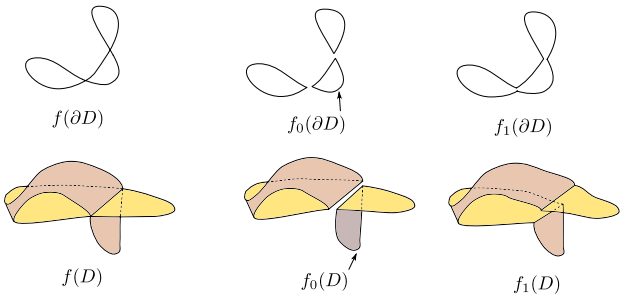

Figure 2: The two possible choices of cut and paste for a simple double arc.

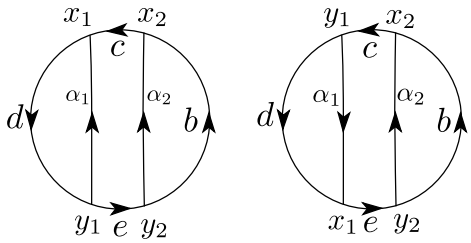

Figure 3: Schematic view of possible orientations. The arcs α_1 and α_2 represent the preimage of the double arc.

pieces, and the other corresponding to f_1 keeps the resulting curve connected. The curve $f_0|\partial D$ is one of the three curves, namely, the one that has pieces defined by both smoothings. The curve $f_1|\partial D$ is the single curve resulting from the two smoothings. The curves $f_0|\partial D$ and $f_1|\partial D$ depend only on $f|\partial D$ and on the preimages, on ∂D, of the endpoints of α.

Now we give details of the formula expressing $[f|\partial D]_F$ in terms of $[f_0|\partial D]_F$ and $[f_1|\partial D]_F$, where F is a regular neighborhood of the curve in ∂M, and $\partial D = S^1$ is given a fixed orientation once and for all. Let α connect points x and y of ∂M, and give α a direction say from y to x. Let $\{x_1, x_2\} = f^{-1}(x)$ and $\{y_1, y_2\} = f^{-1}(y)$. Then depending on the placement of the points x_i, y_i on the circle, two possibilities can happen, depicted in Figure 3. (There are actually two more possibilities, obtained by reversing the orientation of α, but we can actually choose the orientation of α and can thus ignore these two additional cases.) For each possibility, there is a formula expressing $[f|\partial D]_F$ in terms of the $[f_i|\partial D]_F$, $i = 0, 1$, in $\pi_1(F, x_0)$, where the basepoint is the image of the point $b(0)$ in Figure 3. Referring to the figure, let b, c, d, e be arcs $I \to \partial D$ as indicated, and $\beta, \gamma, \delta, \epsilon$ be their images under f. The formula for the left possibility is

(1)
$$[f|\partial D]_F = [f_0|\partial D]_F[\epsilon]_F^{-1}[f_1|\partial D]_F^{-1}[f_0|\partial D]_F[\epsilon]_F,$$
(2) $$f_0|\partial D = \beta\delta,\ f_1|\partial D = \beta\gamma^{-1}\delta\epsilon^{-1},$$

and for the right possibility is

(3)
$$[f|\partial D]_F = [f_0|\partial D]_F[\delta\epsilon]_F^{-1}[f_0|\partial D]_F^{-1}[f_1|\partial D]_F[\delta\epsilon]_F,$$
(4)
$$f_0|\partial D = \beta\delta^{-1},\ f_1|\partial D = \beta\epsilon\delta\gamma.$$

Refer to the proof of [14, Lemma 4.7] for the calculations resulting to the above equations. Note that the placement of the preimages and the case that happens can be deduced from the boundary curve $f|\partial D$. After that, one of the pictures in Figure 3 is fixed. From it the parts $\beta, \gamma, \delta, \epsilon$ of the curve are computed. The *conjugator curve* corresponding to the double arc α is ϵ (in the left possibility) or $\delta\epsilon$ (in the right possibility). These turn out to be essential in our algorithm. We summarize in the following.

Lemma 2.3 *The curves $f_0|\partial D$, $f_1|\partial D$ and the conjugator curve can be computed from $f|\partial D$ if the pair of self-intersections are known. Moreover, a curve representing the right hand side of formula 1 or 3, whichever applies, can be explicitly computed in linear time.*

3 Combinatorial Algorithm

3.1 The Elementary Cut-and-Paste Tree

As described in Section 2, if the map $f : (D, \partial D) \to (M, \partial M)$ is a good general position map then the

double arcs can be removed one by one, giving rise to a collection of maps. In this way, a binary tree is formed whose nodes correspond implicitly to maps and the edges correspond to the two different paste methods for resolving a double arc of the map. This is the background of the following definition of an elementary cut-and-paste tree for a curve $c' : S^1 \to \partial M$.

Let Y be a subset of the set of self-intersection points of c'; assume that $\#Y$ is even. Let $\Delta = \{(y_{2j-1}, y_{2j}), j = 1, \ldots, |Y|/2\}$ be a partition of the set Y into pairs. With this partition as parameter, we describe a procedure that either constructs a labelled binary tree, called an *elementary cut-and-paste tree* for c' with the given parameter, or reports that the set of pairings does not correspond to such a tree. The edges are labelled 0 or 1. We refer to the nodes of the tree using the label of the path from the root to the node; these labels are words on the alphabet $\{0, 1\}$. We denote the empty word by λ. Each node stores some data. If P is a word on $\{0, 1\}$, the data stored on the node described by P, called the *label* of the node, contains an element Δ_P of the set Δ. In addition, it explicitly stores two curves $g_P, r_P : S^1 \to \partial M$.

The tree is constructed as follows. Start by ordering the set Δ arbitrarily, say as indexed by j. We present a recursive procedure. For the root, put the first element of Δ in the label. Set $g_\lambda = c'$, and create two nodes as children of the root and label the edges 0 and 1. The idea is that we assume that the pair is connected by a double arc α in some singular disk, and compute the result of cut-and-paste on the double arc. In more detail, we first smooth the two self-intersection points to get a collection of three curves. If this is not possible, the procedure fails and stops. Note that it could be that the second intersection point to be smoothed is not a self-intersection of the resulting curves after the first smoothing. If this happens then the the procedure stops. Otherwise, we get three curves and g_0 is the curve that is "incident" to two smoothings. See Figure 2. The map g_1 is the single curve resulting from smoothings of the two intersection points. By knowing the pair of self-intersection points we can explicitly compute the curves g_0 and g_1 and store them as labels of the corresponding children. The curve r_λ is the conjugator curve of the imaginary singular disk. This curve can be computed explicitly by Lemma 2.3. For $i = 0, 1$, define $Y_i \subset Y$ to be the set of those elements whose intersection points both are self-intersection points of g_i. We then recurse to the nodes 0 and 1 with the ordered sets Y_0 and Y_1. The recursion stops when there are no more self-intersection points. The leaves therefore contain labels with no self-intersection points from Y and no conjugator curve and a simple curve as g_P. This finishes the construction of the elementary cut-and-paste tree.

Lemma 3.1 *Let $c' : S^1 \to \partial M$ be a PL curve. Given a set of parameters Δ with $n = |\Delta|$, there is an algorithm that runs in time $O(2^n p(n))$ for some polynomial p and constructs an elementary cut-and-paste tree for c' using parameter Δ if such a tree exists, or returns "fail".*

3.2 The General Cut-and-Paste Tree

We define here the general notion of a cut-and-paste tree combinatorially as a data structure. We shall see later that the definition originates from all possible resolutions of certain singular disks in a tower of covering spaces appearing in the proof of the Loop Theorem.

Let $\Sigma \subset \partial M$ be the set of self-intersection points of a curve $c' : S^1 \to \partial M$ and assume $\#\Sigma$ is even. The tree is defined only in this case. Let $\Pi = (\Sigma_1, \ldots, \Sigma_k)$ be an *ordered partition* of Σ, that is, the sets Σ_i are disjoint and their union equals Σ. An ordered partition is called *even* if each of its sets has an even number of elements. Next, for each set Σ_i, let Δ_i be a partition of its elements into ordered pairs, $\Delta_i = \{(y_{i,2j}, y_{i,2j-1}), j = 1, \ldots, |\Sigma_i|/2\}$.

We describe now a procedure that given any Σ, any even ordered partition Π of Σ, and any fixed set of pairs $\Delta_1, \ldots, \Delta_k$ either constructs a labelled binary tree, called the *cut-and-paste tree* with given parameters, or reports that the input does not correspond to such a tree. Like an elementary cut-and-paste tree, each edge is labeled 0 or 1 and the nodes are determined by paths from the root. Let P be the path of a node. The node given by P has a label consisting of the following data: an element Δ_P of Δ_j for some j,

a curve $g_P : S^1 \to \partial M$ which is explicitly stored and self-intersects on the two points in Δ_j, and a second curve $r_P : S^1 \to \partial M$ called the conjugator curve.

The procedure for constructing the tree works inductively as follows. Let $\mathcal{T}_1$ be an elementary cut-and-paste tree constructed for the parameter Δ_1 for the curve c' by Lemma 3.1. If there is no such tree, we stop and the process fails. Now assume $\mathcal{T}_i$ is constructed, and we proceed to construct $\mathcal{T}_{i+1}$, $i < k$. Consider a leaf of $\mathcal{T}_i$ having path P. To this leaf is assigned a curve g_P. Consider the set Δ_{i+1} of pairs of self-intersections of c'. Let Δ'_{i+1} be those intersection points that are also intersection points of g_P. Construct an elementary cut-and-paste tree T for the set Δ'_{i+1} and the curve g_P by Lemma 3.1. Then we merge the root of T with the leaf P. The data of the leaf is kept for the new node other than the pair of self-intersection points that is given by the root of T. The tree $\mathcal{T}_{i+1}$ is obtained after performing this for all the leaves. At the end, the cut-and-paste tree defined by the given parameters is $\mathcal{T} = \mathcal{T}_k$. Observe that the curves of the leaf nodes of $\mathcal{T}$ are simple curves.

3.3 The Class of a Cut-and-Paste Tree

Let G_0 be the space $c'(S^1) \subset \partial M$, i.e., a graph on the surface ∂M. We will assign, to a general cut-and-paste tree $\mathcal{T}$ built for a curve $c' : S^1 \to \partial M$ and a fixed parameter set, an element $e(\mathcal{T}) \in \pi_1(G_0)$. To be precise we have to deal with the basepoint. Let $x_0 \in c'(S^1)$ be an arbitrary point, which will serve as the basepoint for the fundamental group. Now curves g_P and r_P, for all P, have to be *based*, that is, connected to the basepoint. We will do this by defining for every curve g_P and r_P a simple path on $c'(S^1)$ that connects the basepoint to a point on g_P, r_P. If the path for g_P is q_P then define $g'_P = q_P^{-1} g_P q_P$ and similarly for r'_P; note that these new curves can easily be computed when computing g_P, r_P. The class of the tree is defined as a formal product of classes of g'_L and r'_P, where L is a leaf. We start from the leaves and go up the tree, writing a product for a node using those of its children. To each node then is assigned a formal product. To a leaf L is assigned $[g'_L]$. Consider the node P. The pair of self-intersection points in the label of P and the curve g_P fix one of the equations 1 or 3. We consider the formula for the based curves (i.e., the terms in it are based curves). The curves appearing in the chosen formula have been computed already and stored in the labels. We then take the right hand side, and replace the at most two appearances of the symbol $[g'_{P0}]$ by the product computed for the node $P0$. Similarly, for g'_{P1}. The resulting expression is the product assigned to the node P.

We continue in a bottom-up manner. After the root node is processed, we have a formal product of the $[g'_L]$, where L is a path from the root to a leaf, and the $[r'_P]$ for all P and their inverses. We claim that the product has $O(m3^{m/2})$ terms. Consider the longest path from a leaf to the root. Let $a(n)$ be the size of the expression for a node on this path, where $a(0) = 1$ and $a(l)$ is the number of terms for the expression of the root, where l is the size of the path. Then $a(n) \leqslant 3a(n-1)+2$, where we count two conjugator curves for each node. The bound follows since there are at most $m/2$ levels in the tree. The element defined by this product is called the *class* of the cut-and-paste tree and denoted by $e(\mathcal{T})$. A curve representing this class can be found from this formal product by concatenating the curves appearing in it. Observe that the class e is in the normal subgroup generated by the classes g'_L for all leaf paths L.

3.4 Algorithm

Here we present the algorithm for deciding if a given curve with self-intersections on the boundary of a 3-manifold is contractible or not. See Algorithm 1. The input to the algorithm is a triangulation T of a manifold M. The curve is given as a piecewise linear curve $c : S^1 \to \partial T$, where ∂T is the set of those triangles of T that are incident to only one tetrahedron. By convention c is in general position. The contractibility algorithm uses a subroutine SContract; see Algorithm 2.

To finish the presentation of the algorithm we need to explain what we mean by removing and reintroducing self-intersections of a curve $c' : S^1 \to M$. A self-intersection point of c' is *removed* by pushing one branch of the curve slightly into the interior of M so as to remove the self-intersection. The inverse of this is called *reintroducing* the self-intersection point.

Data: The triangulation T of a 3-manifold M and a PL curve c into ∂T
Result: YES if $[c]_M = 1$ and NO otherwise

```
1  if [c]_∂ = 1 then
2  |  return YES;
3  end
4  for each subset X of self-intersection points of c do
5  |  Let c_X be the curve obtained from c by removing the self-intersections not in X;
6  |  Compute a graph G_X isomorphic to c_X(S^1) ⊂ M;
7  |  Let a_1, ..., a_w be a set of simple loops of G_X that form a basis for the loop space;
8  |  For each i ∈ {1, ..., w}, compute a subcurve s_i : S^1 → ∂M of c by reintroducing intersections in the image of a_i in ∂M;
9  |  if SContract(s_i)=YES for all i then
10 |  |  return YES;
11 |  end
12 end
13 return NO;
```

Algorithm 1: The contractibility algorithm.

Note that this is just a simple and convenient way of describing the graph G and it is easy to compute G combinatorially, without reference to M. (The correct way of thinking about the curve c with its self-intersections not in X removed is to assume that there is a covering space of the manifold, or of a neighborhood of the curve in ∂M, such that the curve lifts in that covering space to a curve that has self-intersection from X only. Then the graph G_X is isomorphic with the lift. These notions will be clarified in the rest of the paper.)

The notion of *special curve* is used in the subroutine SContract, and we postpone its definition to Section 4; for now, we note only that every special curve is contractible.

We use two black boxes in the algorithm. The first one is on line 4 of the SContract subroutine where we decide if a simple curve on ∂M is contractible in M, and this can be done using normal surface theory by Haken and Schubert [23] as analysed by Hass et al. [12] and Agol et al [1]. (Agol et al. require the 3-manifold to be orientable, but we can safely assume so from the beginning: Indeed, c is contractible in the non-orientable 3-manifold M if and only if, in the two-sheeted orientable cover of M, the curve c lifts to a closed contractible curve, and we can easily build the two-sheeted orientable cover and the lift.) The second blackbox is on line 1 of the main algorithm and on line 6 of the SContract subroutine, where we check one PL curve on a 2-manifold to be contractible, or (equivalently) two PL curves with basepoint on a 2-manifold to be homotopic; this can be done by using the algorithms by Lazarus and Rivaud [18] or Erickson and Whittlesey [9]. The details of these steps are deferred to Appendix B. The analysis of the running time of the algorithm follows easily, and is also deferred to Appendix B; in short, the first

Data: A triangulation T of a 3-manifold M and a PL curve s on the boundary ∂T
Result: YES or fail. If the algorithm returns YES, then the input curve is contractible. If the algorithm returns "fail", then the input curve is not special.

```
1  for all possible parameters of a general cut-and-paste tree for s do
2  |  Try to construct the cut-and-paste tree and its labels from the given parameters, using the procedure of Section 3.2;
3  |  if the parameters define a general cut-and-paste tree 𝒯 then
4  |  |  if for all leaves defined by paths L, [g_L] = 1 then
5  |  |  |  Write the class of the cut-and-paste tree e = e(𝒯) ∈ π_1(∂T, x_0) as a formal product using Section 3.3;
6  |  |  |  if [s]_∂ = e then
7  |  |  |  |  return YES;
8  |  |  |  end
9  |  |  end
10 |  end
11 end
12 return fail;
```

Algorithm 2: The special contractibility algorithm SContract.

black box is an exponential-time algorithm, run on curves that have complexity linear in the input size, whereas the second one is a linear-time algorithm, run on curves that have complexity exponential in the input size; both black boxes are used exponentially many times.

One part of the correctness proof is easy:

Lemma 3.2 *If the algorithm return YES then the input curve c is contractible in M.*

Proof. First consider SContract. If it returns YES then its input curve s is contractible. This is because then e, which is in the normal subgroup generated by g'_L, is contractible, and so is s by line 6. Therefore, if Algorithm 1 returns YES then, in the notation of the algorithm, there exists some $X \subset \Sigma$ such that the subcurves $s_1, \ldots, s_w$ of G_X are contractible.

Let $G = G_X$ be the graph given by assumption and let $h : G \to c(S^1)$ be the map that is defined by reintroducing self-intersection points. There exists a curve $c' : S^1 \to G$ such that $h \circ c' = c$. Let $a_1, \ldots, a_w$ be as in the algorithm description. Then $\pi_1(G)$ is a free group generated by classes of the form $[b_i a_i b_i^{-1}]_G$ where b_i is a path connecting the basepoint to the loop a_i, for $i = 1, \ldots, w$. Now $[c']_G$ can be written as a product of $[b_i a_i b_i^{-1}]_G$ and their inverses. Let F_0 be a regular neighborhood of the curve in ∂M. Then $[c]_{F_0} = [h(c')]_{F_0}$ can be written as a product of $[h(b_i a_i b_i^{-1})]_{F_0}$ and their inverses. To show that $[c] = 1$ it is enough to show that for all i, $[h(a_i)] = 1$ with respect to any basepoint. However, $h(a_i)$ has been checked by SContract to be contractible. So the given curve c is contractible. □

4 Computational Aspects of the Loop Theorem: Proof of Correctness

The goal in this section is to prove that if $[c] = 1$ then the algorithm returns YES. Together with Lemma 3.2, this proves the correctness of the algorithm. However, we need to introduce some concepts and state some lemmas. From this point, we assume that the reader has a familiarity with the basic notions of combinatorial and algebraic topology, like fundamental groups, covering spaces, regular neighborhoods, etc.; see, e.g., Hatcher [13].

The main theoretical background for the proof of the correctness of our algorithm is the Loop Theorem and the method of its proof. We state a theorem that is essentially the same as the Loop Theorem but extended to provide more information. During the proof of this theorem, we manage to extract combinatorial data from the proof. This information is used to prove the correctness of our algorithm. A version of the classical Loop Theorem was stated in Theorem 1.2. Historically, the Loop Theorem originates as the famous Dehn's Lemma which was stated and used by Dehn [5]. The first correct proof was given by Papakyriakopoulos [19] using the theory of covering spaces. Later Shapiro and Whitehead [24] presented a simpler proof of an extension of the Dehn's Lemma. This proof is the basis for the next extension by Stallings [25]. Indeed, we heavily use the proof of this extended version, which can be found also in [14] and [22]. We refer to [22] for a self-contained proof. We cite [24] as the most geometric and elegant exposition which also requires a reference to the original paper of Papakyriakopoulos for completeness.

Theorem 4.1 *Assume c is a contractible closed curve on the boundary ∂M of a 3-manifold M; let F_0 be a regular neighborhood of c. Then there exist curves $g'_1, \ldots, g'_N$, with $N \leqslant 3(m+n)2^{m/2}$, such that:*

- *$[c]_{F_0} \in ([g'_1]_{F_0}, \ldots, [g'_N]_{F_0})$, i.e., $[c]_{F_0}$ is in the normal subgroup generated by $[g'_1]_{F_0}, \ldots, [g'_N]_{F_0}$,*
- *each g'_i consists of a simple curve g_i, which is obtained by a smoothing of all of the self-intersection points of c, and a simple path that connects g_i to the basepoint,*
- *for each i, g_i (hence g'_i) is contractible in M.*

In the rest of this section, we define certain concepts needed for the proof of Theorem 4.1 and Lemmas 4.3 and 4.4 which contain the main technical arguments for the correctness of the algorithm. The rest of the proofs is deferred to Appendix C. The following lemma is used in the argument.

Lemma 4.2 ([14, Lemma 4.9]) *If the compact 3-manifold M has a boundary component which is not a 2-sphere, then it has a connected double covering.*

The following considerations appear in the proofs of the Loop Theorem, see, e.g., [14], and we briefly indicate them as they form an essential part of the theory behind the algorithm. Since $[c] = 1$, there exists a general position map $f : (D, \partial D) \to (M, \partial M)$ such that $\partial D = S^1$ and $f|\partial D = c$. By standard arguments, we can assume D is triangulated and f is a piecewise linear map since we are allowed to change f by a homotopy. Let $V_0 \subset M$ be a regular neighborhood of the image $f(D)$. Let $F_0 = V_0 \cap \partial M$ be a regular neighborhood of c in ∂M. Note that to define these regular neighborhoods one can use a subdivision of the triangulation T of M that contains the image of f as a subcomplex.

Assume V_0 has a boundary component that is not a 2-sphere. Then there is a double covering space $p_1 : M_1 \to V_0$. By the lifting criterion for covering spaces (see, e.g., [13, Prop. 1.33]), since $\pi_1(D) = 1$, f_0 lifts to a map $f_1 : (D, \partial D) \to (V_0, p_1^{-1}(F_0))$ such that $p_1 f_1 = f_0$. Let V_1 be a regular neighborhood of $f_1(D)$ in M_1 and let $F_1 = V_1 \cap p_1^{-1}(F_0)$ a regular neighborhood of $c_1 = f_1(\partial(D))$ in ∂M_1. The procedure applied to V_0 and F_0 can be repeated as long as there exists a boundary component of V_1 that is not a 2-sphere. In general, there is a double covering $p_{k+1} : M_{k+1} \to V_k$. The map $f_k : (D, \partial D) \to (V_k, F_k)$ lifts to a map $f_{k+1} : (D, \partial D) \to (M_{k+1}, p_{k+1}^{-1}(F_k))$. The regular neighborhood of the image $f_{k+1}(D)$ is denoted V_{k+1}, and $F_{k+1} = V_{k+1} \cap p_{k+1}^{-1}(F_k)$ is a regular neighborhood of $c_{k+1} = f_{k+1}(\partial D)$, a lift of c_k.

It is possible to have all the f_k defined on the same triangulation of D, see the proof in [14]. Then the standard proof of the Loop Theorem shows that the above tower of covering spaces is of finite height and, for some $l \geqslant 0$, all the boundary components of V_l are 2-spheres. The number l is bounded from above by the number of singular double curves of f, see [24]. Therefore, the curve c_l lies on a 2-sphere $S \subset \partial V_l$. Let $F_l \subset p_l^{-1}(F_{l-1})$ be a regular neighborhood of c_l in S. We say that c *lifts to* c_l *on top of a tower of covering spaces* when a tower of double coverings like above exists for the curve c and c_l is a lift of c on top of the tower.

We call a curve c a *special* curve if there exists a tower of covering spaces (of a part of the manifold containing c) for which c lifts to a simple curve on a sphere on top of the tower. This definition suffices for our purposes. Note that a special curve is contractible.

Lemma 4.3 (Reduction to Special Curves) *Let $c : S^1 \to \partial M$. Let Σ be the set of self-intersection points of c. If $[c] = 1$ then there exists a subset $X \subset \Sigma$ with the following property. Let the graph $G = G(X)$ be obtained from $c(S^1)$ by removing self-intersections not in X. Let $a_1, \ldots, a_w$ be simple loops of G generating its loop space and let s_i be obtained from a_i by reintroducing self-intersections not in X. Then, for each i, s_i is a special curve.*

Proof. We have just shown that if $[c] = 1$ then a tower exists and we set X to be the set of self-intersection points of c_l. Removing self-intersections not in X from $c(S^1)$ gives a graph homeomorphic to $c_l(S^1)$. Take a simple loop of $G(X)$ and reintroduce intersections not in X. Since the resulting curve does not have an intersection point in X, it lifts to a simple curve on top of the tower. □

We state the second lemma that is used in the proof of the correctness of the algorithm.

Lemma 4.4 (Existence of a Cut-and-Paste Tree) *Let $s : S^1 \to \partial M$ be a special curve on ∂M. Then there exists a set of parameters $(\Pi, \Delta_1, \ldots, \Delta_k), k = |\Pi|$, for which a labeled cut-and-paste tree is defined by the procedure of Section 3.2. Moreover, the class of the tree expresses $[s]_\partial$ correctly as a product of the based curves g'_L and the based conjugator curves r'_P of the tree. In addition, the curves g_L are all null-homotopic.*

The proof is given in Appendix, Section C. Essentially, each elementary cut-and-paste tree corresponds to the removal of the double arcs of a single good singular disk in one level of the tower of covering spaces above.

Finishing the proof of correctness of the algorithm. We assume $[c] = 1$ and prove that the algorithm then returns YES. If $[c] = 1$ then it follows from Lemma 4.3 that there exists a subset X of the set of self-intersection points of X such that each curve a_j, $j = 1, \ldots, w = w(X)$ defined in the algorithm, s_i is a *special* curve. Then we need to show that, if the given curve is special, then SContract returns YES. Lemma 4.4 shows that a set of parameters exists for which the general cut-and-paste tree for a special curve exists and whose class equals the class of the special curve and all the g_L are trivial. Hence SContract stops and returns YES when this set of parameters is found. This finishes the proof of correctness.

Acknowledgments. The authors would like to thank Jonathan Spreer for useful discussions concerning possible applications of the results of this paper.

References

[1] I. Agol, J. Hass, and W. Thurston. The computational complexity of knot genus and spanning area. *Transactions of the American Mathematical Society*, 358(9):3821–3850, 2006.

[2] M. Aschenbrenner, S. Friedl, and H. Wilton. *3-manifold Groups*. EMS Series of lectures in mathematics. European Mathematical Society, 2015.

[3] M. Aschenbrenner, S. Friedl, and H. Wilton. Decision problems for 3-manifolds and their fundamental groups. *Geometry & Topology Monographs*, 19:201–236, 2015.

[4] B. A. Burton, É. Colin de Verdière, and A. de Mesmay. On the complexity of immersed normal surfaces. *Geometry and Topology*, 20:1061–1083, 2015.

[5] M. Dehn. Über die Topologie des dreidimensionalen Raumes. *Mathematische Annalen*, 69(1):137–168, 1910.

[6] M. Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72:413–421, 1912.

[7] B. Domanski and M. Anshel. The complexity of Dehn's algorithm for word problems in groups. *Journal of Algorithms*, 6:543–549, 1985.

[8] D.B.A. Epstein. *Word Processing in Groups*. Ak Peters Series. Taylor & Francis, 1992.

[9] J. Erickson and K. Whittlesey. Transforming curves on surfaces redux. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1646–1655, 2013.

[10] M. D. Greendlinger. On Dehn's algorithm for the conjugacy and word problems with applications. *Communications on Pure and Applied Mathematics*, 13:641–677, 1960.

[11] W. Haken. Theorie der Normalflächen, ein Isotopiekriterium für den Kreisknoten. *Acta Mathematica*, 105:245–375, 1961.

[12] J. Hass, J. C. Lagarias, and N. Pippenger. The computational complexity of knot and link problems. *J. ACM*, 46(2):185–211, March 1999.

[13] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

[14] J. Hempel. *3-manifolds*. AMS Chelsea Pub., American Mathematical Society, 2004.

[15] W. Jaco and J. L. Tollefson. Algorithms for the complete decomposition of a closed 3-manifold. *Illinois J. Math.*, 39(3):358–406, 09 1995.

[16] G. Kuperberg. Knottedness is in NP, modulo GRH. *Advances in Mathematics*, 256:493–506, 2014.

[17] M. Lackenby. The efficient certification of knottedness and Thurston norm. arXiv:1604.00290, 2016.

[18] F. Lazarus and J. Rivaud. On the homotopy test on surfaces. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 440–449, 2012.

[19] C. D. Papakyriakopoulos. On Dehn's lemma and the asphericity of knots. *Annals of Mathematics*, 66(1):1–26, 1957.

[20] G. Perelman. The entropy formula for the Ricci flow and its geometric application. arXiv:math/0211159, 2002.

[21] G. Perelman. Ricci flow with surgery on three-manifolds. arXiv:math/0303109, 2003.

[22] D. Rolfsen. *Knots and Links*. AMS Chelsea Pub., American Mathematical Society, 1976.

[23] H. Schubert. Bestimmung der Primfaktorzerlegung von Verkettungen. *Mathematische Zeitschrift*, 76(1):116–148, 1961.

[24] A. Shapiro and J. H. C. Whitehead. A proof and extension of Dehn's lemma. *Bulletin of the American Mathematical Society*, 64(4):174–178, 1958.

[25] J. Stallings. On the loop theorem. *Annals of Mathematics*, 72(1):12–19, 1960.

[26] J. Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, second edition, 1993.

A Singular Disks

The definitions and lemmas in this paragraph are based on Hempel [14, Chapter 4]. For any map f the *singular set* of f, $S(f)$, is defined to be the closure of the set $\{x, \#f^{-1}(f(x)) > 1\}$. Let $f : (D, \partial D) \to (M, \partial M)$ be a PL map. The set $S(f)$ can be decomposed as $S(f) = \bigcup_{i \geqslant 1} S_i(f)$ where $S_i = S_i(f) = \{x \in D, \#f^{-1}(f(x)) = i\}$. The elements of $f(S_1)$ are called *branch points* of f, those of $f(S_2)$ *double points*, those of $f(S_3)$ *triple points* and so on.

There exist notions of a "general position" map in 3-manifold topology, and, as is expected, there exist general position maps approximating an arbitrary continuous map. We refer to standard books such as [14] for more details. In this paper we use only the existence of a map $f : (D, \partial D) \to (M, \partial M)$ and we are allowed to change the map by a homotopy. So we can always assume that we are given a general position map. For f a general position map $S_4(f) = \varnothing$, and $S_3(f)$ is finite. Define a *closed double curve* to be a closed curve $a : S^1 \to f(S(f))$ such that the singular set $S(a)$ is finite and $a(S(a)) \subset f(S_3)$. A *double arc* is an *arc* $a' : I \to f(S(f))$ such that $S(a')$ is finite, $a'(S(a')) \subset f(S_1 \cup S_3)$, and moreover a' maps the points in the interior of I into the interior of M and away from any branch point, and the boundary ∂I is mapped to the boundary of M or to a branch point. See Figure 4 for an example. By a *double curve* we mean either a double closed curve or a double arc. Note that the conditions imply that triple points can happen as self-intersections of double curves or as intersection of double arcs and closed double curves. Moreover, a double arc starts and ends at a branch point or at a double point of the $f|\partial D$ on the boundary of M. The main observation here is that the set of singular values $f(S(f))$ can be described using finitely many closed double curves and double arcs. We state the following lemma, see [14, Lemma 4.5]. We denote by $im(a)$ the image of the curve a.

Lemma A.1 *Let $f(D, \partial D) \to (M, \partial M)$ be a general position map. Then there is a finite collection $a_1, \ldots, a_n$ of double arcs and closed double curves of f such that, for $i \neq j$, $im(a_i) \cap im(a_j) \subset f(S_3)$ and moreover f has no singularity other than these curves. The a_i are unique up to parametrization.*

B Complexity Analysis

In this section we analyse the complexity of the algorithm in detail. The input size is expressed by the number of tetrahedra t in T, and the size n of the curve c as a polygonal chain. Recall that m denotes the number of self-intersections of c and is bounded by n^2.

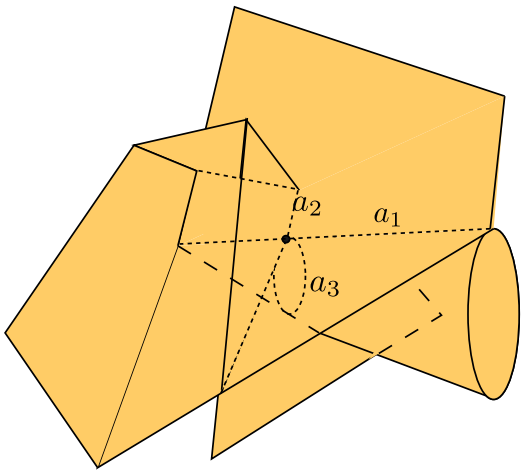

Figure 4: An example of a singular disk with a closed double curve, two double arcs, one branch point and one triple point

In line 4 of the algorithm SContract, we need to decide for a simple closed curve $g_L : S^1 \to \partial T$ if this curve is contractible in the 3-manifold or not. Here is when we use the existing algorithms. Results of the papers [12] and [1] can be used for this purpose. We first translate the problem to an instance of the problem addressed by Agol et al. [1], namely, deciding if a simple closed curve in the interior of a 3-manifold bounds an embedded surface of genus $\leqslant g$. By Dehn's lemma, if the curve g_L (on the boundary) bounds a singular PL disk, it bounds an embedded disk. Then by gluing new tetrahedra to the boundary ∂T we can force g_L to be a knot g'_L in the interior of a manifold M'. Then g'_L is a knot of genus 0 in M' if and only if the curve g_L is contractible in M. Therefore, we can compute the genus of the knot g'_L in M'. Let $t' = m+n+t$. The problem of deciding if the genus of a knot is less than a given number g for an arbitrary manifold is NP-Complete [1]. The certificate is the normal coordinates of a fundamental normal surface. Each normal coordinate is bounded by $O(t'2^{7t'}))$, see [12, Lemma 6.1]. This is because the simple curve g_L, which is of size $O(m+n)$, has to be made part of the triangulation of the manifold M', and this triangulation will have size of order $O(t') = O(m+n+t)$. This certificate can be checked in polynomial time in its bit-length to be a valid certificate by the orbit counting algorithm of [1]. Moreover, there are at most $2^{O(t'2)}$ many normal surfaces that we have to check, by [12, Lemma 6.2]. Therefore, this step of the algorithm takes $2^{O((m+n+t)^2)}$ time deterministically and polynomial non-deterministic time. Note that in the paper [1] the only condition for the input manifold is orientability. We can assume this is the case. If the given manifold is not orientable we construct an orientation double covering of it. Then our input curve (with self-intersections) is contractible in M if and only if it lifts to a contractible curve in the covering, hence we can pass to the orientation double cover which is always orientable.

In line 6 of the algorithm SContract, we need to decide if two elements of the fundamental group of the surface ∂T containing the curve are equal. We have a representative curve of both elements and we want to decide if these two curves define the same element of the fundamental group. This can be approached in two different ways. The conceptually easier approach is to consider instead of ∂T a neighborhood F_0 of the image of the curve $c(S^1)$ and check the equality in the fundamental group $\pi_1(F_0) = \pi_1(c(S^1))$. It follows from the proof of correctness that the equality has to hold in $\pi_1(F_0)$. Then one writes both curves as product of free generators and their inverses and decides if they represent the same element. This can be done in polynomial time in the length of the two words. Another approach is to use the results of Lazarus and Rivaud [18] or Erickson and Whittlesey [9] and check whether the two curves representing the two elements are homotopic on the surface. This can be done in linear time in the size of the input. Note that the proof of correctness guarantees that if $[s] = 1$ then with the right parameters the two sides are homotopic and it is clear that if the two curves are freely homotopic we still can deduce the contractibility of s from that of e. Hence homotopic can be replaced with freely homotopic here. The running time of this step is then $O(n^2 2^{m/2})$ by the algorithms of [18, 9] since the curve on the right hand side is a concatenation of $O(2^{m/2})$ subcurves of c.

Running time of SContract. The total number of possible parameters for a general cut-and-paste tree can loosely be bounded by $m^{m/2}B(m/2)(m/2)!$, where one first chooses a decomposition into pairs of all self-intersection points, partitions the set of pairs and orders the partitions. The number $B(n)$ denotes the number of partitions of a set of n elements and is less than n^n. It follows that there are at most $2^{2m\log m+m}$ different parameters. For each set of parameters the construction of the cut-and-paste tree and a curve representing the class of the tree takes $O(m3^m p'(n))$ time where $p'()$ is a polynomial. There exist at most $2^{m/2}$ leaves and deciding contractibility for each of them takes $2^{O((m+n+t)^2)}$ time by the above. Line 6 takes linear time in size of the curves which is then $O(n^2 m3^{m/2})$. It then follows that the total running time for SContract is $O(2^{2m\log m+m}(2^m p'(n)+2^{m/2}\cdot 2^{O((m+n+t)^2)}+ n^2 m3^{m/2}))$ which is $\exp(O((m+n+t)^2))$.

Complexity of the main algorithm. The main algorithm makes at most $(m+n)^2$ calls to SContract for each choice of the subset, since G_X is a graph on $m+n$ vertices. The rest of the operations performed for each subset X are polynomial. Hence, in total the algorithm complexity is bounded by $\exp(O((m+n+t)^2)))$.

Remark. We note that the only obstruction for the algorithm to be non-deterministically polynomial time is the construction of the general cut-and-paste tree. If in a situation the tree is guaranteed to have polynomial size then the algorithm runs in polynomial non-deterministic time, and the corresponding problem would be in NP.

C Existence of a Labelled Cut-and-Paste Tree

Proof of Theorem 4.1. We use the notation of Section 4. We assume that the given curve c is contractible and a tower of covering spaces is defined for it as in Section 4. Then c lifts to a curve c_l on a sphere on the boundary of V_l. Let G be a graph homeomorphic to $c_l(S^1)$ and ι be such a homeomorphism. Let $a_1,\ldots,a_w$ be a set of simple curves of G generating its loop space. Each curve $\iota(a_i)$ is a simple loop in S hence bounds an embedded disk in V_l. These curves are obviously special. From now on we work with a fixed $a=a_j$ for some j.

We choose a fixed basepoint x_0 on the image of c in ∂M for the fundamental group. For reasons that will become clear later, for each self-intersection point of c in ∂M we fix, once and for all, a simple path that connects it to the basepoint. If $x\in\partial M$ is a self-intersection point, we denote its path by $q_x : I\to c(S^1)\subset\partial M$.

For $j=l,l-1,\ldots,1$ set $\phi_j=(p_j|F_j)_*:\pi_1(F_j)\to\pi_1(F_{j-1})$, where always a lift of $x_0\in F_0$ is used as a basepoint for the fundamental groups. Moreover let $\phi=\phi_1\phi_2\cdots\phi_l:\pi_1(F_l)\to\pi_1(F_0)$. Let $f_a:(D,\partial D)\to(V_l,\partial V_l)$ be the embedding of the disk that a bounds. We now go down the tower. Let $\hat{f}_{l-1}=p_l f_a:(D,\partial D)\to(V_{l-1},F_{l-1})$. We know that

$$[\hat{f}_{l-1}|\partial D]_{F_{l-1}}=(p_l|_{F_l})_*(a)=\phi_l(a). \tag{5}$$

As in the standard proof of the Loop Theorem, the map $\hat{f}=\hat{f}_{l-1}$ can have only simple double curves. Using Lemma 2.1, all the closed double curves of $\hat{f}$ can be removed without introducing any new singularities and such that $\hat{f}$ is unchanged on a neighborhood of ∂D, hence, we can assume that $\hat{f}$ does not contain closed double curves. Then it is a good singular disk. An *abstract* elementary cut-and-paste tree can be defined for the singular disk $\hat{f}$. We call this tree abstract, in contrast to the combinatorial tree defined for a given parameter set. This tree is determined after fixing an ordering of the singular double arcs of $\hat{f}$. Then starting from the root, the two possible cut-and-paste procedures are applied to a node, resulting to its two children. In this way a tree is defined and to each node is associated a singular disk. We say that the abstract elementary tree for $\hat{f}$ has *level* $l-1$ since $\hat{f}$ is a map into V_{l-1}. The maps corresponding to the leaves define simple curves on the boundary in ∂V_{l-1}. Depending on the maps, one of the formulas 1 and 3 apply. It follows that the class $[\hat{f}|\partial D]_{F_{l-1}}$ is in the normal subgroup gen-

erated by $[\hat{f}_0|\partial D]_{F_{l-1}}$ and $[\hat{f}_1|\partial D]_{F_{l-1}}$. By iterating this argument, it follows that $[\hat{f}|\partial D]$ is in the normal subgroup generated by the set of *based* simple curves corresponding to the leaves. Where a based curve we mean when the simple curve is connected to the fixed basepoint by a simple path that is a lift of some q_x for an intersection point x. This finishes the argument for the stage $l-1$ of the tower.

Let $f_{L_1}, \ldots, f_{L_s}$ be the maps corresponding to the leaves of the abstract elementary cut-and-paste tree for $\hat{f}$. The restriction of each of these maps to the boundary is a simple curve that bounds an embedded disk in V_{l-1}. These disks then are projected by p_{l-1} to the level $l-2$, the result is singular disks $\hat{f}_{L_i} = p_{l-1}f_{L_i}, i = 1, \ldots, s$. These maps are treated just like $\hat{f}$ in level $l-1$. So each will generate its own abstract cut-and-paste tree. Note that the self-intersection points of $\hat{f}_{L_i}|\partial D$ are distinct from those which appeared in level $l-1$. Since $\phi_l(a) = [\hat{f}|\partial D]_{F_{l-1}}$ is in the normal subgroup of $\pi_1(F_{l-1})$ generated by the based leaf curves, we can apply the homomorphism $\phi_{l-1} = (p_{l-1}|F_{l-1})_* : \pi_1(F_{l-1}) \to \pi_1(F_{l-2})$ to all the classes and deduce that $(\phi_{l-1}\phi_l)(a)$ is in the normal subgroup generated by the based boundary curves of $\hat{f}_{L_1}, \ldots, \hat{f}_{L_s}$ in $\pi_1(F_{l-2})$, where the basepoint is again a lift of x_0. Note that for each of the curves $\hat{f}_{L_j}$, the image coincides with a part of the image of the lift c_{l-2} of c other than for small neighborhoods of intersection points that are resolved in higher levels.

Each of the maps $\hat{f}_{L_j}$ generates its own abstract elementary cut-and-paste tree and its class is contained in the normal subgroup generated by the based curves of the leaves in that tree. It follows that $(\phi_{l-1}\phi_l)_*(a)$ is in the normal subgroup generated by all the leaves in all of the abstract cut-and-paste trees of various $\hat{f}_{L_i}$. Therefore, leaves of the level-$(l-1)$ abstract elementary cut-and-paste tree correspond to roots of one such tree of level $l-2$. We continue this process down the tower of covering spaces. Any leaf of a level-k abstract cut-and-paste tree corresponds to the root of a level-$(k-1)$ tree, $k = l-1, \ldots, 1$. We delete the leaf and put the root in its place, connecting it to the leaf's parent. Note that the merged node has level one less than that of its parent with respect to the tower. After doing this for all these leaf-root pairs, and repeating for all levels, we obtain a tree that includes all the abstract elementary cut-and-paste trees as subtrees. We call the resulting tree the *abstract cut-and-paste tree* for the singular disk which is the projection of f_a, i.e., $g_a = p_1 p_2 \cdots p_l f_a : (D, \partial D) \to (M, \partial M)$.

On each path from a leaf to the root of the abstract cut-and-paste tree for g_a, each pair of self-intersection points of the curve c_a appears at most once. The self-intersection points of the curves c_{a_j} for various j form a partition of a subset of self-intersection points of c. This is because the lift to distinct points in c_l. It follows that the sum of the numbers of leaves in all of the non-trivial abstract cut-and-paste trees for various g_{a_j} is at most $3(m+n)2^{m/2}$. Since the image of c_l is a graph over at most $m+n$ vertices on the sphere on top of the tower, hence it has at most $3(m+n)$ edges.

In the above procedure, after processing the bottom level of the tower of covering spaces, the class

$$(\phi_1\phi_2\cdots\phi_{l-1})_*(a)$$

is in the normal subgroup generated by based simple curves. Since $[c_l]_{F_l}$ can be written as a product of a_j and their inverses it follows that $[c]_{F_0}$ is in the normal subgroup generated by all the at most $3(m+n)2^{m/2}$ based simple curves appearing in all the leaves of all the abstract cut-and-paste trees g_{a_j}. This completes the proof of Theorem 4.1. □

Lemma 4.4. *Let $s : S^1 \to \partial M$ be a special curve on ∂M. Then there exists a set of parameters $(\Pi, \Delta_1, \ldots, \Delta_k), k = |\Pi|$, for which a labeled cut-and-paste tree is defined by the procedure of Section 3.2. Moreover, the class of the tree expresses $[s]_\partial$ correctly as a product of the based curves g'_L and the based conjugator curves r'_P of the tree. In addition, the curves g_L are all null-homotopic.*

Proof. The existence of a cut-and-paste tree without labels is evident from the proof of Theorem 4.1. Let $\mathcal{T}$ be an abstract cut-and-paste tree associated to a tower of covering spaces for which s lifts to a simple curve on top of the tower as defined above. To each non-leaf node of the tree is associated a pair of self-intersection points, namely the pair connected

by a double arc of some lift of s. Recall that to the node determined by a path P of $\mathcal{T}$ is associated a map $f_P : (D, \partial D) \to (V_{l(P)}, \partial F_{l(P)})$ for some integer $l(P) \geqslant 0$, which is the level of the covering space in which the pair of self-intersection points associated to P are resolved. Moreover, there exists a conjugator curve associated to the node determined by P that we denote by $\rho_P : S^1 \to c_{l(P)}(S^1)$. Let $P0, P1$ be the paths for the children of the node if $f_P|\partial D$ is not simple. Then, two cases can happen. The first case is where $l(P0) = l(P1) = l(P)$. Then we know that $[f_P|\partial D]_{F_{l(P)}}$ properly based is written by one of the formulas 1 or 3 in terms of based curves $[f_{P0}|\partial D]_{F_{l(P)}}$, $[f_{P1}|\partial D]_{F_{l(P)}}$, $[\rho_P]_{F_{l(P)}}$ and their inverses. In the second case, we have $l(P0) = l(P1) = l(P) - 1$. Then again we know that $[f_P|\partial D]_{F_{l(P)-1}}$ properly based is defined by one of the formulas 1 or 3 in terms of $[f_{P0}|\partial D]$, $[f_{P1}|\partial D]$, $[\rho_P]$ and their inverses. Whatever the case may be, let $p_P = p_0 p_1 \cdots p_{l(P)-1} p_{l(P)} : (V_{l(P)}, F_{l(P)}) \to (V_0, F_0)$. Then $g_P = p_P(f_P|\partial D) : S^1 \to \partial M$ is a subcurve of c, the same is true for $r_P = p_P(\rho_P)$.

The tower of covering spaces defines a set of parameters $(\Pi, \Delta_1, \cdots, \Delta_k)$. The partition is given by the level of the covering space on which a pair of self-intersections appear, when projecting the curve from top of the tower to the lower levels. The pairing in the same level is given obviously by the singular arc joining the two self-intersection points.

We now show that the g_P defined here for an abstract cut-and-paste tree coincide with the g_P in the definition of a cut-and-paste tree, defined for the parameters given by the tower of covering spaces. The root of the tree satisfies $g_0 = s$. We show that if for a node defined by the path P the curve g_P has been computed, then we can compute g_{Pi} and r_{Pi}, $i = 0, 1$, with the procedure of Section 3.2. Take the case $i = 0$. Then f_{P0} is defined from f_P by a cut-and-paste process of type 0. Therefore, $f_{P0}|\partial D$ is obtained from $f_P|\partial D$ by the cut-and-paste of type 0. Recall that this is the case where resolving the pair of self-intersections of $f_P|\partial D$ associated to the node results in three curves. Now $g_{P0} = p_{P0}(f_{P0}|\partial D) = p_P(f_{P0}|\partial D)$ or $g_{P0} = p_{P0}(f_{P0}|\partial D)$ where $p_{P0}p_{l(P)} = p_P$, depending on whether P and its children belong to the same level or not. Note that in the second case $f_{P0} = p_{l(P)}(f_P)_0$, and $(f_P)_0$ is a simple curve[2]. In both cases, g_{P0} equals $g_P = p_P(f_P|\partial D)$ everywhere other than a small neighborhood of the self-intersection points resolved at the node P. Moreover, p_P is a local homeomorphism. It is then clear that g_{P0} is obtained from g_P by doing the same resolution on the pair of self-intersection points. This is the same procedure for defining labels of a node in the cut-and-paste tree. The same argument applies to g_{P1}. For $i = 0, 1$, the curve r_{Pi} is uniquely determined from g_P and g_{Pi}. □

D The Case of the Torus Boundary

We apply theorem 4.1 to perhaps the simplest nontrivial case. This is when the boundary component of M containing the curve is a torus. Manifolds with torus boundaries play an important role in the theory of 3-manifolds. For instance, the parts of a JSJ-decomposition have torus boundary [15].

Theorem 1.3. *The following problem is in NP. Given a manifold M and an arbitrary PL curve c on a boundary component of M that is a torus, decide if c is contractible in M.*

Proof. We first claim the following: The number of distinct homotopy classes of subcurves of c is $O(m + n)^2$. To see this, let H be the graph that is the image of c. Let T be a spanning tree of H, and let $H' := H/T$ be obtained from H by contracting T. Each subcurve of c becomes a walk without repeated edges in H', which is a one-vertex graph embedded on the torus. So the loops of H' fall into at most three distinct homotopy classes, of the form a, b, and ab (in multiplicative notation). Moreover, the number of edges in H' is $O(m + n)$. Since the fundamental group of the torus is Abelian, each subcurve of c is homotopic to a loop of the form $a^p b^q$, where $|p|$ and $|q|$ are at most $2(m + n)$. The claim follows.

Next, we show that there exists a certificate of polynomial length for contractibility of the curve c.

[2] See the notation for the cut-and-paste operation.

By Theorem 4.1, if $[c] = 1$ then $[c]_\partial$ is in the normal subgroup group generated by $[g'_1]_\partial, \ldots, [g'_N]_\partial$. By the above argument, polynomially many distinct $[g'_j]_\partial$ exist. A certificate then consists of a certificate for contractibility of polynomially many g'_j as in [1, 12], together with a formula expressing $[c]_\partial$ as a linear combination of the classes of the $[g'_j]_\partial$ in the certificate. The coefficients are at most exponential in size of the curve and the triangulation as they are upper-bounded by the size of the product expression of the class of a cut-and-paste tree. Hence each coefficient can be described by polynomially many bits.

Now assume given a certificate as defined. We can check the contractibility of each g_j appearing in the certificate by the algorithm of Agol et al. [1]. We can also then check if $[c]_\partial$ equals the combination given in the certificate in polynomial time. Hence we can check if the curve is contractible in polynomial time given a certificate of polynomial length. $\square$