Algorithms for Contractibility of Compressed Curves on 3-Manifold Boundaries

Erin Wolf Chambers
St. Louis University, MO, USA

Francis Lazarus
G-SCOP, CNRS, UGA, Grenoble, France

Arnaud de Mesmay
LIGM, CNRS, Université Gustave Eiffel, ESIEE Paris, 77454 Marne-la-Vallée, France

Salman Parsa
St. Louis University, MO, USA

Abstract

In this paper we prove that the problem of deciding contractibility of an arbitrary closed curve on the boundary of a 3-manifold is in **NP**. We emphasize that the manifold and the curve are both inputs to the problem. Moreover, our algorithm also works if the curve is given as a compressed word. Previously, such an algorithm was known for simple (non-compressed) curves, and, in very limited cases, for curves with self-intersections. Furthermore, our algorithm is fixed-parameter tractable in the complexity of the input 3-manifold.

As part of our proof, we obtain new polynomial-time algorithms for compressed curves on surfaces, which we believe are of independent interest. We provide a polynomial-time algorithm which, given an orientable surface and a compressed loop on the surface, computes a canonical form for the loop as a compressed word. In particular, contractibility of compressed curves on surfaces can be decided in polynomial time; prior published work considered only constant genus surfaces. More generally, we solve the following normal subgroup membership problem in polynomial time: given an arbitrary orientable surface, a compressed closed curve γ, and a collection of disjoint normal curves Δ, there is a polynomial-time algorithm to decide if γ lies in the normal subgroup generated by components of Δ in the fundamental group of the surface after attaching the curves to a basepoint.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Mathematics of computing → Graphs and surfaces; Theory of computation → Data compression

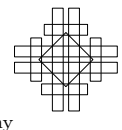
Keywords and phrases 3-manifolds, surfaces, low-dimensional topology, contractibility, compressed curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.23

Related Version *Full Version*: `https://arxiv.org/pdf/2012.02352.pdf`

Funding *Erin Wolf Chambers*: This work was funded in part by the National Science Foundation through grants CCF-1614562, CCF-1907612 and DBI-1759807.
Francis Lazarus: This author is partially supported by the French ANR projects GATO (ANR-16-CE40-0009-01) and MINMAX (ANR-19-CE40-0014) and the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.
Arnaud de Mesmay: This author is partially supported by the French ANR projects ANR-17-CE40-0033 (SoS), ANR-16-CE40-0009-01 (GATO) and ANR-19-CE40-0014 (MINMAX).
Salman Parsa: This work was funded in part by the National Science Foundation through grant CCF-1614562 as well as funding from the SLU Research Institute.

Acknowledgements The authors would like to thank Mark Bell, Ben Burton, and Jeff Erickson for helpful discussions. We also thank an anonymous referee of [11] for suggesting the use of a maximal compression body as a simple exponential-time algorithm for deciding contractibility of arbitrary (non-compressed) curves on the boundary of a 3-manifold.



1 Introduction

In a topological space X, a closed curve $c : \mathbb{S}^1 \to X$ is *contractible* if the map c extends to a map of the standard disk, $f : D \to X$, $f|_{\partial D} = c$; such a curve can be continuously deformed to a point. Dating back to Dehn [12] more than a century ago, the problem of testing contractibility has been an important focus in the development of efficient algorithms in computational topology and group theory. Such problems are now well understood in two dimensions, with recent works of the second author and Rivaud [29] and Erickson and Whittlesey [17] providing linear-time algorithms to test contractibility and free homotopy of curves on surfaces; Despré and the second author [13] also developed efficient algorithms for the closely related problem of computing the geometric intersection number of curves. On the other hand, in higher dimensions, homotopy problems quickly become undecidable: testing contractibility of curves is already undecidable on 2-dimensional simplicial complexes and in 4-manifolds (see Stillwell [39, Chapter 8]).

The aim of this paper is to further improve our understanding of the intermediate 3-dimensional case. In a 3-manifold, testing whether a curve is contractible is known to be decidable, but the best known algorithms are intricate and rely on the proof of the Geometrization Conjecture. We refer to the survey of Aschenbrenner, Friedl and Wilton [2]. When the input manifold is a knot complement, this problem contains as a special case the famous Unknot Recognition problem, which asks whether an input knot in $\mathbb{R}^3$ is trivial. That problem is now known to be in **NP** [19] and co-**NP** [26]. Recently, Colin de Verdière and the fourth author [10] initiated the study of a problem in between those two, which is testing the contractibility of a closed curve γ which lies on the boundary of a triangulated 3-manifold M, and they provided an algorithm to solve this problem in time $2^{O(|M|+|\gamma|)^2}$). They asked whether the problem lies in the complexity class **NP** and proved that it holds in two cases: when the number of self-intersections of γ is at most logarithmic, and when the boundary surface is a torus. We remark that the existence of an algorithm to test contractibility of a closed curve on the boundary actually follows from the earlier results of Waldhausen [40], and an exponential-time algorithm can be derived from the standard 3-manifold literature, as we will explain later in this paper. However, the algorithm of [10, 11] is simpler in the sense that it relies only on the proof of the Loop Theorem.

Our main result is that the problem of contractibility when the curve is on the boundary of the 3-manifold is in **NP**. Furthermore, the algorithm that we provide has two additional features. First, it is fixed parameter tractable in the input manifold: for a given 3-manifold M, after a preprocessing taking exponential time in $|M|$, we can solve any contractibility test for curves on the boundary in polynomial time. Second, our algorithm also works if the input curve is *compressed* via a straight-line program, in particular, it can be exponentially longer than its encoding size (we defer to Section 2 for the precise definitions).

▶ **Theorem 1.** *Let a triangulated 3-manifold M and a curve γ on the boundary of M be given as the input. The problem of deciding whether γ is contractible in M is in **NP**, and there is a deterministic algorithm that solves the problem in time $2^{O(|M|^3)}poly(|\gamma|)$. This is also the case if γ is given as a compressed word, i.e., a straight-line program.*

Our proof of Theorem 1 will reduce the problem to another problem involving only curves on surfaces. However, there is an important subtlety. Many algorithms in 3-dimensional topology involve an exponential blow-up in the size of the objects under consideration. This is the case for example for Unknot Recognition, where the classical algorithm [19] looks for a disk spanning the knot, but this disk might be exponentially large [20]. In order to

get **NP** membership, this exponential blow-up is controlled using a compressed[1] system of coordinates called *normal coordinates* (see Section 2 for definitions). Our approach follows a similar scheme and suffers from a similar blow-up, which forces us to deal with normal curves. Precisely, we show that the problem of contractibility of an arbitrary curve on the boundary of a 3-manifold reduces to the following problem. Given a family of disjoint normal closed curves Δ on a triangulated surface S, and a curve γ on S, decide if γ lies in the normal subgroup of the fundamental group determined by the curves of Δ (appropriately connected to the basepoint). This is equivalent to deciding if the curve γ is contractible in the space resulting from S by gluing a disk to each component of Δ. We call this problem the *disjoint normal subgroup membership* problem, and we provide an algorithm to solve it in polynomial time, despite the highly compressed nature of Δ and γ.

▶ **Theorem 2.** *Let an orientable triangulated surface S, a closed normal multi-curve Δ, and a compressed curve γ be given as the input. There is a polynomial-time algorithm for deciding the disjoint normal subgroup membership problem for γ and Δ on S.*

To solve the disjoint normal subgroup membership problem, one of our technical tools is a polynomial-time algorithm to test triviality of a compressed closed curve on a surface, which might be of independent interest. In fact, we compute a canonical form for such a curve, such that there is a unique canonical representative in each based homotopy class.

▶ **Theorem 3.** *Let an orientable triangulated surface S and a compressed loop γ on S described by a straight-line program be given. Then one can transform γ, in polynomial time, into a canonical form in its based homotopy class, given as a compressed word. In particular, we can test in polynomial time whether γ is contractible.*

We emphasize that in Theorem 3 the surface S is part of the input. The analogous theorem, if we assume that surface S is not part of the input, has been known in a much broader context, as we discuss in the next section.

1.1 Related work

Algorithms in 3-manifold topology. There is now a large body of research on the computational complexity of problems in 3-manifold topology and knot theory. To paint a picture in broad strokes, most topological problems are now known to be decidable, using a combination of geometric, algebraic and topological tools and the complexity of the best known algorithms range from exponential [19, 1] to quite galactic, see for example Kuperberg [25] for the homeomorphism problem. In contrast, only very few complexity lower bounds are known [1, 27, 9] and for many problems, including ours, it is open whether a polynomial-time algorithm exists. One particular tool that our work relies on is *normal surface theory*, which provides a powerful framework to enumerate and analyze the topologically relevant embedded surfaces in a 3-manifold. Actually, tools from normal surface theory [38] provide an off-the-shelf algorithm proving that deciding contractibility of a *simple* curve on the boundary of a 3-manifold is in **NP**, as explained in [10, Section 3]. However, normal surface theory is ill-adapted to study surfaces that are not embedded [5], hence the more technical path that we follow in this paper. We refer to Lackenby [28] for a recent and extensive survey on algorithms in 3-manifold topology.

[1] There are two different forms of compression going on in this paper: normal coordinates and straight line programs. We keep the name *compressed* for the latter, while a curve or a multicurve encoded with normal coordinates will be simply called a *normal (multi-)curve*.

Geometric and combinatorial group theory. The problem of deciding the contractibility of a curve in a topological space, which is given for example as a finite simplicial complex, can be equivalently rephrased as a problem of testing the triviality of a word in a finitely presented group, the fundamental group of the space. In the case of 2- and 3-manifolds, the geometry or topology of the underlying manifold has a strong impact on the properties of the group, and there is a vast amount of literature on this interaction, see for example [8]. We refer to Aschenbrenner, Friedl and Wilton [2, 3] for the case of 3-manifold groups. Of particular interest for the word problem is the notion of a *Dehn function*, which upper bounds the area of a disk certifying the triviality of a word, and thus controls the complexity of a brute-force algorithm to solve the word problem. For 3-manifold groups, this function can be exponential [7, Theorem 8.1.3], making such an approach at least exponentially worse than ours. When the Dehn function of a group presentation is polynomial (as is the case, for instance, for the fundamental group of a compression body), it can be used to prove that the word problem is in **NP**. However, the word problem is defined for a fixed presentation. In our problem, the group presentation itself is part of the input, and we cannot use the bound on the Dehn function directly.

Algorithms for compressed curves and surfaces. As we hinted at in the introduction, 3-dimensional algorithms naturally tend to involve exponentially large objects, which are generally compressed using normal coordinates. This incurs a need for efficient algorithms dealing with normal curves and surfaces, even for the most basic tasks. For example, it is not easy to test whether a curve described by normal coordinates is connected. By now, there are many known techniques to handle topological problems on compressed curves, see for example Agol, Hass and Thurston [1], Schaefer, Sedgwick and Stefankovič [35, 36], Erickson and Nayyeri [16], Bell [4] or Dynnikov [14] and Dynnikov and Wiest [15]. None of these techniques seem to directly solve our disjoint normal subgroup membership problem, but we rely extensively on the work of Erickson and Nayyeri [16] as a subroutine (see Section 4). However, normal coordinates do not describe curves with self-intersections, which are our main object of interest in this paper. This is why we rely on a different compression model in the form of *straight-line programs*. The use of such programs in geometric group theory is not new, and in particular there is a sketch of an algorithm to test triviality of compressed curves on a surface of genus 3 in an appendix of Schleimer [37, Appendix A]. His approach seems to be readily generalizable to higher genus, but the dependency on the genus is not made explicit. More recently, Holt, Lohrey and Schleimer [23] provided a polynomial-time algorithm to test triviality of compressed words in *hyperbolic groups*, of which (most) surface groups are a subclass. The difference with our Theorem 3 is that in their result, the group presentation is not part of the input. While both approaches could plausibly be used in our setting, we rely on different tools to provide a complete proof of Theorem 3: our approach treats the surface as an input and works for any genus, and seamlessly generalizes to the case of wedges of surfaces that we also need.

1.2 Summary of our techniques

The standard methods of normal surface theory allow us to reduce our main contractibility problem to the case where the input manifold belongs to a particular class of manifolds called *compression bodies*, see Section 3. The fundamental group of a compression body is roughly a free product of surface groups, and thus one can rely on known algorithms for surfaces [17, 29] to solve the contractibility problem there. However, due to the exponential size of normal surfaces, this would only yield an exponential time algorithm. In order to

improve on this, we identify the precise problem that we need to solve to be the disjoint normal subgroup membership problem and set out on a quest to solve it efficiently. While an **NP** algorithm would suffice for our main result, we will develop a polynomial-time algorithm.

The disjoint normal subgroup membership problem is made delicate by the compressed nature of the curves in Δ. If they were given as disjoint curves, say, on the 1-skeleton of the surface, we could glue a disk on each of them, and observe that the resulting complex is homotopically equivalent to a wedge of surfaces[2], allowing us to reduce the problem to the triviality of a curve in a surface (see Section 5 for a description of the homotopy equivalence). In order to do so despite the compression, we compute in Section 4 a *retriangulation* of the surface, so that the curves in Δ are only polynomially long in the new triangulation. This is done by tracing the normal curve and building the *street complex* which was specifically designed by Erickson and Nayyeri to handle normal curves on surfaces in polynomial time. We then track how the street complex interacts with our input curve γ. This operation comes at a cost: even if the input curve γ was not compressed, it may become exponentially long after the retriangulation. Since it may be not simple, we rely on straight-line programs instead of normal curves to encode it. At this stage, we note that it might as well have been a compressed straight-line program from the start.

Finally, we want to test the triviality of a compressed curve in a wedge of surfaces. The fundamental group of this wedge is a free product of fundamental groups of surfaces, and thus the crux of the problem is to solve it in a single surface. While there are known techniques to test the contractibility of a curve on a surface, based on local simplification rules [17, 29], the compressed nature of our curves is once again a stumbling block here, as we need to be careful to never apply an exponential number of such simplification rules. The solution sketched by Schleimer [37, Appendix A] to that issue is to introduce a family of *well-tempered paths*, which are carefully designed to not necessitate such exponential simplifications. We use a different approach: we rely on the standard tools developed for the non-compressed case, namely quad systems and turn sequences, and detect exponential simplification and realize them all at once in polynomial time. This relies on recent insights on the structure of these quad systems [13] [30, Section 4.3].

Most proofs are omitted due to space constraints; complete details are provided in the full version, which is appended after this extended abstract.

2 Background

We begin by recalling some key definitions and results, although we assume that the reader is familiar with basic algebraic topology such as homotopy and fundamental groups; we refer to Hatcher [21] or Stillwell [39] for more detailed definitions.

2.1 3-Dimensional manifolds

A 3-manifold is a topological space that is locally homeomorphic to a 3-dimensional ball or a 3-dimensional half-ball. The points of the latter type form the *boundary* of the 3-manifold, which is always a surface. In this paper, 3-manifolds are always assumed to be triangulated, and we use common and a less restrictive definition than simplicial complexes: a *triangulation* of a 3-manifold M is a collection of n abstract tetrahedra, along with a collection of gluing

[2] A *wedge* of a family topological spaces is the space obtained after attaching them all to a single common point. Actually, there are also circle summands here – we do not mention them in this outline to keep the discussion light.

pairs which identify some of the $4n$ boundary triangles in pairs. In particular, we allow two faces of the same tetrahedron to be identified. If we add the further restriction that the neighborhood of each vertex is a 3-ball or a half 3-ball and no edge gets glued to itself with the opposite orientation, then the resulting space is always a 3-manifold [34].

This paper makes heavy use of normal surface theory; see the full version for more definitions and relevant background on this topic.

2.2 Curve and surface representations

All the surfaces in this article are closed, i.e., without boundary. We shall often represent a surface by a graph cellularly embedded in that surface. This embedding is encoded as a *combinatorial* surface thanks to a rotation system over the edges of the graph [33]. Equivalently, one can define a combinatorial surface as a gluing of polygons whose sides are pairwise identified. When all the polygons are triangles, we speak of a *triangulation*, or a *triangulated surface*. Note that we allow a triangle to have two sides identified after the gluing or two triangles to share two vertices but no edge, etc.

The *complexity* of a surface is the number of cells (vertices, edges and faces) of the complex induced by the graph embedding. A curve on such a surface can be represented or encoded in a variety of ways. For instance, any curve is homotopic to a curve defined by a walk on the 1-skeleton of the complex. The complexity of the curve is the number of edges in the walk counted with repetition. One can also consider curves in general position avoiding the vertices of the graph and cutting its edges transversely as in the next section, in which case we say the complexity of a curve is its number of intersection points with the graph.

We next outline two different notions of compressed curve representations. To repeat the footnote of warning in the introduction: normal coordinates and straight line programs are two different methods for compressed representations, both of which we use in this paper. We keep the name *compressed* for straight line programs, while a curve or a multicurve encoded with normal coordinates will be simply called a *normal (multi-)curve*.

Normal curves. Let S be a triangulated surface. A *multi-curve* is a disjoint family of curves embedded on a S. A multi-curve c is *normal* if it is in general position with respect to the triangulation of S and if every maximal arc of c in a triangle has its endpoints on distinct sides. In particular, no component of a normal curve is contained in the interior of a single triangle. A normal curve may have three types of arcs in a triangle, one for each pair of sides. Counting the number of arcs of c of each type in each of the t triangles of the triangulation, we get $3t$ numbers called the *normal coordinates* of c. A *normal isotopy* of S is an isotopy that leaves each cell of the triangulation invariant. Up to a normal isotopy, c is completely determined by its normal coordinates. It is thus possible to encode a multi-curve with exponentially many arcs by storing its normal coordinates, which then have polynomial bit-complexity. A normal multi-curve is *reduced* if no two of its components are normally isotopic.

Straight-line programs. Another method of compressing curves is to think of a curve as a word over an alphabet. In this encoding, a letter of the alphabet corresponds to a directed edge on the surface and juxtaposition of letters corresponds to concatenation of paths. Therefore, we can consider curves as abstract words and to represent them using compressed representations of abstract words. The length of a word w, denoted by $|w|$, is the number of letters in it.

A *straight-line program* is a four-tuple $\mathbb{A} = \langle \mathcal{L}, \mathcal{A}, A_n, \mathcal{P} \rangle$ where:

- $\mathcal{L}$ is a finite alphabet of *terminal* characters,
- $\mathcal{A}$ is a disjoint alphabet of *nonterminal* characters,
- $A_n \in \mathcal{A}$ is the *root*, and
- $\mathcal{P} = \{A_i \to W_i\}$ is a sequence of *production rules*, where W_i is a word in $(\mathcal{L} \cup \mathcal{A})^*$ containing only non-terminals A_j for $j < i$.

A straight-line program is in *Chomsky normal form* if every production rule either has two non-terminals or a single terminal on the right. Every straight-line program can be put in polynomial time into Chomsky normal form by adding intermediate non-terminals, and thus we will always assume that a straight-line program is in Chomsky normal form. We denote by $w(A)$ the word encoded by a non-terminal character A, or sometimes simply by A when there is no confusion. Its length is denoted by $|A|$. We will always use a terminal alphabet $\mathcal{L}$ equipped with an involution $a \mapsto \bar{a}$. The *reversal* of a word w is the word obtained by changing its letter a by its reverse $\bar{a}$ and reversing the order of the letters.

Composition systems are straight-line programs of a more advanced type, where productions of the form $P \to A[i:j]B[k:l]$ are allowed, and the meaning is that $A[i:j]$ represents the subword between the $i+1$th and jth letter (both included, starting the numbering at 1) of the word that A encodes. We take the convention that negative indices count from the end of the word, and that $A[i:]$, $A[i]$ and $A[:j]$ are shorthand respectively for $A[i:-1]$, $A[i:i+1]$ and $A[0:j]$. While composition systems might seem more powerful than straight-line programs, a theorem of Hagenah [18] says that given any composition system, one can compute in polynomial time an equivalent straight-line program. Henceforth, we will slightly abuse language and freely use the $[\cdot:\cdot]$ construct in our straight-line programs.

The following theorem summarizes the algorithms on straight-line programs that we will rely on, see Schleimer [37] or Lohrey [32].

▶ **Lemma 4.** *Let $\mathbb{A}$ and $\mathbb{X}$ be two straight-line programs, i be an integer and e be a letter in the terminal alphabet of $\mathbb{A}$. Then one can, in polynomial-time,*

- *compute the length of $\mathbb{A}$,*
- *output the letter $\mathbb{A}[i]$,*
- *find the greatest j so that $\mathbb{A}[j] = e$,*
- *compute a straight-line program $\overline{\mathbb{A}}$ for the reversal word $\overline{w(\mathbb{A})}$,*
- *decide whether $w(\mathbb{A}) = w(\mathbb{X})$,*
- *find the biggest k such that $\mathbb{A}[:k] = \mathbb{X}[:k]$.*

For a given 2-complex K (in particular if K is a combinatorial surface), a straight-line program with terminal alphabet $\mathcal{L}$ the set of directed edges of K can encode any closed curve on K. We call such an encoding a *compressed curve* or *walk*. Simple operations on K can seamlessly be done while updating a compressed curve appropriately to obtain a homotopic compressed curve in the modified complex. For example, contracting an edge e of K simply boils down to adding a production rule $e \to \varepsilon$, where ε is the empty word. Likewise, deleting an edge e bounding a face $\bar{e}p$, where p is the complementary path in that face, amounts to replacing each occurrence of e by a p via a new production rule $e \to p$. When no encoding is specified for a closed (multi-)curve, then it is simply encoded as a (multi-)walk on the one-skeleton of the underlying surface or complex.

3 From contractibility to normal subgroup membership problem

The following proposition reduces the contractibility problem for closed curves on the boundary of an orientable 3-manifold M to the normal subgroup membership problem for a boundary surface of M, where the multi-curve Δ is a reduced normal curve with polynomial complexity. We note that this reduction is the only place where the algorithm is non-deterministic, the rest of our algorithms run in deterministic polynomial time.

▶ **Proposition 5.** *Let M be an orientable triangulated 3-manifold with boundary. There exists a reduced normal multi-curve Δ on ∂M with normal coordinates of linear bit-size (with respect to $|M|$), so that for any curve γ on ∂M, γ is contractible in M if and only if γ is contained in the normal subgroup N of $\pi_1(\partial M)$ generated by the homotopy classes of the components of Δ.*

Further, given M, Δ, and a cubic-sized certificate (in $|M|$), there is a polynomial-time algorithm to verify that Δ has the property that all curves in N are contractible in M.

Our proof begins by crushing M to get an irreducible manifold [6] with the same triangulated boundary, and then relies on the fact that normal surface theory [24] proves the existence of a complete family of *compression disks*, whose boundary is the reduced multi-curve Δ. Its properties follow from the Loop Theorem [22, Theorem 4.2]. We refer to the full version for details.

Note that we did not strive to optimize the size of the certificate in this proposition, since it does not matter for our application, so further improvements may be possible. In particular, using enumeration techniques for vertex surfaces [19, Section 6], we suspect it can be shown to be quadratic rather than cubic.

▶ **Proposition 6.** *The problem of contractibility of a compressed curve on the boundary of an arbitrary 3-manifold reduces, in polynomial time, to the problem of contractibility of a compressed curve on the boundary of an orientable 3-manifold.*

Relying on Proposition 6, for the rest of the paper we restrict our attention to orientable surfaces.

Hardness of the general normal subgroup membership problem. If the curves Δ in the normal subgroup membership problem are not disjoint then this problem is much harder, and possibly undecidable, even if they have few edges, see the full version for details.

4 Retriangulation

Given a reduced normal multi-curve Δ on a triangulated 2-manifold, in this section we present a way to compute a new triangulation of polynomial complexity, in which the curves of Δ lie in the 1-skeleton and are only polynomially long.

▶ **Proposition 7.** *Let a simplicial complex K triangulating a surface S, a reduced normal multi-curve Δ and a compressed closed walk γ on the 1-skeleton of K be given as input. The size of the input is the summation of the complexities of the curves γ, Δ and complexity of K. There is an algorithm that in polynomial time computes a new simplicial complex K' triangulating the same surface S, a multi-curve Δ', and a walk γ' on the 1-skeleton of K' given as a composition system, such that:*

- *there is a homeomorphism $f : |K| \to |K'|$, such that the images of Δ under f coincides with Δ',*
- *γ' is homotopic to $f(\gamma)$,*
- *the multi-curve Δ' is an embedded multi-curve on the 1-skeleton of K'.*

Consequently, the homotopy class of γ' belongs to the normal subgroup generated by the components of Δ' in $|K'|$ if and only if the homotopy class of γ belongs to the normal subgroup generated by the components of Δ in $|K|$.

Note that in this proposition, the complexities of the inputs γ and Δ are compressed.

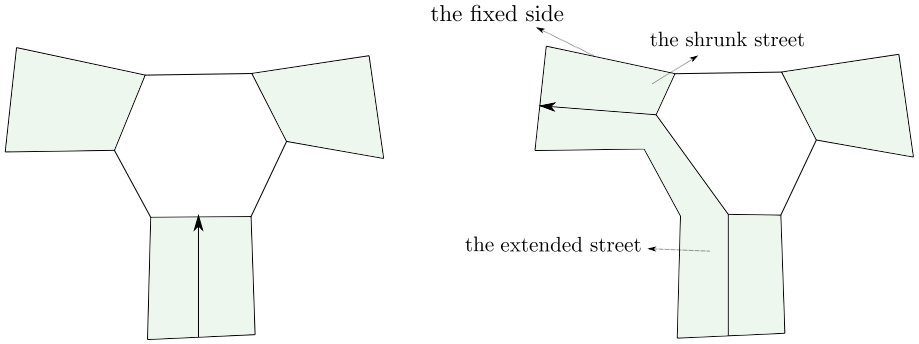


Figure 1 A left turn in the street complex; the arrows depict segments of the edge e.

Sketch of proof. We briefly outline our approach, and again refer the reader to the full version for details. Our algorithm relies heavily on the street complex introduced by Erickson and Nayyeri [16] (see also Bell [4]); we briefly recall its structure here. The overlay of Δ with K cuts the triangles of K into smaller pieces. We abuse notation and call quadrilaterals the pieces bounded by exactly two subedges of K, though they can actually be degenerate triangles. Merging the quadrilaterals whenever they share an edge segment of K results in the *street complex*. The maximal sequences of merged quadrilaterals are called *streets*, and the remaining faces are *junctions*. Quite surprisingly, the size of the street complex is bounded by a linear function of the complexity of K [16, Lem. 2.3], independent of Δ. Note that the curves in Δ appears in the 1-skeleton of this complex at the street boundaries. The complex K' in the proposition is essentially this street complex, further triangulated, and Δ' is the image of Δ in K'.

In [16] the algorithm to construct the street complex works inductively by *tracing* the curves in Δ. Intuitively, the tracing algorithm follows each curve from an arbitrary chosen point and extends one of the incident streets as the curve traverses the current streets and junctions. See Figure 1. An essential step in our algorithm is to somehow reverse the roles of K and Δ to compute for each edge e of K a composition system $\mathbb{E}$ encoding its intersections with Δ. The terminals of this system correspond to the segments of e cut by Δ. We compute $\mathbb{E}$ inductively in parallel to the street complex construction, following the tracing algorithm of Erickson and Nayyeri. Each terminal of $\mathbb{E}$ is contained in some street and can be homotoped to a path in the boundary of this street. We replace the terminal by this path to obtain a composition system encoding a curve homotopic to e, while contained in K'. At a very final step, we replace the terminal edges of the straight line program for γ by the corresponding composition systems. Our algorithm can be executed in polynomial time, by adapting the analysis of Erickson and Nayyeri [16]. ◀

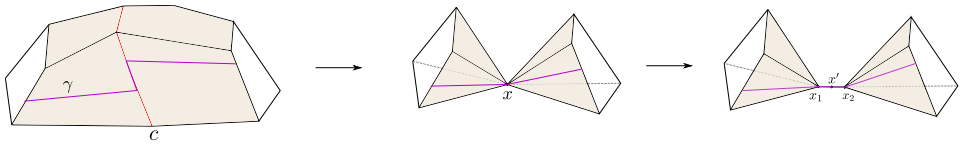


Figure 2 Contraction of a component c of Δ (shown on the left) results in a new complex K' (shown on the right).

5 Capping off

In this section, we show that the complex obtained by gluing disks on a family of disjoint curves on a surface is homotopy equivalent to a wedge of surfaces and circles. We provide a polynomial-time algorithm to compute the resulting wedge, while also tracking what happens to a compressed curve on the original surface.

▶ **Proposition 8.** *Let L be simplicial complex triangulating a surface S of genus ≥ 1, Δ be a family of disjoint embedded closed curves in the 1-skeleton of L and γ be a compressed closed walk on the 1-skeleton of L. The size of the input is the summation of the complexities of L, γ and the number of edges of Δ. We can compute in polynomial time a simplicial complex K which is a wedge of surfaces and circles, and a compressed walk w on K, so that:*

- *K is homotopy equivalent to the complex obtained by gluing disks on each component of Δ,*
- *w is homotopic to a trivial walk in K if and only if γ belongs to the normal subgroup determined by the curves in Δ.*

The proof is deferred to the full version, but the key idea is illustrated in Figure 2: we repeatedly pinch the curves in Δ to a single point, extend this point to a small path and contract a spanning tree in the rest to have a single vertex, and thus a wedge.

6 Triviality for compressed words in free products of surface groups

In this section, we prove the following theorem:

▶ **Theorem 9.** *Let K be a wedge of combinatorial surfaces and circles and w be a compressed walk on K. Then one can compute in polynomial time a canonical form for the homotopy class of w. In particular, we can test in polynomial time whether w is trivial.*

We emphasize that in this theorem, we consider w as a walk with fixed endpoints, and thus we are considering based homotopies (as opposed to free homotopies).

The proof of Theorem 9 is rather involved, and we only outline the main ideas here, in the simpler case where the wedge consists of a single surface of negative Euler characteristic, omitting proofs of the intermediate lemmas. Since the fundamental group of a wedge of spaces is the free product of the fundamental groups of the spaces, the more general version is not much harder. We refer to the full version for complete details.

Our algorithm starts in a similar way as the known linear-time algorithms to test contractibility or homotopy of curves on surfaces [17, 29]: we first turn our surface into a system of quads (Lemma 10), and then compute a canonical form for our curve w (Lemma 13). This canonical form is unique, and therefore the walk w is homotopic to a trivial walk if and only if its reduced canonical form is the empty walk. However, due to the compression of the input, our techniques to reduce the curve w are more involved than those of the aforementioned references.

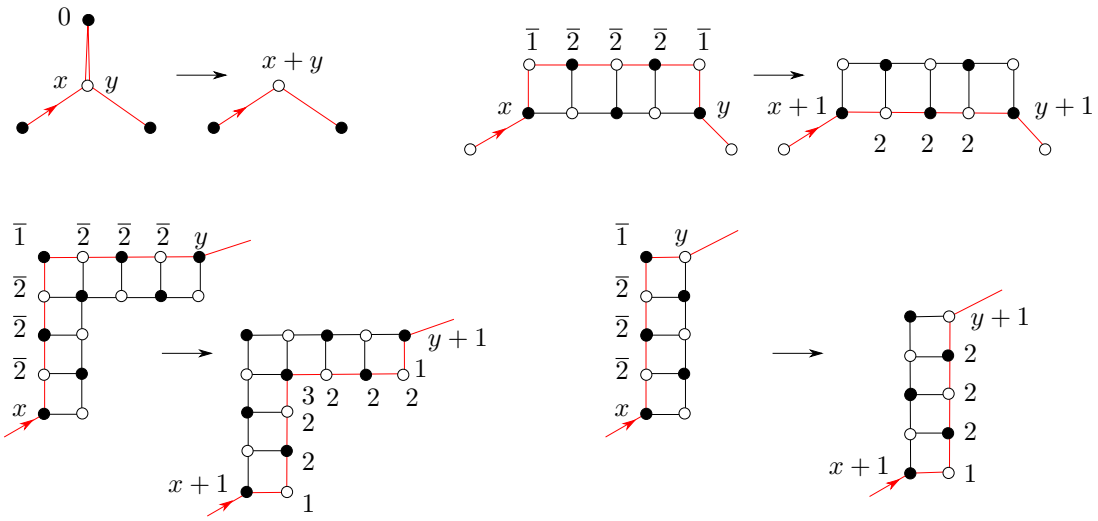


Figure 3 The reductions to eliminate a spur (top left) and a bracket (top right), and the shifting moves to remove a $\bar{1}$ (bottom row).

Our first step is to turn our surface into systems of quads. First, we compute a spanning tree and contract it. Then, we remove edges until there is a single face, we add a new vertex inside the single face, add all the radial edges between this new vertex and the single vertex of S and remove all the previous edges. The resulting complex is called a *quad system*.

▶ **Lemma 10.** *Let K be a surface and let w be a compressed walk on K. In polynomial time we can turn K into a quad system K' and compute a compressed walk w' on K' that is homotopic to w.*

In the next step of our algorithm, we encode words using *turn sequences*. For any two directed edges e and e' on the same surface S, the turn $\tau(e, e')$ is the number of corners, counted with respect to our chosen orientation, between the end of e and the start of e' on S. The turn sequence of our word is the concatenation of all the turn sequences of consecutive edges. However, we lose information by encoding with turn sequences. First, a turn sequence does not specify a starting edge. Second, even if we know the starting edge, it is not immediately clear how to compute an arbitrary intermediate edge along the path efficiently since we encode exponential length paths. The following lemma shows how to do that in polynomial time.

▶ **Lemma 11.** *Let $\mathbb{T}$ be a compressed turn sequence and k be an integer. Then if we are given a starting edge $e_{initial}$, then in polynomial time we can compute the edge of K we are on just after starting at $e_{initial}$ and following the turn sequence up to $\mathbb{T}[k]$.*

The point of turn sequences is that they make it easier to make local simplifications to a contractible curve. Following Lazarus-Rivaud [29] and Erickson-Whittlesey [17], we use the following simplification rules, see Figure 3, which are homotopies.

- A *spur* in a turn sequence w is a 0 turn. Removing a spur is applying the rule $x0y \to x+y$.
- A *bracket* in a turn sequence w is a subword of the form $12\dots21$ or $\bar{1}\bar{2}\dots\bar{2}\bar{1}$. Flattening a bracket is applying the rules $x12\dots21y \to (x-1)\bar{2}\dots\bar{2}(y-1)$ or $x\bar{1}\bar{2}\dots\bar{2}\bar{1}y \to (x+1)2\dots2(y+1)$.

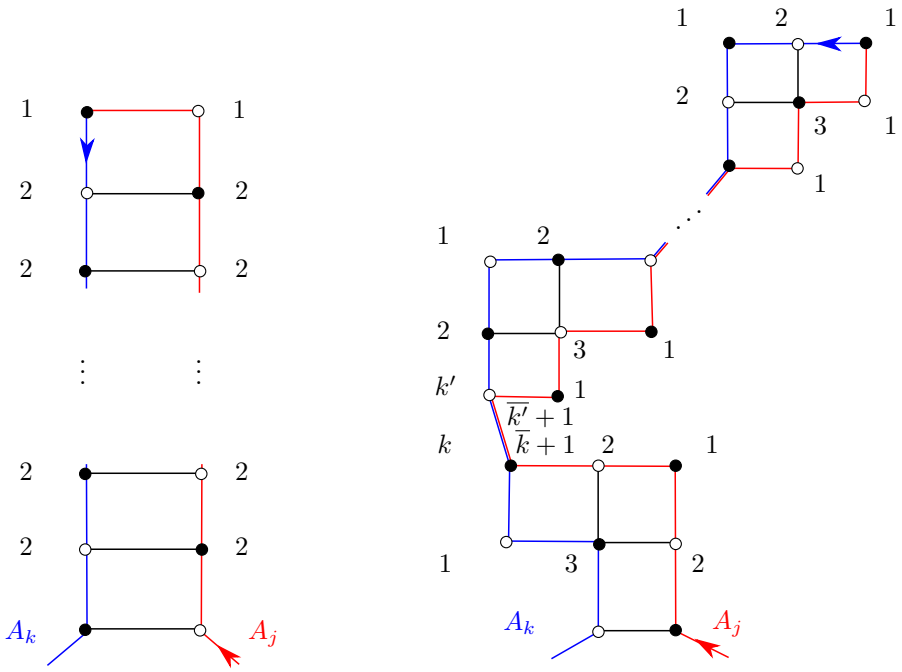


Figure 4 Some production rules $A_i \to A_j A_k$ require an exponential number of bracket flattenings and/or spur removals to be reduced, despite A_j and A_k being already reduced.

We are only considering homotopies of paths in this paper, and thus do not need the other special cases considered in Erickson-Whittlesey [17]. Our goal is to modify the turn sequence so that it is *reduced*, i.e., contains neither spurs nor brackets. However, unlike in the aforementioned works, we cannot apply these rules directly, even inductively: in a production rule $A_i \to A_j A_k$, even when A_j and A_k are reduced, there might be an exponential number of bracket flattenings in A_i, for example in the cases in Figure 4.

There are known techniques to handle exponentially long spurs [31, 37], but for the more intricate cases pictured in Figure 4, especially the kind on the right side, we need to develop our own tools. In order to do that, we rely on stronger inductive forms, similarly to those used in the free homotopy test [17, 29]. A reduced path is *leftmost* (or *rightmost*, respectively) if its turn sequence contains no 1, respectively no $\bar{1}$. One can transform a reduced path into its rightmost (resp. leftmost) form by doing *elementary right-shifts* (resp. *elementary left-shifts*); see Figure 3, bottom. Rightmost and leftmost paths are unique in their homotopy classes [17, 29].

For each character in the straight-line program, we inductively compute both a rightmost and leftmost turn sequence. Then, both are used to carry the induction step. So the next step of our algorithm is the following reduction algorithm. Since turn sequences only encode the turns at the interior vertices of a path and forget where the path starts, we store this information separately: for each character in the straight-line program, we store in a dictionary its starting and ending edges (which might be empty). To keep the description concise, we explain the algorithm in words, and refer to the full version for a much more detailed description.

Algorithm 1 Reduction Algorithm.

Input: A compressed walk on a quad system, as output by Lemma 10.
Output: Two compressed turn sequences on a quad system which are the leftmost and rightmost forms of the input, and their starting and ending edges.

- A leaf of the production tree is a single edge, which is turned to ε, and the edge is stored in the dictionary.
- Let $A_i \to A_j A_k$ be a production rule, and assume that we have inductively computed rightmost and leftmost reduced turn sequences for A_j and A_k, which are denoted by A_j^R, A_j^L, A_k^R and A_k^L, as well as their starting and ending edges. We explain how to compute a rightmost reduced turn sequence A_i^R, the leftmost case being symmetric.
  1. Using Lemma 4 and the dictionary, we compute a maximal common path between the end of A_k^R and the start of A_k^L (let us emphasize that we use the **leftmost** form here). We trim the end of A_j^R and the start of A_k^R by the length of this path, manually reconnecting them if there is some offset.
  2. Using Lemma 4, we remove a maximal spur or ladder between A_j^R and A_k^R (as on the left of Figure 4).
  3. At this stage, no bad cases can happen anymore. We simply perform a single elementary right shift, remove the at most two remaining brackets, and update the starting and ending edge in the dictionary.

Step 3 simply implements the bracket flattenings and the shifts, so the main mysteries in this algorithm are steps 1 and 2. These are tailored to deal with the bad cases pictured in Figure 4, which are the only possible bad cases; see the proof of Lemma 12. Figure 5 illustrates how step 1 reduces (a subset of) the right case of Figure 4 to its left case, and how step 2 deals with that case. Note that in these pictures, A_j and A_k were already rightmost.

We claim that after this procedure, the path that represents the word A_i^R (respectively A_i^L) is homotopic to the path represented by A_i, and is rightmost (respectively leftmost). The following lemma is the crux of the analysis. We state it for rightmost paths but the symmetric version with leftmost paths holds with the same proof.

▶ **Lemma 12.** *At the end of step 2, the paths represented by A_j^R and A_k^R are rightmost. Furthermore, denoting by τ the turn between A_j^R and A_k^R at the end of step 2, the path represented by $A_j^R \tau A_k^R$ is homotopic to the one before these two steps, contains no spurs, contains at most two brackets, and these brackets contain both at least one* 2.

The idea of the proof is that due to structural results on reduced paths on quad systems [13, Theorem 10], paths that would incur long sequences of simplifications necessarily look like a sequence of paths and staircases, as in the right side of Figure 4. These are dealt with a single move in step 1, observing that taking the **leftmost** form of A_k collapses these staircases into a long spur, which we know how to handle. After this step, we prove that there is either a ladder or a spur, which are removed in step 2.

▶ **Lemma 13.** *The Reduction Algorithm runs in polynomial time. The straight-line program that it outputs has the property that every pair of characters A_i^R and A_i^L encode a pair of turn sequence corresponding respectively to the rightmost and leftmost paths homotopic to A_i. In particular, at the top level, $\mathbb{A}^R$ and $\mathbb{A}^L$ are the rightmost and leftmost paths homotopic to the compressed input walk.*

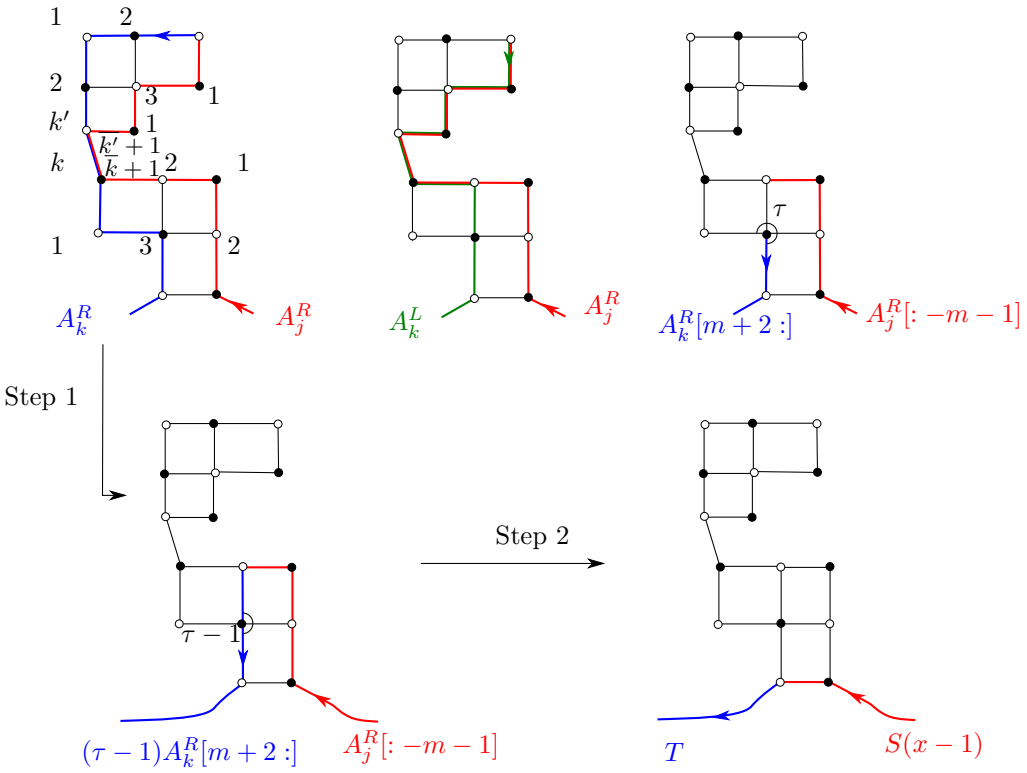


Figure 5 The leftmost form A_k^L forms a long spur with A_j^R. After reducing A_k^R by the length of this spur, all of the staircases get removed, except possibly the last one. It disappears in step 2.

Since rightmost paths are unique, the reduction algorithm applied to the root of the straight-line program produces a unique representative, and testing triviality amounts to testing emptiness. This proves Theorem 9 (see the full version for details).

Proofs of the main theorems

The proof of Theorems 1, 2 and 3 follow from Propositions 6, 5, 7, 8 and Theorem 9. We refer to the full version for details.

References

1 Ian Agol, Joel Hass, and William Thurston. The computational complexity of knot genus and spanning area. *Transactions of the American Mathematical Society*, 358(9):3821–3850, 2006.

2 Matthias Aschenbrenner, Stefan Friedl, and Henry Wilton. Decision problems for 3–manifolds and their fundamental groups. *Geometry & Topology Monographs*, 19(1):201–236, 2015.

3 Matthias Aschenbrenner, Stefan Friedl, Henry Wilton, and Stefan Friedl. *3-manifold groups*, volume 20. European Mathematical Society Zürich, 2015.

4 Mark C Bell. Simplifying triangulations. *arXiv preprint*, 2016. `arXiv:1604.04314`.

5 Benjamin Burton, Éric Colin de Verdière, and Arnaud de Mesmay. On the complexity of immersed normal surfaces. *Geometry & Topology*, 20(2):1061–1083, 2016.

6 Benjamin A Burton. A new approach to crushing 3-manifold triangulations. *Discrete & Computational Geometry*, 52(1):116–139, 2014.

7 James W Cannon, David BA Epstein, Derek F Holt, Silvio VF Levy, Michael S Paterson, and William P Thurston. *Word processing in groups*. CRC Press, 1992.

8 Pierre de La Harpe. *Topics in geometric group theory*. University of Chicago Press, 2000.

9 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. The Unbearable Hardness of Unknotting. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, pages 49:1–49:19, Dagstuhl, Germany, 2019.

10 Éric Colin de Verdière and Salman Parsa. Deciding contractibility of a non-simple curve on the boundary of a 3-manifold. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2691–2704. SIAM, 2017.

11 Éric Colin de Verdière and Salman Parsa. Deciding contractibility of a non-simple curve on the boundary of a 3-manifold: A computational loop theorem. *arXiv preprint*, 2020. `arXiv:2001.04747`.

12 Max Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72(3):413–421, 1912.

13 Vincent Despré and Francis Lazarus. Computing the geometric intersection number of curves. *Journal of the ACM (JACM)*, 66(6):1–49, 2019.

14 Ivan Dynnikov. Counting intersections of normal curves. *arXiv preprint*, 2020. `arXiv:2010.01638`.

15 Ivan Dynnikov and Bert Wiest. On the complexity of braids. *Journal of the European Mathematical Society*, 9:801–840, 2007.

16 Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. *Discrete and Computational Geometry*, 49:823–863, 2013.

17 Jeff Erickson and Kim Whittlesey. Transforming curves on surfaces redux. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1646–1655. SIAM, 2013.

18 Christian Hagenah. Gleichungen mit regulären Randbedingungen über freien Gruppen, 2000.

19 Joel Hass, Jeffrey C Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM (JACM)*, 46(2):185–211, 1999.

20 Joel Hass, Jack Snoeyink, and William P Thurston. The size of spanning disks for polygonal curves. *Discrete and Computational Geometry*, 29(1):1–18, 2003.

21 A. Hatcher, Cambridge University Press, and Cornell University. Department of Mathematics. *Algebraic Topology*. Algebraic Topology. Cambridge University Press, 2002. URL: `https://books.google.com/books?id=BjKs86kosqgC`.

22 John Hempel. *3-Manifolds*, volume 349. American Mathematical Soc., 2004.

23 Derek Holt, Markus Lohrey, and Saul Schleimer. Compressed Decision Problems in Hyperbolic Groups. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126, pages 37:1–37:16, Dagstuhl, Germany, 2019.

24 William Jaco and Jeffrey L Tollefson. Algorithms for the complete decomposition of a closed 3-manifold. *Illinois journal of mathematics*, 39(3):358–406, 1995.

25 Greg Kuperberg. Algorithmic homeomorphism of 3-manifolds as a corollary of geometrization. *Pacific Journal of Mathematics*, 301(1):189–241, 2019.

26 Marc Lackenby. The efficient certification of knottedness and Thurston norm. *arXiv preprint*, 2016. `arXiv:1604.00290`.

27 Marc Lackenby. Some conditionally hard problems on links and 3-manifolds. *Discrete & Computational Geometry*, 58(3):580–595, 2017.

28 Marc Lackenby. Algorithms in 3-manifold theory. *arXiv preprint*, 2020. `arXiv:2002.02179`.

29 Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 440–449. IEEE, 2012.

30 Maarten Löffler, Anna Lubiw, Saul Schleimer, and Erin Moriarty Wolf Chambers. Computation in Low-Dimensional Geometry and Topology (Dagstuhl Seminar 19352). In *Dagstuhl Reports*, volume 9. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

31 Markus Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210–1240, 2006.

32 Markus Lohrey. *The compressed word problem for groups*. Springer, 2014.

33 Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. John Hopkins University Press, 2001.

34 Edwin E Moise. Affine structures in 3-manifolds: V. The Triangulation theorem and Hauptvermutung. *Annals of mathematics*, pages 96–114, 1952.

35 Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Algorithms for normal curves and surfaces. In *International Computing and Combinatorics Conference*, pages 370–380. Springer, 2002.

36 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Computing Dehn twists and geometric intersection numbers in polynomial time. In *CCCG*, volume 20, pages 111–114. Citeseer, 2008.

37 Saul Schleimer. Polynomial-time word problems. *Commentarii Mathematici Helvetici*, 83:741–765, 2008.

38 Horst Schubert. Bestimmung der Primfaktorzerlegung von Verkettungen. *Mathematische Zeitschrift*, 76(1):116–148, 1961.

39 John Stillwell. *Classical topology and combinatorial group theory*, volume 72. Springer Science & Business Media, 2012.

40 Friedhelm Waldhausen. The word problem in fundamental groups of sufficiently large irreducible 3-manifolds. *Annals of Mathematics*, 88(2):272–280, 1968.