Computing Dehn Twists and Geometric Intersection Numbers in Polynomial Time

Marcus Schaefer
Department of Computer Science
DePaul University
243 South Wabash
Chicago, Illinois 60604, USA
mschaefer@cs.depaul.edu

Eric Sedgwick
Department of Computer Science
DePaul University
243 South Wabash
Chicago, Illinois 60604, USA
esedgwick@cs.depaul.edu

Daniel Štefankovič
Department of Computer Science
University of Rochester
Rochester, New York, USA
stefanko@cs.rochester.edu

December 3, 2007

Abstract

Simple curves on surfaces are often represented as sequences of intersections with a triangulation. However, there are much more succinct ways of representing simple curves used in topology such as normal coordinates. In these representations, the length of a curve can be exponential in the size of its representation.

Nevertheless, we show that the following two basic tasks of computational topology, namely

- performing a Dehn-twist of a curve along another curve, and
- computing the geometric intersection number of two curves,

can be solved in polynomial time even in the succinct normal coordinate representation. These are the first algorithms for these two problems that solve these problems in time polynomial in the succinct representations.

As a consequence we can show that a generalized notion of crossing number can be decided in NP, even though the drawings can have exponential complexity.

1 Introduction

In an earlier paper we started an investigation into algorithms for basic problems of computational topology [SSS02]; we extend this work to deal with crossings of curves in surfaces which has applications to graph drawing.

One of the driving problems of computational topology, long before it acquired the name, has been the problem of recognizing the unknot. The story begins with Kneser [Kne30] who, in 1930, introduced a succinct representation for curves and surfaces in which these objects are described by their *normal coordinates*. This led to the theory of normal surfaces which was used by Haken in 1961 to show that the unknot could be recognized by an algorithm (which, much later, was shown to run in exponential time). Haken's approach was pushed further by Hass, Lagarias, and Pippenger who exploited the succinctness of the representation to show that the unknot could be recognized in **NP** [HLP99]. To this end they had to verify in polynomial time that a special type of normal surface was an essential disk, in particular, that is was connected. This result was strengthened by Agol, Hass, and Thurston [AHT02] who showed that the number of connected components of a normal surface can be computed in polynomial time. This immediately implies polynomial time algorithms for checking whether a normal surface is connected, and whether it is orientable.

Independently, we developed a set of tools for algorithms on normal curves in [SSS02], that grew out of our work on the string graph recognition problem [SŠ04, SSŠ03]. We showed how to compute connected components, count them, decide isotopy of curves, and compute the algebraic intersection number of two curves. The novel ingredient of our approach was that it was based on recent developments of algorithms over free monoids (see [Loh03] for a survey) rather than free groups.

In the current paper we continue the study of curves by showing how to efficiently perform Dehn twists, a fundamental topological operation. As a consequence we obtain an algorithm for computing the geometric intersection number of two curves. The following theorem summarizes our main results.

Theorem 1.1. *The tasks of*

- *performing a Dehn twist of a curve along another curve on an arbitrary surface, and*
- *computing the geometric intersection number of two curves on a surface with boundary*

can be solved in polynomial time in the normal coordinate representation (in a given triangulation).

We will make these statements more precise in the upcoming sections; for the time being, however, let us point out that these results are very strong, since the normal coordinate representation is very succinct: the length of a curve is exponential in the size of its representation.

As an application of Theorem 1.1 we show that a generalized notion of crossing number lies in **NP**, which unifies several non-trivial complexity results in graph drawing (Section 7).

There have been previous attempts at calculating the effect of a Dehn twist. Penner [Pen84] gave explicit formulas describing the action of the Lickorish generators on the Dehn-Thurston coordinates for $\mathrm{CS}(M_g)$, the set of all isotopy classes of simple curves. This solves the problem for a very restricted case only, the Lickorish generators. Dehn twists along other curves can be obtained by applying Dehn twists along Lickorish generators, but exponentially many Dehn twists may be needed.

On the other hand, more recently Hamidi-Tehrani and Chen [HTC96] gave an algorithm to compute the action of a set of generators on the space $\mathrm{CS}(M)$ given by measured π-train tracks, but its running time is exponential in the representation.

Hamidi-Tehrani's thesis [HT97] gave an algorithm for the geometric intersection number running in time polynomial in the length of the curves. Again, this translates only to an exponential time algorithm in the succinct representations.

In Section 2 we introduce the basic topological terminology and the tools we use to deal with word equations. We then show that the normal coordinate representation of curves is computationally equivalent to intersection sequences expressed by straight line programs in Section 3. We use this equivalence in Section 4 in an algorithm to compute the Dehn twists of two curves; this algorithm is the basis of computing the geometric intersection number as outlined in Section 5 and 6. Finally, Section 7 gives several applications to graph drawing problems.

2 Preliminaries

2.1 Representations of Surfaces and Simple Curves

By a *surface* M we mean a connected, compact, orientable topological surface with boundary ∂M. For most of this paper we will assume that $\partial M \neq \emptyset$. Let g be the genus of M and let h be the number of boundary components. A simple *multi-curve* α in M is a proper 1-dimensional submanifold of M such that no component of α is null-homotopic or homotopic to the boundary. A connected component of a simple multi-curve is either a *simple closed curve* or a *properly embedded simple arc* (that is, a properly embedded curve with both endpoints on ∂M). By *isotopy* we mean isotopy rel boundary, i. e., a continuous deformation which leaves ∂M fixed. Let $\mathrm{CS}(M)$ be the set of all isotopy classes of simple multi-curves. Let $\mathrm{CS}_0(M) \subseteq \mathrm{CS}(M)$ be the set of isotopy classes of simple multi-curves whose components are simple closed curves.

The *geometric intersection number* of two (isotopy classes of) simple multi-curves $\alpha, \beta \in \mathrm{CS}(M)$ is the minimal number of intersections between any of their representatives.

Surfaces can be presented purely combinatorially, for example, using triangulations, or polygonal schemas. A *triangulation* T of a surface M is a set of points V in M and an embedded collection of arcs E such that each component of $M - E$ is an open disc bounded by three curves from E. A triangulation is *minimal* if the vertices V are on the boundary ∂M. A *polygonal schema* is a polygon with an even number of directed edges together with information on how to pair the edges to obtain the surface.

Let M be given by a triangulation. A simple multi-curve γ is *normal* w.r.t. T if all the intersections of γ with T are transversal and if γ enters a triangle t via an edge e then it leaves t via an edge different from e. If γ is normal, then the vector $(|\gamma \cap e|)_{e \in T}$ is called *normal coordinates* of γ. The *complexity* of the normal coordinates is the number of bits needed to encode the vector $(|\gamma \cap e|)_{e \in T}$, by writing each coordinate in binary. Any two simple multi-curves with the same normal coordinates are isotopic. If the triangulation is minimal then the converse is true—any isotopic curves have the same normal coordinates. (Note that only surfaces with non-empty boundary have minimal triangulations.)

Let M be given by a triangulation T. Arbitrarily fix an orientation $\vec{e}$ of each edge $e \in T$. Given an oriented simple curve γ let w be a word obtained by traversing γ and appending e to w if $\vec{e}$ is crossed from left to right and appending e^{-1} is $\vec{e}$ is crossed from right to left. Then w is called an *intersection sequence of γ with the triangulation*. If γ is a simple closed curve the intersection sequence is not unique, since we did not specify the starting point of our walk (note that any cyclic shift of an intersection sequence is also an intersection sequence). The length of an intersection sequence can be exponentially longer than the complexity of normal coordinates. Fortunately, for simple curves the sequence can always be compressed, as we show in Section 3.

Also note that while normal coordinates can encode multi-curves (which can have many connected components), the intersection sequences are restricted to simple curves (which have a single component). The number of components of a multi-curve can be exponential in the complexity of its normal coordinates but, fortunately, they fall into a small number of isotopy classes [SSS02].

Let α be a simple closed curve in M. A *Dehn twist* $D_\alpha : M \to M$ along α is a homeomorphism of M obtained by cutting M along α, rotating one of the copies of α by 360 degrees and gluing the two copies back together. More precisely, if an annulus around α is parameterized by $\{(x, \varphi), 1 \leq x \leq 2, 0 \leq \varphi \leq 2\pi\}$ then D_α is $(x, \varphi) \mapsto (x, \varphi + 2\pi(x-1) \bmod 2\pi)$ on the annulus and the identity elsewhere. Dehn twists are a fundamental object in the topology of surfaces: they generate the *mapping class group* $\mathcal{M}_g^s$ (the group of orientation preserving homeomorphisms of the surface M_g^s to itself modulo isotopy).

2.2 Compressed representation of words and quadratic word equations

Let Σ be an *alphabet*. A word in Σ^* can be represented by a *straight-line program* (SLP), which is a sequence of assignments $x_i := \text{EXPR},\ i = 1, \ldots, n$, where EXPR is a either a symbol from Σ or $x_j x_k, 1 \leq j, k < i$. Note that the length n of a straight line program representing a word w can be exponentially smaller than $|w|$, the length of w.

Given a word w it is **NP**-hard to find the shortest SLP generating w. In fact, the length of the shortest SLP is **NP**-hard to approximate within a factor of 1.0001 ([LS02]), but an approximation algorithm with factor $O(\log |w|)$ is known ([CLL+02, Ryt02]).

Equality of two words given by SLPs can be tested in time polynomial in the length of the SLPs [HJM96, Pla94]. More precisely the following is known:

Theorem 2.1. *Let u and v be words represented by straight-line programs P and P' of lengths m and n. We can decide whether $u = v$*

- *in time $O(m^2 n^2)$ by a deterministic algorithm [MST97].*
- *in time $O(m + n)$ by a randomized algorithm with 1-sided error: If $u = v$ then the algorithm always gives the correct answer, if $u \neq v$ it errs with probability at most δ [GKPR96b]. (The model used assumes that arithmetic operations on numbers of size $O(\log 1/\delta)$ have unit cost.)*

Let Σ be an alphabet. Let Θ be an alphabet of variables disjoint from Σ. A *word equation* $u = v$ is a pair of words $u, v \in (\Sigma \cup \Theta)^*$. The *size* of the equation is $|u| + |v|$. A *solution of the word equation* $u = v$ is a morphism $h : (\Sigma \cup \Theta)^* \to \Sigma^*$ such that $h(a) = a$ for all $a \in \Sigma$ and $h(u) = h(v)$ (h being a morphism means that $h(wz) = h(w)h(z)$ for all $w, z \in (\Sigma \cup \Theta)^*$.) A *word equation with specified lengths* is a word equation $u = v$ and a function $\ell : \Theta \to \mathbb{N}$. The solution h has to respect the lengths, i. e., we require $|h(x)| = \ell(x)$ for all $x \in \Theta$. An SLP for solution h is an SLP for $h(u)$. The *complexity* of a word equation $u = v$ with specified lengths ℓ is $|u| + |v| + \sum_{x \in \Omega} \log \ell(x)$.

Finding a solution of a word equation is **NP**-hard. The situation changes when the lengths are given. Plandowski and Rytter [PR98] found an algorithm which finds an SLP of a solution of a word equation with specified lengths in time polynomial in the complexity of the equation.

A word equation is *quadratic* if every variable occurs at most twice in the equation. The word equations arising in our context will be quadratic and hence we will be able to use the following, faster (and simpler) algorithm:

Theorem 2.2 (Robson, Diekert [RD99]). *Quadratic word equations $u = v$ with specified lengths can be decided in time linear in the complexity of the equation. Moreover, if there exists a solution, a linear-size SLP for the solution can be found in linear time.*

3 Converting between compressed representations

In this section we show that for simple curves the normal coordinates and intersection sequences represented by SLPs are polynomially related, that is, we can convert from one representation to the other one in polynomial time.

Moving from an SLP representation of an intersection sequence to normal coordinates is simple—we just need to count the number of occurrences of each symbol in the compressed word, a task that can be performed in time $O(n)$ for each symbol, see [GKPR96a]. Hence we have the following result.

Lemma 3.1. *Let M be a surface given by a triangulation T. Let $\alpha \in \mathrm{CS}(M)$ be a simple curve in M given by an intersection sequence of α with T, encoded by an SLP of length n. The normal coordinates of α can be computed in time $O(n \cdot |T|)$.*

Surprisingly, the inverse conversion can be computed efficiently, too:

Lemma 3.2. *Let M be a surface given by a triangulation T. Let $\alpha \in \mathrm{CS}(M)$ be a simple curve in M given by normal coordinates on T. Let n be the complexity of the normal coordinates. An SLP of size $O(n)$ for an intersection sequence of α with T can be obtained in time $O(n)$.*

Proof. We will create a system of quadratic word equations with given lengths such that the solution will give all the intersection sequences of α with T. Let L be the sum of all coordinates of α. Note that L is the length of any intersection sequence of α with T.

For the proof we add the following curves to the surface. Let t be a triangle with vertices a, b, c and edges ab, bc, ca. Fix a vertex v_t inside t and connect v_t to $u, u = a, b, c$ by a simple curve γ_u. For $e = ab, bc, ca$ add a curve η_e within t connecting the endpoints of e. Clearly, we can add the curves so that any two are disjoint (except at the endpoints) and they are disjoint from T (except at vertices a, b, c), see Figure 1. We will refer to the curves added as "added curves".

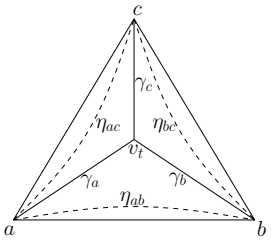

Figure 1: Adding curves γ_u and η_e (dashed).

Now we isotope α so that it intersects the added curves minimally. Thus the curve η_e is intersected $|e \cap \alpha|$ times by α. The number of intersections of α with the γ curves can be easily computed, e.g., $|\gamma_a \cap \alpha| = (|ab \cap \alpha| + |ca \cap \alpha| - |bc \cap \alpha|)/2$.

For each of the added curves we have two variables in the system of quadratic word equations, one for each orientation of the curve. In the system of quadratic word equations we will seek a solution where the variables for curve β (where β is one the added curves) are assigned strings of length $L \cdot |\alpha \cap \beta|$.

Fix an orientation $\vec{\alpha}$ on α. Let $\vec{\beta}$ be one of the added curves, together with an orientation. For each intersection point p of α and $\vec{\beta}$ take the intersection sequence of α with T starting at p leaving from p to the left of $\vec{\beta}$ (note that the direction of travel depends on the sign of the intersection of

$\vec{\alpha}$ and $\vec{\beta}$ in p). We set up the equations so that in the solution the variable $X(\vec{\beta})$ for $\vec{\beta}$ will be the concatenation of the intersection sequences for points p in the order they occur on $\vec{\beta}$.

Our system of equations will have six equations for each triangle of T and two additional equations for each edge of T. The equations for a triangle $t \in T$ correspond to the triangles inside T formed by the added curves. For example for the triangle formed by a, b, v_t we will have equations $X(ab) = X(av_t)X(v_tb)$ and $X(ba) = X(bv_t)X(v_ta)$. Finally, for each edge of the triangulation we add two equations to our system (one for each way of orienting e). Let η_e and η'_e be a pair of η curves arising from the same oriented edge $\vec{e}$ and having the same orientation. Suppose that η_e is to the left of η'_e. We add equation $X(\eta'_e)s = sX(\eta_e)$ where $s = e$ or $s = e^{-1}$ depending on which symbol represents the crossing of e from left to right.

There is a unique solution of the described system of equations, as can be seen by starting at the leftmost intersection of α with β on $\vec{\beta}$ and following the curve α.

Since the system of equations is quadratic, we can apply Theorem 2.2 to obtain a linear-sized SLP for $\vec{\alpha}$ in time $O(n)$. ■

4 Computing Dehn twists

In this section we show how a Dehn twist of a curve given by normal coordinates can be computed in time polynomial in the size of the representation. The algorithm works for all surfaces (that is, with or without boundary).

Theorem 4.1. *Let M be a surface of genus g given by a triangulation T. Let be $\alpha \in \mathrm{CS}(M)$ and $\beta \in \mathrm{CS}_0(M)$ be simple multi-curves in M given by normal coordinates of complexity $\leq n$. The normal coordinates of a representative of the Dehn twist $D_\beta(\alpha)$ can be computed*

- *in time $O(g \cdot n^3)$ by a randomized algorithm with small probability of error; and*
- *in time $O(g \cdot n^{10})$ by a deterministic algorithm.*

Proof. We first handle the case that β is connected. The isotopy class of a Dehn twist $D_\beta(\alpha)$ is independent of the choice of the representatives for α and β, so we can isotope both curves to simplify the computation. For each edge of $e \in T$ define three segments on e, the left, middle and right windows. Isotope α, β so that β always crosses the triangulation in middle windows, α so that it always crosses the triangulation in left and right windows and so that all the crossings of α and β happen between the η edges defined in the proof of Lemma 3.2, see Figure 2.

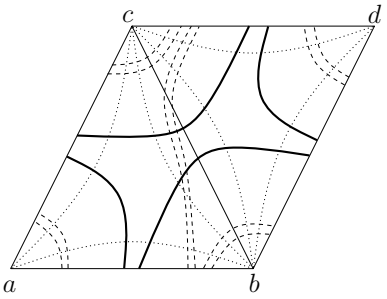

Figure 2: The choice of representatives of α (dashed) and β (heavy solid) for computing $D_\beta(\alpha)$. The η lines are dotted.

Let a, b, c, d be vertices of triangles abc and dbc sharing the edge bc. The edge bc is enclosed within a region formed by the two η curves belonging to it. The only segments of α which intersect β in that region correspond to sequences $ab\,bc\,cd$, $ac\,bc\,bd$ (and their inverses).

A Dehn twist along β transforms the sequence $ab\,bc\,cd$ to $ab\,X(bc)\,bc\,cd$ where X is the variable from the proof of Lemma 3.2, containing the sequence of intersection sequences of β with T. All these transformations can be performed simultaneously on the SLP representation in linear time. The resulting SLP has size at most $3n$. The resulting intersection sequence is a representative of $D_\beta(\alpha)$ but it is not in normal position w.r.t. T. To obtain a representative in normal position we need to cyclicly reduce the word which can be done in polynomial time by Corollary 6.2. Once in normal position, we can compute its normal coordinates as in Lemma 3.1.

If β is not connected then it has $O(g)$ isotopy classes of components. These can be identified, and the number of copies of each class counted in time $O(n^2 g)$ (see [SSS02]). Now we can perform the Dehn twists for each isotopy class separately. If there are c copies of a simple closed curve γ in β then twisting along all of them simultaneously is similar to twisting along a single curve (the only difference is that we use $X(\eta'_e)s^c = s^c X(\eta_e)$ in the proof of Lemma 3.2). ■

5 Computing geometric intersection numbers

For surfaces with a boundary we can compute geometric intersection numbers in polynomial time once we can compute Dehn twists in polynomial time. The reason is that the complexity of Dehn twists is closely related to the geometric intersection number. Considering representatives of β and γ which intersect minimally shows that for any simple curve α

$$|D^n_\gamma(\beta) \cap \alpha| \leq |\beta \cap \alpha| + n \cdot i(\beta, \gamma)|\gamma \cap \alpha|,$$

in other words, $D^n_\gamma(\beta)$ grows by a rate of at most $i(\beta, \gamma)|\gamma \cap \alpha|$ in n. For sufficiently large n, it grows exactly at that rate:

Lemma 5.1. *Let M be a surface. Let α, β, γ be simple curves on M. Suppose that γ is closed. Let $n = 2i(\alpha, \beta)$. Then*

$$i(\gamma, \beta) = \frac{i(\alpha, D^{n+1}_\gamma(\beta)) - i(\alpha, D^n_\gamma(\beta))}{i(\alpha, \gamma)}.$$

We will use following two results about geometric intersection numbers to prove Lemma 5.1.

Theorem 5.2 ([FLP79]). *Let M be a surface. Let α, β, γ be simple curves on M. Suppose that γ is closed. Then*

$$n \cdot i(\alpha, \gamma) i(\gamma, \beta) - i(\alpha, \beta) \leq i(\alpha, D^n_\gamma(\beta)) \leq n \cdot i(\alpha, \gamma) i(\gamma, \beta) + i(\alpha, \beta). \tag{1}$$

Theorem 5.3 ([Luo01]). *Let M be a surface. Let α, β, γ be simple curves on M. Suppose that γ is closed. Then*

$$f(n) = i(\alpha, D^n_\gamma(\beta)) \tag{2}$$

is a convex function of $n \in \mathbb{Z}$.

Proof of Lemma 5.1. Let $A = i(\alpha, \gamma) i(\gamma, \beta)$ and $B = i(\alpha, \beta)$. We have $kA - B \leq f(k) \leq kA + B$ for any k. Let $n = 2B$.

Suppose that $f(n+1) - f(n) \leq A - 1$. Then by convexity

$$f(n+1) \leq (A-1)(n+1) + f(0) \leq (n+1)A - B - 1,$$

which contradicts $(n+1)A - B \leq f(n+1)$.

Suppose that $f(n+1) - f(n) \geq A + 1$. Then by convexity for $k \geq 0$

$$f(n+k) - f(n) \geq k(A+1) + f(n) \geq k(A+1) + nA - B = (n+k)A + k - B,$$

which contradicts $f(n+k) \leq (n+k)A + B$ for $k = 2B + 1$. Hence $f(n+1) - f(n) = A$. ■

Lemma 5.4. *Let M be a surface given by a triangulation T. Let β, γ be simple curves on M given by normal coordinates. Assume that γ is closed. Then $i(\beta, \gamma)$ can be computed in polynomial time.*

Proof. Let e be an edge of triangulation T which is intersected by γ. Note that $i(e, \gamma)$ is the normal coordinate $\gamma(e)$ of γ on e. Let $n = 2i(e, \alpha) = 2\alpha(e)$. Compute $\beta'' = D_\gamma^{n+1}(\beta)$ and $\beta' = D_\gamma^{n+1}(\beta)$. By Lemma 5.1 we have

$$i(\beta, \gamma) = \frac{\beta''(e) - \beta'(e)}{\gamma(e)}.$$

■

If both β and γ are simple arcs then we cannot use Lemma 5.4 to compute $i(\beta, \gamma)$. We will first show how to compute $i(\beta, \gamma)$ if the endpoints of β and γ are all on different components of ∂M.

Let D_1, D_2 be the connected components of ∂M which contain the endpoints of β. Let $C\beta$ be the closed simple curve which is the boundary of the set of points within ε distance of $\beta \cup D_1 \cup D_2$ for sufficiently small ε.

Lemma 5.5. $i(C\beta, \gamma) = 2i(\beta, \gamma)$.

Proof. Note that $i(\beta, C\beta) = 0$. Consider a drawing minimizing all the intersections. Cut M along $C\beta$. Let M' be the component containing D_1 and D_2. Note that M' is a pair of pants (that is, a 2-sphere with 3 holes) and β connects D_1 and D_2. The drawing of γ in M' consists of essential simple arcs with both boundary points on $C\beta$. Each intersection of γ and β corresponds to two intersections of γ and $C\beta$. ■

Lemma 5.5 solves the case when all the endpoints are on different components. The case when some endpoints of β and γ share boundary components can be reduced to the case when they don't. The reduction changes the surface but it will not change the geometric intersection number. We increase the number of holes in the surface so that each endpoint is on its own component of ∂M by "adding a bridge" to the surface which takes two segments on a component of ∂M and identifies the segments (see Figure 3). We require that the segments do not contain the endpoints of β and γ.

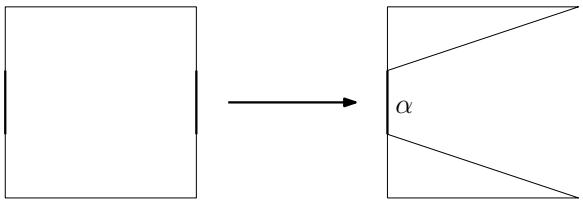

Figure 3: Adding a bridge to a surface (the inside of the square is the hole, the outside is the surface).

Lemma 5.6. *Let M be a surface with boundary. Let β, γ be simple curves on M. Let M' be a surface obtained by adding a bridge in M, where the bridge does not touch β and γ. Then $i_M(\beta, \gamma) = i_{M'}(\beta, \gamma)$.*

Proof. The drawing of β and γ in M which minimizes the number of their intersections is a drawing of β and γ in M'. Hence $i_{M'}(\beta,\gamma) \leq i_M(\beta,\gamma)$.

Let α be the bridge curve in M' (see Figure 3). Note that $i(\alpha,\beta) = i(\alpha,\gamma) = 0$. Thus in the drawing which simultaneously minimizes all geometric intersection numbers (see, e. g., Lemma 3.2 of [SSŠ03]) both β and γ do not intersect α. Hence we can cut along α to obtain a drawing of β and γ in M with $i_{M'}(\beta,\gamma)$ intersections. Hence $i_M(\beta,\gamma) \leq i_{M'}(\beta,\gamma)$. ■

Theorem 5.7. *The geometric intersection number of two curves given by normal coordinates on a surface with boundary can be computed in polynomial time.*

Proof. If β or γ is simple closed curves then we use Lemma 5.4. If both β and γ are properly embedded arcs then we add bridges to the surface so that the endpoints of β and γ are all on different components. The normal coordinates of β and γ in the new triangulation are easy to compute (they agree on all but the new edges, where they are zero). By Lemma 5.6 this does not change the geometric intersection number of β and γ. Finally we compute the normal coordinates of $C\beta$ and use Lemma 5.4. ■

6 Cancellation and Reduction

As we saw in the previous section, normalizing a curve is closely related to finding the reduction of a compressed word in a free group. In this section we discuss the algorithm used for this.

Let Γ be a set of generators of a free group and let $\Sigma = \Gamma \cup \Gamma^{-1}$ be an alphabet.

Lemma 6.1. *Let w be a word over $\Sigma = \Gamma \cup \Gamma^{-1}$ given by an SLP P of length n. An SLP P' of the reduction w' of w can be computed*

- *in time $O(n^3)$ by a randomized algorithm; and*
- *in time $O(n^{10})$ by a deterministic algorithm. In both cases the resulting SLP P' has length $O(n^2)$*

Note that the general SLP equality testing problem can be reduced to the problem of computing a reduction of a word over $\Sigma = \Gamma \cup \Gamma^{-1}$ in linear time: $u = v$ if and only if the reduction of uv^{-1} is the empty word. Hence, unless there is an improvement in SLP equality testing, a randomized algorithm for the reduction problem will be faster. It might be that the reduction problem is easier for intersection sequences since their equality can be tested in $O(n \cdot t)$ time (by transforming to normal coordinates).

Note that we can compute the cyclic reduction of a word w by computing reductions w', u' of w and $u = ww$ and then taking $w'[1 + r...|w'| - r]$ where $r = |w'| - |u'|/2$. Hence we have the following observation.

Corollary 6.2. *Let w be a word over $\Sigma = \Gamma \cup \Gamma^{-1}$ given by an SLP P of length n. An SLP P' of the cyclic reduction w' of w can be computed by a randomized algorithm in time $O(n^3)$. The length of P' is $O(n^2)$.*

Let $\mathrm{SL}_2(K)$ be the group of unimodular 2×2 matrices over a commutative ring K. The modular group $\mathrm{PSL}_2(\mathbb{Z})$ is the factor group $\mathrm{SL}_2(\mathbb{Z})/(\pm I)$. There is a homeomorphism from $\mathrm{SL}_2(\mathbb{Z})/(\pm I)$ to $\mathrm{SL}_2(\mathbb{Z}/p\mathbb{Z})/(\pm I)$ obtained by reducing each entry modulo p (where p is a prime).

The modular group is a free product of finite groups generated by

$$S = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}.$$

Lemma 6.3 ([Imr84]). *Let B_m be a set of elements*

$$S^{a_1}(R^{a_2}S)\dots(R^{a_m}S)RSR(SR^{-a_m})\dots(SR^{-a_2})S^{a_1},$$

where $a_1 \in \{0,1\}$ and $a_2,\dots,a_m \in \{-1,1\}$. Elements of B_m are free generators of a free subgroup of $\mathrm{PSL}_2(\mathbb{Z})$. Moreover any $X \in B_m$ has $\|X\|_2 \le \phi^{2m}$ where $\phi = 1.618...$ is the golden ratio.

Proof (sketch) of Lemma 6.1. Pick a random prime $p, 2 \le p \le N$ where N will be specified later. To each symbol from Γ assign a matrix from the set B_m in Lemma 6.3. This defines a map $\psi : \Sigma^* \to \mathrm{PSL}_2(\mathbb{Z})$, a word w maps to 1 if and only if its reduction is the empty word.

Let $x_1,\dots,x_n$ be the variables in the straight-line program P. In time $O(n)$ we can compute $\psi(x_i) \bmod p$ for $i = 1,\dots,n$.

Claim: Let $y_1,\dots,y_k$ be the reductions of $x_1,\dots,x_k$. Suppose that we know the lengths of the y_i. Let $a, 1 \le a \le |y_k|$. In time $O(k)$ we can compute $\psi(y_k[1\dots a]) \bmod p$ and a' such that $y_k[1\dots a]$ is the reduction of $x_k[1\dots a']$.

Proof: Let $y_k = y_i y_j$. If $a \le |y_i|$ then we just need to compute $\psi(y_i[1\dots a]) \bmod p$ and a' such that $y_i[1\dots a]$ is the reduction of $x_i[1\dots a']$. If $a > |y_i|$ we compute $\psi(x_i)\psi(y_j[1\dots(a-|y_i|+c)]) \bmod p$, where $c = (|y_i|+|y_j|-|y_k|)/2$ is the length of "cancellation" between y_i and y_j. We also find a'' such that $y_j[1\dots(a-|y_i|+c)]$ is the reduction of $x_j[1\dots a'']$ and let $a' = |x_i| + a''$.

In both cases we spent only constant time and the index of the variable for which we need to recursively solve the subword problem decreased by at least 1. □

Suppose that we computed the lengths of the reductions $y_1,\dots,y_{k-1}$ of $x_1,\dots,x_{k-1}$. Let $x_k = x_i x_j$ be an assignment. Using binary search and the previous claim we can in time $O(n^2)$ find the length ℓ of the longest common prefix of y_j and y_i^{-1}. Now we can compute $|y_k| = |y_i| + |y_j| - 2\ell$.

After we computed lengths of the y_i, $i = 1,\dots,n$ we will generate the program for y_n. For k from n to 1 we do the following. Let $x_k = x_i x_j$ be an assignment. Let a' be such that y_i is the reduction of $x_i[1\dots a']$ and let b' be such that y_j is the reduction of $x_j[b'\dots|x_j|]$. In time $O(n)$ we can compute $a_1,\dots,a_m,b_1,\dots,b_{m'}$ such that $x_i[1\dots a'] = x_{a_1}\dots x_{a_m}$ and $x_j[b'\dots|x_j|] = x_{b_1}\dots x_{b_{m'}}$ where $m, m' \le n$. We replace $x_k = x_i x_j$ by $x_n = x_{a_1}\dots x_{a_m} x_{b_1}\dots x_{b_{m'}}$. The resulting SLP has size $O(n^2)$ and it represents the reduction w' of w.

The algorithm executes $O(n^2)$ randomized identity tests. It will fail if for x that was tested $\psi(x) \ne 1$ but $\psi(x) \bmod p = 1$. Note that $\|\psi(x)\|_2 \le |\Sigma|^{2\cdot 2^n}$ and hence for a random prime $\le N$ for $N = O(n)$ the algorithm fails with probability $O(2^{-n})$.

To get a deterministic algorithm for the problem we consider the assignments in the order they occur in the program and compute SLPs for the reductions. Suppose that we have an SLP for $x_1,\dots,x_{k-1}$ of size m. Let $x_k = x_i x_j$ be an assignment. We can then compute the length of the reduction between x_i and x_j in time $O(n \cdot m^4)$ using binary search and the deterministic equality testing algorithm. In time $O(n^2)$ we can compute $a_1,\dots,a_t,b_1,\dots b_{t'}$ such that the reduction of $x_i x_j$ is $x_{a_1}\dots x_{a_t} x_{b_1}\dots x_{b_{t'}}$ and $t, t' \le n$. We increased the size of the program by $O(n)$. Hence $m = O(n^2)$ and the total running time is $O(n^{10})$. ■

7 Generalized Crossing Number

In this section we apply our algorithm for calculating the geometric intersection number of two curves to a notoriously intractable graph drawing problem: the crossing number. The *crossing number*, $\mathrm{cr}(G)$, of a graph $G = (V, E)$ is the smallest number of intersections in a drawing of G in the plane (making certain standard assumptions about the drawing). Given a weight function $w : E^2 \to \mathbb{N}$, we can define a generalization, $\mathrm{cr}_w(G)$ of the crossing number as the minimum value of

$$\sum_{e,f \in E} i_D(e, f) \cdot w(e, f)$$

over all drawings D of G in the plane. For $w(e, f) = 1$ we obtain the usual crossing number.

Theorem 7.1. *Deciding whether* $\mathrm{cr}_w(G) \leq k$ *lies in* **NP** *for any polynomial-time computable weight function* w*.*

The generalized crossing number is a powerful modelling tool. For example, given a graph $G = (V, E)$ with a symmetric relationship $\mathcal{R} \subseteq E^2$, the weak realizability problem in topological graph theory asks whether G can be drawn in the plane so that for any e and f that intersect $(e, f) \in \mathcal{R}$. Defining $w(e, f) = 0$ if $(e, f) \in \mathcal{R}$ and 1 otherwise, and choosing $k = 0$ shows that the weak realizability problem is a special case of the generalized crossing number and therefore lies in **NP**. This, in turn implies that the string graph problem, topological inference, and several other problems also lie in **NP**, see [SSŠ03]). As another application, we consider the recently introduced simultaneous graph drawing model SCM^+ introduced by Chimani, Jünger, and Schulz [CJS07]: given two planar graphs on the same vertex set, a *simultaneous drawing with fixed edges* is a drawing in which each graph by itself is planar and shared edges are drawn identically. Now, if we relax the condition that the two graphs be planar, we get the *simultaneous crossing number*, which is the smallest number of crossings in a drawing of the union of the two graphs that occur between edges of the same graph. If, in addition, we require that the total number of crossings be minimized, we get the SCM^+ model. It is easy to see that this problem is a special case of our generalized crossing number problem.

For the proof of Theorem 7.1 we need the following result from [SŠ04]: if we have a drawing of a graph in the plane in some edge e has more than 2^m intersections, where $m = |E|$, then we can redraw the graph so that the total number of crossings with e becomes strictly smaller and the numbers $i(f, g)$ do not increase for any $f, g \in E$.

We can now sketch the proof of the theorem. Start with a triangulated plane with $|V|$ punctures and $m' = O(|V|)$ edges. Suppose we are given a drawing of G realizing $\mathrm{cr}_w(G)$ with endpoints of edges on the boundary of the puncture representing the vertex. We can assume that all edges are represented by normalized curves. We can now apply the result from [SŠ04] to conclude that any edge of the graph intersects any edge of the triangulation at most $2^{m+m'}$. Hence, all normal coordinates are bounded by $2^{m+m'}$, and, in **NP**, we can guess the normal coordinates of each edge of G. One we know the normal coordinates of each edge, we can use our result that the geometric intersection number of two curves can be computed in polynomial time, to compute $\mathrm{cr}_w(e, f)$ for any pair of edges, and, thereby, $\mathrm{cr}_w(G)$. Hence, $\mathrm{cr}_w(G) \leq k$ can be decided in **NP**.

References

[AHT02] Ian Agol, Joel Hass, and William Thurston. 3-manifold knot genus is NP-complete. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing (STOC-2002)*, 2002.

[CJS07] Markus Chimani, Michael Jünger, and Michael Schulz. Crossing minimization meets simultaneous drawing. Technical report, Zentrum für Angewandte Informatik Köln, Lehrstuhl Jünger, May 2007.

[CLL+02] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, April Rasala, Amit Sahai, and Abhi Shelat. Approximating the smallest grammar: Kolmogorov complexity in natural models, 2002. In Proc. 29th Ann. Sympo. on Theory of Computing, 792-801.

[FLP79] A. Fathi, F. Laudenbach, and V. Poénaru. Travaux de Thurston sur les surfaces. *Astérisque*, 66–67, 1979.

[GKPR96a] Leszek Gąsieniec, Marek Karpinski, Wojciech Plandowski, and Wojciech Rytter. Efficient algorithms for lempel-ziv encoding. *in Proceedings of SWAT'96, LNCS 1097*, pages 392–403, 1996.

[GKPR96b] Leszek Gąsieniec, Marek Karpinski, Wojciech Plandowski, and Wojciech Rytter. Randomized efficient algorithms for compressed strings: The finger-print approach (extended abstract). In *Proceedings of the 7th Annual Symposium on Combinatorial*, pages 39–49, 1996.

[HJM96] Yoram Hirshfeld, Mark Jerrum, and Faron Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158(1-2):143–159, 1996.

[HLP99] Joel Hass, Jeffrey Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of ACM*, 46(2):185–211, 1999.

[HT97] Hessam Hamidi-Tehrani. *Algorithms in the Mapping Class Groups*. PhD thesis, Columbia, 1997.

[HTC96] Hessam Hamidi-Tehrani and Zong-He Chen. Surface diffeomorphisms via train-tracks. *Topology and its Applications*, 73:141–167, 1996.

[Imr84] Wilfried Imrich. Explicit construction of regular graphs without small cycles. *Combinatorica*, 4(1):53–59, 1984.

[Kne30] H. Kneser. Geschlossene Flächen in dreidimensionalen Mannigfaltigkeiten. *Jahresbericht der Deutschen Mathematikver-Vereinigung*, pages 248–260, 1930.

[Loh03] Markus Lohrey. *Computational and logical aspects of infinite monoids*. PhD thesis, Universitat Stuttgart, 2003. Habilitation.

[LS02] Eric Lehman and Abhi Shelat. Approximations algorithms for grammar-based compression, 2002. In Thirteenth Annual Symposium on Discrete Algorithms (SODA'02).

[Luo01] Feng Luo. Some applications of a multiplicative structure on simple loops in surfaces. In *Knots, braids, and mapping class groups—papers dedicated to Joan S. Birman (New York, 1998)*, volume 24 of *AMS/IP Stud. Adv. Math.*, pages 123–129. Amer. Math. Soc., Providence, RI, 2001.

[MST97] M. Miyazaki, A. Shinohara, and M. Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. In A. Apostolico and J. Hein, editors, *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching*, volume 1264, pages 1–11, Aarhus, Denmark, 1997. Springer-Verlag, Berlin.

[Pen84] Robert C. Penner. The action of the mapping class group on curves in surfaces. *L'Enseignement Mathématique*, 30:39–55, 1984.

[Pla94] Wojciech Plandowski. Testing equivalence of morphisms on context-free languages. In *Algorithms—ESA '94 (Utrecht)*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470. 1994.

[PR98] Wojciech Plandowski and Wojciech Rytter. Application of Lempel-Ziv encodings to the solution of words equations. In *Automata, Languages and Programming*, pages 731–742, 1998.

[RD99] John M. Robson and Volker Diekert. On quadratic word equations. *Lecture Notes in Computer Science*, 1563:217–226, 1999.

[Ryt02] Wojciech Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression, 2002. In Proceedings 13th Annnual Symposium Combinatorial Pattern Matching, 20-31, 2002.

[SŠ04] Marcus Schaefer and Daniel Štefankovič. Decidability of string graphs. *J. Comput. System Sci.*, 68(2):319–334, 2004.

[SSS02] Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Algorithms for normal curves and surfaces. In Oscar H. Ibarra and Louxin Zhang, editors, *COCOON*, volume 2387 of *Lecture Notes in Computer Science*, pages 370–380. Springer, 2002.

[SSŠ03] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *J. Comput. System Sci.*, 67(2):365–380, 2003. Special issue on STOC2002 (Montreal, QC).