Topological Algorithms for Graphs on Surfaces

(Algorithmes topologiques pour les graphes tracés sur des surfaces)

par

Éric COLIN DE VERDIÈRE

Mémoire pour l'obtention de

l'Habilitation à Diriger des Recherches

de l'École normale supérieure

(Spécialité Informatique)

Soutenue le 29 mai 2012 devant le jury composé de :

M. Jean-Daniel BOISSONNAT, *Directeur de Recherche*
M. Victor CHEPOI, *Professeur*
M. Michel HABIB, *Professeur*
M. Sylvain PETITJEAN, *Directeur de Recherche*
M. Jean PONCE, *Professeur*
M. Gilles SCHAEFFER, *Directeur de Recherche*
M. Jean-Marc SCHLENKER, *Professeur*

au vu des rapports de :

M. Victor CHEPOI, *Professeur*
M. Bojan MOHAR, *Professor*
M. Jack SNOEYINK, *Professor*

Date of this version: May 21, 2012.

ACKNOWLEDGMENTS

I wish to express my gratitude to Victor Chepoi, Bojan Mohar, and Jack Snoeyink, who accepted to review this habilitation thesis in spite of their busy schedule; I thank them for the interest they expressed in reading my manuscript. Thanks also to the other members of the examining committee: Jean-Daniel Boissonnat, Michel Habib, Sylvain Petitjean, Jean Ponce, Gilles Schaeffer, and Jean-Marc Schlenker.

I would like to thank my colleagues at ENS, and in particular Jean Ponce, chair of the laboratory, for his help and encouragements. I gratefully acknowledge Jean Vuillemin, the past chair of the laboratory, Michel Pocchiola, my Ph.D. advisor, and David Pointcheval for his help in case of rain! Many thanks also to the persons from the administrative, technical, and library staff at ENS for the great and efficient work they are doing.

Let me thank the co-authors of the research articles presented in this thesis: Alexandre Boulch, Sergio Cabello, Erin Chambers, Jeff Erickson, Grégory Ginot, Xavier Goaoc, Sylvain Lazard, Francis Lazarus, Arnaud de Mesmay, Atsuhiro Nakamoto, Alexander Schrijver, Shripad Thite, and Kim Whittlesey. In particular, thanks to my students Alexandre Boulch and Arnaud de Mesmay for their efforts and patience. This manuscript has also benefited from fruitful discussions with Guillaume Chapuy, Jeff Erickson, and Arnaud de Mesmay, and I gratefully thank them.

Enfin, je remercie de tout cœur ma famille, et tout particulièrement ma femme Mathilde et nos enfants François, Cyprien, Thibaut et Maylis pour leur soutien et leur affection au quotidien.

Contents

CHAPTER 1

INTRODUCTION

In this habilitation thesis, we describe algorithms for solving topological problems for surfaces and graphs drawn on them. Many graphs, in theory and in practice, are inherently geometric; they can represent traffic roads and junctions, computer networks, vertices and edges of a polytope, or islands and bridges connecting them, as in the historically famous problem of the seven bridges of Königsberg studied by Euler. Furthermore, the easiest way to describe a graph is not by its formal definition (adjacency or incidence matrix), but rather by a drawing of the graph in the plane, possibly with crossings; *graph drawing* [89] and *geometric graph theory* [197] are entire fields of research.

Here, we consider graphs embedded (drawn without crossings) on surfaces "more complicated" than the plane. Examples of "more complicated" surfaces include the torus, the Möbius strip, the projective plane, and other surfaces of higher genus. Unlike the plane or the sphere, such surfaces are rich from a topological viewpoint: For example, they admit closed curves that are non-contractible—in other words, that cannot be continuously shrunk to a point while staying on the surface.

Therefore, in addition to algorithmic tools, we will use some tools from topology. Thus, our work belongs to *computational topology* [82, 244, 251], a growing subfield of computational geometry, and also has tight links with related communities such as topological graph theory and graph algorithms. As we shall see, the study of algorithms and combinatorial properties for graphs on surfaces is far from new and is represented in several disciplines. However, within the computational geometry and computational topology communities, this line of research emerged in the late 1980s and swarmed in the 2000s.

In the field of topology, surfaces have been successfully studied for more than one century and are very well-understood. Mathematical objects built upon surfaces, such as mapping class groups and Teichmüller spaces, are the subject of active research, as a useful step to consider more complicated topological objects such as three-manifolds, which are, in contrast, much deeper and not completely understood. While such contemporary mathematics are useful to know about for some of the problems we study, we almost exclusively build upon the very classical results on surfaces such as their classification up to homeomorphism and the properties of their homotopy groups. On the other hand, the algorithms we describe require new combinatorial and structural properties of curves on surfaces.

Why are algorithms for graphs on surfaces interesting? In contrast to planar graphs, *every* graph is embeddable on some surface; graphs on surfaces are natural generalizations of graphs embedded in the plane and share many combinatorial properties with them. When studying graphs embeddable on surfaces, we are not really restricting ourselves to a particular class of graphs; rather, graphs are classified according to their *genus*, or informally to the "simplest" surface they can be embedded in.

Not surprisingly, many general graph problems can be solved more efficiently using dedicated algorithms assuming that the input graph is planar. It turns out that many computational problems are solvable on graphs embedded on a fixed surface within the same asymptotic time as for planar graphs. In particular, such results apply to "almost-planar graphs", for instance, graphs obtained from planar ones by adding a few edges.

At a very high level, a general paradigm for solving such problems is to cut the surface to make it planar; this process transforms the graph into a planar one, allowing to apply algorithmic techniques for planar graphs. In other words, algorithms are needed to simplify the topology of the surface.

On another note, given a graph G, the data of an embedding of G on a surface defines topological properties that a closed walk in G may or may not have, such as being contractible or not; it also partitions the set of closed walks into *homotopy classes*, where two closed walks are in the same class if and only if one can be deformed continuously to the other on the surface. As we shall see, many standard questions on closed walks in graphs admit natural and interesting variants that take these properties into account.

Shortest curves with prescribed topological properties. Most of this document is devoted to the study of algorithms to compute shortest curves that satisfy some desired topological properties on a surface. This includes the problem of simplifying or decomposing the surface topologically, by cutting along curves drawn on the surface. Furthermore, we insist in computing *shortest* curves satisfying the desired conditions. Besides being a natural requirement—the shortest path problem is fundamental, and we are considering a topological variation of it—, this provides "canonical" curves satisfying these conditions, which also bear structural properties useful in other algorithms.

On the application side, simplifying or decomposing topologically a surface is useful in computer graphics and mesh processing, among others, to remove irrelevant topological features of a mesh or to parameterize a surface. In such applications, it is important to make the length of the curves as small as possible. For example, one would like to cut a surface obtained by a 3d scanner along short non-contractible closed curves, because the scanning process may create spurious handles, which need to be removed. When putting a texture on the surface of a three-dimensional object is desired, usually a parameterization step is needed, which requires cutting the surface; the cut should be as short as possible (with respect to some length function that is not necessarily the Euclidean distance), because artefacts in the texture may appear at such places. Some of the algorithms that we describe can be used for such purposes.

Structure of this document. The idea I had in mind when writing this habilitation thesis was to present my results on algorithms for graphs on surfaces, together with a glimpse at the main proof techniques. In particular, many proofs are sketched or omitted, and some results are presented in a weakened form; some variants are completely dismissed. The reader is referred to the research articles for more details.

This document is largely self-contained, and could hopefully serve as an introduction to computational topology of graphs on surfaces, albeit heavily biased towards my own research results.

Part I first presents the preliminary notions of surfaces and graphs drawn on them, together with some important topological notions such as homotopy and homology. Then we survey existing results about graphs on surfaces in various disciplines, mostly from a computational perspective.

In Part II, we consider the problem of computing shortest curves with certain topological properties on a surface; in that part, the material is somehow more developed, in particular in Chapter 5, where several results are presented in a unified way. We first consider the problem of computing shortest non-separating or non-disk-bounding (non-contractible) closed curves, because such curves bear important structural properties and are basic building blocks; we also describe algorithms for computing short cut graphs. Then, in Chapter 7, we describe algorithms to compute shortest curves with other prescribed topological properties: a shortest curve within a given homotopy class, or a shortest splitting closed curve.

In Part III, we show other algorithmic and combinatorial results related to graphs on surfaces, on isotopies between two embedded graphs, on cycles without repeated vertices on such a graph, and on the complexity of irreducible triangulations of surfaces.

The first chapter of Part IV surveys three of my recent results not related to surface-embedded graphs. All of them consist of new algorithms or combinatorial properties obtained with topological tools. They are concerned, respectively, with a topological variation of the notion of Fréchet distance to measure the similarity between curves, a graph algorithm to compute vertex-disjoint paths of minimum total length in a planar graph, and a topological extension of Helly's theorem with applications to geometric transversal theory. We conclude with some perspectives for further research in Chapter 12.

The chapters of this manuscript are quite independent, with the notable exceptions of Chapters 2, 4, and 5, which introduce concepts and algorithms used in many subsequent chapters.

Part I

Preliminaries and Survey on Graphs on Surfaces

CHAPTER 2

PRELIMINARIES

In this chapter, we present the main actors of this manuscript: surfaces, graphs, and embeddings, together with some topological notions. For more details, standard references on topology of surfaces include Armstrong [14], Henle [139], and Stillwell [229]. As usual, for a subspace X of Y, we denote by $\overline{X}$ the closure of X in Y and by ∂X its boundary.

2.1 Graphs

We use standard terminology for graphs, see, e.g., Diestel [90]. All the graphs we consider are finite and may have loops and multiple edges (i.e., they are actually *multigraphs*); they are undirected unless noted otherwise.

Let G be such a graph; $V(G)$ and $E(G)$ denote respectively the set of vertices and edges of G. Furthermore, if $V' \subseteq V(G)$ and $E' \subseteq E(G)$, then $G - V'$ denotes the graph G where the vertices in V' and their incident edges have been removed; furthermore, $G - E'$ denotes the graph G where the edges in E' have been removed.

A ***walk*** in G is an alternating sequence $v_0 e_0 v_1 e_1 \ldots e_{k-1} v_k$ $(k \geq 0)$ of edges and vertices of G, such that the initial endpoint of e_i is v_i and its final endpoint is v_{i+1}, for each i, $0 \leq i \leq k-1$. If $v_0 = v_k$, the walk is ***closed***.

2.2 Surfaces

2.2.1 Definition

A ***surface*** Σ, or compact 2-dimensional topological manifold with boundary, is a (Hausdorff) compact topological space in which every point has a neighborhood homeomorphic to the plane or to the closed half-plane. The points that have no neighborhood homeomorphic to the plane comprise the ***boundary*** of Σ. Equivalently, a ***surface*** is a topological space Σ obtained from finitely many disjoint triangles by gluing together some edges of the triangles by pairs; the boundary points of Σ arise from the edges of the triangles that are not identified with any other edge.

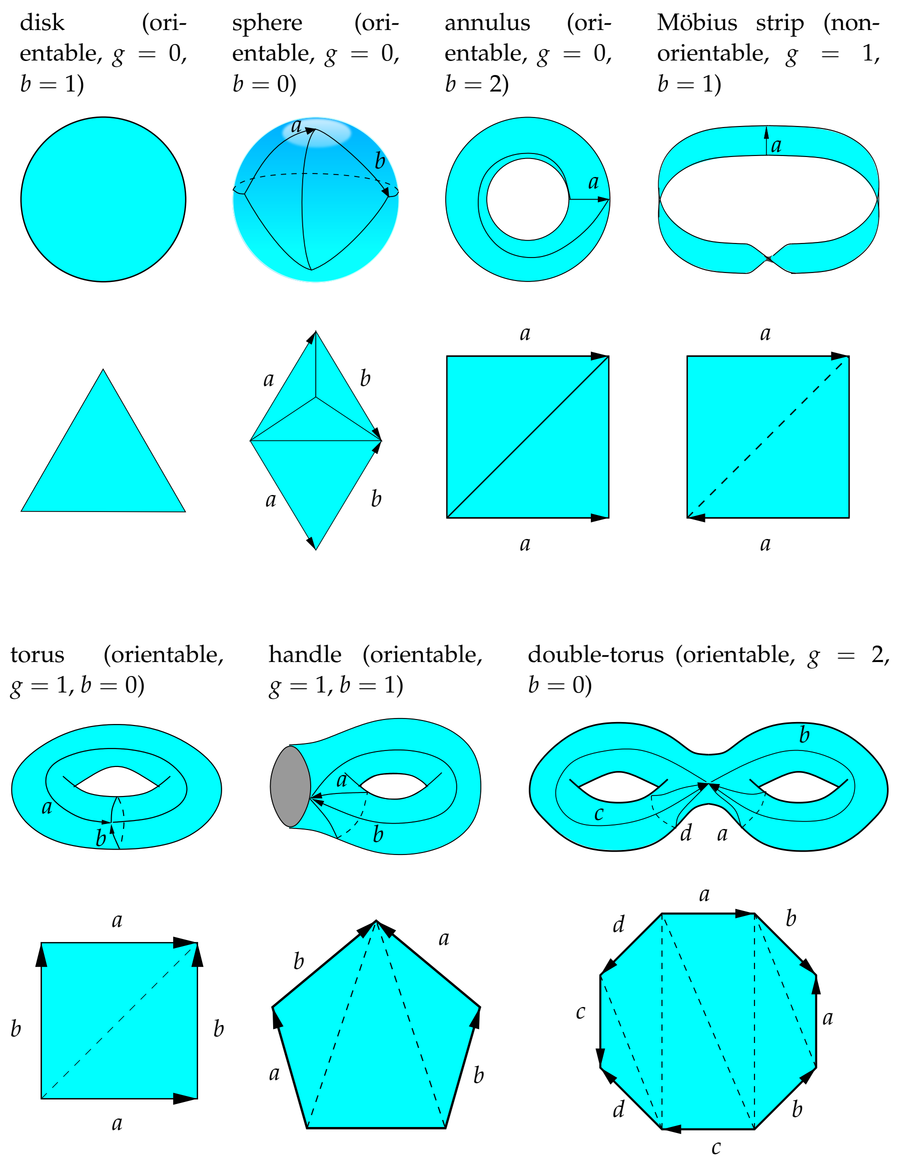

disk (orientable, $g = 0$, $b = 1$)
sphere (orientable, $g = 0$, $b = 0$)
annulus (orientable, $g = 0$, $b = 2$)
Möbius strip (non-orientable, $g = 1$, $b = 1$)
torus (orientable, $g = 1, b = 0$)
handle (orientable, $g = 1, b = 1$)
double-torus (orientable, $g = 2$, $b = 0$)
a
b
c
d

Figure 2.1. Examples of surfaces. Each surface (top rows) comes with a way to obtain it by gluing together triangles (bottom rows). Since we only deal with topological properties, the triangles on the surfaces may not be flat. In the bottom rows, some triangles are already glued together, and the pairs of edges labeled should be glued, respecting the orientation of the arrows. In the bottom rows, some edges are dashed; these edges are not shown in the corresponding surface, for clarity. The genus g and number of boundary components b is specified, as well as whether the surface is orientable.

The first definition is topological, while the second is more combinatorial. The equivalence between both definitions is non-trivial and due to Kerékjártó [161] and Radó [206]. Henceforth, we always assume additionally that a surface is, by definition, ***connected***.

2.2.2 Examples

Examples of surfaces include the disk, the sphere, the annulus (or cylinder), the Möbius strip, the torus, the handle (or torus with one open disk removed), and the double-torus; Figure 2.1 shows these surfaces, together with possible gluings of triangles to obtain them. In particular, note that the boundary of the Möbius strip is connected. In most cases, we consider surfaces up to homeomorphism; for example, a ***disk*** is just a topological space homeomorphic to the standard closed disk. We emphasize that the surfaces we consider need not be embedded in $\mathbb{R}^3$ (actually, the non-orientable surfaces without boundary cannot be embedded in $\mathbb{R}^3$), but should be regarded as abstract topological spaces.

2.2.3 Classification

A surface is ***orientable*** if it does not contain a subspace homeomorphic to a Möbius strip. The following *classification theorem* describes the homeomorphism classes of surfaces.

Theorem 2.1. *Each surface is homeomorphic to exactly one of the following surfaces:*

- *the orientable surface of* **genus** $g \geq 0$ *with* $b \geq 0$ **boundary components**, *obtained from the sphere by removing g open disks with disjoint closures, attaching a handle to each of the resulting g circles, and finally removing b open disks with disjoint closures;*
- *the non-orientable surface of* **genus** $g \geq 1$ *with* $b \geq 0$ **boundary components**, *obtained from a sphere by removing g open disks with disjoint closures, attaching a Möbius strip to each of the resulting g circles, and finally removing b open disks with disjoint closures.*

Thus a surface is uniquely determined by its genus, by its number of boundary components, and by whether it is orientable.

2.3 Curves and Embedded Graphs

Now we consider curves and graphs drawn on a surface Σ. To avoid topological pitfalls, we assume that every surface is defined by a gluing of triangles, as explained above, and that all curves are piecewise-linear with respect to that triangulation.

2.3.1 Paths, Loops, and Cycles

A ***path*** on Σ is a continuous map $p : [0,1] \to \Sigma$ that is ***piecewise-linear*** with respect to the triangulation of Σ: It intersects the edges of the triangulation finitely many times, and it is piecewise-linear when restricted to a subinterval whose image lies in a single triangle. The ***endpoints*** of p are $p(0)$ (its ***initial endpoint***) and $p(1)$ (its ***final endpoint***). A ***loop*** is a path whose endpoints coincide; that common endpoint is the ***basepoint*** of the loop. A ***cycle*** is a piecewise-linear continuous map from the circle $\mathbb{S}^1 = \mathbb{R}/\mathbb{Z}$ to Σ; thus, a cycle is the same as a loop, except that a cycle has no distinguished basepoint.

A path or cycle is ***simple*** if it is one-to-one; a loop is ***simple*** if its restriction to $[0,1)$ is one-to-one. Here we would like to warn the reader that the term *path* is used in the topological sense: In general, a path may self-intersect. In contrast, for graph theorists, paths in graphs have no repeated vertex.

A ***curve*** is a path (possibly a loop) or a cycle. By abuse of language, we sometimes identify a curve with its image on Σ. Also, the exact parameterization of a curve does not matter; for example, a path p can be regarded as equivalent to $p \circ \varphi$, where $\varphi : [0,1] \to [0,1]$ is bijective and increasing.

The ***reversal*** of a path p is the path $q : [0,1] \to \Sigma$ defined by $q(t) = p(1-t)$. The ***concatenation*** $\boldsymbol{p \cdot q}$ of two paths p and q such that $p(1) = q(0)$ is the path r defined by

$$r(t) = \begin{cases} p(2t) & \text{if} \quad t \leq 1/2 \\ q(2t-1) & \text{if} \quad t \geq 1/2. \end{cases}$$

2.3.2 Graph Embeddings

Let G be a graph. An *embedding* of G on Σ is, intuitively, a crossing-free drawing of G on Σ. More precisely, G can be viewed as a topological space, by taking a disjoint union of segments, one segment per edge of G, and identifying the endpoints of the segments where the corresponding edges share endpoints. An ***embedding*** of G is a continuous, piecewise-linear, one-to-one map from G (viewed as a topological space) into Σ. (Piecewise-linearity means that the restriction of the embedding to each edge is piecewise-linear.) When no confusion can arise, we sometimes identify G with its embedding on Σ, or with the image of that embedding.

Let G be a graph embedded on Σ. The ***faces*** of G are the connected components of the complement of the image of G in Σ. The set of faces of G is denoted by $\boldsymbol{F(G)}$. A vertex or edge is ***incident*** to that face if it lies on the boundary of that face. The ***degree*** of a face is the number of edges incident with that face, counted with multiplicity (it may happen that an edge has the same face on both sides).

Assume now that G has no isolated vertex. An important consequence of the piecewise-linearity of the embeddings is that ***cutting*** Σ along G is a well-defined operation that results in one or several surfaces. The faces of G are in bijection with these surfaces, and are actually homeomorphic to the interiors of these surfaces. We sometimes use the notation $\boldsymbol{\Sigma \backslash\backslash G}$ to denote the surface Σ cut along the graph embedding G. If $\Sigma \backslash\backslash G$ is a disk, then G is a ***cut graph*** of Σ.

For example, the (easier piecewise-linear version of the) Jordan–Schönflies theorem asserts that cutting a sphere along a simple cycle yields two disks. In general, cutting a surface along a simple cycle γ yields one or two surfaces. If it yields two surfaces, γ is ***separating***. If, furthermore, (at least) one of these two surfaces is a disk, γ is ***disk-bounding***.

We also say that a graph is ***planar*** if and only if it can be embedded in the plane, or, equivalently, in the sphere. By abuse of language, we also say that a graph is planar if it comes with an actual embedding in the plane.

2.4 Cellular Graph Embeddings

A graph embedding is ***cellular*** if all its faces are open disks.[1] On every surface, there exists a cellular graph embedding. Furthermore, any graph that embeds cellularly on a surface is connected. In this section, we discuss representations and properties associated with cellular graph embeddings.

2.4.1 Combinatorial Representations

Let us assume for now that Σ has no boundary; let G be a graph cellularly embedded on Σ. In general, we are not interested in the actual geometric position of G on Σ; for example, it is irrelevant to consider the description of the vertices and edges of G in terms of coordinates in a fixed triangulation of Σ. Therefore, we need a combinatorial description of cellular graph embeddings. Essentially, it suffices to specify the way the faces of G are glued together on the surface; for example, if the graph has no loop and the surface is oriented, it suffices to store the cyclic order of the edges around each vertex on the surface. Such combinatorial representations of cellularly embedded graphs are usually called *maps* [169], although essentially equivalent variants appear in the litterature, like *rotation systems* [191] or *fat graphs* [126].

2.4.2 Data Structures

However, to develop efficient algorithms on cellular graph embeddings, it is necessary to describe actual data structures. Here also, there exist many possibilities, all of which are more or less equivalent; a popular choice is the *doubly-connected edge list* [74], or the closely related *halfedge data structure* of the CGAL computational geometry algorithms library[2]; see Kettner [162] for a comparison of such data structures for orientable surfaces, with implementation details. We choose a representation close to the *graph-encoded map* [100,179]. Two advantages are that it can represent non-orientable surfaces, and that the representation is canonical (for example, it does not depend on an orientation of the surface, if the surface is orientable). We still assume that Σ has no boundary for simplicity of exposition.[3]

The basic building block is the ***flag***, which represents an incidence between a vertex, an edge, and a face of the embedding; see Figure 2.2. Each edge e bears four flags. If e is incident to two different vertices v_1 and v_2, two flags are incident to v_1 and the two others are incident

[1] In particular, the boundary of Σ has to be entirely covered by the image of G.

[2] http://www.cgal.org

[3] We also assume that the graph is not reduced to a single vertex, which may in principle be the case if Σ is a sphere.

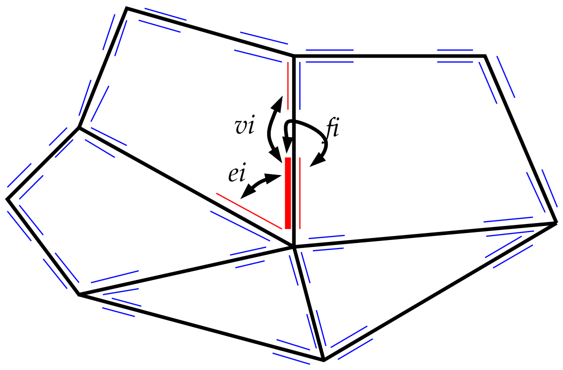


Figure 2.2. A part of a cellular graph embedding on a surface. The flags are represented as short line segments parallel to the edges; there are four flags per edge. The involutions *vi*, *ei*, and *fi* on the thick flag are also shown.

to v_2. Similarly, if e is incident to two different faces f_1 and f_2, two such flags are incident to f_1 and the two others are incident to f_2. If these two assumptions are satisfied for all edges e of G (which is not the case in general, e.g., when the graph contains a loop), then a flag is uniquely determined by the data of its incident vertex, edge, and face.

Three involutions without fixed point[4] allow to move to a nearby flag, and, by iterating the process, to visit the whole cellular graph embedding. If φ is a flag,

- $vi(\varphi)$ is the flag with the same edge-face incidence as φ, but with a different vertex incidence;
- $ei(\varphi)$ is the flag with the same vertex-face incidence as φ, but with a different edge incidence;
- $fi(\varphi)$ is the flag with the same vertex-edge incidence as φ, but with a different face incidence.

In an actual implementation, each flag φ can be represented as an object storing three pointers to $vi(\varphi)$, $ei(\varphi)$, and $fi(\varphi)$.

All our analyses of complexity of algorithms are implicitly done in the *real RAM* model [3], as is customary in computational geometry. (For the algorithms that do not manipulate real numbers, the RAM model would be sufficient.) In particular, for all our purposes, we can assume that a pointer can be stored in constant space and can be used in constant time; also, a real number can be stored in constant space, and arithmetic operations on real numbers take constant time. It follows that our data structure has a size that is linear in the number of edges of the graph.

Within the data structure described above, all reasonable operations on the embedding can be performed efficiently: For example, one can easily traverse the vertices, edges, and faces of G in $O(|E(G)|)$ time (since G is cellularly embedded and thus connected). Given a flag φ incident to a vertex v, one can determine the degree of v in linear time in that degree:

[4]An involution without fixed point is a map $f : X \to X$ such that f has no fixed point but $f \circ f$ is the identity.

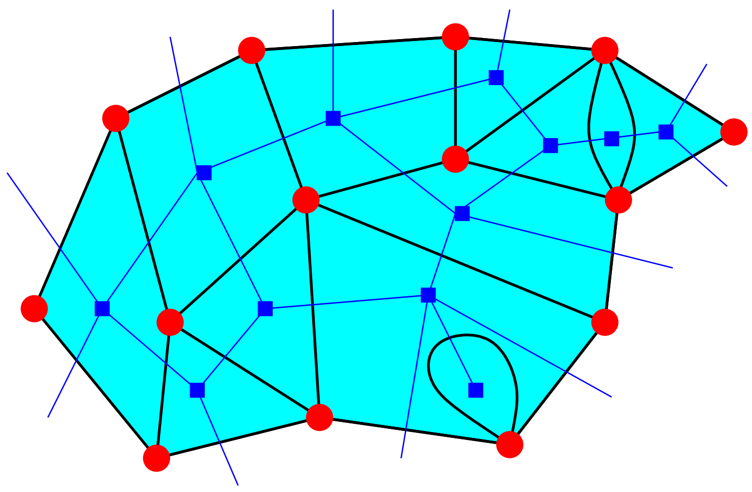


Figure 2.3. Duality. One of the graphs (primal or dual) is depicted with rounded vertices, and the other with squared vertices. Note that the graphs may have loops and multiple edges.

It is the number of iterations of the map $ei \circ fi$ that are needed, starting on φ, to reobtain φ. Similarly the degree of a face can be computed (with $ei \circ vi$ instead).

Each flag also has a pointer to the underlying vertex, edge, and face (stored as separate objects). This allows to test in $O(1)$ time whether two flags are incident to the same vertex, edge, or face. Storing additional information in the vertices, edges, and faces is also useful in algorithms. It is also possible to store information in the vertex-edge, edge-face, vertex-face, and vertex-edge-face incidences, though we will not get at this level of detail here. Note that the complexity of the data structure is linear in the number of edges of G.

If the surface Σ is oriented, we can define the ***orientation*** of a flag φ intuitively as follows: Start from the vertex incident to φ, move along the edge incident to φ, and turn towards the face incident to φ. The flag φ has clockwise or counterclockwise orientation depending on the direction of the turn. For example, the thick flag in Figure 2.2 is oriented counterclockwise. Note that, with this definition, each involution vi, ei, and fi *reverses* the orientation of a flag. To determine in linear time whether the surface is orientable, one can choose an arbitrary orientation of a flag and propagate the orientation information to each flag via the involutions; this can be done consistently if and only if the surface is orientable.

2.4.3 Duality

Let G be a graph cellularly embedded on a surface Σ without boundary. A ***dual graph*** of G is a graph embedding G^* defined as follows: Put one vertex f^* of G^* in the interior of each face f of G; for each edge e of G, create an edge e^* in G^* crossing e and no other edge of G (if e separates faces f_1 and f_2, then e^* connects f_1^* and f_2^*). See Figure 2.3.

A dual graph embedding is also cellular. The combinatorial representation of the dual graph is unique. Actually, with our above data structure, dualizing is easy: Simply replace *fi* with *vi* and vice-versa. This in particular proves that duality is an involution: $G^{**} = G$.

2.4.4 Euler's Formula

Let Σ be a surface with genus g and b boundary components. ***Euler's formula*** states that, for every graph G cellularly embedded on Σ with $v = |V(G)|$ vertices, $e = |E(G)|$ edges, and $f = |F(G)|$ faces, the quantity $v - e + f$ does not depend on G, but merely on the surface Σ: It equals $2 - 2g - b$ if Σ is orientable, and $2 - g - b$ if Σ is non-orientable. It is sometimes useful to define the ***Euler genus*** $\tilde{g}$ of Σ as $2g$ if Σ is orientable and g if Σ is non-orientable, in which case Euler's formula rewrites as $v - e + f = 2 - \tilde{g} - b$, regardless of the orientability character of Σ.

Let G be a graph cellularly embedded on Σ. Euler's formula directly implies that the genus of Σ and the complexity of G are (asymptotically) bounded from above by the number of edges of G. Conversely, it is often useful to bound the number of edges of a graph in terms of its number of vertices. Let G be a graph embedded (not necessarily cellularly) on a surface Σ. For an integer $k \geq 1$, a ***k-gon*** of G is a face of G of degree k that is a disk. A ***monogon*** is a 1-gon; a ***bigon*** is a 2-gon.

Lemma 2.2. *Let G be a graph embedded on a surface of genus g without boundary. If G has no monogon or bigon, then $|E(G)| = O(|V(G)| + g)$.*

Proof. If G is cellularly embedded, this follows from Euler's formula and from the relation $2|E(G)| \geq 3|F(G)|$, the latter being obtained by double-counting the incidences between the edges and faces of G and using the absence of monogons and bigons.

In the general case, one can augment G to a cellular graph G' by adding only edges, without creating monogons or bigons, and apply the previous reasoning to G'. □

In particular, note that if G is a simple graph (without loops or multiple edges), then G has no monogon or bigon, so the hypotheses of the lemma are fulfilled.

2.5 Homotopy and Homology

2.5.1 Homotopy

Informally, two paths on Σ are ***homotopic*** if one can be deformed into the other continuously without moving the endpoints. More formally, a ***homotopy*** between two paths p_0 and p_1 is a continuous map $h : [0,1] \times [0,1] \to \Sigma$ such that $h(0,\cdot) = p_0$, $h(1,\cdot) = p_1$, and $h(\cdot,0)$ and $h(\cdot,1)$ are constant maps. In such a case, p_0 and p_1 are ***homotopic***; in particular, they have the same initial and final endpoints. Being homotopic is an equivalence relation.

This notion applies in particular to loops with a given basepoint b. The homotopy classes of loops with basepoint b form a group, called

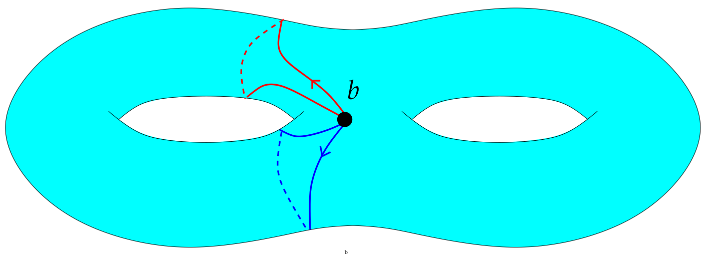


Figure 2.4. These two loops with the same basepoint b on a double-torus are not homotopic. However, when regarded as cycles, they are freely homotopic, since one can be detached away from b, moved "around the left handle", and finally pushed onto the other cycle.

the ***fundamental group***, or ***homotopy group***, of Σ (the definition does not actually depend on b): The unit element is the class of the constant loop, the multiplication corresponds to the concatenation of two loops, and the inverse operation corresponds to the reversal of a loop. A loop is ***null-homotopic***, or ***contractible***, if it is homotopic to the constant loop.

A ***homotopy*** between two cycles γ_0 and γ_1 is a continuous map $h : [0,1] \times S^1 \to \Sigma$ such that $h(0,\cdot) = \gamma_0$ and $h(1,\cdot) = \gamma_1$. Figure 2.4 highlights a difference between homotopy of loops and cycles: Two loops can be non-homotopic while the corresponding cycles are homotopic, because in homotopy of loops, the basepoint is not allowed to move. In contrast, homotopy of cycles is sometimes called ***free homotopy*** because no point is fixed.

We will also occasionally encounter the concept of ***isotopy***. An isotopy between two *simple* paths or cycles is a homotopy h such that $h(t,\cdot)$ is simple for each $t \in [0,1]$. An ***isotopy*** between two homeomorphisms from Σ to Σ is a continuous family of homeomorphisms.

2.5.2 Homology

Some results of Chapter 5 rely on the concept of *homology*. For convenience, we define this concept in the simplest framework that is sufficient for our purposes: *one-dimensional cellular homology on surfaces over the field* $\mathbb{Z}/2\mathbb{Z}$. Furthermore, we restrict ourselves to a surface Σ without boundary.

Cycles, boundaries, and homology. Let Γ be the *image* of a graph G embedded on Σ. We say that Γ is a ***homology cycle*** if every vertex of G has even degree. We say that Γ is a ***homology boundary*** if the faces of G can be colored in two colors, say black and white, such that G is the "boundary" of the black faces: Each edge of G is incident to a black face and a white face. (These notions are well-defined, because G is uniquely defined by Γ, up to the insertion or deletion of vertices of degree two on edges of G.) Clearly, a homology boundary is a homology cycle, as in a homology boundary, the colors black and white must alternate around any vertex, and therefore every vertex has even degree.

Let Γ and Γ' be two images of embedded graphs on Σ. Since the embeddings are piecewise-linear, the symmetric difference of Γ and Γ' is the image of an embedded graph, called their ***sum***, denoted by $\Gamma + \Gamma'$. Moreover, the sum of two homology cycles is again a homology cycle (as the symmetric difference of two images of embedded graphs with only even degrees is the image of an embedded graph with only even degrees). It is also not hard to see that the sum of two homology boundaries is again a homology boundary.

In other words, the sets Z of homology cycles and B of homology boundaries form vector spaces over the field $\mathbb{Z}/2\mathbb{Z}$. (The multiplication by one is the identity, and the multiplication by zero always gives the empty set.) Moreover, B is a vector subspace of Z. The ***homology*** vector space of Σ, denoted by $\boldsymbol{H_1(\Sigma)}$, is now simply the vector space that is the quotient of Z by B.

We can define the homology class of a loop or cycle γ as follows. The image of γ is the image of an embedded graph where each vertex has even degree; the homology class of γ is the homology class of that (image of) embedded graph.

Application and extension. So far, this is just an algebraic construction. To illustrate one possible use of homology in our context, consider a simple cycle γ on Σ. We claim that γ is separating if and only if the homology class of (the image of) γ is zero. Indeed, γ is a homology cycle. If γ is separating, it splits Σ into two pieces; coloring one piece in black and one piece in white shows that γ is a homology boundary, hence homologically trivial. Conversely, if γ is non-separating, there are exactly two possible colorings of the faces of γ: One can color the unique face in black or in white; in either case, γ obviously cannot be incident to a white face and a black face. So γ is not a homology boundary, and is therefore homologically non-zero.

Relative homology with respect to a finite set P of points of Σ is defined similarly, except that a homology cycle is defined as the image Γ of a graph G in which every vertex of G has even degree, except possibly for those vertices that belong to a point in P in the embedding. This yields the ***relative homology*** vector space $\boldsymbol{H_1(\Sigma, P)}$. The algebraic properties are similar; we omit the details.

CHAPTER 3

GRAPHS ON SURFACES: A SHORT SURVEY

In this chapter, we survey the numerous incarnations of graphs on surfaces in various disciplines. To keep this chapter at a reasonable size, we focus on intrinsic topological surfaces, and omit results considering surfaces embedded in $\mathbb{R}^3$ or in a 3-manifold. Furthermore, we omit the vast literature on planar graphs. Finally, the chapter is biased towards algorithmic results. To avoid repetition, some of the results that are the closest to our work are not cited here, but will be cited in forthcoming chapters.

Also, our survey is deliberately focused on theoretical aspects; we omit occurrences of graphs on surfaces in applied fields. Let us only mention here that some topological algorithms developed in the subsequent chapters are especially relevant for a variety of applications (a more detailed argumentation can be found in the author's Ph.D. thesis [F]): computer graphics and geometry processing, for texture mapping [177, 205], surface correspondence [178], and morphing [171]; mesh processing, for parameterization [128], remeshing [5], topological simplification [129, 247], approximation [62], and compression [6]. Other related application areas include computer-aided geometric design, geometric modeling, and topological data analysis [252]. Techniques for general surfaces apply in particular to surfaces of genus zero with boundaries, and are thus useful in VLSI design [176] and map simplification [46, 75].

The present chapter is split into three sections. In the first one, we consider graphs on surfaces in non-algorithmic fields (topological graph theory, topology, and enumerative combinatorics). Then we describe algorithms for solving problems with a topological flavor on graphs on surfaces; most of the remaining results of this manuscript fall into this category. Finally, we consider algorithms for solving general graph problems, in the special case where the input graph is embedded on a surface. We implicitly only consider surfaces without boundary for simplicity of exposition.

3.1 Graphs on Surfaces in Non-Algorithmic Fields

3.1.1 Topological Graph Theory

In combinatorics, one of the pillars of *topological graph theory* is the investigation of structural properties of graphs on surfaces. We can only refer to existing books and surveys by Mohar and Thomassen [191], Gross and Tucker [126], or Archdeacon [13] for details, and describe a few themes in the field.

The (orientable) ***genus*** of a graph G is the minimum genus of an orientable surface on which G can be embedded; an embedding of minimum genus is necessarily cellular. There are also the variants of the ***Euler genus*** (the minimum Euler genus of a surface on which G can be embedded) and the ***non-orientable genus*** (restricting to non-orientable surfaces). These are important concepts, and the genera of some graphs like the complete graphs are known; see, e.g., Archdeacon [13, Section 4.2] or Mohar and Thomassen [191, Section 4.4]. Similarly, the evolution of the genus of a graph under simple modifications of the graph has been studied [12, 187].

Another line of research in this area is the definition of polynomials for surface-embedded graphs, which are variations of the Tutte polynomial and enjoy the same properties as that polynomial, but depend on the embedding of the graph on a surface [32, 99]. Determining the number of curves in different homotopy classes that can pairwise intersect a bounded number of times is another interesting (and largely open) problem [10, 149, 183, 199].

Graph coloring on surfaces has also been much studied; see, e.g., Mohar and Thomassen [191, Chapter 8] for a survey. In particular, for every surface Σ of positive genus, the number of colors needed to color the vertices of any graph on Σ, in such a way that adjacent vertices have different colors, is known; see Ringel [210]. For graphs on Σ that are sufficiently "locally planar", the number of colors needed drops to five [239]. Surprisingly, the general problem is harder in the plane, but Appel and Haken solved it with the famous four-color theorem [11].

Last but not least, one of the main reasons for studying graphs embeddable on a fixed surface is that they form a minor-closed family. A simple graph H is a ***minor*** of another simple graph G if H can be obtained from G by removing edges and contracting edges (after which loops are removed and multiple edges are identified). A family of graphs $\mathscr{F}$ is ***minor-closed*** if every minor of a graph in $\mathscr{F}$ is also in $\mathscr{F}$. This leads us to the theory of Robertson and Seymour on graph minors, one of the most important results in graph theory. In particular [214], for each surface Σ, there exists a finite, very large but theoretically computable, set of graphs $\mathscr{O}_\Sigma$ such that a graph G is embeddable on Σ if and only if no graph in $\mathscr{O}_\Sigma$ is a minor of G (see Mohar and Thomassen [191, Chapter 6] for a proof of this result). Graphs on surfaces play also a central role in the proof of the more general *graph minor theorem* of Robertson and Seymour, stating that every minor-closed family is characterized by a *finite* set of excluded minors. More precisely, a crucial ingredient of that result is a structure theorem for H-minor-free graphs [213], the statement of which involves graphs embedded on (possibly non-orientable) surfaces in a crucial way.

3.1.2 Topology

Many topology papers study graphs and curves on surfaces. Most, if not all, textbooks in topology include the classification of surfaces (Theorem 2.1) [14, 49, 139, 229]; the proof of this classification theorem consists of transforming an embedded graph into a canonical form with some elementary operations.

Various interesting mathematical structures can be defined on surfaces and have been much studied. The ***mapping class group*** of an orientable surface Σ is, roughly, the group of isotopy classes of orientation-preserving self-homeomorphisms of Σ (see Farb and Margalit [116] for a recent book on this topic). In other words, one regards two such self-homeomorphisms as equivalent if one can be deformed into the other. Thus, the definition of the mapping class group is in the same spirit as the one of the fundamental group, where two homotopic loops are regarded as equivalent. The ***complex of curves*** (see Schleimer [224]) of a surface Σ is (essentially) the simplicial complex with vertex set the set of isotopy classes of simple cycles; a subset of vertices forms a simplex if and only if the corresponding isotopy classes can be realized by pairwise disjoint cycles.

A ***pair of pants*** is an orientable surface of genus zero with three boundary components; a ***pants decomposition*** of a surface Σ is a family of simple, pairwise disjoint cycles Γ such that $\Sigma \backslash\backslash \Gamma$ is a disjoint union of pairs of pants. The ***pants complex*** [134] is a two-dimensional cell complex whose vertices are the pants decompositions of a surface; two pants decompositions are connected with an edge if they differ by one of two elementary moves that change exactly one cycle in the decomposition; and similarly two-cells can be defined using more complicated operations. Variations on this idea exist [22].

Normal curves constitute a compact way of representing families of simple, pairwise disjoint cycles on a given surface Σ. Essentially, given a triangulation of Σ, such a family of cycles can be represented by recording its number of crossings with each edge of the triangulation. If no cycle crosses twice a given edge consecutively in opposite directions, the isotopy class of the family is uniquely determined. Actually, this representation is the toy version of ***normal surfaces***, used in particular in algorithms for 3-manifold and unknot recognition [132]. Other kinds of representations of curves on surfaces are *train tracks* [192, 201] and *Dehn-Thurston coordinates* [240].

A very important feature of surfaces is that they can be endowed with a Riemannian metric of constant curvature. Assume that Σ is an orientable surface of genus g without boundary. Then, depending on whether $g = 0$, $g = 1$, or $g \geq 2$, the surface Σ can be endowed with a

Riemannian metric of constant curvature $+1$ (spherical), 0 (flat), or -1 (hyperbolic). Moreover, if a metric is given on Σ, that metric is conformally equivalent to a metric of constant curvature: This is called the ***uniformization theorem***.

Related and central is the notion of ***Teichmüller space*** of a surface Σ. Assuming $g \geq 2$, this is roughly the set of all "essentially different" hyperbolic metrics one can put on Σ (in the following natural but technical sense: two hyperbolic metrics on Σ are "essentially different" if there exists no self-diffeomorphism of Σ isotopic to the identity that maps one metric to the other). This set can naturally be endowed with a topology, and is homeomorphic to $\mathbb{R}^{6g-6}$. Explicit coordinates can be given. Teichmüller spaces are strongly related to mapping class groups [116]. See also the classical exposition of Thurston's works on surfaces [117] for more details.

Three-dimensional topology is a deep and rich field of study; Perelman's resolution of the Poincaré conjecture [202–204] is a clear indication of the current importance of the field. Understanding structural properties for curves on surfaces can be an important ingredient for discovering more elaborate results in three-dimensional topology. For example, a Heegaard splitting of a 3-manifold is obtained by attaching two handlebodies along the same orientable surface. Thus, the resulting 3-manifold can be understood by the way the gluing is done. That gluing can be expressed by a homeomorphism of the surface, and is in particular a motivation for studying mapping class groups and pants decompositions [118].

3.1.3 Enumerative, Bijective, and Asymptotic Combinatorics

The problem of counting maps with given properties in the plane or on surfaces has been much studied. The field was initiated by Tutte [241] in the special case of the plane, and includes numerous developments in the case of positive genus, one of the earliest one being by Bender and Canfield [19], who study generating series associated to the number of maps on a surface of genus g.

More recently, bijective results were found on the number of rooted maps or quadrangulations of surfaces [56] and of maps representing cut graphs on surfaces [55], as well as refined asymptotic estimates in some cases [20]. (Usually, one considers *rooted* maps to factor out possible symmetries of maps.) One ultimate goal is to define a well-behaved random model for maps on surfaces. It is possible to describe the asymptotic number of such maps and some topological and metric properties of scaling limits of random quadrangulations [27, 186].

Matrix integral techniques are very different methods, originally from two-dimensional quantum gravity in theoretical physics, to enumerate such maps on a surface. Although these methods apparently need hard work to be made rigorous, they are very powerful; the idea is that certain Gaussian integrals over the space of Hermitian matrices can be interpreted as sums of some quantities over all maps with given properties, for example all cut graphs with a given number of edges on a given surface. Lando and Zvonkin [169, Chapter 3] and Zvonkin [253] are good introductions to this topic.

3.2 Algorithms for Topological Problems

The last few decades saw the development of algorithms for topological problems for curves and graphs on surfaces. This is also one of the directions of *computational topology*, a recent subfield of computational geometry aiming at revisiting topological questions through an algorithmic lens and at focusing on applications of topology in computer science. See e.g., Dey et al. [82], Vegter [244], or Zomorodian [251] for surveys of the field with various points of view; for a more in-depth introduction to computational topology, we recommend the course notes by Erickson [104].

Thomassen [237] proves that computing the genus of a graph is NP-hard; some hardness and approximation results are known [60]. On the other hand, for every fixed integer g, there exists a linear-time algorithm that embeds an input graph into a surface of genus g, or "certifies" that it is impossible [158, 189], generalizing the planarity test problem [143]. At the other extreme, determining the *maximum* genus of a surface on which a graph can be cellularly embedded is feasible in polynomial time [123].

The (planar) ***crossing number*** of a graph G is the minimum number of crossings of a drawing of G in the plane. Like the genus, it can be seen as a measure of the "topological complexity" of a graph. There are several variants on this notion, and connections between them are known [200]. Determining the crossing number of a graph is NP-hard, even in very restricted, "nearly-planar", cases of a planar graph with an additional edge [48], where the genus is either zero or one. On the positive side, for fixed k, there is a linear-time algorithm to determine whether a graph has crossing number at most k [159]; there are also recent approximation algorithms [68]. One can also define the crossing number of a graph on a surface different from the plane; in this setting also, approximation algorithms have been found recently [142].

The following questions are fundamental, and were already tackled by Dehn [77] at the beginning of the 20th century: Determine whether a given closed walk on a surface-embedded graph is contractible, or whether two given closed walks are homotopic. These are closely related to two central problems of *combinatorial group theory*, where the corresponding algebraic problems are the ***word problem*** and the ***conjugacy problem*** respectively: Determine whether an element of a group (given by generators and relations) is trivial, and whether two such elements are conjugates. In our context, the group considered is the fundamental group of a surface (made of $2g$ generators and a single relation, for an orientable surface without boundary). After a series of intermediate results [87, 222, 223], these problems were shown to be solvable in optimal linear time by Lazarus and Rivaud [173] (see also Dey and Guha [84]). The most basic algorithm for this problem, Dehn's algorithm [77], has been largely generalized to other groups; see, e.g., Lyndon and Schupp [182, Chapter 5]. The special case of the plane with obstacles, where the input is a set of obstacle points and one or two piecewise-linear curves avoiding these points, has also been investigated [24,25,46]; these papers consider explicitly only the case of homotopy of paths, but some of these results extend with little modifications to the free homotopy setting.

A quite old topic, also connected to combinatorial group theory, is the study of algorithms to determine whether a given set of cycles can be moved by a homotopy so that they become simple and/or pairwise disjoint. This has been a large subject of investigation [29,65,66,69,181, 209,235].

There are several polynomial-time algorithms to deal with families of curves encoded by normal coordinates [113,219,220], for example, to compute the number of curves in such a representation. Such results are impressive because the normal coordinates are compact: They can be exponentially smaller than a naïve representation that would record the ordered list of crossings of the edges of the triangulation along the curves of the family.

There are also various computational topology results that explore topological concepts like Reeb graphs and Morse theory in the special case of surfaces [1,70,96].

3.3 Graph Algorithms for Embedded Graphs

In this section, we consider graph problems that can be solved more efficiently when the graph is embedded in a low-genus surface.

First, the theory of Robertson and Seymour, and the concepts and properties of tree-width, branch-width, and relatives, have algorithmic implications for graphs embedded on a fixed surface; see, e.g., Bienstock and Langston [28] or Kawarabayashi and Mohar [156]. Examples of algorithmic problems on graphs on surfaces exploiting successfully these concepts include graph [157] and subgraph [33] isomorphism, contraction-checking [152], induced cycles [163], and optimization connectivity problems like minimum Steiner tree and traveling salesman problems for graphs of small branch-width using dynamic programming [216] and other decompositions [34,80].

As mentioned above, for a fixed surface Σ, there exists a finite set of graphs $\mathscr{O}_\Sigma$ such that a graph embeds on Σ if and only if it has no minor in $\mathscr{O}_\Sigma$. One strong motivation for developing algorithms for surface-embedded graphs is that they can be viewed as an indication of the difficulty of building algorithms for H-minor-free graphs, in the same way that algorithms for planar graphs can sometimes be generalized to graphs on surfaces, possibly using very different techniques. There are indeed results for graphs embedded on any fixed surface that generalize to H-minor-free graphs, for any fixed H. For example, the existence and linear-time construction of small separators for planar graphs by Lipton and Tarjan [180] has been extended to graphs embeddable on a fixed surface [4,100,124], and has then been shown to "almost" hold (i.e., with a near-linear complexity) for H-minor-free graphs, for any fixed H [160]. Shortest paths can be computed in linear time in planar graphs [140], and the approach extends to H-minor-free graphs [236]. There are other examples of extensions of results on surface-embedded graphs to H-minor free graphs [52,78,215].

The recent *bidimensionality theory* provides efficient approximation or fixed-parameter tractable algorithms for a host of graph problems on some graph classes; such classes include the families of H-minor-free graphs and bounded-genus graphs. This is a vast and developing theory, so we refer to Demaine and Hajiaghayi [79] for a survey; results of this theory for graphs of bounded genus include efficient fixed-parameter algorithms and kernelization results for domination, vertex cover, matching, and feedback vertex set problems [78, 120], and subexponential algorithms for Hamiltonian cycle and related problems [92].

There exist other efficient approximation algorithms for graphs on surfaces that rely on random partitions [175] and stochastic planarization with low distortion [35,145,227].

As mentioned earlier, some algorithms of Part II allow to compute shortest curves or graphs that simplify the topology of the surface. Such curves are used as building blocks for computing efficiently maximum flows [53], minimum s–t cuts [51], global minimum cuts [108], replacement paths [110], traveling salesman tours [196], approximate distance oracles [154], bipartite matchings [73], and some expansion parameters [198] for graphs on surfaces. Such techniques have proven useful also for planar graph problems [105,112].

Finally, other results on surface-embedded graphs include computing the girth of the graph [91], vertex-disjoint paths and cycles [226], possibly in prescribed homotopy classes [225] (though in the latter, no explicit algorithm is given). Rather different problems are concerned with the compact encoding of a surface mesh [50], the maintenance of a minimum spanning tree of an evolving graph on a surface [100], and the computation of the genus of a surface-embedded graph by a simple randomized, local process [21].

Part II

Shortest Curves and Graphs on Surfaces

Chapter 4

Models: Combinatorial and Cross-Metric Surfaces

The material of this chapter was first introduced in an article co-authored with Jeff Erickson [G].

This part contains our algorithmic results for computing shortest paths, cycles, and graphs with prescribed topological properties on surfaces. How can we efficiently compute shortest non-null-homotopic or non-null-homologous cycles, or shortest paths or cycles homotopic to given paths or cycles? How can we compute shortest cut graphs on a given surface?

One difficulty for developing algorithms for such purposes is that one needs a convenient framework to describe the input and output of the algorithms. In this chapter, we introduce the discrete models used in our algorithms to represent curves on surfaces. These models have become a standard formalism for representing possibly crossing curves on a surface when studying topological problems [G,H,L,N,R,45,47,51,108,111,115,121,167]. Less powerful models appeared earlier [C,F]. For simplicity, we only describe these models on surfaces without boundary; although they extend to surfaces with boundary, this extension is slightly more complicated and does not bring much new insight.

Since we want to compute shortest curves and graphs with given topological properties on surfaces, we need to be able to describe which (sets of) curves are allowed and the way to measure their length. As the previous chapter witnesses, graphs on surfaces appear in various disciplines. The way to represent a surface algorithmically, as well as curves on that surface, may depend on the viewpoint. Consider for example the problem of computing a shortest non-null-homotopic cycle on a surface. In topological graph theory and in the context of algorithms for graphs on surfaces, one is usually given a graph cellularly embedded on a surface, and one wants to determine the length of the shortest non-null-homotopic closed walk in that graph, which is called the *edge-width* (see Chapter 6). In computational geometry, computer graphics, and geometry processing, a surface is often described as a triangular mesh, namely, a set of flat triangles in $\mathbb{R}^3$, the vertices of which are given with explicit coordinates. In mathematics, it may be more relevant to consider an arbitrary Riemannian surface and to study the properties of shortest curves under some topological constraints.

Therefore, a variety of models could be relevant for the problems we consider. In this document, we restrict ourselves to two models, the *combinatorial surface model* and the *cross-metric surface model*, which we describe next. This has several advantages. First, all the results and proof techniques in Part II can be rigorously given within these models. Also, these models do not require excessive formalism. Finally, they are algorithmically simple; in particular, computing shortest paths on such surfaces amounts to computing shortest paths in (weighted) graphs. Shortest path computations are a basic building block used by all our algorithms, and we prefer to avoid such a fundamental primitive to rely on complicated shortest path computations on piecewise-linear surfaces.

As a drawback, these models may appear a bit far from computational geometry or application viewpoints. However, in Section 4.3, we briefly indicate that our results extend to other, more geometric models, at the price of additional technicalities and some increase in the complexity bounds.

4.1 Combinatorial Surfaces

The *combinatorial surface model* is probably the simplest conceivable algorithmic model for considering curves on surfaces, and is natural when the problem involves a graph on a surface. A ***combinatorial surface*** (Σ, G) is the data of a surface Σ (without boundary), together with a cellular embedding G of a graph with non-negative weights on the edges. In this model, the only allowed curves are walks in G; the length of a curve is the sum of the weights of the edges traversed by the curve, counted with multiplicity. Therefore, the curves on a combinatorial surface are just walks in the graph G. In general, a given vertex or edge of G may be used by several curves (or several pieces of the same curve).

The ***complexity*** of a combinatorial surface (Σ, G) is the number of edges of G, which is also, up to a multiplicative constant, the size of a data structure needed to represent that combinatorial surface (in the real RAM model—refer back to Section 2.4.2).

4.2 Cross-Metric Surfaces

While relatively natural, the combinatorial surface model is rather weak, in particular because the notion of (self-)crossing between curves is not well-defined. Imagine that two curves on a combinatorial surface (Σ, G) share an edge of G. It is possible to define in a natural way whether these two curves cross or whether they “run along” without crossing; however, in the first case, the location of the crossing point is not well-defined.

The *cross-metric surface* model is a dual and refined formulation of the combinatorial surface model that allows to encode precisely where curves cross on the surface, which is needed for almost all results in Part II. For example, the main result of Section 7.1 is an algorithm for computing shortest homotopic curves in combinatorial surfaces; however, in the intermediate steps of the algorithm, switching to the cross-metric surface model is crucial. In Chapter 8, we give an algorithm to determine whether two graph embeddings on a surface are isotopic; the (unweighted version of the) cross-metric surface model is used, because it contains exactly the information needed by the algorithm.

4.2.1 Definition and Properties

A ***cross-metric surface*** (Σ, G^*) is a surface Σ together with a cellular embedding G^* of a graph with non-negative weights on the edges. Unlike the combinatorial surface model, however, the only paths and cycles on Σ that we consider are those in ***general position*** with respect to G^*, meaning that they intersect the edges of G^* only transversely and away from the vertices. The ***length*** of a curve γ is defined to be the sum of the weights of the dual edges that γ *crosses*, counted with multiplicity. To emphasize this usage, we sometimes refer to the weight of a dual edge as its ***crossing weight***.

In other words, a cross-metric surface is a usual topological surface with a discrete notion of metric: Two points within the same face of G^* are at distance zero, and the length of a path that crosses an edge of G^* is the weight of that edge.

To any combinatorial surface (Σ, G), we can associate by duality a cross-metric surface (Σ, G^*), where G^* is (as notation suggests) the dual graph of G. To any curve on a combinatorial surface, traversing edges $e_1, \ldots, e_p$, we can associate a curve in the corresponding cross-metric surface, crossing edges $e_1^*, \ldots, e_p^*$, and conversely. This transformation preserves the lengths of the curves. We can easily construct shortest paths on a cross-metric surface by restating the usual algorithms (for example, Dijkstra's algorithm) on G in terms of the dual graph G^*.

The ***complexity*** of a cross-metric surface (Σ, G^*) is, as one can expect, the number of edges of G^*; it is, up to a multiplicative constant, the space needed to represent the cross-metric surface.

4.2.2 Curves on Cross-Metric Surfaces, Algorithmically

We can represent an arbitrary set of possibly (self-)intersecting curves in general position on a cross-metric surface (Σ, G^*) by maintaining the combinatorial ***arrangement*** of G^* and of the curves, i.e., the map associated with the union of the curves. This data structure thus encodes the crossings between curves and their relative positions unambiguously.

The initial arrangement is just the graph G^*, without any additional curve. We embed each new curve such that every crossing point of the new curve and the existing arrangement, and every self-crossing of the new curve, creates a vertex of degree four.

Whenever we split an edge e^* of G^* to insert a new curve, we give both sub-edges the same crossing weight as e^*. Each segment of the curve between two intersection points becomes a new edge, which is, unless noted otherwise, assigned weight zero.

4.3 Discussion

In this section, we briefly discuss other possible computational models for curves on surfaces.

4.3.1 Other Graph-Based Models

In graph algorithms, many problems deal with *vertex-disjoint* paths and cycles. When studying such problems in the specific case of graphs on surfaces, it would make sense to restrict the combinatorial surface model described above by requiring each vertex of the graph to carry *at most one* piece of curve. In other words, one would consider vertex-disjoint curves in a cellular graph.

While this model is very relevant for topological graph theory, it has several important limitations. First, it is obviously restricted to curves without crossings. Moreover, important topological objects such as one-vertex cut graphs (Chapter 5) do not always exist, since the number of curves passing through a given vertex is limited by half of the degree of that vertex. Also, subpaths of shortest homotopic paths may not be shortest homotopic paths, because of the restriction that a path must have no repeated vertex. In short, most basic properties used in subsequent chapters do not extend to that setting. This does not mean that nothing is possible algorithmically; for example, vertex-disjoint paths can be computed in polynomial time for a fixed number of pairs of endpoints [226]; see also the results of Chapter 9.

A variant of this setting would be to forbid crossings between curves but to allow "non-crossing" curves. In other words, one would keep track of the ordering of the curves running along each edge of the graph, and pretend that no crossing occurs at a vertex if the cyclic ordering of the curves around that vertex is parenthesized. This is what was done in the (weaker) predecessors of the cross-metric surface model [C, F].

4.3.2 More Geometric Models

From a computational geometry or mathematical perspective, more geometric models would be preferable, where curves can go anywhere on the surface, and where their length would be measured in a more "continuous" way.

Ideally, one would consider a general Riemannian surface and compute, e.g., shortest non-null-homotopic or non-null-homologous cycles.

Of course, the difficulty here is that it is not clear how to specify the input surface, and that computing shortest paths in such a generality is unfeasible. However, several of the structural results we prove on shortest curves and graphs with prescribed topological properties in the cross-metric model hold also for smooth Riemannian surfaces, by similar arguments. Related problems, such as the existence of a shortest cut graph, have been studied earlier in a Riemannian setting [67].

An apparently more tractable model would be to represent a surface by a finite family of Euclidean triangles (specified by the coordinates of their vertices in the Euclidean plane) together with the information of the pairs of edges that are attached together. However, no exact algorithm for computing shortest paths in this setting is known: All shortest path algorithms in piecewise-linear surfaces [61, 188] implicitly assume that a shortest path on the surface between two points in the same triangle belongs to that triangle, and that property does not hold in general; see the discussion by Erickson [107, Section 2.3].

So let us restrict ourselves to cases where the previous property holds; this is, for example, always the case if the triangulated surface is embedded in some Euclidean space. Under this restriction, it turns out that the algorithms in Part II extend, with slightly higher, but still polynomial-time complexities, to this setting (with the exception of the results in Chapter 6, for which such a model is irrelevant). However, proving our results in such a setting leads to additional technicalities which we do not want to describe here in detail. One difficulty is that two shortest paths in this model may overlap along common subpaths and then diverge. Defining whether such paths cross, and where, requires additional, rather painful formalism. Even defining the desired output of the algorithms is rather tedious. For example, among all cut graphs with a single given vertex, a shortest one may not exist: There exists a sequence of cut graphs converging towards a shortest one, but the limit object may contain two edges that partly overlap. One benefit of the cross-metric surface model is that it provides a unified framework that avoids such pathological phenomena and their irrelevant technical distractions.

4.4 Notational Conventions for Part II

We now introduce common notations for the material in Part II. We denote by Σ a surface without boundary of genus g. We use the convention that (Σ, G) is a combinatorial surface and that (Σ, G^*) is the dual cross-metric surface, and we freely switch between both models by duality. Furthermore, n denotes the complexity of (Σ, G) or (Σ, G^*).

CHAPTER 5

BASICS: CUT LOCI AND APPLICATIONS

This chapter presents material originally from the author's course notes [K, Chapter 4], to which the reader is referred for more details. These results were included in two subsequent articles [L, N] (one written with Sergio Cabello and Francis Lazarus).

In this chapter, we give algorithms to compute various shortest objects with prescribed topological properties on surfaces: a shortest non-disk-bounding loop, a shortest non-separating loop, or a shortest cut graph with a given vertex set. These basic building blocks are used in subsequent parts of this manuscript as well as in other works; their interest stems from the fact that cutting along them simplifies the topology of the surface. Considering shortest such curves is useful both for theoretical and applied purposes; these curves enjoy structural properties that are exploited later.

Some of the results in this chapter were proved before by other authors. However, for those results, we give different proofs and algorithms. This has the advantage that all algorithms in this chapter are presented in a simple and unified way and rely on a single common tool, the *cut locus*.

In this chapter and in the whole Part II, (Σ, G^*) is a cross-metric surface *without boundary* of genus g and complexity n. Here are the results of this chapter.

Theorem 5.1 (Erickson and Har-Peled [109]). *In (Σ, G^*), computing a shortest non-disk-bounding or non-separating loop based at a given point can be done in $O(n \log n)$ time.*

Recall that, by definition, the adjectives "disk-bounding" and "separating", and there negations, apply only to *simple* loops. Also, we implicitly assume in this theorem that the surface is not a sphere; otherwise, every loop is disk-bounding and separating.

This result was first proved by Erickson and Har-Peled [109]. The algorithms and the proofs we present, taken from the author's course notes [K] and published in a subsequent article [L], are different, but the new technique is needed for the results in Chapter 6; furthermore, it avoids a clever but tedious complexity analysis in one case [109, Lemma 5.4].

Computing a shortest non-disk-bounding or non-separating cycle (without prescribed basepoint) is then easy: Run $O(n)$ instances of the algorithm in Theorem 5.1, each with a basepoint in a different face of G^*, and return the shortest overall loop.

Here is another result proved in this chapter.

Theorem 5.2 ([N]). *In (Σ, G^*), computing a shortest cut graph with a given vertex set of cardinality k can be done in $O(n \log n + (g+k)n)$ time.*

In particular, a shortest cut graph with a single prescribed vertex—also known as a ***system of loops***—can be computed in $O(n \log n + gn)$ time. This very important special case was known before, see Erickson and Whittlesey [114]; our proof of the more general theorem is also simpler, and one part of this proof was further simplified by Erickson [107]; we present only the simplest proof.

5.1 Topology: Cut Loci

5.1.1 Definition

Let P be a set of vertices of G. A *cut locus* of P is a subgraph C of G^* that partitions the surface Σ into a set of disks, each disk containing the set of points of the surface with the same closest point in P. Informally, let water flood the faces of G^*, starting simultaneously from the source set P. Every edge of G^* acts as a barrier and delays the flood by a time equal to its crossing weight; after this delay is elapsed, water crosses that edge only if the opposite face has not yet been flooded. Then the cut locus C is the set of edges of G^* not crossed by water during this process.

More precisely (see Figures 5.1(a–b) and 5.2(a–b)), let F be a spanning forest of shortest paths in G, starting from every vertex of P simultaneously. In other words, F is a spanning subgraph of G, each connected component of F is a tree containing exactly one point of P, and F contains the shortest path from every vertex of G to its nearest vertex in P. A ***cut locus*** of P is, by definition, obtained from G^* by removing the duals of the edges of F.

One can view $\Sigma \backslash\backslash C$ as the set of faces of G^* attached together along the duals of the edges in the forest F. Since the faces of G^* are disks, and since attaching disks in a tree-like fashion gives a family of disks, we see that $\Sigma \backslash\backslash C$ is a union of disjoint disks. Furthermore, each disk contains a single point of P.

5.1.2 Remarks

The cut locus is not uniquely defined, because there may be several shortest path forests F. However, if the weights of G^* are generic (which the

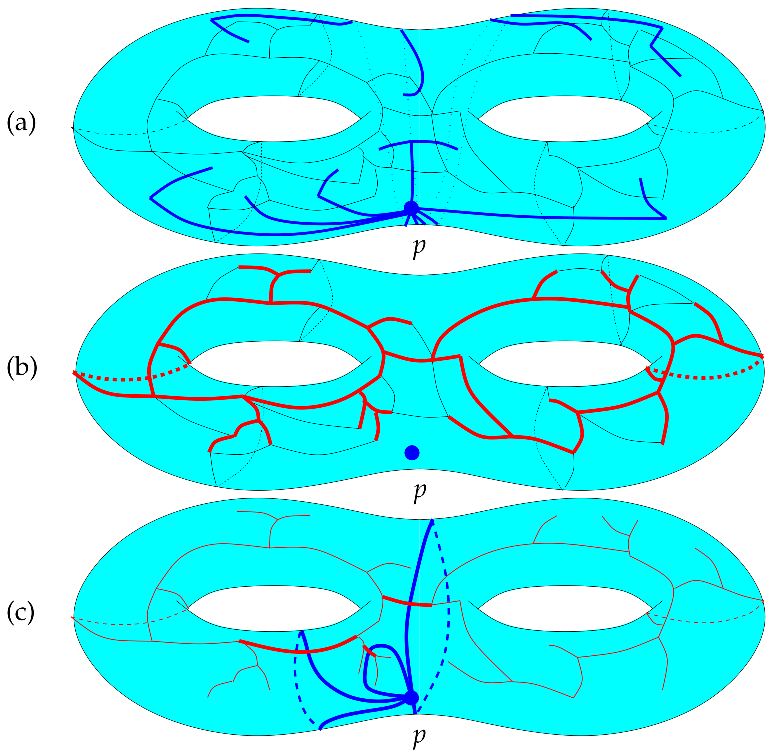


Figure 5.1. A cut locus of a single point $P = \{p\}$ on a double-torus. (a) The graph G^* is shown in light lines, and the visible part of the spanning tree F of G is highlighted. (b) The cut locus C is highlighted; its edge set is the edges of G^* not crossed by F. (c) The primitive loops $\sigma(e)$, for three edges e of C.

reader may safely assume to help intuition), then there is a unique cut locus.

Also, the cut locus resembles the Voronoï diagram of P, widely used in computational geometry [31]. However, each edge of the Voronoï diagram bounds two cells containing *different* points of P, while an edge of the cut locus can have the same edge on both sides (cf. the case where P is a single vertex). So the Voronoï diagram is a subset of the cut locus.

5.1.3 Primitive Paths

For $e \in E(C)$, let $\boldsymbol{\sigma(e)}$ be a shortest path among all paths with endpoints in P that cross edge e; see Figures 5.1(c) and 5.2(c). By construction of the cut locus, $\sigma(e)$ crosses only edge e of C, exactly once, and is the concatenation of two shortest paths, one on each side of e. Such a path $\sigma(e)$ is called ***primitive***. We may furthermore assume that all the primitive paths are pairwise disjoint, except at common endpoints: Indeed, let p be a point of P, in face f of G^*; within f, the primitive paths are all shortest paths with p as an endpoint, and can therefore be chosen so as to be disjoint except at p.

The following crucial lemma relates, for any $A \subseteq E(C)$, the topology of $\Sigma \setminus (P \cup \sigma(A))$ to that of $C - A$.

Lemma 5.3. *Let $A \subseteq E(C)$. Then there exists a deformation retraction from $X := \Sigma \setminus (P \cup \sigma(A))$ to $Y := C - A$. In other words, there exists a continuous*

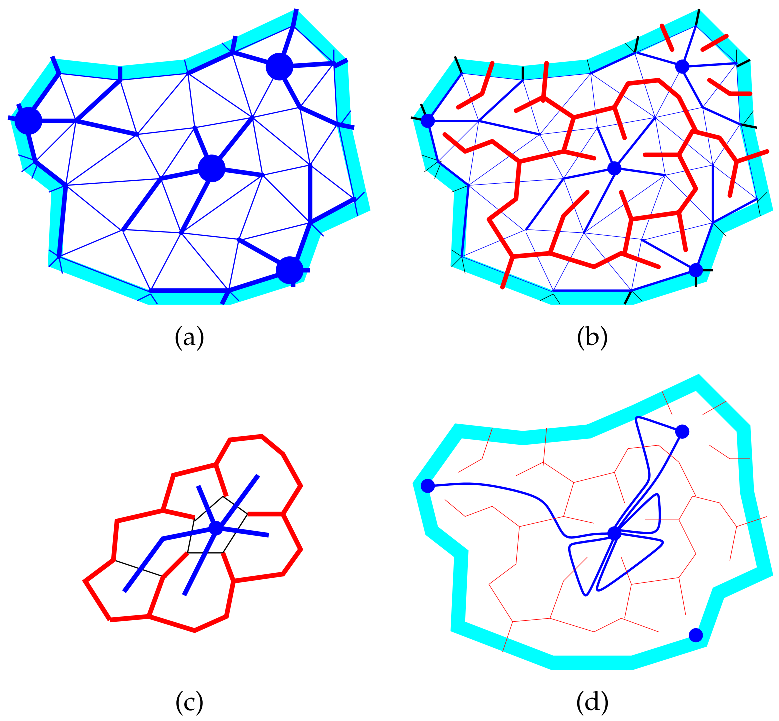


Figure 5.2. A closeup of the construction of a cut locus on a surface with a "dense" set of points P. The graph G is a triangulation. (a) The graph G, with vertex set P (shown as disks) and the spanning forest F (highlighted). (b) The cut locus C. (c) Each face of C is obtained by attaching faces of G^* in a tree-like fashion, and is therefore a disk. (d) A set of primitive paths.

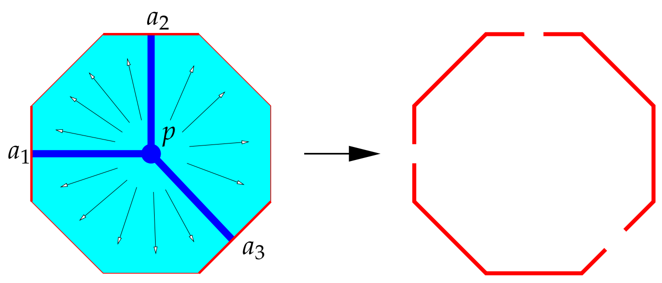


Figure 5.3. A schematic view of the retraction in the proof of Lemma 5.3, within a single face f of C. Here, $A = \{a_1, a_2, a_3\}$.

map $h : [0,1] \times X \to X$ such that $h(0,\cdot)$ is the identity, $h(1,X) \subseteq Y$, and $h(t,\cdot)|_Y$ is the identity for each $t \in [0,1]$.

Proof. Consider a face f of the cut locus C (Figure 5.3); it is a disk containing a single point p of P. We can retract $\overline{f} \setminus (\sigma(A) \cup \{p\})$ onto $(\partial f) \setminus (A \cap C)$. Gluing these retractions together gives us a deformation retraction of $\Sigma \setminus (P \cup \sigma(A))$ onto $C \setminus (A \cap C)$, which in turn retracts onto $C - A$. $\square$

Computing the cut locus can be done in $O(n \log n)$ time using Dijkstra's algorithm in G to compute the shortest path forest F. In the same amount of time, one can label each edge e of C by its ***σ-weight***, namely, the length of $\sigma(e)$.

5.2 Algorithms

Recall that we aim at computing various shortest objects on (Σ, G^*):

1. a shortest non-disk-bounding loop based at a given point p of Σ;
2. a shortest non-separating loop based at a given point p of Σ;
3. a shortest cut graph with vertex set *exactly* P, where P is a given non-empty set of points in Σ. (This implies that each point of P must be incident to at least one edge of the cut graph.)

For consistency of notation, in the first two cases, we let $P = \{p\}$. Furthermore, to simplify the exposition, we assume that P is a subset of vertices of G.

At a high level, the algorithms for all three problems are all the same: They compute a shortest *primitive* object of the desired type. We focus on this problem now, and later justify the correctness of the algorithms by proving that the shortest desired objects are primitive. As we shall see, the key to compute shortest primitive objects is Lemma 5.3: It allows to read off the topology of a (set of) primitive path(s) directly on the cut locus.

5.2.1 Shortest Non-Disk-Bounding Loop

A primitive loop $\sigma(e)$ bounds a disk if and only if $\Sigma \setminus \sigma(e)$ has a component that is an open disk; equivalently, by Lemma 5.3, $C - \{e\}$ has a connected component that is a tree. (See Figure 5.1(c).) Therefore, it suffices to find all *candidate* edges e, those such that $C - \{e\}$ has no tree component; assuming we can do this, the algorithm will return $\sigma(e')$, where e' is the minimum-σ-weight candidate edge.

The condition of e being candidate does not depend on the surface, but merely on the "position" of e in the cut locus C. In other words, we have turned a topological problem on a surface into a graph problem. Not surprisingly, this problem can be solved in linear time: After iteratively removing a degree-one vertex of the graph with its incident edge, exactly the candidate edges remain. From the minimum-σ-weight candidate edge e, computing the corresponding loop $\sigma(e)$ is easy.

5.2.2 Shortest Non-Separating Loop

A primitive loop $\sigma(e)$ separates Σ if and only if $\Sigma \setminus \sigma(e)$ has two connected components; equivalently, by Lemma 5.3, $C - \{e\}$ has two connected components. (Refer again to Figure 5.1(c).) In other words, e is a *bridge* of the graph C. Again, it now suffices to solve a graph problem: Our candidate edges are now the non-bridge edges of C; finding them can be done in linear time using depth-first search, relying on a similar technique as in the computation of the biconnected components [3, Section 5.3].

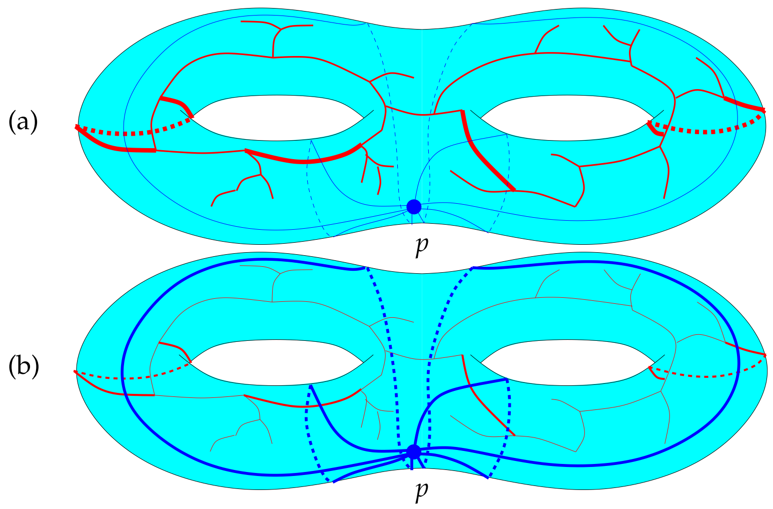


Figure 5.4. Continuation of Figure 5.1. (a) The complement of a spanning tree of the cut locus C. (b) The shortest cut graph with vertex p.

5.2.3 Shortest Cut Graph with Prescribed Vertex Set

A set of primitive paths $\sigma(A)$, $A \subseteq E(C)$, is a cut graph including every vertex of P if and only if $\Sigma \setminus (P \cup \sigma(A))$ is an open disk; equivalently, by Lemma 5.3, $C - A$ is a tree; this means that $E(C) \setminus A$ is the edge set of a spanning tree of C. So the algorithm computes the edge set A of a *maximum-σ*-weight spanning tree of C (so that the σ-weight of $E(C) \setminus A$ is minimized), and returns $\sigma(E(C) \setminus A)$. See Figure 5.4 for an illustration in the case of a single vertex.

Computing a maximum-weight (or minimum-weight) spanning tree in a graph can be done in linear randomized time [153] or near-linear deterministic time [58]. Computing the paths $\sigma(E(C) \setminus A)$ from A is not difficult and can be done in time linear to their total complexity; each of the paths has complexity $O(n)$, and Euler's formula implies that there are $g + k - 1$ paths, so this last step takes $O((g + k)n)$ time.

5.3 Optimality via Algebraic Structures

In this section, we explain why the key structural result holds: The shortest object that we want to compute is *primitive*, for algebraic reasons.

5.3.1 Shortest Non-Disk-Bounding Loop

The relevant algebraic structure here is *homotopy*. The following lemma is standard and intuitive (though a formal proof requires some machinery such as the universal cover):

Lemma 5.4. *Let ℓ be a simple loop. Then ℓ is null-homotopic if and only if it bounds a disk.*

We also have:

Lemma 5.5. *Some shortest non-null-homotopic loop is primitive.*

Proof. Let ℓ be a shortest non-null-homotopic loop; let $e_1, \ldots, e_k$ be the edges of C crossed by ℓ, in this order. The loop ℓ is homotopic to $\sigma(e_1) \cdot \sigma(e_2) \cdots \sigma(e_k)$;[1] therefore, one of the loops $\sigma(e_i)$ is non-null-homotopic. Furthermore, by definition, $\sigma(e_i)$ is no longer than ℓ. So $\sigma(e_i)$ is a shortest non-null-homotopic loop. □

So let ℓ be a shortest non-null-homotopic loop, which we can assume to be primitive (Lemma 5.5). It is simple, and is therefore non-disk-bounding (Lemma 5.4). On the other hand, any non-disk-bounding loop is non-null-homotopic (again by Lemma 5.4), hence longer than ℓ. Therefore, ℓ is a shortest non-disk-bounding loop, and is primitive, which had to be proved.

5.3.2 Shortest Non-Separating Loop

The strategy follows the very same principles, with homology in place of homotopy, so we omit the proof. We indicate only that the homological counterpart of Lemma 5.4 is stated as follows, as mentioned in Section 2.5.2.

Lemma 5.6. *Let ℓ be a simple loop. Then ℓ is null-homologous if and only if it is separating.*

5.3.3 Shortest Cut Graph with Prescribed Vertex Set

Here, the relevant algebraic structure is the homology vector space relatively to the point set P, denoted by $H_1(\Sigma, P)$. Define a *P-path* to be a path intersecting P exactly at its endpoints. The crux of the matter is to prove the following topological lemma, generalizing Lemma 5.6.

Lemma 5.7 ([N, Proposition 3])**.** *Let Q be a set of P-paths that are pairwise disjoint except at common endpoints. Then Q is a cut graph with vertex set exactly P if and only if the relative homology classes of Q constitute a basis of $H_1(\Sigma, P)$.*

The proof of the following lemma is identical to that of Lemma 5.5.

Lemma 5.8. *A shortest P-path among all P-paths whose relative homology class is outside a given vector subspace of $H_1(\Sigma, P)$ is primitive.*

Now comes the simplified argument by Erickson [107] (removing a matroid argument used earlier [N]): Let $q_1, \ldots, q_j$ be a shortest family of P-paths forming a basis of $H_1(\Sigma, P)$. If one of these P-paths, say q_i, is not primitive, we may replace it with a shortest P-path homologically independent with $q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_j$; that P-path is primitive (Lemma 5.8). Continuing, we may assume that all the q_i's are primitive. To summarize, some shortest family of P-paths forming a basis of $H_1(\Sigma, P)$ is primitive.

Since these P-paths are primitive, they are simple except at common endpoints, so Lemma 5.7 implies that they form a cut graph with vertex

[1]This is the only place where the algebraic structure of the homotopy group is used!

set exactly P. On the other hand, any cut graph with vertex set exactly P corresponds to a basis of $H_1(\Sigma, P)$ (again by Lemma 5.7), so it must be longer than $q_1, \dots, q_j$. Bottom line: A shortest cut graph with vertex set exactly P is made of primitive paths.

5.4 Remarks and Extensions

5.4.1 Reformulations

Homotopy and homology. In proving the above structural results, we have shown a few results that are interesting in their own right and worth emphasizing. A shortest non-null-homotopic loop is simple (and therefore a shortest non-disk-bounding loop). A shortest non-null-homologous loop is simple (and therefore a shortest non-separating loop). A shortest family of P-paths forming a basis of $H_1(\Sigma, P)$ is made of disjoint simple paths (and therefore a shortest cut graph with vertex set exactly P).

Combinatorial surfaces. Although this chapter focuses on the cross-metric setting, it is worth reformulating some results in the combinatorial surface model. Given a surface Σ with a weighted cellularly embedded graph G, we can compute a shortest non-null-homotopic (or non-null-homologous) closed walk (without basepoint) in G in $O(n^2 \log n)$ time. Furthermore, it is easy to prove that this closed walk has no repeated vertices, so that it is also a shortest non-disk-bounding (or non-separating) closed walk in G.

Similarly, if P is a subset of vertices of G, we can compute a shortest set of walks in G with endpoints in P that forms a basis of $H_1(\Sigma, P)$. In general, the walks are neither simple nor pairwise disjoint; however, their union forms a cut graph of Σ, and actually a shortest cut graph among all cut graphs obtained by taking unions of P-paths (where the length of each edge of G is counted with multiplicity).

5.4.2 Extensions

Improvements in special cases. One of the bottlenecks in our algorithms is the $O(n \log n)$-term for computing the cut locus. However, computing the cut locus can be done in $O(n)$ time in two special cases, and the overall complexity can be reduced accordingly:

- if the graph is unweighted, because Dijkstra's algorithm can be replaced by breadth-first search;
- if the surface is fixed, because $O(\sqrt{n})$-size separators exist and can be computed in $O(n)$ time (see Gilbert et al. [124] and Eppstein [100]), so that the linear-time shortest path algorithm by Henzinger et al. [140] applies.

Surfaces with boundary. We have entirely dismissed the case of surfaces with boundary above (as will also be the case in most forthcoming chapters). Indeed, surfaces with boundary are usually no harder to deal with than surfaces without boundary. For example, to compute a shortest non-separating loop on a surface with boundary, attach a disk to each boundary component, with infinite crossing weight, and compute a shortest non-separating loop in the resulting surface; this is a shortest non-separating loop in the original surface. To compute a shortest non-disk-bounding loop, use a similar construction, but attaching a handle (instead of a disk) to each boundary component.

However, the case of the annulus is important. As it turns out, on an annulus, a shortest non-disk-bounding cycle corresponds to a minimum cut in the dual graph, which can be computed in $O(n \log\log n)$ time with a recent algorithm [147]. Here is an equivalent reformulation:

Theorem 5.9 (Italiano et al. [147])**.** *Assume Σ is the sphere. Computing a shortest closed walk in G separating two given faces of G can be done in $O(n \log\log n)$ time. The output cycle has no repeated vertices.*

5.4.3 Notes

Other decompositions. In the algorithm for computing a shortest cut graph, prescribing the vertex set of the cut graph is crucial; indeed, in general, Erickson and Har-Peled [109] prove that computing the shortest cut graph is NP-hard, and show how to compute a cut graph whose length is within an $O(\log^2 g)$-factor of the optimum in $O(g^2 n \log n)$ time.

In the case of one single vertex, the algorithm computes a shortest set of loops with a fixed basepoint that generates the homology group. Erickson and Whittlesey [114] give an algorithm for the same problem, removing the basepoint condition; in other words, they compute the shortest family of cycles generating the first homology group. Extensions are known for arbitrary simplicial complexes [40,88]; variants are solvable in arbitrary dimension [59].

Besides cut graphs, there are other types of topological decompositions of surfaces. Lazarus et al. [172], based on earlier work by Vegter and Yap [245], describe an optimal algorithm to compute a (not necessarily shortest) ***canonical system of loops***: a one-vertex cut graph where the loops appear around the vertex in a prescribed cyclic order. Brahana [37], early in the 1920's, describes a combinatorial algorithm for this purpose; in fact, the classification theorem for surfaces is usually proved by showing that every surface without boundary has a canonical system of loops.

Other decompositions are conceivable. No algorithm for computing the shortest pants decomposition is known, but Eppstein [101] gives efficient approximation algorithms, in the case of the punctured Euclidean or hyperbolic plane. ***Octagonal decompositions*** will be introduced in Section 7.1.3. One could also consider decompositions such sets of disjoint simple cycles cutting the surface into a 0-genus surface (also called *cut systems* [136]), or into punctured tori. In all cases, no algorithm is known to compute such shortest decompositions, even approximately (except in the case mentioned above), and no NP-hardness proof is known.

Bibliographic notes. The expression *cut locus* used in this chapter has been used in this context by Erickson and Whittlesey [114]. It comes from Riemannian geometry: The cut locus of a point p in a Riemannian manifold is essentially the set of points q with several shortest paths from p. Our notion of cut locus extends the standard one to several points, and translates the Riemannian case to the cross-metric surface setting.

Variants of the cut locus have been used in enumerative combinatorics for graphs on surfaces (also called *maps* in this context). In particular, Miermont [186], building upon previous works by Schaeffer [221] and several other authors [36, 56], exhibits a bijection between bipartite quadrangulations with a given marked subset of vertices and cellularly embedded graphs (both structures need some additional information). At a very high level, the cellularly embedded graph is a cut locus of the quadrangulation with respect to the marked vertices.

The length of the shortest non-disk-bounding cycle on a surface is a quantity that appears in different scenarios. In topological graph theory, in the context of embedded graphs, it is called the *edge-width* (see Chapter 6). In Riemannian geometry, it is called the *systole* of the surface; see, e.g., Berger [23].

Eppstein [100], with his *tree-cotree decomposition*, introduces ideas used later by Erickson and Whittlesey [114]. In particular, Eppstein notes that the minimum-weight spanning tree in a graph and the maximum-weight spanning tree in the dual are disjoint.

The argument of Lemma 5.5 appears, in a different form, in a paper by Thomassen [238]; see also Mohar and Thomassen [191, Section 4.3]. More abstractly, Thomassen defines the *3-path condition* for a family of cycles in a graph, and shows that the shortest cycle in every family satisfying that condition can be computed in cubic time. In our terms, the shortest cycle in the family is primitive. Shortest cycles with an odd number of edges, and shortest one-sided cycles, can also be computed in cubic time. Incidentally, a simple variant of our algorithm computes a shortest one-sided loop through a given vertex in $O(n \log n)$ time.

CHAPTER 6

MORE SHORTEST NON-TRIVIAL CYCLES

The results of this chapter appeared in joint articles with Sergio Cabello and Francis Lazarus [L, M].

In this chapter, we carry on with algorithms to compute shortest non-disk-bounding or non-separating cycles on a cross-metric surface (Σ, G^*). In both algorithms, the techniques are very similar, so it is useful to define a ***non-trivial*** cycle as a non-disk-bounding or non-separating cycle, depending on the context. (The terminology comes from the fact that a non-disk-bounding loop is non-trivial in homotopy, while a non-separating loop is non-trivial in homology.) We again assume that the surface Σ is not the sphere, for otherwise every cycle is disk-bounding and separating.

In graph algorithms, problems generally have different flavors depending on which kind of graph is considered: Is the graph weighted or unweighted, is it directed or undirected? In Chapter 5, we considered only the "standard" case of graphs where edges are undirected and have non-negative weights. We now study the other possibilities, heavily relying on the techniques from the previous chapter.

In Section 6.1, we consider a more restricted case, namely, the case where the graph is unweighted and undirected; we show how to compute shortest non-trivial cycles more efficiently than in the standard case, and also develop output-sensitive and approximation algorithms. In Section 6.2, we consider the more general setting of weighted, directed graphs, and provide efficient algorithms.

6.1 Unweighted, Undirected Case

In this section, we assume that the edge weights of the graph G (and consequently also of G^*) are all equal to one, and we study the computation of shortest non-trivial loops and cycles.

A first remark is that one can compute shortest non-trivial loops in (Σ, G^*) in $O(n)$ time: Indeed, the algorithms of Chapter 5 apply, but an $O(\log n)$-factor can be removed, because that factor is due to the application of Dijkstra's algorithm when computing the cut locus, and this algorithm can be replaced by a breadth-first search in the unweighted case. Consequently, we can compute shortest non-trivial cycles in $O(n^2)$ time. We shall give more efficient algorithms in some cases.

6.1.1 Our Results

Theorem 6.1 ([L]). *Assume G is unweighted. Given an integer k, we can compute in $O(gnk)$ time a shortest non-trivial cycle in (Σ, G^*) if it has length at most k, or correctly report that the length of every non-trivial cycle is larger than k.*

By running the algorithm of Theorem 6.1 for exponentially increasing values of k, it is straightforward to obtain an efficient output-sensitive algorithm:

Corollary 6.2. *Assume G is unweighted. We can compute a shortest non-trivial cycle in (Σ, G^*) in $O(gnk)$ time, where k is the length of the output cycle.*

We can also compute almost-shortest non-trivial cycles. Call a non-trivial cycle ***α-short*** if its length is at most α times the length of the shortest non-trivial cycle. Erickson and Har-Peled [109, Corollary 5.8] give an algorithm to compute a 2-short non-trivial cycle in $O(gn)$ time (see Lemma 6.5 below). The following theorem is an enhancement over their result.

Theorem 6.3 ([L]). *Assume G is unweighted. Let ε be such that $0 < \varepsilon < 1$. We can compute a $(1 + \varepsilon)$-short non-trivial cycle in (Σ, G^*) in $O(gn/\varepsilon)$ time.*

6.1.2 Applications

The *edge-width* and *face-width* of an unweighted cellularly embedded graph G on a surface Σ are important parameters introduced in the field of topological graph theory [191, Chapter 5]. The ***edge-width*** of G is the length of the shortest non-disk-bounding cycle in G. The ***face-width*** of G is a variant of this concept: It is the minimum number of points of the image of G met by a non-disk-bounding closed curve on Σ. Is is easy to see that the face-width of G equals half the edge-width of the vertex-face incidence graph of G, also called ***radial graph*** [191, Proposition 5.5.4]. One also has the concept of ***non-separating edge-width*** or ***face-width***, where non-disk-bounding is replaced by non-separating. Our results imply that all these parameters can be computed efficiently.

Our Theorem 6.1 also strengthens a result by Kawarabayashi and Mohar [157, Corollary 1], where it is shown that there is a linear-time algorithm to decide whether the face-width of a graph on a fixed surface is bounded from above by a constant. Their approach is based on graph minors, has an exponential dependency on the genus and the face-width, relies on an unknown, though theoretically computable, list of minimal graphs, and deals only with the face-width, not the edge-width.

Algorithms for computing or approximating the edge-width or face-width of a graph are also useful for planar problems. Kawarabayashi and

Reed [159] prove that, given a fixed integer k, there exists a linear-time algorithm to determine whether a given graph can be drawn in the plane with at most k crossings. As a subroutine, they have a graph embedded with bounded face-width on a surface with bounded genus, and they need to compute a short non-contractible cycle in the radial graph. Our Corollary 6.2 allows to compute a shortest such cycle in linear time, while they rely on a 2-approximation algorithm [109, Corollary 5.8] (Lemma 6.5 below). Using this 2-approximation instead of the real face-width as computed by our algorithm affects exponentially the running-time; however, this is hidden in the $O(\cdot)$-notation because the face-width is $O(1)$ since the surface is considered to be fixed.

In some special cases, it is known that the edge-width or face-width k of an embedded graph is always sublinear in n; as the running-time of the algorithm of Corollary 6.2 is $O(gnk)$, this allows us to give a better bound on the running-time of that algorithm (compared to the naïve bound $O(gn^2)$) in those cases. For example, the edge-width of a triangulation and the face-width of an arbitrary cellularly embedded graph can be computed in $O(n^{3/2}g^{1/2}\log g)$ time, by results by Hutchinson [144] and Cabello and Mohar [47, Lemma 13 and Theorem 14], respectively.

Other results involving the edge-width or face-width have the following form: For every fixed surface Σ, if an embedded graph G has edge-width or face-width large enough, then it has some particular properties [81, 157, 190, 212, 239, 249]. Our Theorem 6.1 provides a linear-time algorithm to test whether such hypotheses are fulfilled. See also Fiedler et al. [119] for another result relating the face-width of a graph in the projective plane to its orientable genus.

6.1.3 Output-Sensitive Algorithm: Proof of Theorem 6.1

In a nutshell, the algorithm for Theorem 6.1 goes as follows. (1) We first compute a set of vertices A of G such that a non-trivial cycle has to use some vertex of A. (2) For every vertex a in A, we compute the shortest non-trivial cycle passing through a that stays within a distance at most $k/2$ from a (if it exists). The key idea in our approach is to find an efficient way to carry Step (2) simultaneously for several basepoints that are far apart on the surface. This idea, in turn, requires to choose A in Step (1) adequately. Here is the proof in more detail.

The tool to work with several basepoints in parallel is contained in the following lemma.

Lemma 6.4. *Let $U_1, \dots, U_t$ be each a* union *of closed faces of G^*, such that the U_i's have disjoint interiors. For each i, let s_i be a vertex of G in U_i. In $O(n)$ time, we can compute a shortest loop in (Σ, G^*) among all non-trivial loops that, for some i, are based at s_i and stay in U_i, or correctly report that no such loop exists.*

Proof. Let P be the set of all vertices s_i. We define crossing weights on the edges of G^* (or weights on G, by duality) as follows: The weight of an edge of G^* is one if its incident faces are in some common set U_i, and (symbolically) infinite otherwise. Then we compute the cut locus C of P with respect to these weights; this takes $O(n)$ time since Dijkstra's algorithm can be replaced by a variant of breadth-first search. We can also compute, in the same amount of time, the length of each path $\sigma(e)$, again called the *σ-weight* of $e \in E(C)$. One easily proves that the shortest non-trivial loop we seek (if it exists at all) is primitive. In other words, we are looking for the minimum-σ-weight edge among all *candidate* edges: the edges e of C with weight one such that $\sigma(e)$ is a non-trivial loop.

Now, in $O(n)$ time, we remove some edges of C to transform it into a cut graph C'. The edges removed from C cannot be incident to faces in the same set U_i, so they are not candidate edges. As in Section 5.2, we can determine which edges e of C' are such that $\sigma(e)$ is a trivial loop in $O(n)$ total time. So we can determine all candidate edges in $O(n)$ time. □

Proof of Theorem 6.1. Let r be an arbitrary vertex of G. We apply Theorem 5.2 to compute a shortest cut graph C with vertex set $P = \{r\}$ (here all weights are unit weights). Let A be the set of vertices of G in a face of G^* meeting C. Since every cycle in the disk $\Sigma \backslash\backslash C$ is trivial, every non-trivial cycle crosses C; hence, a shortest non-trivial cycle meets A.

From Section 5.3.3, we know that C is made of primitive loops. In particular, for each integer i, the set A_i of vertices of A at distance exactly i from r has cardinality $O(g)$. For each j, $0 \le j \le k$, we put the points of

$$A_j, A_{(k+1)+j}, A_{2(k+1)+j}, \ldots$$

into $O(g)$ batches, each containing at most one element from each of these A_i. Doing this for each j, we obtain a partition of A into $O(gk)$ batches such that any two vertices in the same batch are at distance at least $k+1$ from each other.

Now, consider a single batch $\{a_1, \ldots, a_t\}$. For each i, let U_i be the union of the faces of G^* at distance at most $k/2$ from a_i; the U_i's are pairwise disjoint. We can thus apply Lemma 6.4: If there exists a non-trivial loop based at some a_i that has length at most k, we obtain a shortest such loop.

We apply this step for every batch. If a non-trivial loop of length at most k was found during at least one step, the algorithm returns the shortest one. Otherwise, each non-trivial cycle has length at least $k+1$.

The set A and the $O(gk)$ batches can be computed in $O(n)$ time. Then the $O(n)$-time algorithm of Lemma 6.4 is applied once for each of the $O(gk)$ batches; thus the total running-time is $O(gnk)$. □

6.1.4 Approximation Algorithm: Proof of Theorem 6.3

We will need the following lemma from Erickson and Har-Peled [109]; we sketch a proof for convenience.

Lemma 6.5 (Erickson and Har-Peled [109, Corollary 5.8])**.** *In $O(gn)$ time, one can compute a 2-short non-trivial cycle in (Σ, G^*).*

Proof. Again, let C be the shortest cut graph with a single vertex r, chosen arbitrarily. Every non-trivial cycle must cross C; since each loop in C is primitive, every non-trivial cycle must cross one of the $O(g)$ shortest paths whose union is C. Given a shortest path p, we show below how to compute in $O(n)$ time a non-trivial cycle crossing p and whose length is at most twice the length of a shortest non-trivial cycle crossing p; this concludes.

Let p be a shortest path. One can compute a short non-trivial cycle crossing p by putting infinitesimal crossing weights on the edges of G^* crossed by p, computing a shortest non-trivial loop ℓ based at some given point of p, shortening it by moving the basepoint along p, and restoring the original weights on the edges. This takes $O(n)$ time; furthermore, in the original metric, a simple argument shows that the resulting non-trivial cycle has length at most twice the length of a shortest non-trivial cycle intersecting p. □

Proof of Theorem 6.3. Let k be the (unknown) length of a shortest non-trivial cycle, and let k' ($k \leq k' \leq 2k$) be the 2-approximation computed by Lemma 6.5. We compute the set A (here we freely reuse the notations of the proof of Theorem 6.1). A shortest non-trivial cycle has to intersect A. We add a detour to it so that it intersects A_i, for some i multiple of $\lceil \varepsilon k'/4 \rceil$; the length of the detour is at most $2\lceil \varepsilon k'/4 \rceil - 2 \leq \varepsilon k'/2$. This detour makes the cycle longer by a factor of at most $1+\varepsilon$, since the initial cycle has length $k \geq k'/2$.

Hence, it suffices to compute a shortest non-trivial loop of length at most $(1+\varepsilon)k$ based at some vertex of A_i, where i is a multiple of $\lceil \varepsilon k'/4 \rceil$. The same batching technique as in the proof of Theorem 6.1 will work, but now only $O(g/\varepsilon)$ batches are needed. □

6.2 Weighted, Directed Case

In this section, we assume that the edges of G are weighted, but that the weights are ***asymmetric***: Each edge of G bears two weights, one for each of its two orientations. The length of an (oriented) walk in (Σ, G) is the sum of the weights of the *oriented* edges of G traversed by the walk, counted with multiplicity. Some weights can be infinite, so that shortest curves will avoid at all costs using the edge in the corresponding direction; hence we can represent weighted directed graphs in this model.

This asymmetric set-up may seem rather unnatural if we think of edge weights to be lengths; on the other hand, many graph problems consider weighted, directed graphs (flow and cut problems, for example). If we want to develop algorithms for directed graphs on surfaces, it is reasonable to study basic topological tasks in this setting.

In the cross-metric model, this translates as follows: The length of an (oriented) curve in (Σ, G^*) is still the sum of the weights of the edges of G^* crossed by this curve, but now each weight depends on the orientation of each crossing.

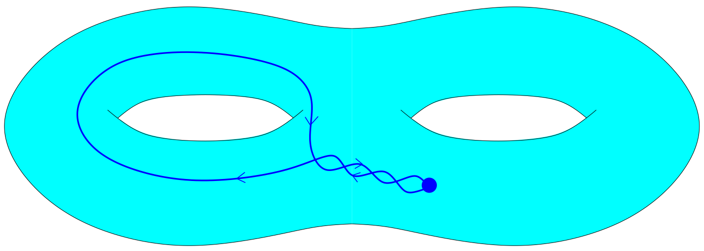


Figure 6.1. A shortest non-trivial directed loop may self-cross. (In this example, all the weights are infinite except those used by the oriented loop.)

Our first algorithm computes shortest non-null-homotopic or non-null-homologous loops in $O(n \log n)$ time, and therefore has the same complexity as in the undirected case (Theorem 5.1). Hence shortest non-trivial cycles can be computed in $O(n^2 \log n)$ time. Furthermore, we give another algorithm to compute shortest non-trivial cycles in $O(g^{1/2}n^{3/2} \log n)$ time; this is more efficient in the case of low genus. For such low genus surfaces, the current best algorithms, by Erickson and Nayyeri [111] and Fox [121], have respective complexities $O(g^2 n \log n)$ for the shortest non-separating cycle and $O(g^3 n \log n)$ for the shortest non-disk-bounding cycle.

We note that our latter algorithm applies in particular to the undirected setting, and is better than naïvely applying $O(n)$ times Theorem 5.1. However, such a subquadratic result (for fixed surfaces) was already known [43, 47, 167], the best record, due to Cabello et al., being $O(g^2 n \log n)$ [44].

The combinatorial model is more adequate for the proofs, so we state our results in this model, although the theorems translate immediately to the cross-metric model.

6.2.1 General Algorithm

Theorem 6.6 ([M]). *Assume G has asymmetric lengths. In $O(n^2 \log n)$ time, we can compute a shortest non-trivial closed walk in (Σ, G).*

The proof of Theorem 6.6 is a direct consequence of the following lemma.

Lemma 6.7. *Assume G has asymmetric lengths; let s be a vertex of G. In $O(n \log n)$ time, we can compute, in (Σ, G), a shortest non-null-homotopic or non-null-homologous closed walk based at s.*

Such closed walks may have self-crossings, see Figure 6.1; this is of course in contrast with the undirected set-up, and partly explains why it is simpler to stay in the combinatorial model. However, it is easy to prove that a shortest non-null-homotopic (or non-null-homologous) closed walk has no repeated vertices, if no basepoint is fixed.

Proof of Lemma 6.7. For concreteness, we focus on the homotopic case first, the homological case being a simple variant. Let T be a shortest path tree in G from the source s; it contains a shortest path from s to every vertex of G. Similarly, let R be a *reversed* shortest path tree in G to the sink s: it contains a shortest path from every vertex of G to s. Since the weights are asymmetric, T and R can be very different. If e is an oriented edge of G, with initial endpoint x and final endpoint y, we let $\tau_T(e)$ be the closed walk based at s that follows the unique path in T from s to x, traverses edge e, and follows the unique path in T from y to s. Similarly, $\tau_{T,R}(e)$ is the closed walk based at s that follows the unique path in T from s to x, traverses e, and follows the unique path in R from y to s.

It turns out that some shortest non-null-homotopic closed walk passing through s is the shortest closed walk of the form $\tau_{T,R}(e)$ such that $\tau_T(e)$ is non-null-homotopic. This is quite surprising: In general, $\tau_{T,R}(e)$ can be non-null-homotopic while $\tau_T(e)$ is null-homotopic and vice-versa. The proof relies on a sequence of exchange arguments similar to but more complicated than those used in the proof of Lemma 5.5. While highly non-trivial, it does not contain any new major ingredient, so we omit the proof.

There remains to show that we can determine in $O(n)$ time which edges e are such that $\tau_T(e)$ is non-null-homotopic. This is easy using the same techniques as in Section 5.2.2. More precisely, let C be the graph obtained from G^* by removing the dual of the edges in T; C is a cut graph, which is the analog of the cut locus. Thus the edges e such that $\tau_T(e)$ is null-homotopic are those such that the dual of e splits C into two components, one of which is a tree, and we know how to find them in $O(n)$ time.

If a shortest non-null-homologous closed walk is desired, the only difference is that one needs to compute the bridge edges of C instead. □

6.2.2 Algorithm for Low-Genus Graphs

Theorem 6.8 ([M]). *Assume G has asymmetric lengths. In $O(g^{1/2}n^{3/2}\log n)$ time, we can compute a shortest non-trivial closed walk in (Σ, G).*

Proof. We use a divide-and-conquer approach using separators. If G has $O(1)$ vertices, we can just apply Lemma 6.7 for each vertex in turn. For the general case, we compute an $O(\sqrt{gn})$-size separator for G in $O(n)$ time [100,124]. This is a set $W \subseteq V(G)$ of vertices of G such that $G - W$ is the disjoint union of two subgraphs H_1 and H_2, each with at most $2n/3$ vertices. We can compute a shortest non-trivial closed walk intersecting W in $O(g^{1/2}n^{3/2}\log n)$ time by applying Theorem 6.6 for each vertex of W in turn. If a shortest non-trivial closed walk does not meet W, then it belongs entirely to H_1 or H_2.

To compute the shortest non-trivial closed walk in H_1 (and similarly in H_2), the idea is to erase from G all the vertices not in H_1 together with their incident edges, and to apply induction in this new graph. However, by doing this, the graph may become non-cellular, so one needs to add edges with infinite weights. With some care, this step can be achieved in $O(n)$ time.

The analysis of the recursion yields a running-time of $O(g^{1/2}n^{3/2} \times \log n)$, as claimed. However, a subtlety arises here: To obtain this complexity, it is necessary that, at each step, the number of edges of the graph

	undirected	directed
weighted	$\boldsymbol{O(n^2 \log n)}$ [109]	$O(n^2 \log n)$ [M]
	$\left.\begin{array}{ll} O(g^{3/2}n^{3/2}\log n) & \text{non-sep} \\ g^{O(g)}n^{3/2} & \text{non-db} \end{array}\right\}$ [47]	$\boldsymbol{O(g^{1/2}n^{3/2} \log n)}$ [M]
	$g^{O(g)} n \log n$ [167]	$2^{O(g)} n \log n$ non-sep [111]
	$O(g^3 n \log n)$ [43]	$\left.\begin{array}{ll} \boldsymbol{O(g^2 n \log n)} & \text{non-sep} \\ g^{O(g)} n \log n & \text{non-db} \end{array}\right\}$ [106]
	$\boldsymbol{O(g^2 n \log n)}$ [44]	$\boldsymbol{O(g^3 n \log n)}$ non-db [121]
	$\boldsymbol{g^{O(g)} n \log\log n}$ [147]	
unweighted	$O(n^3)$ [238] (see [191])	$\boldsymbol{O(n^2)}$ [M]
	$\boldsymbol{O(n^2)}$ [L]	$\boldsymbol{O(gnk)}$ [L, M, in preparation]
	$\boldsymbol{O(gnk)}$ [L]	
	$O(gn)$ for 2-short [109]	
	$\boldsymbol{O(gn/\varepsilon)}$ for $(1+\varepsilon)$-short [L]	

Table 6.1. Time complexities of the various algorithms to compute a shortest non-trivial cycle (in both the combinatorial and cross-metric models). "Non-sep" and "non-db" mean non-separating and non-disk-bounding, respectively; k is the size of the output. Within each category, the results are listed in chronological order. The best complexities known to date are in bold (there can be several of them in each category due to the tradeoff between g, n, and k).

be linear in the number of its vertices. This is the case if the graph has no face of degree one or two (Lemma 2.2). So an easy but crucial preprocessing step is necessary at each call of the recursion: It consists of modifying the input graph to remove such faces. We omit the details. $\square$

The $O(g^{1/2})$ factor appears only because genus g graphs have separators of size $O(\sqrt{gn})$. The same algorithm works (with a different running-time) for surface-embedded graphs with small separators, for example, graphs with bounded tree-width [M, Theorem 6].

6.3 Concluding Remarks

We have omitted the case of unweighted, directed graphs in our description, because it is perhaps more esoteric. However, applying the techniques of this chapter, we can obtain the counterparts of Theorem 6.1 and Corollary 6.2 in the directed case also. Table 6.1 summarizes the known algorithms to compute shortest non-separating or non-disk-bounding cycles in all four settings (directed or undirected, weighted or unweighted).

Some open cases remain open. For example, no approximation algorithm is known to compute efficiently an almost-shortest non-trivial cycle in the directed, unweighted case. In the directed setting, it is not known how to compute a shortest non-trivial *simple* loop through a given basepoint.

CHAPTER 7

OTHER SHORTEST CURVES WITH PRESCRIBED TOPOLOGICAL PROPERTIES

The first section of this chapter is a condensed version of an article written with Jeff Erickson [G]. The second section presents results obtained with Erin Chambers, Jeff Erickson, Francis Lazarus, and Kim Whittlesey [H]; the NP-hardness proof in that section is taken from a joint article with Sergio Cabello and Francis Lazarus [O].

In the last two chapters, we have studied algorithms to compute shortest non-disk-bounding (or, equivalently, non-null-homotopic) and non-separating (or, equivalently, non-null-homologous) loops and cycles. In this chapter, we consider the computation of shortest curves with other prescribed topological properties.

In the first section, we study the following problems: Given a path, compute a shortest path homotopic to it. Similarly, given a cycle, compute a shortest cycle (freely) homotopic to it. We give efficient algorithms for both problems. The techniques rely on the use of (a finite part of) the universal cover of Σ, which we build using techniques from combinatorial hyperbolic geometry.

Then, in Section 7.2, we consider the problem of computing a shortest ***splitting*** cycle, i.e., a cycle that is simple, separating, but non-disk-bounding. It turns out that this problem is NP-hard; however, we give a near-linear algorithm for fixed genus. At a high level, the algorithm enumerates $g^{O(g)}$ homotopy classes, one of which necessarily contains a shortest splitting cycle; this technique has been recycled later [51,112].

It is worth noting at this point that other topological types of cycles have been considered. Erickson and Worah [115] give algorithms, using rather different techniques, to compute a shortest ***essential*** cycle on a surface possibly with boundary: a simple cycle that is neither disk-bounding nor homotopic to a boundary component. Their algorithms run in $O(n^2 \log n)$ time, or $O(n \log n)$ time if the surface is fixed. Cabello et al. [45] compute a non-separating cycle that is as short as possible within its homotopy class in $O(n \log n)$ time (one has no control over the homotopy class of the output cycle, however).

The ***multiplicity*** of a curve on (Σ, G^*) is the maximum number of times an edge of G^* is crossed by that curve.

7.1 Tightening Curves

In this section, we consider the problem of computing shortest homotopic paths and cycles on a combinatorial surface. This is a shortest path problem with a natural topological constraint, and it has been studied in the case of the plane with obstacles [24, 63, 98], or, more generally, in locally Euclidean triangulated manifolds [141]. While these special cases are inspiring, they all consider surfaces with boundary; as it turns out, the case of surfaces without boundary requires additional tools. Other works (not presented here) solve the problem for *simple* curves [C, E, F], with worse time complexities. A subsequent more heuristic work [248] considers the same problem, but the complexity of the algorithm is not studied and turns out to be exponential in the worst case.

7.1.1 Our Results

For simplicity of exposition, in this section, we essentially focus on orientable surfaces of genus at least two, without boundary; this is the most instructive case. (We will occasionally digress on the case of the torus.)

Theorem 7.1 ([G]). *Assume Σ is orientable, of genus at least two. Let p be a walk in (Σ, G) with complexity k. After a preprocessing step that takes $O(gn \log n)$ time, we can compute a shortest walk in (Σ, G) homotopic to p in $O(gnk)$ time.*

Theorem 7.2 ([G]). *Assume Σ is orientable, of genus at least two. Let γ be a closed walk with complexity k. After a preprocessing step that takes $O(gn \log n)$ time, we can compute a shortest closed walk freely homotopic to γ in $O(gnk \log \log(nk))$ time.*[1]

It should be emphasized that these theorems are stated in the combinatorial model: The input and output curves are walks on the graph G, possibly with repeated vertices and edges. (Actually, the output of the algorithms may have self-crossings, and it is not clear how to convert it to the cross-metric setting within the indicated time bounds.) However, as we shall see, the proof techniques heavily rely on the cross-metric setting.

In the case of genus one (the torus), after some adaptations in the proofs, the results above are still valid, but with an additional $O(k)$ factor.[2] While Theorem 7.1 extends to non-orientable surfaces (by taking their orientable double cover), Theorem 7.2 does not. Specifically, the technique does not seem to extend to one-sided closed walks.

[1]The published version indicates a running-time of $O(gnk \log(nk))$. However, the recent result by Italiano et al. [147], stated in Theorem 5.9, improves this to $O(gnk \log \log(nk))$.

[2]We believe that this factor can be removed (unpublished work with Laurent Jouhet during his Master's internship).

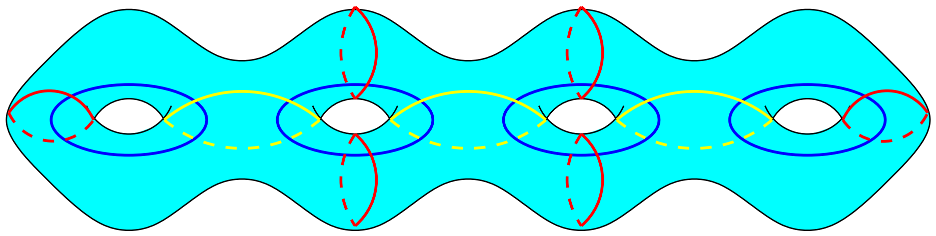


Figure 7.1. An octagonal decomposition built by our algorithm.

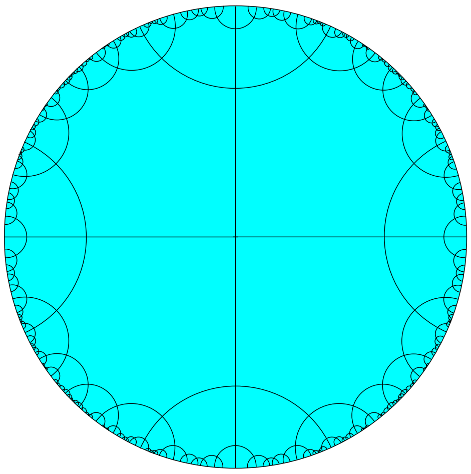


Figure 7.2. Universal cover of an octagonal decomposition.

7.1.2 Overview of the Techniques

Let us say that a family of curves is ***tight*** if each curve in the family is as short as possible in its homotopy class.

The preprocessing step consists of computing a tight ***octagonal decomposition***. An octagonal decomposition is an arrangement of simple cycles on the surface in which every vertex has degree four and every face has degree eight. See Figure 7.1. (As a side remark, we note that similar tight decompositions arise in the context of Teichmüller theory; specifically, if we start with a pants decomposition Γ and add, for each cycle γ in Γ, a new cycle crossing γ twice, we obtain a hexagonal decomposition [117, Chapter 3]. We could consider hexagonal decompositions instead of octagonal ones without any modification.)

On the other hand, consider a tiling of the unit open disk with octagons, where each vertex has degree four; see Figure 7.2. (Such a tiling is a tiling of the hyperbolic plane with regular right-angled octagons, but we will not need this fact.) The surface equipped with the octagonal decomposition, and the unit disk equipped with the octagonal tiling, look alike locally. Actually, there is a well-defined ***projection*** π from the unit disk onto the surface that is, locally, a homeomorphism; in particular, it maps the interior of each octagon in the plane bijectively onto the interior of an octagon on the surface. In technical terms, the octagonal tiling of the disk is a ***covering space*** of the surface Σ (see any textbook on algebraic topology, e.g., Stillwell [229, Section 1.4], for the formal definition

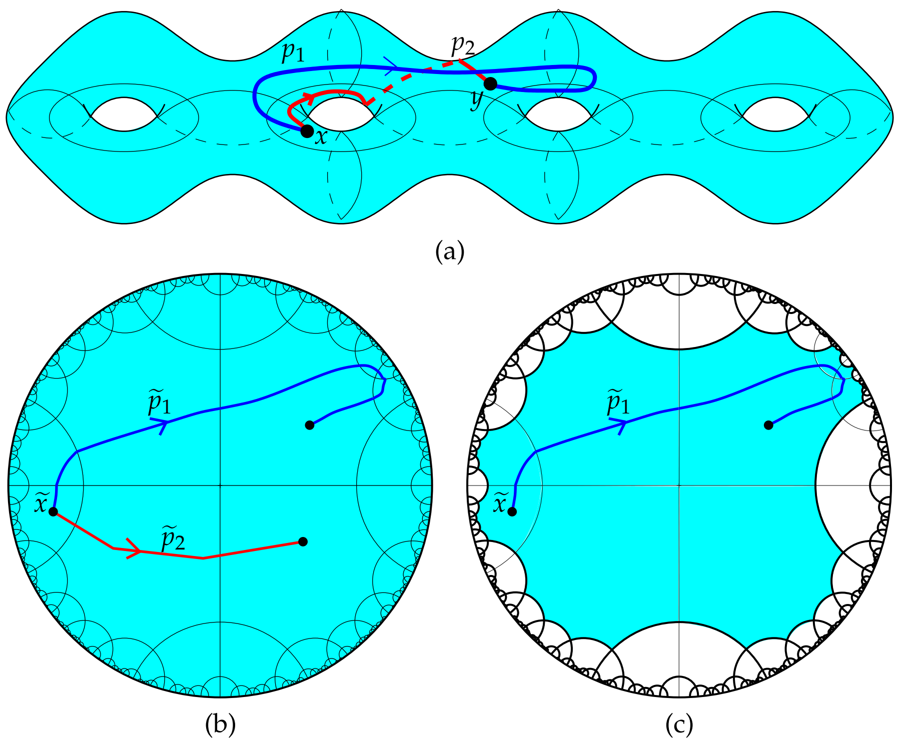


Figure 7.3. (a) Two paths p_1 and p_2, with endpoints x and y. (b) Two lifts $\widetilde{p}_1$ and $\widetilde{p}_2$ of these paths in the universal cover, starting from the same lift $\widetilde{x}$ of x. The fact that these two lifts have different final endpoints witnesses the fact that p_1 and p_2 are not homotopic. (c) The relevant space of $\widetilde{p}_1$.

and properties). Actually, since every loop in a disk is contractible, the octagonal tiling of the disk is a model for the ***universal cover*** $\widetilde{\Sigma}$ of the surface Σ.

A ***lift*** of a path p in Σ is a path $\widetilde{p}$ in $\widetilde{\Sigma}$ such that $\pi \circ \widetilde{p} = p$. If p is a path in Σ, with initial endpoint x, and $\widetilde{x} \in \pi^{-1}(\{x\})$, there is a unique lift $\widetilde{p}$ of p with initial endpoint $\widetilde{x}$. Furthermore, two paths p_1 and p_2 in Σ with the same endpoints are homotopic if and only if they have lifts in $\widetilde{\Sigma}$ with common endpoints; see Figure 7.3(a–b).

Let us focus on tightening walks first (Theorem 7.1). The problem can be reformulated as computing a shortest path in $\widetilde{\Sigma}$ between the endpoints of a lift of p. Of course, this does not help much, since $\widetilde{\Sigma}$ is infinite. The general approach to solve this problem is as follows:

1. Compute a lift $\widetilde{p}$ of the input walk p in the universal cover $\widetilde{\Sigma}$ of Σ;
2. build a suitable part $\widetilde{R}$ of the universal cover of Σ, containing the endpoints of $\widetilde{p}$;
3. compute the shortest path $\widetilde{p}'$ in $\widetilde{R}$ between the endpoints of $\widetilde{p}$;
4. return the projection of $\widetilde{p}'$ in Σ.

Of course, $\widetilde{R}$ has to be "large enough" to contain the (unknown) shortest path in $\widetilde{\Sigma}$ between the endpoints of $\widetilde{p}$, but "small enough" to obtain a good complexity. This is where the tight octagonal decomposition is useful: As it turns out, $\widetilde{p}'$ can cross a lift of a cycle in the tight octagonal decomposition only if $\widetilde{p}$ does.

The free homotopy version is more delicate, because the lift of a cycle in the universal cover is not a cycle, but a path (unless the input cycle is contractible), so a different strategy is needed.

7.1.3 Preprocessing Step: Tight Octagonal Decompositions

The preprocessing step consists in applying the following result:

Theorem 7.3. *Assume Σ is orientable, of genus at least two. In $O(gn \log n)$ time, we can compute an octagonal decomposition of (Σ, G^*) made of tight cycles. Furthermore, the multiplicity of each cycle is $O(1)$.*

We will omit the proof of this result, which is quite long and technical. The main steps of the construction are shown in Figure 7.4. Essentially, the algorithm combines the following operations (all ultimately relying on shortest paths computations with Dijkstra's algorithm):

i. Cut the surface along some already computed curves, obtaining a surface with boundary;

ii. paste a disk or a torus along a boundary component of a surface;

iii. compute a shortest path;

iv. compute a shortest non-separating loop with a given basepoint (Theorem 5.1);

v. on a topologically simple surface, compute a shortest cycle homotopic to a given boundary (we can handle this case separately).

Moreover, to get an efficient running-time, algorithms by Reif and Frederickson [122, 208] and by Cabello et al. [45] are used as subroutines.

The main difficulties for the proof of Theorem 7.3 are (1) to bound the multiplicity of each cycle, and (2) to prove that every output cycle is tight. (1) follows from a careful analysis of the construction and from technical lemmas bounding the multiplicity during the operation (v). The following central lemma ensuring (2) is a consequence of Hass and Scott [133] (see Farb and Margalit [116, Lemma 3.16] for a related result in the area of mapping class groups):

Lemma 7.4. *Let α and β be each either a cycle or a path with endpoints on the boundary of Σ. Assume that α and β are simple and disjoint. Furthermore, assume that α is tight in Σ, and, if it is a cycle, assume it is non-contractible. Then β is tight in $\Sigma \setminus\!\!\setminus \alpha$ if and only if β is tight in Σ.*

7.1.4 Shortest Homotopic Walks

Let D be a tight octagonal decomposition of a cross-metric surface (Σ, G^*) without boundary, and let $\widetilde{D}$ be the set of all lifts of all cycles in D to the

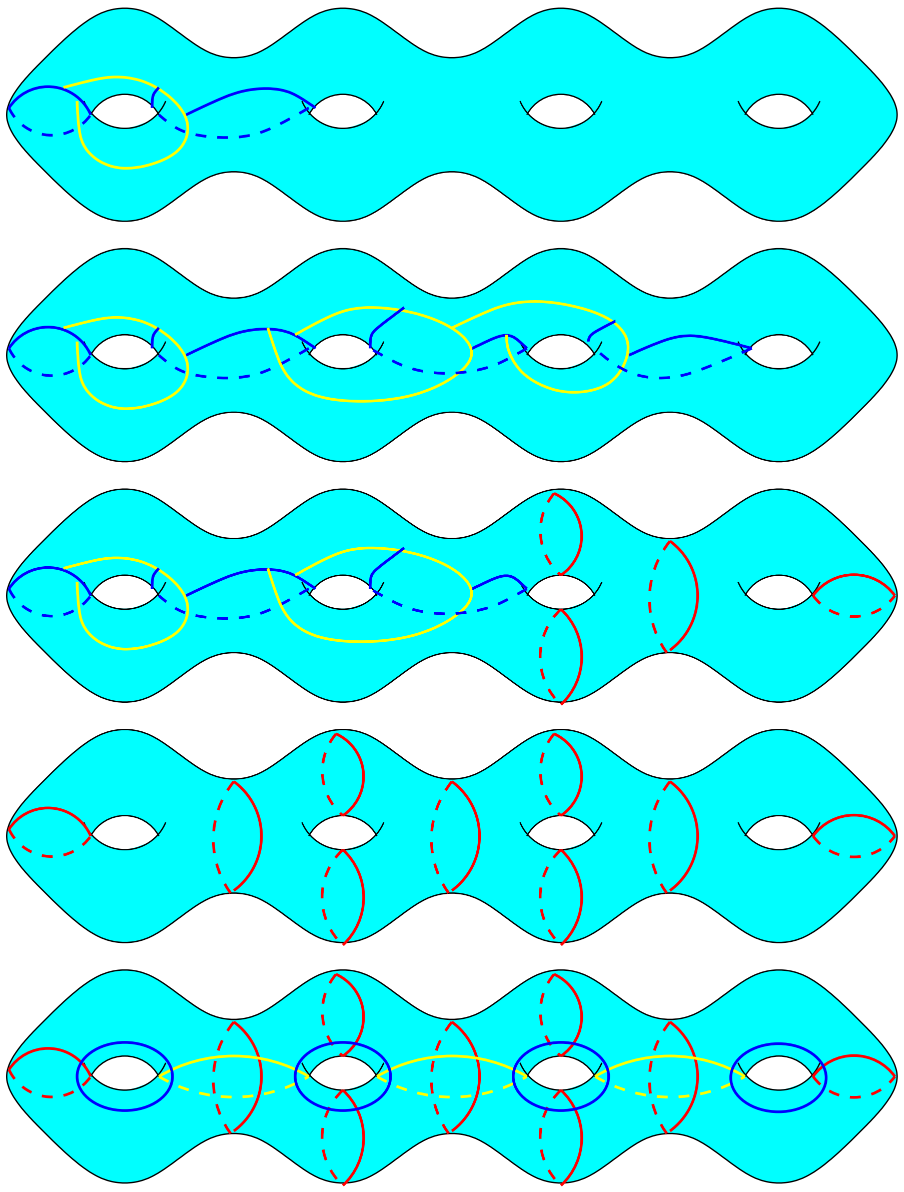


Figure 7.4. The construction of an octagonal decomposition. In a first step, the algorithm computes tight paths that split the surface into an annulus. Then, going “backwards”, a pants decomposition is computed. Finally, some cycles must be computed within pairs of adjacent pairs of pants.

universal cover $\widetilde{\Sigma}$; refer back to Figure 7.2. As noted above, combinatorially, $\widetilde{D}$ is a tessellation of the hyperbolic plane with right-angled octagons, meeting four at a vertex. With this analogy in mind, we call any lift of a cycle in D to the universal cover $\widetilde{\Sigma}$ a ***line***. A line separates the plane into two ***half-spaces***.

Let $\widetilde{p}$ be a lift of the input path p, with endpoints $\widetilde{x}$ and $\widetilde{y}$. Define the ***relevant space*** of $\widetilde{p}$, denoted by $\widetilde{R}$, to be the union of the octagons accessible from $\widetilde{x}$ by crossing only lines of $\widetilde{D}$ that are crossed by $\widetilde{p}$; see Figure 7.3(c). In other words, it is the intersection of all half-spaces bounding $\widetilde{\ell}$ and containing $\widetilde{x}$, for all lines $\widetilde{\ell}$ *not* crossed by $\widetilde{p}$, and is thus a ***convex polygon***.

Lemma 7.5. *Some shortest path in $\widetilde{\Sigma}$ between $\widetilde{x}$ and $\widetilde{y}$ is contained in $\widetilde{R}$.*

Proof. Let $\widetilde{p}'$ be a shortest path between $\widetilde{x}$ and $\widetilde{y}$, with as few crossings with $\widetilde{D}$ as possible. It turns out that $\widetilde{p}'$ crosses each line at most once. This concludes, because every line $\widetilde{\ell}$ separates the plane, and thus every path from $\widetilde{x}$ to $\widetilde{y}$ has to cross $\widetilde{\ell}$ with the same parity (an odd number of times if $\widetilde{\ell}$ separates $\widetilde{x}$ and $\widetilde{y}$, even otherwise).

To prove that fact, two side results are needed, whose proofs are omitted:

i. Every subpath of a line is tight;

ii. two distinct lines cross at most once.

Now, assume that $\widetilde{p}'$ crosses a line $\widetilde{\ell}$ at least twice, at points $\widetilde{u}$ and $\widetilde{v}$. By (i), the subpath of $\widetilde{\ell}$ between $\widetilde{u}$ and $\widetilde{v}$ is a shortest path, so we can replace the part of $\widetilde{p}'$ between $\widetilde{u}$ and $\widetilde{v}$ with a path running along the corresponding part of $\widetilde{\ell}$, without increasing the length of $\widetilde{p}'$. Furthermore, using (ii), one proves that this exchange results in a path with fewer line crossings, which contradicts our choice of $\widetilde{p}'$. □

The algorithm computes the relevant space $\widetilde{R}$, the shortest path between $\widetilde{x}$ and $\widetilde{y}$ in that space, and projects that path back onto the surface. Correctness follows directly from Lemma 7.5. We omit the construction of the relevant space and several other details.

Once $\widetilde{R}$ is built, there remains to compute a shortest path in that space, which takes linear time because it amounts to computing a shortest path in a planar graph, which is feasible in linear time [140]. So the complexity is linear in the size of $\widetilde{R}$, which is $O(n)$ times the number of octagons in it. As it turns out, with some technical tweaks, the number of lines crossed by the input path p is $O(gk)$. It then follows from combinatorial properties of the regular tiling of the hyperbolic space, already noted by Dehn [77], that the number of octagons in $\widetilde{R}$ is also $O(gk)$, whence the overall $O(gnk)$ time complexity.

As mentioned above, surprisingly, the case of the torus ($g = 1$) is a priori more expensive computationally. To see this, consider the analog of an octagonal decomposition in the case of genus one. In this case, the decomposition used is a pair of cycles crossing exactly once on the torus, and they lift to a tessellation of the Euclidean plane with squares. If $\widetilde{p}$ is a path in this plane, with endpoints $\widetilde{x}$ and $\widetilde{y}$, the relevant space of $\widetilde{p}$ is the intersection of all half-spaces bounding $\widetilde{\ell}$ and containing $\widetilde{x}$, for all lines $\widetilde{\ell}$ not crossed by $\widetilde{p}$. There are cases where the relevant space consists of $\Omega(k^2)$ squares: for example, if the lift $\widetilde{p}$ crosses $k/2$ "horizontal" lines and $k/2$ "vertical" lines.

This also explains why the complexity of the algorithm (Theorem 7.1) is linear in k in the case of genus two or more, but its analog for the torus has (a priori) a running-time that is quadratic in k. (Not surprisingly, the torus has no octagonal decomposition.)

7.1.5 Shortest Freely Homotopic Closed Walks

Let γ be a closed walk in (Σ, G). To compute a shortest closed walk freely homotopic to γ, the previous approach has to be adapted. Instead of lifting γ to the universal cover $\widetilde{\Sigma}$ of Σ, we lift it to its ***annular cover*** $\widehat{\Sigma}_\gamma$: this is a covering space that is homeomorphic to an open annulus, in which (at least) one lift $\widehat{\gamma}$ of γ is a cycle.

One can still define a subspace $\widehat{R}$ of $\widehat{\Sigma}_\gamma$, a subset of octagons containing $\widehat{\gamma}$ that must contain a shortest cycle homotopic to $\widehat{\gamma}$. This space $\widehat{R}$ is actually the *projection* onto $\widehat{\Sigma}_\gamma$ of a convex polygon in $\widetilde{\Sigma}$. More specifically, let γ^5 denote the fifth power of γ, i.e., the cycle winding five times around γ, and let $\widetilde{p}$ be a path that is a lift of γ^5 in the universal cover. The relevant space of $\widehat{\gamma}$ in $\widehat{\Sigma}_\gamma$ is the projection onto $\widehat{\Sigma}_\gamma$ of the relevant space of $\widetilde{p}$.

Then, combinatorially, $\widehat{R}$ is a gluing of octagons that is an annulus, and $\widehat{\gamma}$ is a cycle that separates the two boundaries of the annulus. We have to compute a shortest cycle homotopic to $\widehat{\gamma}$ in this annulus; this is a shortest simple cycle separating the two boundaries of the annulus, which can be done efficiently, using the result of Theorem 5.9 by Italiano et al. [147].

7.2 Shortest Splitting Cycles

In Chapter 5, we have seen algorithms to compute shortest non-disk-bounding and non-separating cycles. The efficiency of the algorithms follows from the fact that whether a cycle is non-disk-bounding or non-separating can be seen in a suitable homotopy or homology group. Computing a shortest disk-bounding or a shortest separating cycle is trivial: Just return a small cycle not intersecting G^*.

Henceforth, let us assume that Σ is orientable. For the purpose of this discussion, let us say that two simple cycles on Σ have the same *topological type* if there is a self-homeomorphism of Σ that maps one to the other. It is not hard to prove that two simple cycles have the same topological type if and only if both are non-separating, or both are separating and the genera of the surfaces cut by these cycles correspond. It is natural to ask for algorithms that compute shortest cycles of a given topological type. So far, we know how to compute a shortest non-separating cycle; the case of a shortest disk-bounding cycle is trivial. Therefore, there remains to study the complexity of computing a shortest separating cycle that splits the surface into two subsurfaces of given non-zero genera.

Let us say that a cycle is ***splitting*** if it is simple and separating, but does not bound a disk. (Since Σ is orientable, splitting cycles exist only when the genus is at least two.) We study the problem of computing a shortest splitting cycle. (Computing a splitting cycle, without length constraint, is easy.) In particular, we prove that this problem is NP-hard, but that it becomes easy if g is fixed, by giving an $O(n \log n + g^{O(g)} n \log\log n)$-time algorithm. Furthermore, it is easy to see that the algorithm extends without modification if one requires a cycle splitting the surface into two subsurfaces of prescribed genera.

The crucial property of shortest splitting cycles that we prove and use in the algorithm is (roughly) the following: There exists a bound (in our case, $O(g)$) on the number of crossings between a shortest splitting cycle and an arbitrary shortest path. This structural property is basically the only one used; it implies that one can enumerate a set of $g^{O(g)}$ *candidate* free homotopy classes, one of which is guaranteed to contain a shortest splitting cycle.

In particular, the technology developed for our algorithm was applied subsequently in other contexts. Chambers et al. [51] use it to compute a minimum (s,t)-cut in surface-embedded graphs (they note that a minimum (s,t)-cut corresponds to a certain shortest null-homologous subgraph in the dual graph, and show that such subgraphs can be decomposed into cycles, each of which crosses a shortest path $O(g)$ times). Erickson and Nayyeri [112] study the following computational geometry problem: In the plane, given a set of p polygonal obstacles, a set of distinct pairs of points (s_i, t_i) on the boundary of these polygonal obstacles, compute a set of shortest disjoint paths connecting the pairs (s_i, t_i) and avoiding the obstacles.[3] Although the problem seems to be unrelated to surface graphs, the same techniques apply. Erickson and Nayyeri prove that a path in the solution crosses a shortest path $2^{O(p)}$ times, and this bound is enough to apply the technique above.

7.2.1 NP-hardness

Theorem 7.6 ([H]). *Computing a shortest splitting loop or cycle in an arbitrary cross-metric surface is NP-hard.*

The proof technique was later refined [O] to prove Theorem 9.1. We present an outline of the proof of Theorem 7.6 using the refined technique. The NP-hardness proof proceeds by reduction from the following NP-complete problem: Determine whether a given planar graph H with maximum degree 3 has a Hamiltonian cycle [146, Lemma 2.1]. So let H be such a graph. See Figure 7.5 for an overview of the reduction.

[3]More precisely, instead of disjointness, one requires the paths to be non-crossing, because otherwise shortest disjoint paths do not exist: In the limit case, the paths are

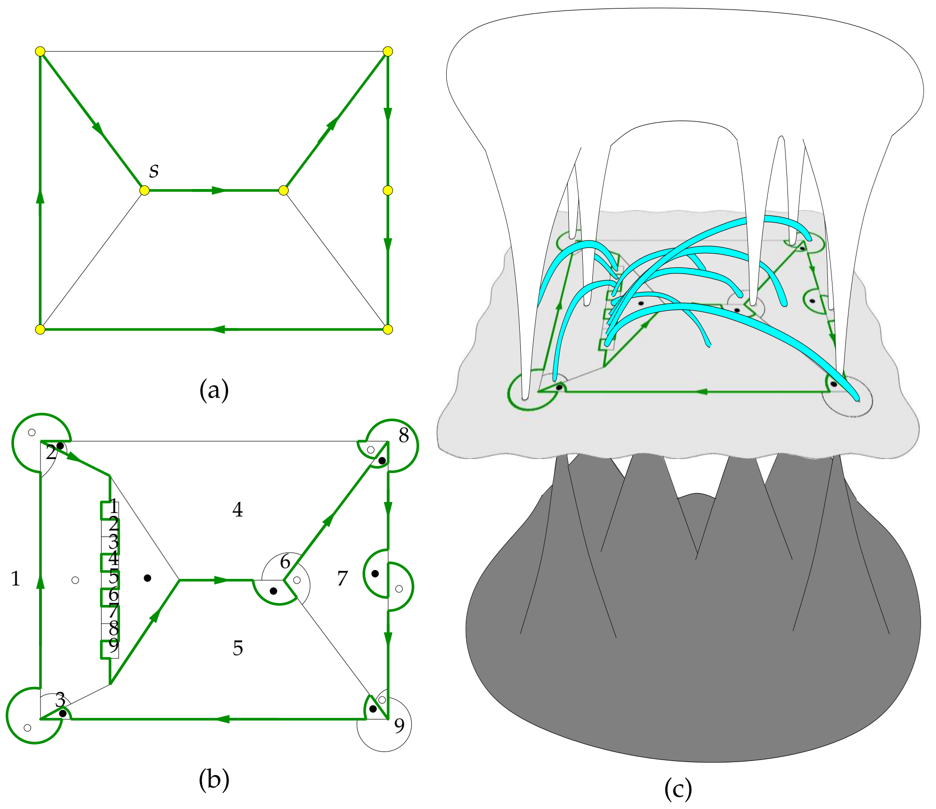


Figure 7.5. Overview of the reduction from Hamiltonian cycle in planar graphs with maximum degree 3. (a) An original instance H with a solution; s is an arbitrary vertex of H. (b) The corresponding graph H'; the small black and white disks inside the faces indicate their color. (c) A part of the corresponding surface (only a part of the middle gray area is shown; the zig-zag boundary is actually closed with a disk).

We transform the instance into another planar graph H' (see Figure 7.5(b)), each face of which is labeled with either a number (a given number appearing exactly twice), or a color, "black" or "white". It turns out that H has a Hamiltonian cycle if and only if there is an oriented closed walk without repeated vertices in H' that has the faces with the same color on the same side, and that has both faces with the same number on the same side.

Then a surface Σ containing H' is built (Figure 7.5(c)) by connecting all the white faces of H' with tunnels to a "top" sphere, connecting all the black faces of H' with tunnels to a "bottom" sphere, and connecting the pairs of faces of H' with the same number using annuli. Now, H has a Hamiltonian cycle if and only if H' has a closed walk without repeated vertices that splits (or separates) Σ.

H' is not cellular on Σ, but we can make it cellular by adding edges; let G be the resulting graph. We assign appropriate weights on G: essentially, unit weights on each edge coming from H, infinitesimally small weights on the edges in H' but not in H, and infinitely large weights on the edges in G but not in H'. Then H has a Hamiltonian cycle if and

non-crossing, but not strictly disjoint.

only if the length of a shortest splitting cycle in the cross-metric surface (Σ, G^*) is at most $|V(H)| + 1/2$, which concludes the high-level description of the proof of Theorem 7.6.

As a side note, it is also easy to deduce that the same problem, restricted to unweighted cross-metric surfaces, is NP-hard. To see this, after the reduction above, subdivide each edge of G (with infinitesimally small weight, unit weight, or infinitely large weight) with an adequate number of unweighted subedges. On the other hand, we can see two open problems:

- Is it NP-hard to compute a shortest loop or cycle in a cross-metric surface that splits Σ into surfaces of prescribed genera?
- Does the shortest splitting cycle problem admit good approximation algorithms?

7.2.2 Algorithm: Enumeration of Homotopy Classes

Theorem 7.7 ([H]). *In an orientable cross-metric surface (Σ, G^*), computing a shortest splitting cycle can be done in $O(n \log n) + g^{O(g)} n \log \log n$ time.*[4]

The main structural property used by our algorithm is the following:

Proposition 7.8. *Let C be a shortest cut graph with a single arbitrary vertex. Let γ be a shortest splitting cycle, crossing C a minimum number of times. Then γ crosses $O(g)$ times each loop in C.*

This bound is tight: loops in C can be crossed $\Omega(g)$ times by γ.

Proof. From Section 5.3.3, we know that each loop ℓ in C is primitive, and can therefore be split into two shortest paths. So let p be a shortest path that is a subpath of some loop ℓ; we prove that γ crosses p at most $O(g)$ times.

The intersection points of $\gamma \cap p$ partition γ into *segments*. Contracting p on the surface to a single point p_0, the segments now become a set of pairwise disjoint simple loops L with basepoint p_0. The goal is to bound the number of these loops.

As a warm-up, let us prove that no face of the embedded graph L is a monogon. If it were the case, before contracting p to a point, we would have a disk on Σ bounded by a subpath of p and a subpath of γ. Pushing the subpath of γ across the subpath of p would result in a splitting cycle that has fewer crossings with C, and is no longer than γ (since p is a shortest path). So L has no monogon.

At this point, it would suffice to prove that L has no bigon. Because then, Lemma 2.2 implies that $|L| = O(g)$, as desired. This statement is not quite true. What is true, and sufficient for our purposes, is that L has no loop, the two incident faces of which are both bigons. This is proved by

[4]The running-time is $g^{O(g)} n \log n$ in the published version, but Theorem 5.9 by Italiano et al. [147] allows to decrease the complexity by using their algorithm as a subroutine in the algorithm of Kutz [167].

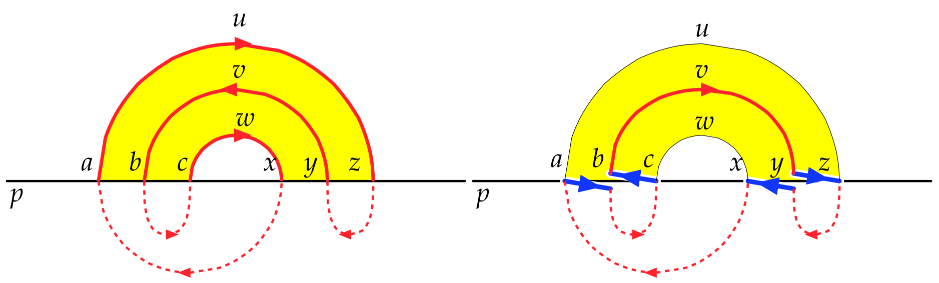


Figure 7.6. One case of the exchange argument. The paths u, v, and w correspond to loops in the surface where path p is contracted to a point; in that surface, v has one bigon on each side. The exchange argument shows that such a situation is impossible, by finding a shorter splitting cycle with fewer crossings with p.

an exchange argument similar to, but more complicated than, the monogon argument above: If we have a loop L with two incident bigons, we can modify γ by removing some segments from γ and connecting the resulting endpoints with paths running along p; the orientation of some segments is reversed. We omit the full proof, as it is rather technical and distinguishes several subcases; one of the simplest cases is illustrated in Figure 7.6. □

The *crossing sequence* of a cycle γ records the intersections of γ with the loops in C, in cyclic order along γ. Any two cycles with the same crossing sequence are homotopic, although two homotopic cycles can have different crossing sequences. A crossing sequence is *simple* if it can be generated by a simple cycle; non-simple cycles can have simple crossing sequences.

Proposition 7.8 implies that some shortest splitting cycle γ crosses $O(g)$ times each loop in C. Our algorithm enumerates a superset of all simple crossing sequences that satisfy this property. For this purpose, cut Σ along C, obtaining a *polygonal schema*: a $4g$-gon D where sides need to be glued by pairs to reobtain Σ. This cutting operation also cuts the unknown splitting cycle γ into *segments* that cut across D. Because γ is simple, no two of these segments cross. The segments of γ can be grouped into subsets according to which pair of loops of C they meet on the boundary of D (Figure 7.7). We abstract and dualize the polygonal schema by replacing each edge of the polygonal schema with a vertex and connecting vertices that correspond to consecutive edges. Now, each subset of segments corresponds to a diagonal between two vertices of the dual $4g$-gon. Since no two segments cross, these diagonals cannot cross. In particular, all the diagonals belong to some triangulation of the dual polygon.

Thus the candidate crossing sequences of a shortest splitting cycle are described by *weighted triangulations*, which consist of a triangulation of the dual polygon, each of whose edges is weighted with an integer between 0 and $O(g)$. The label of an edge in the triangulation represents the number of times that the cycle runs along that edge. There are $g^{O(g)}$ such labelings, which we can enumerate in constant amortized time per

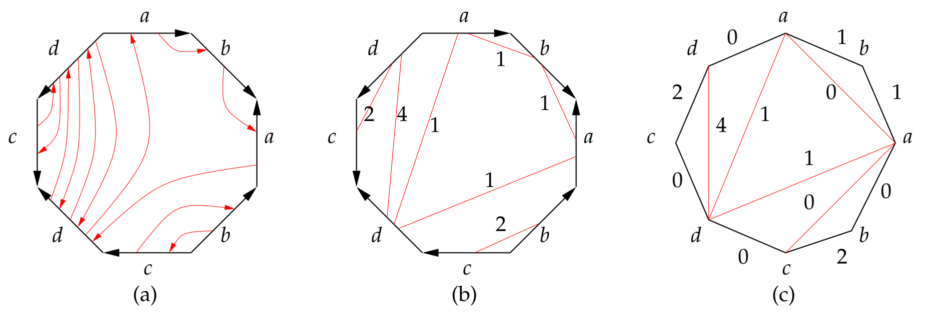


Figure 7.7. (a) A splitting cycle on a double-torus ($g = 2$). (b) The corresponding subsets of segments as weighted diagonals of the dual polygon; each label indicates the number of segments contained in a subset. (c) The corresponding weighted triangulation.

labeling.

A weighted triangulation corresponds to a splitting cycle if and only if it corresponds to a *single* cycle that is both separating and non-disk-bounding. These conditions can be tested by computing the topology of each component of the surface cut along the cycle(s). Moreover, this can be done in $O(g^2)$ time per weighted triangulation, by performing the computations in the abstract surface obtained by gluing the edges of the polygonal schema D by pairs, ignoring the $O(n)$ internal complexity of this polygonal schema.

For each weighted triangulation corresponding to a splitting cycle, we need to compute a shortest cycle with that crossing sequence. Our method is inspired from Hershberger and Snoeyink [141] (compare also with Kutz [167]) and similar in spirit to that used to compute shortest freely homotopic closed walks (Section 7.1.5). Essentially, one can build a part of the annular cover of the candidate splitting cycle, obtained by gluing together copies of the polygonal schema D, compute a shortest closed walk in that annulus that separates the two boundaries of the annulus (Theorem 5.9), and project back to Σ; this takes $O(g^2 n \log\log n)$ time. One subtlety is that the cycle found is not necessarily simple, though it has the correct homotopy type; with some work, one can remove self-intersections in $O(g^2 n \log g)$ time by combining a result by Takahashi et al. [233, Theorem 2] with a linear-time algorithm for shortest paths in planar graphs by Henzinger et al. [140].

Part III

More Results for Curves and Graphs on Surfaces

CHAPTER 8

TESTING ISOTOPY OF GRAPHS

The results of this chapter were obtained with Arnaud de Mesmay during his Master's internship and Ph.D. [R].

In this chapter, we give an algorithm for the following problem: Given a graph embedded on an orientable surface in two different ways, is there a continuous motion from the first embedding to the second that keeps the graph embedded at all times during the deformation? In other words, does there exist an ***isotopy*** between these two graph embeddings?

8.1 Context and Result

8.1.1 Motivations

As an important special case, consider a finite set of obstacle points P in the plane, and two embeddings G_1 and G_2 of the same graph G into $\mathbb{R}^2 \setminus P$. Does there exist a "morph" between G_1 and G_2 (possibly moving the vertices and bending the edges) that avoids passing over any obstacle? Such questions are relevant for morphing applications: To compute a morph between two images, it is helpful to first build a deformation between compatible graphs representing the most salient features of the images (see, e.g., Gotsman and Surazhsky [125]). It is sometimes desirable to add some topological requirements on the morph, e.g., to force some area of the deforming image to always cover a fixed point of the plane during the deformation. Such requirements can be encoded using obstacle points, since a face of the graph containing an obstacle point has to contain it during the whole deformation. Another potential application area is cartography and geographic information systems, where it is often needed to simplify a map, but one needs to check that the simplified map is still correct topologically. In this field, testing homotopy for paths in the punctured plane is already useful [46], but we believe it is even more relevant to test for isotopy of graphs. For example, does a simplified version of a road network leave the houses or villages on the appropriate side of the simplified roads?

More generally, assume that we have a triangulated surface in $\mathbb{R}^3$, and two embeddings G_1 and G_2 of the same graph G on that surface: Each graph G_i is encoded by its combinatorial arrangement with the triangulation. Can we continuously move G_1 to G_2? In this setting, the graphs G_1 and G_2 might represent textures on the surface, and the question is whether one can continuously move one texture so that it coincides with the other.

8.1.2 Our Result

Let G be a graph, and G_1 and G_2 be two embeddings of G on an orientable surface Σ. (In particular, the correspondence between the vertices and edges of G_1 and G_2 is known.)

G_1 and G_2 are encoded as follows. We assume that a fixed surface Σ is given, together with a fixed cellular graph embedding H on Σ. The embedding G_1 is assumed to be in general position with respect to H, and is described by the combinatorial arrangement of G_1 with H, and similarly for the embedding G_2. Equivalently, though we will not use this terminology, one can assume that (Σ, H) is an unweighted cross-metric surface, and consider that G_1 and G_2 are graphs drawn (not simultaneously) in this cross-metric surface. We emphasize that our algorithm does not consider the intersections between G_1 and G_2, which are not given in the input. Here is our main result.

Theorem 8.1 ([R]). *Let Σ be an orientable surface, possibly with boundary; let H be a fixed graph cellularly embedded on Σ. Let G_1 and G_2 be two graph embeddings of the same graph G on Σ, each in general position with respect to H. Given the combinatorial arrangements of G_1 with H (respectively, G_2 with H), of complexity k_1 (respectively, k_2), we can determine whether G_1 and G_2 are isotopic in $O(k_1 + k_2)$ time.*

In particular, the general position assumption implies that G_1 and G_2 are in the interior of Σ. The role of H is solely to provide a reference framework on which all graphs are drawn; its complexity is necessarily smaller than k_1 (or k_2). We emphasize that, in the conclusion of the theorem, the isotopy has to map each vertex or edge in G_1 to the corresponding vertex or edge in G_2.

8.1.3 Known Variants

Theorem 8.1 can be derived easily from known techniques when G is a cycle. Indeed, as mentioned in Section 3.2, an algorithm by Lazarus and Rivaud [173] allows to decide in optimal linear time whether two cycles on an orientable surface are homotopic and, on the other hand, homotopy and isotopy are almost the same concept for simple cycles [103]: Two simple cycles are isotopic if and only if either (1) they are non-contractible and homotopic or (2) they are contractible (disk-bounding) and they are the boundaries, with the same orientation (clockwise or counterclockwise), of disks on the surface.

Another variant is known, where the input is the data of two embeddings G_1 and G_2 of G such that each vertex of G is at the same position in G_1 and G_2, and one wants to decide whether there exists an isotopy between G_1 and G_2 that does *not* move the vertices. Although never described explicitly, it follows from previous works [F, G] that this can be done in polynomial time.

An ***ambient isotopy*** of a surface is a self-homeomorphism of Σ that is isotopic to the identity;[1] namely, a homeomorphism $h_1 : \Sigma \to \Sigma$ such that there exists a continuous family of self-homeomorphisms $(h_t)_{t\in[0,1]} : \Sigma \to \Sigma$ where h_0 is the identity. Two graph embeddings are ***ambient isotopic*** if there exists an ambient isotopy of the surface that takes one to the other. It is known that two graph embeddings in the interior of Σ are isotopic if and only if they are ambient isotopic (this fact actually *follows* from our proof).

We mention in passing that our problem is quite related to the theory of mapping class groups: If G_1 and G_2 are cellularly embedded on Σ, a self-homeomorphism of Σ that maps G_1 to G_2 represents a unique element of the mapping class group, and our problem amounts to testing whether this element is the identity. This problem has already been tackled computationally [26, 234], but the inputs of these algorithms are different and their complexities are not made explicit and are much higher than linear, so that these results do not help.

Alexander's theorem (see, e.g., Farb and Margalit [116, Lemma 2.1]) is the most basic result on mapping class groups. We mention it for further use:

Theorem 8.2 (Alexander's theorem)**.** *Let D be a disk and $h : D \to D$ be a self-homeomorphism that restricts to the identity on the boundary ∂D of D. Then h is an ambient isotopy.*

8.2 Main Ideas for the Proof

8.2.1 Overview

If two graph embeddings G_1 and G_2 of the same graph G are (ambient) isotopic, then clearly (1) some oriented self-homeomorphism of the surface maps G_1 to G_2; and (2) if γ is a cycle in G (possibly with repeated vertices), its images in G_1 and G_2 are homotopic.

It was shown by Ladegaillerie [168] that such necessary conditions are, in fact, sufficient. However, the second condition is not algorithmic, since there are infinitely many cycles in G. The complexity of the cycles in Ladegaillerie's proof is not made explicit, but would be too large to obtain a linear complexity.[2] We therefore reprove Ladegaillerie's characterization, also providing an explicit set of cycles Γ in G of linear overall

[1]This is not the standard definition, but a slight adaptation more suitable for our purposes.

[2]Actually, one of our earlier attempts followed Ladegaillerie's technique more closely and had an additional $O(g)$ factor in the complexity.

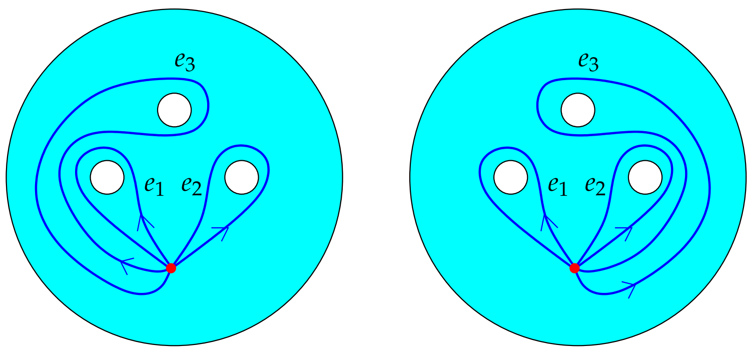


Figure 8.1. Two embeddings G_1 and G_2 of a one-vertex graph with three loop edges on the sphere with four punctures. These two embeddings are not isotopic (even if the vertex is allowed to move), although there exists an oriented self-homeomorphism of the surface sending one to the other, and the four cycles following the boundaries of the faces are homotopic in G_1 and G_2.

complexity that have to be tested for homotopy. Our algorithmic results then follow relatively easily using known techniques.

We note that finding such a set Γ is not straightforward. In particular, a natural candidate for Γ would be the set of all facial cycles in G_1 (and thus in G_2). However, Figure 8.1 shows that the condition is not fulfilled, even in the case where the surface is the sphere with four punctures.

Before reproving Ladegaillerie's characterization, we first need another characterization, of independent interest, on the existence of an isotopy between certain *stable* curves.

8.2.2 Stable Curves

Let Γ be a family of cycles in general position on Σ. A ***0-gon*** in Γ is a simple cycle in Γ that bounds a disk containing no piece of Γ in its interior. Recall that a k-gon of Γ, for $k \geq 1$, is a face of the arrangement of Γ of degree k that is a disk. We say that Γ is ***stable*** if its arrangement contains no k-gon for $k \leq 3$.

Proposition 8.3. *Let Σ be an orientable surface and let $\Gamma_1 = (\gamma_{1,1}, \dots, \gamma_{1,n})$ and $\Gamma_2 = (\gamma_{2,1}, \dots \gamma_{2,n})$ be two* stable *families of cycles on Σ. There exists an ambient isotopy of Σ mapping each curve $\gamma_{1,j}$ of Γ_1 to the corresponding curve $\gamma_{2,j}$ of Γ_2, not necessarily pointwise, but preserving the orientations of the curves, if and only if the following conditions are satisfied:*

- *There exists an oriented self-homeomorphism h of Σ mapping each curve in Γ_1 to the corresponding curve in Γ_2, not necessarily pointwise, but preserving their orientations, and*
- *each curve of Γ_1 is homotopic to the corresponding curve of Γ_2.*

It is obvious that, if there exists an ambient isotopy, then both conditions are satisfied; the hard part is to prove the converse.

It follows from a result by de Graaf and Schrijver [76] that each stable family is *minimally crossing*: Each cycle has the minimum number of self-intersections among all cycles in general position in its homotopy class, and similarly each pair of cycles has the minimum number of intersections among all pairs of cycles in general position in their respective homotopy classes. Indeed, de Graaf and Schrijver prove that one can make any family of cycles in general position minimally crossing via a sequence of *Reidemeister moves* that do *not* increase the number of crossings; the definition of stability implies that no Reidemeister move at all is possible. Our proof uses some techniques of that paper. Let us sketch it for the case $g \geq 2$, $b = 0$.

Under this assumption, by the uniformization theorem, Σ can be endowed with a *hyperbolic metric* of constant curvature -1. It is standard [41] that every free homotopy class contains a unique geodesic (a cycle that is a shortest path in the neighborhood of all its points), which is also a shortest homotopic cycle. Recycling the technique from de Graaf and Schrijver [76], we may assume, after applying an ambient isotopy of the surface, that each cycle in Γ_1 (or Γ_2) is in a neighborhood of the unique geodesic homotopic to that cycle. The arrangement of these geodesics forms a graph K on Σ. It turns out that the vertices of K have degree four; at such places, exactly two geodesics cross. (If there were a vertex of degree at least six, then three pieces of cycles would cross pairwise in the neighborhood of that vertex, which would imply the existence of a k-gon for $k \leq 3$.) If every edge of K is used exactly once by exactly one cycle in Γ_1 (respectively, Γ_2), then it is not hard to build an isotopy from Γ_1 to Γ_2, by pushing the piece of Γ_1 that runs along an edge of K to the corresponding piece of Γ_2. This is the gist of the proof, though there are technical details in the case where several pieces of cycles run along a given edge of K; we omit these details.

8.2.3 Ladegaillerie's Characterization Revisited

We reprove and strengthen Ladegaillerie's result [168]:

Proposition 8.4. *Let G_1 and G_2 be two graph embeddings of a graph G on Σ. Assume that there is an oriented self-homeomorphism h of Σ mapping G_1 to G_2. There exists a family Γ of cycles in G such that the following holds: If, for each cycle γ in Γ, the images of γ in G_1 and G_2 are homotopic, then there exists an ambient isotopy of Σ mapping G_1 to G_2 (pointwise). Furthermore, the cycles in Γ use each edge of G $O(1)$ times times in total and can be computed in linear time.*

(By virtue of our definition of *cycle*, the cycles in Γ may repeat edges and vertices.)

The idea for the proof of Proposition 8.4 is to build a family of *stable* curves Γ_1 in a tubular neighborhood of G_1 such that G_1 is included in the union of Γ_1 and of the faces of the arrangement of Γ_1 that are disks. It is essentially possible to build such a family in linear time, but we omit the construction; see Figure 8.2 for an example. We obtain a family of cycles Γ in G whose images under G_1 are slight perturbations of Γ_1, and that use each edge of G at most twice in total.

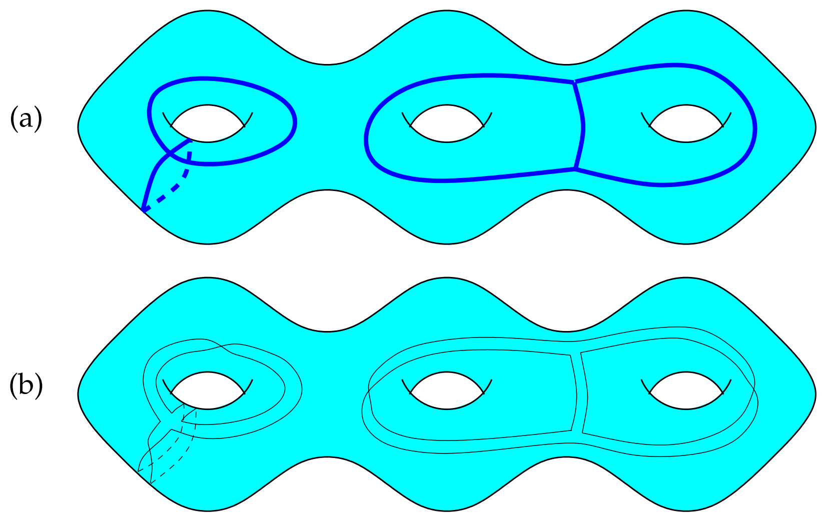


Figure 8.2. (a) A graph embedding G_1. (b) The corresponding family Γ, made of three cycles.

Recall that h is an oriented self-homeomorphism of Σ mapping G_1 to G_2. Therefore, $\Gamma_2 := h(\Gamma_1)$ is a stable family as well. Assume that the images of Γ in G_1 and G_2 are homotopic. By construction, this implies that each cycle in Γ_1 is homotopic to the corresponding cycle in Γ_2. Our goal is to prove that some ambient isotopy maps G_1 to G_2; in other words, that there is an ambient isotopy i such that $i|_{G_1} = h|_{G_1}$.

The above homotopy assumption implies with Proposition 8.3 that some ambient isotopy maps each curve in Γ_1 to the corresponding curve in Γ_2, preserving the orientations of these curves. It does *not*, in general, imply that some ambient isotopy maps each curve in Γ_1 to the corresponding curve in Γ_2 *pointwise*. However, it turns out that, by adding a few curves in Γ, we can make sure that such an isotopy exists; we omit the details.

In other words, if we denote by j this isotopy, we have that $j|_{\Gamma_1} = h|_{\Gamma_1}$. Furthermore, since h and j have the same orientation, they map each face of Γ_1 to the corresponding face of Γ_2.

Let f be a face of Γ_1 that is a disk. Since $j|_{\partial f} = h|_{\partial f}$, Alexander's theorem (Theorem 8.2) implies that there is another isotopy j' that is the identity outside f and such that $j' \circ j|_f = h|_f$. By applying this operation for each face f that is a disk, we obtain the existence of an ambient isotopy i such that $i|_f = h|_f$ for each such face f, and still $i|_{\Gamma_1} = h|_{\Gamma_1}$. Finally, since G_1 is included in the union of Γ_1 and of the faces of Γ_1 that are disks, we obtain $i|_{G_1} = h|_{G_1}$, which proves Proposition 8.4.

8.2.4 Algorithm

It is now time to describe the algorithm for Theorem 8.1. Given the combinatorial arrangements of G_1 and H (respectively, G_2 and H) on Σ, the algorithm first checks that there exists an oriented self-homeomorphism of Σ mapping G_1 to G_2. This is relatively easy; let us illustrate first the case where G_1 and G_2 are cellularly embedded. In this case, this amounts to checking that the combinatorial maps of G_1 and G_2 are *isomorphic* and have the same *orientation*. To check that they are isomorphic, if we are using for example the flag representation as described in Section 2.4.2, we know the bijection between the flags of G_1 and those of G_2, and we have to check that they commute with the three involutions vi, ei, and fi. To check that the maps have the same orientation, we have to check that a flag in G_1 has the same orientation as the corresponding flag in G_2. All of this can be done in linear time.

In the general case, G_1 and G_2 are not necessarily cellular; in particular, one has to keep track of the topology of each face of G_1 and of the list of boundary components of each such face, and similarly for G_2. However, this can still be done in linear time with a bit more bookkeeping.

Assuming the oriented homeomorphism test succeeds, by Proposition 8.4, there remains to perform homotopy tests for cycles in G_1 and G_2, which can be done in linear time in their complexities using the aforementioned algorithm by Lazarus and Rivaud [173]. In total, our algorithm is linear in the complexity of the combinatorial arrangements of G_1 with H and of G_2 with H.

8.3 Conclusion

It is also interesting to test isotopy of graphs, where some vertices of the graph are required to be fixed. Also, if we allow the input graph embeddings to meet the boundary, there is a difference between ambient isotopy fixing the boundary, ambient isotopy that allows to move the boundary points, and graph isotopy. It appears that all these cases are tractable using simple variations of our algorithm.

There remain several open problems. First, our algorithm only works in the orientable case, as it relies on the existence of an orientable homeomorphism. Most works on mapping class groups consider the orientable case only [116]; as the question we study is rather related, this may be an indication that the problem is much harder in the non-orientable case.

It may be natural to consider testing isotopy of graphs on surfaces, when the mapping between the edges and vertices of both embeddings is not specified. In other words, we have two embedded graphs G_1 and G_2 on Σ, and we want to determine whether there is an isotopy of the surface that maps the image of G_1 to the image of G_2. If the graph is connected, then there is a linear number of possible ways of choosing the isomorphism between G_1 and G_2, because the isomorphism between the two embedded graphs is uniquely determined once the correspondence between two given flags is chosen; therefore, there is a quadratic-time algorithm. However, in the disconnected case, this is not true anymore, and the problem seems related to the isomorphism test problem.

Finally, it is tempting to generalize the results of Section 7.1 from cycles to graphs, namely, to compute shortest graph embeddings within a given isotopy class; we suspect the problem to be much harder, and solving it, even for graphs with one vertex, would be very interesting.

CHAPTER 9

TESTING THE EXISTENCE OF CYCLES WITHOUT REPEATED VERTICES

This chapter presents the results of an article written with Sergio Cabello and Francis Lazarus [O].

In this chapter, we consider closed walks without repeated vertices on graphs; this corresponds to the usual graph-theoretic notion of cycle. To avoid confusion, let us call a ***real cycle*** a closed walk in a graph that has no repeated vertex and is not reduced to a single vertex.

Let G be a graph cellularly embedded on a surface Σ. It is not always the case that G contains a *real* cycle of a given topological type (e.g., disk-bounding, separating, etc.); for example, if G is a cut graph, it contains no real disk-bounding or separating cycle. It is clear from the results in Chapter 5 that there always exists a real cycle that is non-disk-bounding (respectively, non-separating). However, given a vertex v of G, there does not always exist such a cycle containing v (for a trivial example, consider the case where v has degree one); so what if we want to determine whether there exists a non-separating (or non-contractible) real cycle passing through a given vertex? We exhibit a strong dichotomy in the complexity of these problems, depending on the topological property required: These problems are either NP-hard or solvable in (optimal) linear time.

In the same spirit but for weighted graphs, Cabello [42] studies the complexity of computing a *shortest* real cycle of a given topological type. He provides a polynomial-time algorithm to compute a shortest disk-bounding real cycle in a surface-embedded graph (if such a cycle exists), and proves that it is NP-hard to compute a shortest disk-bounding real cycle through a given vertex, or a shortest separating real cycle. In contrast to Part II and to the result by Cabello [42], here we do not insist in finding *shortest* real cycles.

Another related result by Kobayashi and Kawarabayashi [163] is an algorithm to determine whether there exists an *induced* real cycle through a given set of k vertices of G; in particular, if G is embedded on a fixed surface and k is constant, the problem is solvable in linear time.

		(a) no restriction	(b) one prescribed vertex
(1)	disk-bounding	linear-time	linear-time
(2)	non-disk-bounding	(linear-time)	linear-time
(3)	non-separating	(linear-time)	linear-time
(4)	separating	NP-hard	NP-hard
(5)	splitting	NP-hard	NP-hard

Table 9.1. Complexities of computing real cycles of a given topological type (Theorem 9.1).

9.1 Results

Theorem 9.1 ([O]). *Let G be a graph cellularly embedded on a surface Σ without boundary. The problem of computing a real cycle of a given topological type, possibly requiring the cycle to pass through a prescribed vertex of G, has the complexity indicated in Table 9.1. If no such real cycle exists, it is possible to decide it within the same time bounds.*

(In the table, the known results, following from Chapter 5, are denoted in parentheses.) Case (4a) solves an open problem mentioned by Mohar and Thomassen [191, Problem 4.3.3(b)].

We also prove that the NP-hard problems of Table 9.1 are somehow fixed-parameter tractable:

Theorem 9.2 ([O]). *Let G be a graph of complexity n, cellularly embedded on a surface Σ of genus g without boundary. Let $k \geq 1$ be an integer. The problem of determining a real cycle of length at most k that is separating (respectively, splitting), possibly requiring it to pass through a given vertex, can be solved in $2^{O(g+k)} n \log n$ time.*

9.2 Proof Overview

9.2.1 Proof Overview of Theorem 9.1

The NP-completeness proof in Theorem 9.1 (Cases (4) and (5)) follows from the same ideas as the ones in Section 7.2.1, used to prove NP-hardness of computing the shortest splitting cycle; we omit the details.

Case (1) of Theorem 9.1 (disk-bounding case) is proved by the following simple but delicate argument; for concreteness, let us focus on Case (1a). If there exists a disk-bounding real cycle, let e be an edge of the boundary of the disk, incident to a face f that lies inside the disk. Every cycle that is included in the boundary of f is contractible. It is not always the case that the boundary of f is a real cycle; however, there exists a real cycle, the edge set of which contains e and is included in the boundary of f. The algorithm therefore computes, for each edge e and each face f incident to e, a real cycle containing e and included in the boundary of f (if it exists); if that cycle is contractible (equivalently, disk-bounding), it returns it and halts; if no such cycle is contractible, then the graph has no contractible real cycle.

This algorithm can be implemented so as to run in linear time: Without getting into the details, note that the overall complexity of the cycles is linear; testing for contractibility can be done in overall linear time by a result by Lazarus and Rivaud [173] (though we provide a more direct algorithm for our specific case).

Cases (2b) and (3b) of Theorem 9.1 (the non-disk-bounding and non-separating cases) can be proved as follows. For concreteness, let us assume that we want to determine whether there exists a real non-disk-bounding cycle passing through vertex s. Using an exchange argument, it is equivalent to determine whether there exists a real non-disk-bounding cycle (not necessarily passing through s) in the subgraph H of G that is the union of the blocks of G containing s. (Recall that a *block* of G is a maximal set of edges of G in which every pair of edges is contained in some real cycle.) Then the problem becomes to determine whether H contains a non-disk-bounding cycle, which can be done using cut locus techniques, and specifically Lemma 6.4.

9.2.2 Proof Overview of Theorem 9.2

The proof of Theorem 9.2 relies on the color-coding paradigm of Alon et al. [7]. The basic idea is the following: Randomly color the vertices of G in one of k colors; determine whether there exists a separating or splitting cycle that is *colorful*, namely, has all its vertices colored differently. Repeating this algorithm $2^{\Theta(k)}$ times ensures that, if a separating or splitting real cycle of length at most k exists, it will be colorful for at least one coloring with probability $\Omega(1)$. Furthermore, this simple randomized strategy can be derandomized. So we are left with determining, given a coloring, whether there exists a separating or splitting colorful cycle. Note that all colorful cycles are real by construction, so we do not have to care about repeated vertices anymore; this is the whole point of using color-coding.

As it turns out, determining a separating or splitting colorful cycle (if it exists) can now be done using dynamic programming. For the (easier) separating case, we build a graph with $2^k 2^g |V|$ vertices, where V is the vertex set of G; each vertex of that graph corresponds to a path using one of the 2^k subsets of the k colors and corresponding to one of the 2^g possible homology classes. Our problem amounts to decide the existence of certain paths of length k in this graph.

9.3 Conclusion

Our results extend to surfaces with boundary. It would be interesting to generalize these results to graphs that are embedded on a surface Σ, but possibly not cellularly embedded. Using the same data structure as the one sketched in Chapter 8 for storing non-cellularly embedded graphs, we believe that the results of this chapter extend without much modifications to this setting.

There remain a couple of interesting open questions. We only considered the existence of real cycles passing through a single prescribed vertex. What if we prescribe two (or more) vertices? Our techniques seem to break down. The techniques by Kobayashi and Kawarabayashi [163] may be useful. Finally, in the splitting case, we did not consider the problem of requiring, in addition, that the splitting cycle cuts Σ into surfaces of prescribed genera.

Chapter 10

Combinatorics of Irreducible Triangulations

The material of this chapter is a submitted joint work with Alexandre Boulch and Atsuhiro Nakamoto [Q]; these results were obtained during Alexandre's pre-Master (M1) internship.

Let Σ be a surface of genus g with b boundary components. A ***triangulation*** of Σ is an embedded graph G without loops or multiple edges, each face of which is a 3-gon. (It should also be required that moreover, two distinct triangles do not share three vertices and three edges, though this only happens when Σ is the sphere, which is irrelevant for the rest of the discussion.) More concisely, it is the one-dimensional skeleton of some simplicial complex homeomorphic to Σ.[1]

Contracting an edge of the triangulation (identifying two adjacent vertices in the simplicial complex; see Figure 10.1) is allowed if this results in another triangulation of the same surface. An ***irreducible*** triangulation, sometimes called *minimal* triangulation, is a triangulation in which no edge can be contracted. Every triangulation can be reduced to an irreducible triangulation by iteratively contracting some of its edges.

We provide a bound on the size of an irreducible triangulation of Σ. We first put our result in context and then give a glimpse at the proof technique.

10.1 Previous Results and Motivations

Irreducible triangulations have been much studied, but only in the context of surfaces without boundary, with results of two types. First, the (numbers of) irreducible triangulations of low-genus surfaces have been computed [16, 170, 228, 230–232]. A second series of results proves that finitely many irreducible triangulations exist [17, 18, 149] and shows that the number of vertices of an irreducible triangulation is linear in the genus [148, 187, 193].

[1]In some contexts, a triangulation of a surface is simply an embedded graph such that each face is a 3-gon. Our definition is more restrictive, since on every surface there exists

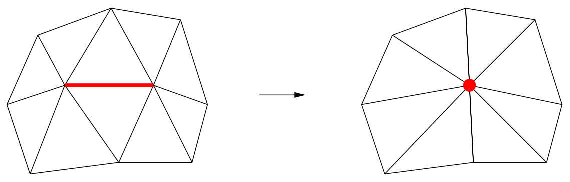


Figure 10.1. Contraction of an edge of a triangulation.

The interest of irreducible triangulations also lies in the fact that some problems on triangulations of surfaces are automatically solved if one can solve them for irreducible triangulations. (This was actually one of our initial motivations for studying irreducible triangulations and their structural properties, though we did not succeed in proving such a result.) For example, Barnette conjectured in 1982 [191, Conjecture 5.9.3] that, on a triangulation of an orientable surface with genus $g \geq 2$, there always exists a splitting cycle without repeated vertices. It is easy to see that if one can prove this fact for irreducible triangulations, then it holds for all triangulations. More generally, again on an orientable surface of genus g, Mohar and Thomassen [191, Conjecture 5.9.5] conjectured that for every h, $0 < h < g$, there exists a cycle without repeated vertices that splits the surface into two surfaces of genus h and $g - h$, respectively. (See also the discussion by Sulanke [231, Sect. 5].) Irreducible triangulations have also been used in the context of diagonal flips on surfaces [194,195].

10.2 Our Result

Here is the result of this chapter.

Theorem 10.1 ([Q]). *Let Σ be a surface of genus g with b boundary components. Then every irreducible triangulation of Σ has $O(g + b)$ vertices.*

Before, this result was only known in the case $b = 0$ (albeit with a smaller multiplicative constant hidden in the $O(\cdot)$ notation). This bound is asymptotically tight. Furthermore, the previous techniques used to prove Theorem 10.1 in the case of surfaces without boundary do not seem to extend to the case $b \geq 1$. In fact, our proof, when specialized to the case $b = 0$, is significantly different from and simpler and more natural than the other techniques. These former proofs, by Nakamoto and Ota [193] and Joret and Wood [148], rely on a deep theorem by Miller [187] stating that the Euler genus of a graph is additive over 2-vertex amalgams (identification of two vertices of disjoint graphs). While the method yields the current best bounds on the number of vertices, it seems a bit unnatural to use the genus of a graph to derive a result on graphs embedded on a fixed surface. Another paper [62] claims a linear bound without using Miller's theorem, but this part of their paper has a flaw (personal communication with the authors).[2]

an embedded graph made of a *single* vertex and only 3-gons.

[2]Specifically, in the proof of their Lemma 3, the authors incorrectly claim that there are at most g pairwise non-homologous cycles on an orientable surface of Euler genus g.

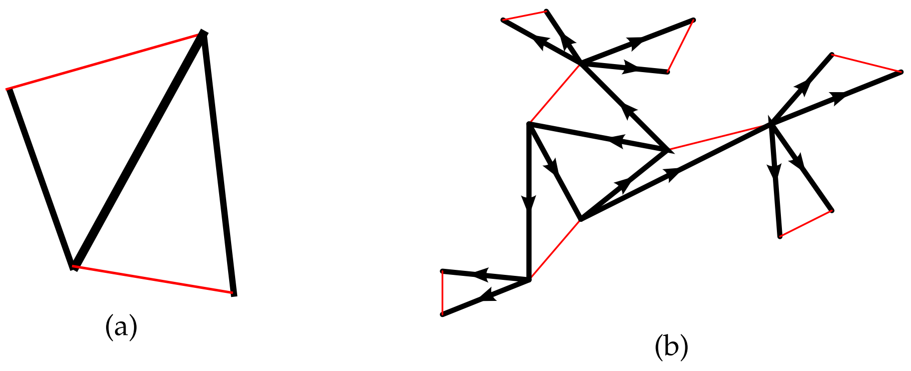


Figure 10.2. (a) In thin lines, two edges e_1 and e_2 of the matching M, with the other edges of the 3-cycles γ_{e_1} and γ_{e_2} in thick lines; one of the edges belongs to both 3-cycles. (b) In bold lines, a connected component of the graph P, a tree plus one edge. In thin and bold lines, a connected component of the graph Γ.

10.3 A Glimpse at the Proof

Since our goal is only to convey the main ideas, we restrict ourselves to the case $g \geq 1$, $b = 0$: Even though our result is not new in this special case, our proof for the general case requires only some technical adjustments. Some refinements of the technique also allow for a better numerical constant in the $O(\cdot)$ bound.

So let G be an irreducible triangulation of Σ. Actually, the only (simple, and essentially already known) properties of G that we use are the following:

1. the degree of every vertex of G is at least four [230, Theorem 1];

2. every edge of G belongs to a non-contractible (equivalently, non-disk-bounding) 3-cycle (cycle of length three) [17, Lemma 1];

3. no ten 3-cycles of G are pairwise edge-disjoint, pairwise homotopic, and non-contractible (equivalently, non-disk-bounding) [17, Lemma 9].

The main idea is to bound the number of edges of a ***matching*** of G (a set of edges with pairwise disjoint endpoints) by $O(g)$; once this is done, a simple argument involving Euler's formula and using Property 1 concludes. So let M be a matching of G.

By Property 2, each edge e of G belongs to some non-contractible 3-cycle γ_e. Since M is a matching, every edge of G belongs to at most two cycles γ_{e_1} and γ_{e_2} (Figure 10.2(a)). It is easy to see that, up to removing at most half of the elements of the matching M, we can assume that the cycles γ_e, $e \in M$, are edge-disjoint. It now suffices to prove that the number of edges of this new matching (still denoted by M) is $O(g)$.

We then partition the cycles γ_e, $e \in M$, according to their homotopy classes. Since, by Property 3, there are at most ten cycles γ_e in each homotopy class, then, again up to removing a constant fraction of the elements in the matching M, we can assume that the cycles γ_e, $e \in M$, are in distinct homotopy classes.

Finally, for each edge e of M, let p_e be the path of length two defined by $\gamma_e \setminus e$. Note that the paths p_e are edge-disjoint. We orient each edge of each path p_e towards the endpoint of p_e it is incident to. Let P be the union of the paths p_e. Since M is a matching, every vertex of P is the final endpoint of at most one oriented edge of P. It follows that P is a *pseudoforest* (Figure 10.2(b)): Every connected component of P is a tree plus possibly a single additional edge. Up to removing one edge e of M per connected component of P, we can assume that P is a forest; again, it is not too hard to see that this removes at most a constant fraction of M.

Now, consider the graph Γ that is the union of the cycles γ_e, for $e \in M$; this is an embedded graph containing P. We contract each tree of P in the graph Γ on the surface Σ. (This operation is different from the contraction of an edge of a triangulation, as described in the beginning of this chapter: Here we do *not* remove loops or multiple edges. It is always legal to contract the edges of a forest in this sense; in contrast, it is never legal to contract a loop, or the edges of a cycle.) This yields a graph Γ' where each edge is a loop and corresponds to an edge of M. Let T be a tree on Σ meeting Γ' exactly at its vertex set. Contracting T, we transform Γ' into a one-vertex graph Γ''. Each edge of Γ'' is a loop, which corresponds to some cycle γ_e and has been obtained from γ_e by a homotopy. So Γ'' cannot contain a monogon or a bigon, for otherwise one cycle γ_e would be contractible, or two cycles γ_{e_1} and γ_{e_2} would be homotopic, which is not the case by one of the earlier steps of our construction.

Bottom line: By Lemma 2.2, Γ'' has $O(g)$ edges. So M also has $O(g)$ edges; since we successively reduced M, but removing at most a constant fraction of the edges of M at each step, this is also true for the original matching. Hence every matching of G has $O(g)$ edges, which concludes.

Part IV

Other Works and Perspectives

CHAPTER 11

OTHER WORKS

This chapter presents three results, obtained with Alexander Schrijver [I]; with Erin Chambers, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite [J]; and with Grégory Ginot and Xavier Goaoc [P].

Here, we present succinctly three works that are not concerned with graphs on surfaces. All of them make a crucial use of some topological tools at various levels of sophistication.

11.1 Shortest Vertex-Disjoint Paths in Planar Graphs

In this section, we consider a special case of the *shortest vertex-disjoint paths problem*, which is stated as follows. Given

- a directed graph G with n arcs,
- non-negative weights (lengths) on the edges of G,
- an arbitrary number k of pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$,

compute vertex-disjoint paths in G connecting the pairs (s_1, t_1), ..., (s_k, t_k), of minimum total length.

The problem is known to be NP-hard. Actually, even *deciding* the existence of such vertex-disjoint paths is NP-hard, even when the graph is planar and undirected [165,207].

11.1.1 Our Result

We give an efficient algorithm in a special case (see Figure 11.1(a)):

Theorem 11.1 ([I]). *The shortest vertex-disjoint paths problem described above is solvable in $O(kn \log n)$ time if we assume that:*

- *G is planar and*
- *there exist distinct faces $s \neq t$ of G such that $s_1, \ldots, s_k$ are incident to s, and similarly $(t_1, \ldots, t_k)$ are incident to t.*

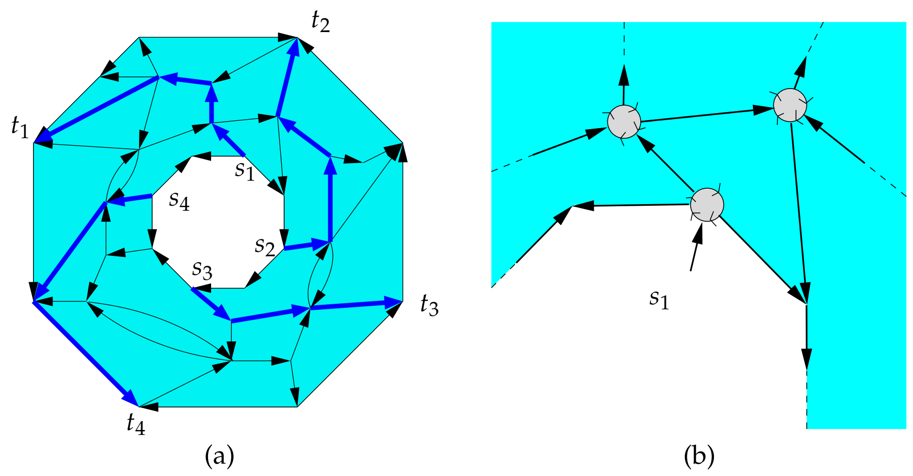


Figure 11.1. (a) An instance of the problem in Section 11.1 and a solution (in bold lines). (b) A portion of the graph G' obtained from the graph G; each vertex of G is replaced by a "ring", and each new vertex has degree at most three. The thin arcs on the rings have length zero.

In particular, we may assume without loss of generality that G is connected and that t is the unbounded face of G. Furthermore, up to renumbering, we can assume that $s_1, \dots, s_k$ appear in clockwise order around face s. Then clearly, the terminals $t_1, \dots, t_k$ must appear also in clockwise order around face t, for otherwise no vertex-disjoint paths can exist.

While these restrictions may seem rather strict, it is not clear at all that it is possible to obtain a polynomial-time algorithm if they are relaxed in any way. A subsequent paper [164] extends this result to the case where the terminals s_i and t_i lie on the boundaries of at most two faces (a face may possibly mix some s_i's and some t_i's), but only in the case $k \in \{2, 3\}$.

11.1.2 Sketch of Proof

The first step to prove Theorem 11.1 is to transform G into another graph G' so that now the problem becomes a problem of *edge*-disjoint paths in G'; without entering into the details, we illustrate the construction in Figure 11.1(b). The proof of this reduction relies on an (elementary but non-trivial) topological argument.

Edge-disjoint paths problem can be modeled by flows, and this allows us to use classical tools from combinatorial optimization. One first computes a minimum-cost flow in G', with unit capacities and with costs equal to lengths; this corresponds to vertex-disjoint paths in G of minimum total length between $\{s_1, \dots, s_k\}$ and $\{t_1, \dots, t_k\}$; however, we have no control over the connections: s_1 may be connected to an arbitrary t_i. Therefore, in a second step, we need to "rotate" the flow clockwise or counterclockwise to modify these connections. It follows from standard techniques that the initial minimum-cost flow can be computed in $O(kn \log n)$ time using k shortest path computations in the residual graph. To rotate the flow, one needs to compute a minimum-cost cycle separating the two faces s and t in G, oriented in the appropriate direction (clockwise or counterclockwise), in the residual graph; this is possible using a minimum cut in the dual residual graph in $O(n \log n)$ time. A convexity argument shows that the number of rotations needed is at most $2k$, which overall gives an $O(kn \log n)$-time algorithm.

11.1.3 Final Remark

It is interesting to note that minimum-cost flows with given "rotation" correspond to minimum-cost flows within a given homology class; each rotation transforms a flow that has minimum cost in its homology class into another. This is reminiscent from subsequent algorithms that compute maximum flows in surface-embedded graphs [53].

11.2 Homotopic Fréchet Distance in the Plane With Obstacles

11.2.1 Fréchet Distance

Let $a, b : [0,1] \to M$ be two curves in an arbitrary metric space M. A well-studied measure of similarity between two such curves is the ***Fréchet distance*** [8]. While other metrics, such as the Hausdorff distance, exist, they are not specific to the case of curves and not well-adapted to them in practice, because they only consider the images of the curves as sets and are oblivious to the "order" in which the image of each curve is traversed. The Fréchet distance has been much studied, extended, and applied; we refer to a recent sample of the existing literature and references therein [15,38,39,71,93,131].

The ***Fréchet distance*** $\boldsymbol{F}(a,b)$ between two curves a and b is also called the *dog-leash distance* because of its following informal definition: It is the minimum length of a leash required to connect a dog and its owner as they walk along their respective curves a and b from one endpoint to the other. (Both the man and the dog are allowed to adjust their speed, but not to backtrack.) More formally, let $\mathrm{dist}(u,v)$ denote the distance between points u and v in the metric space M. A ***reparameterization*** of a curve c has the form $c \circ \alpha$, where α is a bijective, increasing map from $[0,1]$ onto $[0,1]$. The Fréchet distance is defined as follows:

$$F(a,b) = \inf_{\substack{a' \text{ reparameterization of } a \\ b' \text{ reparameterization of } b}} \left(\max_{0 \le t \le 1} \mathrm{dist}\left(a'(t), b'(t)\right) \right).$$

This definition allows the "leash" to switch discontinuously, without penalty, from one side of an obstacle or a mountain to another.

11.2.2 Our Result

We introduce a continuity requirement on the motion of the leash. We require that the leash cannot switch discontinuously from one position to another; in particular, the leash cannot jump over obstacles, and can sweep over a mountain only if it is long enough. We define the *homotopic*

Fréchet distance between two curves a and b as the Fréchet distance with this additional continuity requirement. More formally, a ***leash map*** is a continuous function $\ell\colon [0,1]^2 \to M$ such that $\ell(\cdot,0)$ is a reparameterization of a, and $\ell(\cdot,1)$ is a reparameterization of b. A leash map describes the continuous motion of a leash between a dog walking along a and its owner walking along b; the curve $\ell(t,\cdot)$ is the leash at time t. The ***length*** of a leash map ℓ, denoted by ***len***(ℓ), is the maximum length of any leash $\ell(t,\cdot)$. Finally, the ***homotopic Fréchet distance*** $\overline{\mathbf{F}}(a,b)$ between two curves a and b is the infimum, over all leash maps ℓ between a and b, of the length of ℓ:

$$\overline{F}(a,b) = \inf_{\substack{\text{leash map} \\ \ell\colon [0,1]^2 \to M}} \left(\max_{0 \le t \le 1} \operatorname{len}(\ell(t,\cdot)) \right).$$

When M is a simple polygon, both the standard and the homotopic Fréchet distance coincide, but not in more general environments like the boundary of a convex polyhedron or the plane with obstacles. In particular, when M is the plane minus a finite set of obstacle *points*, these obstacles are completely ignored by the Fréchet distance, while our definition takes them into account.

The motion of the leash defines a correspondence between the two curves that can be used to morph between the two curves—two points joined by a leash morph into each other [97]. Thus, the homotopic Fréchet distance can be thought of as the minimal amount of deformation needed to transform one curve into the other.

Efficiently computing the homotopic Fréchet distance in general metric spaces is an open problem. We present a polynomial-time algorithm for a special case of this problem, which is to compute the homotopic Fréchet distance between two polygonal curves in the plane minus a set of polygonal obstacles.

Theorem 11.2 ([J]). *Let M be the plane $\mathbb{R}^2$ minus a set of polygonal obstacles. Given two polygonal curves a and b, we can compute the homotopic Fréchet distance between a and b in M in $O(n^9 \log n)$ time, where n is the total size of the input.*

11.2.3 Sketch of Proof

Here is a very high-level description of our algorithm. Assume first that we somehow know a single leash, say $\ell(0,\cdot)$, of a leash map ℓ realizing the homotopic Fréchet distance. Then we can lift $\ell(0,\cdot)$ to the universal cover $\widetilde{M}$ of M to a lift $\widetilde{\ell}(0,\cdot)$, and similarly build the lifts $\widetilde{a}$ and $\widetilde{b}$ of a and b that touch the endpoints of $\widetilde{\ell}(0,\cdot)$. As it turns out, it now suffices to compute the homotopic Fréchet distance between $\widetilde{a}$ and $\widetilde{b}$ in $\widetilde{M}$: Essentially, $\widetilde{M}$ is an "infinite simple polygon", in which the standard and homotopic Fréchet distances coincide. We can compute this Fréchet distance in polynomial time by adapting the seminal paper by Alt and Godau [8]; however, for this extension, a couple of additional technical details are needed, including a careful convexity argument and an adaptation of Alt and Godau's use of the *parametric search* technique originally by Megiddo [185] (see also van Oostrum and Veltkamp [243]).

There remains to find the adequate lifts $\widetilde{a}$ and $\widetilde{b}$ in $\widetilde{M}$, since we do not know *a priori* how the optimal leash winds around the obstacles, or more formally the homotopy class of the leash map *relatively* to the curves a and b. For this purpose, we prove that, in the relative homotopy class of an optimal leash map, there exists a path p from a point on a to a point on b that has a special structure: It consists of either a straight-line segment, or of the concatenation of a straight-line segment to a vertex of a polygon obstacle, a shortest path to another vertex of a polygon obstacle, and a straight-line segment. This characterization, which heavily relies on properties of shortest paths and funnels [57, 141, 174] in the universal cover $\widetilde{M}$, shows that only a polynomial number of candidate relative homotopy classes of leash maps need to be considered, at least one of which is guaranteed to realize the homotopic Fréchet distance. This concludes a high-level view of the algorithm.

11.2.4 Other Recent Results

Independently and almost simultaneously to this paper, Cook and Wenk presented an algorithm to compute the Fréchet distance between two curves in the case where the ambient metric space M is a simple polygon [71]. Our and their papers rely on some common technical lemmas; it is also possible that their method allows to improve the running-time of our algorithm. Finally, at least two recent works study further the homotopic Fréchet distance: Chambers and Letscher [54] develops a variant, the *height* of a homotopy, that is, in our notations above, the infimum, over all leash maps ℓ, of the maximum length of $\ell(\cdot, t)$. Har-Peled et al. [130] provide a polynomial-time algorithm to approximate the homotopic Fréchet distance in another case, where the ambient metric space is a obtained by attaching together Euclidean triangles to give a topological disk.

11.3 Helly-Type Theorems for Families of Disconnected Sets

Last but not least, we provide a new topological Helly-type theorem and apply it to problems in combinatorial geometry.

11.3.1 Our Main Result

Helly's original theorem [137] states that, whenever a finite family $\mathscr{F}$ of convex sets in $\mathbb{R}^d$ has empty intersection, then some subset of $\mathscr{F}$ of size at most $d+1$ has empty intersection. In such a situation, we say that the ***Helly number*** of $\mathscr{F}$ is at most $d+1$. Helly himself gave a more general, topological version of this theorem [138], showing that the same conclusion holds if $\mathscr{F}$ is a ***good cover*** in $\mathbb{R}^d$, meaning a finite family of

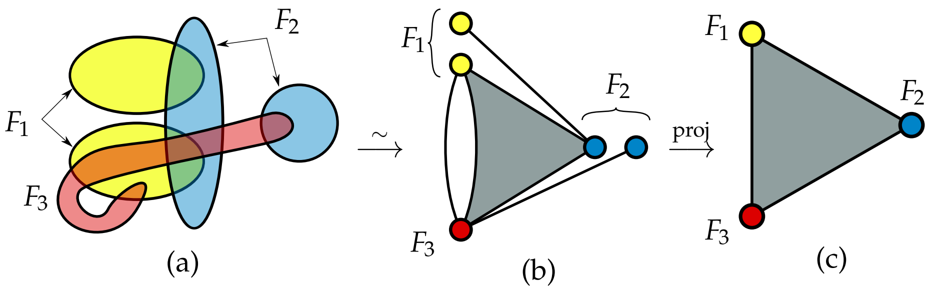


Figure 11.2. (a) A family $\mathscr{F} = \{F_1, F_2, F_3\}$ of objects in the plane, satisfying the hypotheses of Theorem 11.3. (b) The multinerve $M(\mathscr{F})$ of $\mathscr{F}$, a simplicial poset with the same homology as the union of $\mathscr{F}$. Each vertex of $M(\mathscr{F})$ corresponds to a connected component of one object in $\mathscr{F}$. Each edge of $M(\mathscr{F})$ corresponds to a connected component of the intersection of two objects in $\mathscr{F}$. More generally, each simplex of dimension i of $M(\mathscr{F})$ corresponds to a connected component of the intersection of $i+1$ objects in $\mathscr{F}$. (c) The nerve $N(\mathscr{F})$ of F, a "squashed" version of $M(\mathscr{F})$.

open sets such that the intersection of every subfamily of $\mathscr{F}$ either is empty or has the homotopy type of a point.

We show an extended version of this result, allowing each subfamily to intersect in more than one connected component. Recall that a ***homology cell*** is a topological space that has the $\mathbb{Q}$-homology of a point. (We point out that the definition of homology we consider here is much more general than that defined in Chapter 2; we refer the reader to any textbook in algebraic topology, for example Hatcher [135], for an introduction to homology. Also, technically we should consider *reduced* homology, but we omit this precision for simplicity.)

Theorem 11.3 ([P])**.** *Let $\mathscr{F}$ be a family of open sets in $\mathbb{R}^d$ such that the intersection of every subfamily of $\mathscr{F}$ is the disjoint union of at most r homology cells. Then the Helly number of $\mathscr{F}$ is at most $r(d+1)$.*

The bound $r(d+1)$ is tight. This theorem generalizes results obtained in the last twenty years by Amenta [9], Matoušek [184], and Kalai and Meshulam [150].

11.3.2 Sketch of Proof

Our proof combines several ingredients. One ingredient is an extension of the *nerve theorem*. The ***nerve*** $N(\mathscr{F})$ of a family $\mathscr{F}$ of sets is a simplicial complex that encodes the intersection patterns of its subfamilies. More precisely, it is a simplicial complex with vertex set $\mathscr{F}$; a subfamily $\mathscr{G} \subseteq \mathscr{F}$ is a simplex if and only if $\mathscr{G}$ has non-empty intersection. The nerve theorem states that, if $\mathscr{F}$ is a good cover, then its nerve has the same homotopy type as the union of the objects in $\mathscr{F}$. We generalize the nerve theorem to handle families $\mathscr{F}$ that satisfy the hypotheses of Theorem 11.3. More precisely, we define the ***multinerve*** $M(\mathscr{F})$ of such a family $\mathscr{F}$ to be (essentially) the set of all connected components of the intersections of any subfamily of $\mathscr{F}$, ordered by reverse inclusion. One can "almost" represent $M(\mathscr{F})$ by a simplicial complex: The simplices of dimension i are the connected components of the intersections of subfamilies of cardinality $i+1$; see Figure 11.2. However, it may happen that, for example, two edges are connected to the same vertices. Therefore, $M(\mathscr{F})$ is a more general ***simplicial poset*** that is, intuitively, a blown-up version of the nerve $N(\mathscr{F})$. We generalize the nerve theorem and prove that $M(\mathscr{F})$ is homologically equivalent to the union of the objects in $\mathscr{F}$, using spectral sequence arguments from algebraic topology. (We suspect that such a result can also be obtained by extending more combinatorial proofs of the nerve theorem [30].)

It is not too hard to prove that, if $N(\mathscr{F})$ has trivial homology in dimension k and higher, then the Helly number of $\mathscr{F}$ is at most $k+1$. In particular, if $\mathscr{F}$ is a good cover, then $N(\mathscr{F})$ has the same homology as the union of the objects in $\mathscr{F}$ (by the nerve theorem), hence has trivial homology in dimension d and higher; so the Helly number of $\mathscr{F}$ is at most $d+1$. Thus Helly's topological theorem follows quite directly from the nerve theorem. The strategy of the proof of Theorem 11.3 is similar: We show that $N(\mathscr{F})$ has trivial homology in dimension $r(d+1)-1$ and higher. One thing that we know is that $M(\mathscr{F})$ has trivial homology in dimension d and higher, by our generalization of the nerve theorem. Therefore, the missing link is (roughly) a connection between the homology of $M(\mathscr{F})$ and that of $N(\mathscr{F})$. We obtain such a relation by extending a proof by Kalai and Meshulam [151], who prove it in the case where $M(\mathscr{F})$ is a simplicial complex. Their rather technical proof, using Leray numbers and also spectral sequences, extends for our purposes, although some steps need to be adapted in a non-trivial way.

Incidentally, Eckhoff and Nischke [95] have given an alternative, more combinatorial proof of Kalai and Meshulam's result. While we are enthusiastic about this simpler proof technique, it does not seem that adapting it instead of that by Kalai and Meshulam would allow us to obtain Theorem 11.3.

11.3.3 Applications

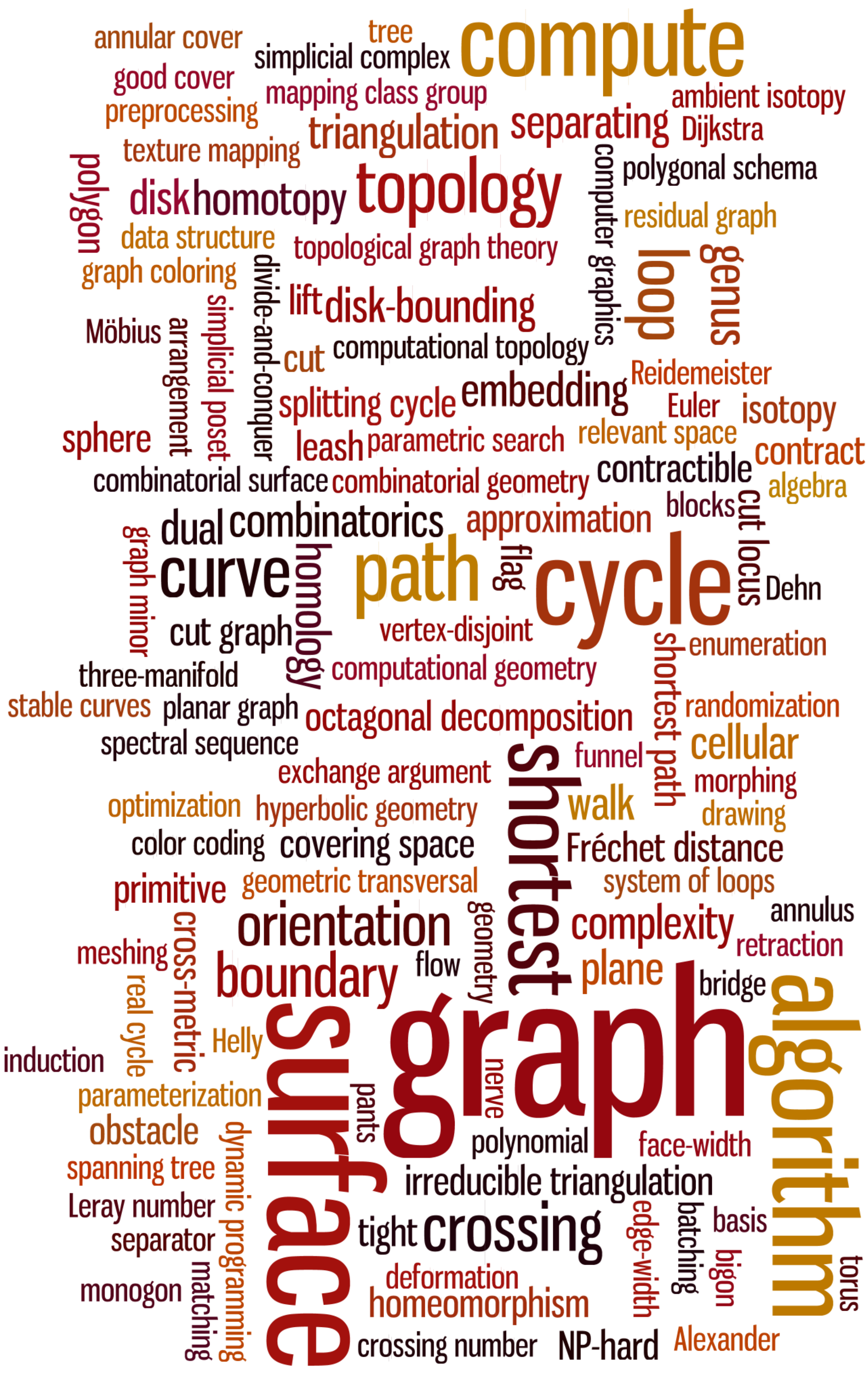
The original motivation for this work lies in *geometric transversal theory*. Typical results in this area are of the following form: Let $\mathscr{F} = \{F_1, \ldots, F_n\}$ be a family of convex sets in $\mathbb{R}^d$. For each i, let $\varphi(F_i)$ be the set of lines that intersect F_i. Under which circumstances can one ensure that the Helly number of the $\varphi(F_i)$'s is bounded? In other words, is there some number k such that, if every k-tuple of objects in $\mathscr{F}$ is pierced by some line, then all the objects of $\mathscr{F}$ are simultaneously pierced by some line? Many such results have been found in this field, often using ad hoc techniques, in particular in the case where $\mathscr{F}$ is a set of parallelotopes in $\mathbb{R}^d$ [218], of disjoint translates of a planar convex set [242], or of disjoint unit balls in $\mathbb{R}^d$ [64,72].

We give a generic method that handles all these cases simultaneously, sometimes giving a better bound on the Helly number than previously known. The idea is that, in many cases, the sets of lines $\varphi(F_i)$ "almost" satisfy the hypotheses of Theorem 11.3. In particular, let $\mathscr{G} \subseteq \mathscr{F}$. If the

F_i's are pairwise disjoint, then the intersection of $\varphi(\mathscr{G})$ can have several connected components, corresponding (under mild conditions) to the *geometric permutations* of the objects in $\mathscr{G}$, namely, the possible orderings of $\mathscr{G}$ along a line that pierces all of them. Theorem 11.3 thus applies to the family $\varphi(\mathscr{G})$, except in two respects:

- the $\varphi(F_i)$'s do not lie in some space $\mathbb{R}^k$, but in some Grassmannian manifold of dimension $2d-2$;
- for small subfamilies $\mathscr{G} \subset \mathscr{F}$, the topology of each connected component of $\varphi(\mathscr{G})$ is not necessarily a homology cell.

We can further extend Theorem 11.3 to handle these cases; while the first point is easy to take care of, the second point is more delicate and requires additional technicalities, which we omit.


annular cover
tree
simplicial complex
compute
good cover
mapping class group
ambient isotopy
preprocessing
triangulation
separating
Dijkstra
texture mapping
computer graphics
polygonal schema
polygon
disk
homotopy
topology
residual graph
data structure
topological graph theory
loop
genus
graph coloring
divide-and-conquer
lift
disk-bounding
Möbius
arrangement
simplicial poset
cut
computational topology
Reidemeister
splitting cycle
embedding
Euler
isotopy
sphere
leash
parametric search
relevant space
contract
combinatorial surface
combinatorial geometry
contractible
algebra
blocks
cut locus
dual
combinatorics
approximation
graph minor
curve
homology
path
flag
cycle
Dehn
cut graph
vertex-disjoint
shortest path
enumeration
three-manifold
computational geometry
stable curves
planar graph
octagonal decomposition
randomization
spectral sequence
shortest
funnel
cellular
exchange argument
morphing
optimization
hyperbolic geometry
walk
drawing
color coding
covering space
Fréchet distance
primitive
geometric transversal
system of loops
orientation
geometry
complexity
annulus
meshing
cross-metric
retraction
boundary
flow
plane
bridge
real cycle
Helly
surface
graph
algorithm
induction
nerve
parameterization
pants
obstacle
polynomial
face-width
dynamic programming
spanning tree
irreducible triangulation
Leray number
tight
crossing
edge-width
batching
basis
separator
bigon
torus
deformation
monogon
matching
homeomorphism
crossing number
NP-hard
Alexander

CHAPTER 12

PERSPECTIVES

We conclude with some open questions and perspectives on topological algorithms for graphs on surfaces, along with some more long-term research directions on surfaces embedded in $\mathbb{R}^3$.

Other shortest decompositions. As already mentioned in Section 5.4.3, there are many conceivable ways of decomposing a surface topologically; for many purposes, it is useful to compute shortest such decompositions. Rather surprisingly, this is currently feasible only for cut graphs with prescribed vertex set (Theorem 5.2) and shortest homology bases [114, Theorem 4.3]. In both cases, an underlying algebraic structure allows for a greedy strategy to yield the optimal decomposition. Are some other optimal decompositions computable in polynomial time? Are the remaining ones NP-hard to compute, and, in the affirmative, does there exist efficient approximation strategies? In particular, is computing the shortest cut graph, without prescribing the vertex set, fixed-parameter tractable with respect to the genus? Is it W[1]-hard?

Graph drawing on surfaces. Here is an interesting but not well-defined problem, generalizing the graph drawing problem to surfaces: Given a graph G and a surface Σ, build a "nice" embedding of G on Σ. If Σ is fixed, one can check in linear time whether G can be embedded on Σ, and, in the affirmative, build an embedding [158, 189]; can we somehow compute an embedding with some bounds on its length? To have an "aesthetic" embedding, probably other objective functions than the sum of the lengths of the edges should be optimized.

Deformations. We can determine whether two curves or graphs are homotopic or isotopic. Assuming this is the case, how hard is it to actually compute a homotopy or isotopy? For most purposes, one would like to determine, in some sense, an "optimal" deformation. For example, let us measure the complexity of an isotopy on a cross-metric surface (Σ, G^*) by the number of times the evolving curve has to pass over a

Wordle on left page created with http://www.wordle.net/.

vertex of G^*. Can we compute (possibly approximately) a minimum-complexity isotopy between two given isotopic simple cycles? Related results may prove useful [246]. Alternatively, the complexity of a homotopy could be measured by the homotopic Fréchet distance (see Section 11.2), for which there exists a recent approximation algorithm on topological disks [130].

Implementation. All the polynomial-time algorithms we described for graphs on surfaces are, in principle, not hard to implement, and should run efficiently in practice, as they do not require heavy data structures, and as there is no hidden huge constant in the $O(\cdot)$ notations. While this document was slanted towards theoretical results and not towards possible applications, computer graphics (among others) use basic topological operations on surfaces; one can expect that they would benefit from the possibility of computing optimal curves or graphs satisfying some topological properties, as these and related problems are considered from a somewhat more heuristic perspective in this field [D, 127–129, 178, 247, 250]. Developing a unified implementation of these algorithms to make them directly useable would hopefully arouse interest from other communities, in which such results are probably regarded by many as too theoretical. I implemented some years ago (partly with Laurent Jouhet during his Master's internship) a prototype for computing a shortest cut graph with one specified vertex, using a generic data structure for cross-metric surfaces that would be also suitable for the other algorithms; much to my regret, the project did not go further due to lack of time and manpower, but I believe it would be useful to have an off-the-shelf implementation, probably as a CGAL package, of (a subset of) the algorithms described in Part II.

Surfaces embedded in $\mathbb{R}^3$. As a more long-term goal, I would like to develop algorithms for graphs and surfaces embedded in $\mathbb{R}^3$ (or $\mathbb{S}^3$). Probably the most central open problem in computational topology in $\mathbb{R}^3$ is the complexity of the *unknotting problem*: determine whether a closed polygonal curve in $\mathbb{R}^3$ is a trivial knot. The unknotting problem is known to be in NP [132], and it has been open for a long time whether it lies in P, although a very recent preprint claims that it is the case if the Generalized Riemann Hypothesis holds [166].

Studying the complexity of the unknotting problem directly seems hard; instead, some related problems can be expected to be more tractable algorithmically. There exist recent topological algorithms in three dimensions, and we only mention a couple of examples from computational geometry and related fields that we find inspiring: knot invariants [2, 211], linklessly embeddable graphs [155], homology generators of geometric objects in $\mathbb{R}^3$ [83], homology handle and tunnel loops [86], and reconstruction of surfaces in $\mathbb{R}^3$ from their boundaries under some assumptions [102]. In general, topological problems in three dimensions have been studied from a practical point of view, notably by the computer graphics community, but remain relatively unexplored from a more theoretical viewpoint.

For example, given a surface Σ in $\mathbb{R}^3$, one may try to compute a simple cycle γ on Σ that bounds a compressing disk (namely, γ is not contractible in Σ but bounds a disk intersecting Σ exactly at its boundary); similarly, one could aim at finding several "independent" such cycles. (As a preliminary study reveals, computing a single such cycle is feasible in polynomial time.) An aforementioned recent work [86] solves the problem where the condition of having compressing disks is replaced with arbitrary compressing surfaces.

Recent papers consider a discrete version of the Plateau problem, where the goal is to compute a minimum-area surface whose boundary is a closed polygonal curve in $\mathbb{R}^3$ [85, 94] (see also Erickson [107] for a survey of older related results). The problem, adequately discretized, can be solved in polynomial time. Assuming a "certificate" that the closed curve is unknotted, can we compute the minimum-area *disk* bounded by that curve? Can normal surfaces prove useful for studying this problem?

In general, topological questions on two-dimensional simplicial complexes are undecidable (notably from the homotopy viewpoint). Which such questions become decidable, or even solvable in polynomial time, for two-dimensional simplicial complexes embedded in $\mathbb{R}^3$?

Index

References

Articles (co-)signed by the author are listed alphabetically; other references are listed numerically.

List of Publications

The list is ordered by date of first publication. Each paper is listed once, even if it appears in multiple versions.

[A] Éric Colin de Verdière, Michel Pocchiola, and Gert Vegter. Tutte's barycenter method applied to isotopies. *Computational Geometry: Theory and Applications*, 26(1):81–97, 2003.
Conference version in *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG)*, pages 57–60, 2001.

[B] David Cohen-Steiner, Éric Colin de Verdière, and Mariette Yvinec. Conforming Delaunay triangulations in 3D. *Computational Geometry: Theory and Applications*, 28(2–3):217–233, 2004.
Conference version in *Proceedings of the 18th Annual ACM Symposium on Computational Geometry (SOCG)*, pages 199–208, 2002.

[C] Éric Colin de Verdière and Francis Lazarus. Optimal system of loops on an orientable surface. *Discrete & Computational Geometry*, 33(3):507–534, 2005. [pp. 27, 30, and 52]
Conference version in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 627–636, 2002.

[D] Pierre Alliez, Éric Colin de Verdière, Olivier Devillers, and Martin Isenburg. Centroidal Voronoi diagrams for isotropic surface remeshing. *Graphical Models*, 67(3):204–231, 2005. [p. 96]
Conference version in *Proceedings of the International Conference on Shape Modelling and Applications (SMI)*, pages 49–58, 2003.

[E] Éric Colin de Verdière and Francis Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *Journal of the ACM*, 54(4):Article 18 (27 pages), 2007. [p. 52]
Conference version in *Proceedings of the 12th International Symposium on Graph Drawing (GD)*, volume 2912 of *Lecture Notes in Computer Science*, pages 478–490, 2003.

[F] Éric Colin de Verdière. *Raccourcissement de courbes et décomposition de surfaces*. PhD thesis, Université Paris 7, 2003. [pp. 17, 27, 30, 52, and 69]

[G] Éric Colin de Verdière and Jeff Erickson. Tightening nonsimple paths and cycles on surfaces. *SIAM Journal on Computing*, 39(8):3784–3813, 2010. [pp. 27, 51, 52, and 69]
Conference version in *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 192–201, 2006.

[H] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. *Computational Geometry: Theory and Applications*, 41(1–2):94–110, 2008. [pp. 27, 51, 59, and 61]

Conference version in *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SOCG)*, pages 421–429, 2006.

[I] Éric Colin de Verdière and Alexander Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms*, 7(2), Article 19 (13 pages), 2011. [p. 85]

Conference version in *Proceedings of the 25th Annual International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 181–192, 2008.

[J] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite. Homotopic Fréchet distance between curves—or, walking your dog in the woods in polynomial time. *Computational Geometry: Theory and Applications*, 43(295–311), 2010. [pp. 85 and 88]

Conference version in *Proceedings of the 24th Annual ACM Symposium on Computational Geometry (SOCG)*, pages 101–109, 2008.

[K] Éric Colin de Verdière. *Algorithms for graphs on surfaces*. Course notes, 2008 (first version published). Current version available at http://www.di.ens.fr/~colin/cours/algo-graphs-surfaces.pdf. [p. 33]

[L] Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Algorithms for the edge-width of an embedded graph. *Computational Geometry: Theory and Applications*, 45:(215–224), 2012. [pp. 27, 33, 43, 44, and 50]

Conference version in *Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG)*, pages 147–155, 2010.

[M] Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG)*, pages 156–165, 2010. [pp. 43, 48, 49, and 50]

[N] Éric Colin de Verdière. Shortest cut graph of a surface with prescribed vertex set. In *Proceedings of the 18th European Symposium on Algorithms (ESA), part 2*, volume 6347 of *Lecture Notes in Computer Science*, pages 100–111, 2010. [pp. 27, 33, 34, and 39]

[O] Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding cycles with topological properties in embedded graphs. *SIAM Journal on Discrete Mathematics*, 25:1600–1614, 2011. [pp. 51, 59, 75, and 76]

[P] Éric Colin de Verdière, Grégory Ginot, and Xavier Goaoc. Multinerves and Helly numbers of acyclic families. In *Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG)*, 2012, to appear. [pp. 85 and 90]

ArXiv preprint 1101.6006 [math.CO], 2011.

[Q] Alexandre Boulch, Éric Colin de Verdière, and Atsuhiro Nakamoto. Irreducible triangulations of surfaces with boundary. Submitted to journal. ArXiv preprint 1103.5364 [math.CO], 2011. [pp. 79 and 80]

[R] Éric Colin de Verdière and Arnaud de Mesmay. Testing graph isotopy on surfaces. In *Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG)*, 2012, to appear. [pp. 27, 67, and 68]

References by Other Authors

[1] Pankaj K. Agarwal, Herbert Edelsbrunner, John Harer, and Yusu Wang. Extreme elevation on a 2-manifold. *Discrete & Computational Geometry*, 36(4):553–572, 2006. [p. 22]

[2] Pankaj K. Agarwal, Herbert Edelsbrunner, and Yusu Wang. Computing the writhing number of a polygonal knot. *Discrete & Computational Geometry*, 32(1):37–53, 2004. [p. 96]

[3] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The design and analysis of computer programs*. Addison-Wesley, 1974. [pp. 12 and 37]

[4] Lyudmil Aleksandrov and Hristo Djidjev. Linear algorithms for partitioning embedded graphs of bounded genus. *SIAM Journal on Discrete Mathematics*, 9(1):129–150, 1996. [p. 23]

[5] Pierre Alliez, Ucelli Giulana, and Marco Attene. Recent advances in remeshing of surfaces. In Leila De Floriani and Michela Spagnuolo, editors, *Shape analysis and structuring*. Springer-Verlag, 2007. [p. 17]

[6] Pierre Alliez and Craig Gotsman. Recent advances in compression of 3D meshes. In Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 3–26. Springer-Verlag, 2005. [p. 17]

[7] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. [p. 77]

[8] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. [pp. 87 and 88]

[9] Nina Amenta. A new proof of an interesting Helly-type theorem. *Discrete & Computational Geometry*, 15:423–427, 1996. [p. 90]

[10] James W. Anderson, Hugo Parlier, and Alexandra Pettet. Small filling sets of curves on a surface. *Topology and its Applications*, 158:84–92, 2011. [p. 18]

[11] Kenneth Appel and Wolfgang Haken. *Every planar map is four-colorable*. AMS, Providence, Rhode Island, 1989. [p. 18]

[12] Dan Archdeacon. The nonorientable genus is additive. *Journal of Graph Theory*, 10(3):363–383, 1986. [p. 18]

[13] Dan Archdeacon. Topological graph theory. A survey. *Congressus Numerantium*, 115:5–54, 1996. [p. 18]

[14] Mark Anthony Armstrong. *Basic topology*. Undergraduate Texts in Mathematics. Springer-Verlag, 1983. [pp. 7 and 19]

[15] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *Proceedings of the 14th European Symposium on Algorithms (ESA)*, pages 52–63, 2006. [p. 87]

[16] David Barnette. Generating the triangulations of the projective plane. *Journal of Combinatorial Theory, Series B*, 33:222–230, 1982. [p. 79]

[17] David W. Barnette and Allan Edelson. All orientable 2–manifolds have finitely many minimal triangulations. *Israel Journal of Mathematics*, 62:90–98, 1988. [pp. 79 and 81]

[18] David W. Barnette and Allan Edelson. All 2–manifolds have finitely many minimal triangulations. *Israel Journal of Mathematics*, 67:123–128, 1989. [p. 79]

[19] Edward A. Bender and E. Rodney Canfield. The number of rooted maps on an orientable surface. *Journal of Combinatorial Theory, Series B*, 53(2):293–299, 1991. [p. 20]

[20] Edward A. Bender, Zhicheng Gao, and L. Bruce Richmond. The map asymptotics constant t_g. *Electronic Journal of Combinatorics*, 15:Article R51, 2008. [p. 20]

[21] Itai Benjamini and László Lovász. Global information from local observation. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 701–710, 2002. [p. 23]

[22] Sylvia Benvenuti and Riccardo Piergallini. The complex of pants decompositions of a surface. *Topology and its Applications*, 156:399–419, 2008. [p. 19]

[23] Marcel Berger. What is a systole? *Notices of the AMS*, 55(3):374–376, 2003. [p. 42]

[24] Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. *Journal of Algorithms*, 49(2):284–303, 2003. [pp. 22 and 52]

[25] Sergei Bespamyatnikh. Encoding homotopy of paths in the plane. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium*, volume 2976 of *Lecture Notes in Computer Science*, pages 329–338. Springer-Verlag, 2004. [p. 22]

[26] Mladen Bestvina and Michael Handel. Train-tracks for surface homeomorphisms. *Topology*, 34(1):109–140, 1995. [p. 69]

[27] Jérémie Bettinelli. The topology of scaling limits of positive genus random quadrangulations. *Annals of Probability*, 2012. To appear. [p. 20]

[28] Daniel Bienstock and Michael A. Langston. Algorithmic implications of the graph minor theorem. In *Handbook of operations research and management science*, volume Networks and distribution. Elsevier, 2003. [p. 22]

[29] Joan S. Birman and Caroline Series. An algorithm for simple curves on surfaces. *Journal of the London Mathematical Society, Second Series*, 29:331–342, 1984. [p. 22]

[30] Anders Björner. Nerves, fibers and homotopy groups. *Journal of Combinatorial Theory, Series A*, 102(1):88–93, 2003. [p. 91]

[31] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. [p. 35]

[32] Béla Bollobás and Oliver Riordan. A polynomial of graphs on surfaces. *Mathematische Annalen*, 323:81–96, 2002. [p. 18]

[33] Paul Bonsma. Surface split decompositions and subgraph isomorphism in graphs on surfaces. In *Proceedings of the 29th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 531–542, 2012. [p. 22]

[34] Glencora Borradaile, Erik D. Demaine, and Siamak Tazari. Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. In *Proceedings of the 26th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 171–182, 2009. [p. 22]

[35] Glencora Borradaile, James R. Lee, and Anastasios Sidiropoulos. Randomly removing g handles at once. *Computational Geometry: Theory and Applications*, 43(8):655–662, 2010. [p. 23]

[36] Jérémie Bouttier, Philippe Di Francesco, and Emmanuel Guitter. Planar maps as labeled mobiles. *Electronic Journal of Combinatorics*, 11:Article 69, 2004. [p. 42]

[37] Henry R. Brahana. Systems of circuits on 2-dimensional manifolds. *Annals of Mathematics*, 23:144–168, 1921. [p. 41]

[38] Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 645–654, 2009. [p. 87]

[39] Kevin Buchin, Maike Buchin, and Carola Wenk. Computing the Fréchet distance between simple polygons in polynomial time. In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SOCG)*, pages 80–87. ACM, 2006. [p. 87]

[40] Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. arXiv:1107.3793, 2011. [p. 41]

[41] Peter Buser. *Geometry and spectra of compact Riemann surfaces*, volume 106 of *Progress in Mathematics*. Birkhäuser, 1992. [p. 71]

[42] Sergio Cabello. Finding shortest contractible and shortest separating cycles in embedded graphs. *ACM Transactions on Algorithms*, 6(2), 2010. [p. 75]

[43] Sergio Cabello and Erin W. Chambers. Multiple source shortest paths in a genus g graph. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 89–97, 2007. [pp. 48, 50, and 105]

[44] Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple source shortest paths in embedded graphs. arXiv:1202.0314. Full version of [43], 2012. [pp. 48 and 50]

[45] Sergio Cabello, Matt DeVos, Jeff Erickson, and Bojan Mohar. Finding one tight cycle. *ACM Transactions on Algorithms*, 6(4):Article 61, 2010. [pp. 27, 51, and 55]

[46] Sergio Cabello, Yuanxin Liu, Andrea Mantler, and Jack Snoeyink. Testing homotopy for paths in the plane. *Discrete & Computational Geometry*, 31:61–81, 2004. [pp. 17, 22, and 67]

[47] Sergio Cabello and Bojan Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discrete & Computational Geometry*, 37(2):213–235, 2007. [pp. 27, 45, 48, and 50]

[48] Sergio Cabello and Bojan Mohar. Adding one edge to planar graphs makes crossing number hard. In *Proceedings of the 26th Annual Symposium on Computational Geometry (SOCG)*, pages 68–76. ACM, 2010. [p. 21]

[49] J. Scott Carter. *How surfaces intersect in space: an introduction to topology*. World Scientific, 1993. [p. 19]

[50] Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. Schnyder woods for higher genus triangulated surfaces, with applications to encoding. *Discrete & Computational Geometry*, 42(3):489–516, 2009. [p. 23]

[51] Erin Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th Annual Symposium on Computational Geometry (SOCG)*, pages 377–385. ACM, 2009. [pp. 23, 27, 51, and 59]

[52] Erin W. Chambers and David Eppstein. Flows in one-crossing-minor-free graphs. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC), part 1*, number 6506 in Lecture Notes in Computer Science, pages 241–252, 2010. [p. 23]

[53] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Homology flows, cohomology cuts. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 273–282, 2009. [pp. 23 and 87]

[54] Erin W. Chambers and David Letscher. On the height of a homotopy. In *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG)*, pages 103–106, 2009. [p. 89]

[55] Guillaume Chapuy. A new combinatorial identity for unicellular maps, via a direct bijective approach. *Advances in Applied Mathematics*, 47(4):874–893, 2011. [p. 20]

[56] Guillaume Chapuy, Michel Marcus, and Gilles Schaeffer. A bijection for rooted maps on orientable surfaces. *SIAM Journal on Discrete Mathematics*, 23(3):1587–1611, 2009. [pp. 20 and 42]

[57] Bernard Chazelle. A theorem on polygon cutting with applications. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 339–349, 1982. [p. 89]

[58] Bernard Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000. [p. 38]

[59] Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Computational Geometry: Theory and Applications*, 43(2):169–181, 2010. [p. 41]

[60] Jianer Chen, Saroja P. Kanchi, and Arkady Kanevsky. A note on approximating graph genus. *Information Processing Letters*, 61:317–322, 1997. [p. 21]

[61] Jindong Chen and Yijie Han. Shortest paths on a polyhedron. *International Journal of Computational Geometry & Applications*, 6:127–144, 1996. [p. 31]

[62] Siu-Wing Cheng, Tamal K. Dey, and Sheung-Hung Poon. Hierarchy of surface models and irreducible triangulations. *Computational Geometry: Theory and Applications*, 27(2):135–150, 2004. [pp. 17 and 80]

[63] Siu-Wing Cheng, Jiongxin Jin, Antoine Vigneron, and Yajun Wang. Approximate shortest homotopic paths in weighted regions. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC)*, volume 6507 of *Lecture Notes in Computer Science*, pages 109–120, 2010. [p. 52]

[64] Otfried Cheong, Xavier Goaoc, Andreas Holmsen, and Sylvain Petitjean. Hadwiger and Helly-type theorems for disjoint unit spheres. *Discrete & Computational Geometry*, 1–3:194–212, 2008. [p. 91]

[65] David R. J. Chillingworth. Simple closed curves on surfaces. *Bulletin of the London Mathematical Society*, 1:310–314, 1969. [p. 22]

[66] David R. J. Chillingworth. An algorithm for families of disjoint simple closed curves on surfaces. *Bulletin of the London Mathematical Society*, 3:23–26, 1971. [p. 22]

[67] Jaigyoung Choe. On the existence and regularity of fundamental domains with least boundary area. *Journal of Differential Geometry*, 29(3):623–663, 1989. [p. 31]

[68] Julia Chuzhoy. An algorithm for the graph crossing number problem. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 303–312, 2011. [p. 21]

[69] Marshall Cohen and Martin Lustig. Paths of geodesics and geometric intersection numbers. I. In *Combinatorial group theory and topology (Alta, Utah, 1984)*, volume 111 of *Annals of Mathematical Studies*, pages 479–500. Princeton University Press, 1987. [p. 22]

[70] Kree Cole-McLaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Loops in Reeb graph of 2-manifolds. *Discrete & Computational Geometry*, 32(2):231–244, July 2004. [p. 22]

[71] Atlas F. Cook IV and Carola Wenk. Geodesic Fréchet distance inside a simple polygon. *ACM Transactions on Algorithms*, 7(1):Article 9, 2010. [pp. 87 and 89]

[72] Ludwig Danzer. Über ein Problem aus der kombinatorischen Geometrie. *Archiv der Mathematik*, 8:347–351, 1957. [p. 91]

[73] Samir Datta, Arjun Gopalan, Raghav Kulkarni, and Raghunath Tewari. Improved bounds for bipartite matching on surfaces. In *Proceedings of the 29th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 254–265, 2012. [p. 23]

[74] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997. [p. 11]

[75] Mark de Berg, Marc van Kreveld, and Stefan Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and GIS*, 25:243–257, 1998. [p. 17]

[76] Maurits de Graaf and Alexander Schrijver. Making curves minimally crossing by Reidemeister moves. *Journal of Combinatorial Theory, Series B*, 70(1):134–156, 1997. [pp. 70 and 71]

[77] Max Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72:413–421, 1912. [pp. 21, 22, and 57]

[78] Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005. [p. 23]

[79] Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292–302, 2008. [p. 23]

[80] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Bojan Mohar. Approximation algorithms via contraction decomposition. *Combinatorica*, pages 533–552, 2010. [p. 22]

[81] Matt DeVos, Ken-ichi Kawarabayashi, and Bojan Mohar. Locally planar graphs are 5-choosable. *Journal of Combinatorial Theory, Series B*, 98(6):1215–1232, 2008. [p. 45]

[82] Tamal K. Dey, Herbert Edelsbrunner, and Sumanta Guha. Computational topology. In Bernard Chazelle, Jacob E. Goodman, and Richard Pollack, editors, *Advances in Discrete and Computational Geometry – Proc. 1996 AMS-IMS-SIAM Joint Summer Research Conf. Discrete and Computational Geometry: Ten Years Later*, number 223 in Contemporary Mathematics, pages 109–143. AMS, 1999. [pp. 1 and 21]

[83] Tamal K. Dey and Sumanta Guha. Computing homology groups of simplicial complexes in R^3. *Journal of the ACM*, 45(2):266–287, 1998. [p. 96]

[84] Tamal K. Dey and Sumanta Guha. Transforming curves on surfaces. *Journal of Computer and System Sciences*, 58:297–325, 1999. [p. 21]

[85] Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 221–230, 2010. [p. 97]

[86] Tamal K. Dey, Kuiyu Li, Jian Sun, and David Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3D models. *ACM Transactions on Graphics*, 27(3), 2008. [pp. 96 and 97]

[87] Tamal K. Dey and Haijo Schipper. A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. *Discrete & Computational Geometry*, 14(1):93–110, 1995. [p. 21]

[88] Tamal K. Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In *Proceedings of the 26th Annual Symposium on Computational Geometry (SOCG)*, pages 166–175. ACM, 2010. [p. 41]

[89] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph drawing*. Prentice Hall, Upper Saddle River, NJ, 1999. [p. 1]

[90] Reinhard Diestel. *Graph theory*. Springer-Verlag, 2000. Available at http://diestel-graph-theory.com/. [p. 7]

[91] Hristo N. Djidjev. A faster algorithm for computing the girth of planar and bounded genus graphs. *ACM Transactions on Algorithms*, 7(1):Article 3, 2010. [p. 23]

[92] Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Fast subexponential algorithm for non-local problems on graphs of bounded genus. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 4059 of *Lecture Notes in Computer Science*, pages 172–183. Springer-Verlag, 2006. [p. 23]

[93] Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. In *Proceedings of the 26th Annual Symposium on Computational Geometry (SOCG)*, pages 365–374. ACM, 2010. [p. 87]

[94] Nathan M. Dunfield and Anil N. Hirani. The least spanning area of a knot and the optimal bounding chain problem. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SOCG)*. ACM, 2011. [p. 97]

[95] Jürgen Eckhoff and Klaus-Peter Nischke. Morris's pigeonhole principle and the Helly theorem for unions of convex sets. *Bulletin of the London Mathematical Society*, 41:577–588, 2009. [p. 91]

[96] Herbert Edelsbrunner, John Harer, and Afra Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete & Computational Geometry*, 30:87–107, 2003. [p. 22]

[97] Alon Efrat, Leonidas J. Guibas, Sariel Har-Peled, Joseph S. B. Mitchell, and T. M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete & Computational Geometry*, 28:535–569, 2002. [p. 88]

[98] Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry: Theory and Applications*, 35:162–172, 2006. [p. 52]

[99] Joanna A. Ellis-Monaghan and Irasema Sarmiento. A recipe theorem for the topological Tutte polynomial of Bollobas and Riordan. *European Journal of Combinatorics*, 32(6):782–794, 2011. [p. 18]

[100] David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 599–608, 2003. [pp. 11, 23, 40, 42, and 49]

[101] David Eppstein. Squarepants in a tree: sum of subtree clustering and hyperbolic pants decomposition. *ACM Transactions on Algorithms*, 5(3), 2009. [p. 41]

[102] David Eppstein and Elena Mumford. Self-overlapping curves revisited. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 160–169, 2009. [p. 96]

[103] David B. A. Epstein. Curves on 2-manifolds and isotopies. *Acta Mathematica*, 115:83–107, 1966. [p. 68]

[104] Jeff Erickson. Computational topology, 2009. Course notes available at http://compgeom.cs.uiuc.edu/~jeffe/teaching/comptop/. [p. 21]

[105] Jeff Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 794–804, 2010. [p. 23]

[106] Jeff Erickson. Shortest non-trivial cycles in directed surface graphs. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SOCG)*, pages 236–243. ACM, 2011. [p. 50]

[107] Jeff Erickson. Combinatorial optimization of cycles and bases. In Afra Zomorodian, editor, *Computational topology*, Proceedings of Symposia in Applied Mathematics. AMS, 2012. [pp. 31, 34, 39, and 97]

[108] Jeff Erickson, Kyle Fox, and Amir Nayyeri. Global minimum cuts in surface embedded graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1309–1318, 2012. [pp. 23 and 27]

[109] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004. [pp. 33, 34, 41, 44, 45, 46, and 50]

[110] Jeff Erickson and Amir Nayyeri. Computing replacement paths in surface-embedded graphs. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1347–1354, 2011. [p. 23]

[111] Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1166–1176, 2011. [pp. 27, 48, and 50]

[112] Jeff Erickson and Amir Nayyeri. Shortest non-crossing walks in the plane. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 297–308, 2011. [pp. 23, 51, and 59]

[113] Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. In *Proceedings of the 28th Annual Symposium on Computational Geometry (SOCG)*. ACM, 2012. To appear. [p. 22]

[114] Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1038–1046, 2005. [pp. 34, 41, 42, and 95]

[115] Jeff Erickson and Pratik Worah. Computing the shortest essential cycle. *Discrete & Computational Geometry*, 44(4):912–930, 2010. [pp. 27 and 51]

[116] Benson Farb and Dan Margalit. *A primer on mapping class groups*. Princeton University Press, 2011. [pp. 19, 20, 55, 69, and 73]

[117] Albert Fathi, François Laudenbach, and Valentin Poénaru, editors. *Travaux de Thurston sur les surfaces*. Société Mathématique de France, 1991. Séminaire Orsay, Reprint of the 1979 edition, Astérisque No. 66-67. [pp. 20 and 53]

[118] Anna Felikson and Sergey Natanzon. Double pants decompositions of 2-surfaces. *Moscow Mathematical Journal*, 11(2):231–258, 2011. [p. 20]

[119] J. R. Fiedler, J. P. Huneke, R. B. Richter, and N. Robertson. Computing the orientable genus of projective graphs. *Journal of Graph Theory*, 20(3):297–308, 1995. [p. 45]

[120] Fedor V. Fomin and Dimitrios M. Thilikos. Fast parameterized algorithms for graphs on surfaces: linear kernel and exponential speed-up. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 581–592, 2004. [p. 23]

[121] Kyle Fox. Faster shortest non-contractible cycles in directed surface graphs. arXiv:1111.6990, 2011. [pp. 27, 48, and 50]

[122] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. [p. 55]

[123] Merrick L. Furst, Jonathan L. Gross, and Lyle A. McGeoch. Finding a maximum-genus graph imbedding. *Journal of the ACM*, 35(3):523–534, 1988. [p. 21]

[124] John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984. [pp. 23, 40, and 49]

[125] Craig Gotsman and Vitaly Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25(1):67–75, 2001. [p. 67]

[126] Jonathan L. Gross and Thomas W. Tucker. *Topological graph theory*. Wiley, 1987. [pp. 11 and 18]

[127] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 355–361, 2002. [p. 96]

[128] Xianfeng Gu and Shing-Tung Yau. Global conformal surface parameterization. In *Proceedings of the Eurographics/ACM Symposium on Geometry Processing*, pages 127–137, 2003. [pp. 17 and 96]

[129] Igor Guskov and Zoë J. Wood. Topological noise removal. In *Proceedings of Graphics Interface*, pages 19–26, 2001. [pp. 17 and 96]

[130] Sariel Har-Peled, Amir Nayyeri, Mohammad Salavatipour, and Anastasios Sidiropoulos. How to walk your dog in the mountains with no magic leash. In *Proceedings of the 28th Annual Symposium on Computational Geometry (SOCG)*. ACM, 2012. To appear. [pp. 89 and 96]

[131] Sariel Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SOCG)*, pages 448–457. ACM, 2011. [p. 87]

[132] Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM*, 46(2):185–211, 1999. [pp. 19 and 96]

[133] Joel Hass and Peter Scott. Intersections of curves on surfaces. *Israel Journal of Mathematics*, 51(1–2):90–120, 1985. [p. 55]

[134] Allen Hatcher. Pants decompositions of surfaces. Manuscript available at http://www.math.cornell.edu/~hatcher/Papers/pantsdecomp.pdf, 2000. [p. 19]

[135] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002. Available at http://www.math.cornell.edu/~hatcher/. [p. 90]

[136] Allen Hatcher and William Thurston. A presentation for the mapping class group of a closed orientable surface. *Topology*, 19(3):221–237, 1980. [p. 41]

[137] Eduard Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923. [p. 89]

[138] Eduard Helly. Über Systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monatshefte für Mathematik und Physik*, 37:175–176, 1930. [p. 89]

[139] Michael Henle. *A combinatorial introduction to topology*. Dover Publications, 1994. [pp. 7 and 19]

[140] Monika R. Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1, part 1):3–23, 1997. [pp. 23, 40, 57, and 63]

[141] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4:63–98, 1994. [pp. 52, 63, and 89]

[142] Peter Hliněný and Marcus Chimani. Approximating the crossing number of graphs embeddable in any orientable surface. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 918–927, 2010. [p. 21]

[143] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974. [p. 21]

[144] Joan P. Hutchinson. On short noncontractible cycles in embedded graphs. *SIAM Journal on Discrete Mathematics*, 1(2):185–192, 1988. [p. 45]

[145] Piotr Indyk and Anastasios Sidiropoulos. Probabilistic embeddings of bounded genus graphs into planar graphs. In *Proceedings of the 23rd Annual Symposium on Computational Geometry (SOCG)*, pages 204–209. ACM, 2007. [p. 23]

[146] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982. [p. 59]

[147] Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for Min Cut and Max Flow in undirected planar graphs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 313–322, 2011. [pp. 41, 50, 52, 58, and 61]

[148] Gwenaël Joret and David R. Wood. Irreducible triangulations are small. *Journal of Combinatorial Theory, Series B*, 100:446–455, 2010. [pp. 79 and 80]

[149] Martin Juvan, Aleksander Malnič, and Bojan Mohar. Systems of curves on surfaces. *Journal of Combinatorial Theory, Series B*, 68:7–22, 1996. [pp. 18 and 79]

[150] Gil Kalai and Roy Meshulam. Intersections of Leray complexes and regularity of monomial ideals. *Journal of Combinatorial Theory, Series A*, 113:1586–1592, 2006. [p. 90]

[151] Gil Kalai and Roy Meshulam. Leray numbers of projections and a topological Helly-type theorem. *Journal of Topology*, 1(3):551–556, 2008. [p. 91]

[152] Marcin Kamiński and Dimitrios M. Thilikos. Contraction checking in graphs on surfaces. In *Proceedings of the 29th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 182–193, 2012. [p. 22]

[153] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, 1995. [p. 38]

[154] Ken-ichi Kawarabayashi, Philip N. Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP), part 1*, pages 135–146, 2011. [p. 23]

[155] Ken-ichi Kawarabayashi, Stephan Kreutzer, and Bojan Mohar. Linkless and flat embeddings in 3-space and the unknot problem. In *Proceedings of the 26th Annual Symposium on Computational Geometry (SOCG)*, pages 99–106. ACM, 2010. [p. 96]

[156] Ken-ichi Kawarabayashi and Bojan Mohar. Some recent progress and applications in graph minor theory. *Graphs and combinatorics*, 23:1–46, 2007. [p. 22]

[157] Ken-ichi Kawarabayashi and Bojan Mohar. Graph and map isomorphism and all polyhedral embeddings in linear time. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 471–480, 2008. [pp. 22, 44, and 45]

[158] Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 771–780, 2008. [pp. 21 and 95]

[159] Ken-ichi Kawarabayashi and Bruce Reed. Computing crossing number in linear time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 382–390, 2007. [pp. 21 and 45]

[160] Ken-ichi Kawarabayashi and Bruce Reed. A separator theorem in minor-closed classes. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 153–162, 2010. [p. 23]

[161] Béla Kerékjártó. *Vorlesung über Topologie*. Springer-Verlag, 1923. [p. 8]

[162] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry: Theory and Applications*, 13:65–90, 1999. [p. 11]

[163] Yusuke Kobayashi and Ken-ichi Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1146–1155, 2009. [pp. 22, 75, and 78]

[164] Yusuke Kobayashi and Christian Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7(4):234–245, 2010. [p. 86]

[165] M. R. Kramer and J. van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. In Franco P. Preparata, editor, *VLSI-Theory*, volume 2 of *Advances in Computing Research*, pages 129–146. JAI Press, Greenwich, Connecticut, 1984. [p. 85]

[166] Greg Kuperberg. Knottedness is in NP, modulo GRH. arXiv:1112.0845, 2011. [p. 96]

[167] Martin Kutz. Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time. In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SOCG)*, pages 430–438. ACM, 2006. [pp. 27, 48, 50, 61, and 63]

[168] Yves Ladegaillerie. Classes d'isotopie de plongements de 1-complexes dans les surfaces. *Topology*, 23(3):303–311, 1984. [pp. 69 and 71]

[169] Sergei K. Lando and Alexander K. Zvonkin. *Graphs on surfaces and their applications*. Springer-Verlag, 2004. [pp. 11 and 21]

[170] Serge Lawrencenko. The irreductible triangulations of the torus. *Ukrainskiĭ Geometricheskiĭ Sbornik*, 30:52–62, 1987. [p. 79]

[171] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14:373–389, 1998. [p. 17]

[172] Francis Lazarus, Michel Pocchiola, Gert Vegter, and Anne Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. In *Proceedings of the 17th Annual Symposium on Computational Geometry (SOCG)*, pages 80–89. ACM, 2001. [p. 41]

[173] Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. arXiv:1110.4573, 2011. [pp. 21, 68, 73, and 77]

[174] D. T. Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. [p. 89]

[175] James R. Lee and Anastasios Sidiropoulos. Genus and the geometry of the cut graph. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 193–201, 2010. [p. 23]

[176] Charles E. Leiserson and F. Miller Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC)*, pages 69–78, 1985. [p. 17]

[177] Bruno Lévy and Jean-Laurent Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *Proceedings of the 25th Annual Conference on Computer Graphics (SIGGRAPH)*, pages 343–352, 1998. [p. 17]

[178] Xin Li, Xianfeng Gu, and Hong Qin. Surface mapping using consistent pants decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 15:558–571, 2009. [pp. 17 and 96]

[179] Sóstenes Lins. Graph-encoded maps. *Journal of Combinatorial Theory, Series B*, 32:171–181, 1982. [p. 11]

[180] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. [p. 23]

[181] Martin Lustig. Paths of geodesics and geometric intersection numbers. II. In *Combinatorial group theory and topology (Alta, Utah, 1984)*, volume 111 of *Annals of Mathematical Studies*, pages 501–543. Princeton University Press, 1987. [p. 22]

[182] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*, volume 89 of *A Series of Modern Surveys in Mathematics*. Springer-Verlag, 1977. [p. 22]

[183] Justin Malestein, Igor Rivin, and Louis Theran. Topological designs. arXiv:1008.3710, 2010. [p. 18]

[184] Jiří Matoušek. A Helly-type theorem for unions of convex sets. *Discrete & Computational Geometry*, 18:1–12, 1997. [p. 90]

[185] Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30:852–866, 1983. [p. 89]

[186] Grégory Miermont. Tessellations of random maps of arbitrary genus. *Annales Scientifiques de l'École normale supérieure, Quatrième série*, 42:725–781, 2009. [pp. 20 and 42]

[187] Gary L. Miller. An additivity theorem for the genus of a graph. *Journal of Combinatorial Theory, Series B*, 43(1):25–47, 1987. [pp. 18, 79, and 80]

[188] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. [p. 31]

[189] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999. [pp. 21 and 95]

[190] Bojan Mohar and Neil Robertson. Flexibility of polyhedral embeddings of graphs in surfaces. *Journal of Combinatorial Theory, Series B*, 83(1):38–57, 2001. [p. 45]

[191] Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001. [pp. 11, 18, 42, 44, 50, 76, and 80]

[192] Lee Mosher. What is a train track? *Notices of the AMS*, 50(3):354–356, 2003. [p. 19]

[193] Atsuhiro Nakamoto and Katsuhiro Ota. Note on irreducible triangulations of surfaces. *Journal of Graph Theory*, 20(2):227–233, 1995. [pp. 79 and 80]

[194] Seiya Negami. Diagonal flips of triangulations on surfaces, a survey. *Yokohama Mathematical Journal*, 47:1–40, 1999. [p. 80]

[195] Seyia Negami. Diagonal flips in triangulations of surfaces. *Discrete Mathematics*, 135(1–3):225–232, 1994. [p. 80]

[196] Shayan Oveis Gharan and Amin Saberi. The asymmetric traveling salesman problem on graphs with bounded genus. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 967–975, 2011. [p. 23]

[197] János Pach. *Towards a theory of geometric graphs*. Number 342 in Contemporary Mathematics. AMS, 2004. [p. 1]

[198] Viresh Patel. Determining edge expansion and other connectivity measures of graphs of bounded genus. In *Proceedings of the 18th European Symposium on Algorithms (ESA)*, number 6346 in Lecture Notes in Computer Science, pages 561–572, 2010. [p. 23]

[199] Stephen Patrias. Simple closed curves on surfaces with intersection number at most one. Manuscript available at http://www.math.uchicago.edu/~may/VIGRE/VIGRE2009/REUPapers/Patrias.pdf, 2009. [p. 18]

[200] Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing even crossings on surfaces. *Electronic Notes in Discrete Mathematics*, 29:85–90, 2007. [p. 21]

[201] Robert C. Penner. *Combinatorics of train tracks*. Princeton University Press, 1992. [p. 19]

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

[203] Grisha Perelman. Finite extinction time for the solutions to the Ricci flow on certain three-manifolds. arXiv:math/0307245, 2003. [p. 20]

[204] Grisha Perelman. Ricci flow with surgery on three-manifolds. arXiv:math/0303109, 2003. [p. 20]

[205] Dan Piponi and George Borshukov. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. In *Proceedings of the 27th Annual Conference on Computer Graphics (SIGGRAPH)*, pages 471–478, 2000. [p. 17]

[206] Tibor Rado. Über den Begriff der Riemannschen Fläche. *Acta scientiarum mathematicarum (Szeged)*, 2:101–121, 1924. [p. 8]

[207] Prabhakar Raghavan. *Randomized rounding and discrete ham-sandwich theorems: provably good algorithms for routing and packing problems*. PhD thesis, University of California, Berkeley, California, 1986. Report UCB/CSD 87/312. [p. 85]

[208] John H. Reif. Minimum $s - t$ cut of a planar undirected network in $O(n \log^2(n))$ time. *SIAM Journal on Computing*, 12(1):71–81, 1983. [p. 55]

[209] Bruce L. Reinhart. Algorithms for Jordan curves on compact surfaces. *Annals of Mathematics*, 75(2):209–222, 1962. [p. 22]

[210] Gerhard Ringel. *Map color theorem*. Springer-Verlag, 1974. [p. 18]

[211] Neil Robertson, Paul Seymour, and Robin Thomas. Sachs' linkless embedding conjecture. *Journal of Combinatorial Theory, Series B*, 64:185–227, 1995. [p. 96]

[212] Neil Robertson and Paul D. Seymour. Graph minors. VII. Disjoint paths on a surface. *Journal of Combinatorial Theory, Series B*, 45:212–254, 1988. [p. 45]

[213] Neil Robertson and Paul D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003. [p. 19]

[214] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004. [p. 18]

[215] Juanjo Rué, Ignasi Sau, and Dimitrios M. Thilikos. Dynamic programming for minor-free graphs. In preparation, cited in [217]. [p. 23]

[216] Juanjo Rué, Ignasi Sau, and Dimitrios M. Thilikos. Dynamic programming for graphs on surfaces. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6198 of *Lecture Notes in Computer Science*, pages 372–383, 2010. [p. 22]

[217] Juanjo Rué, Ignasi Sau, and Dimitrios M. Thilikos. Dynamic programming for graphs on surfaces. arXiv:1104.2486, 2011. [p. 116]

[218] Luis Santaló. Un theorema sobre conjuntos de paralelepipedos de aristas paralelas. *Publ. Inst. Mat. Univ. Nat. Litoral*, 2:49–60, 1940. [p. 91]

[219] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Computing Dehn twists and geometric intersection numbers in polynomial time. In *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG)*, pages 111–114, 2008. [p. 22]

[220] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Algorithms for normal curves and surfaces. In *Proceedings of the 8th International Conference on Computing and Combinatorics (COCOON)*, pages 370–380, 2002. [p. 22]

[221] Gilles Schaeffer. *Conjugaison d'arbres et cartes combinatoires aléatoire*. PhD thesis, Université Bordeaux I, 1998. [p. 42]

[222] Haijo Schipper. Determining contractibility of curves. In *Proceedings of the 8th Annual Symposium on Computational Geometry (SOCG)*, pages 358–367. ACM, 1992. [p. 21]

[223] Haijo Schipper. The word problem: a geometric approach. In *Proceedings of the 4th Canadian Conference on Computational Geometry (CCCG)*, pages 59–65, 1992. [p. 21]

[224] Saul Schleimer. Notes on the complex of curves. Expository notes available on the author's webpage, 2006. [p. 19]

[225] Alexander Schrijver. Disjoint circuits of prescribed homotopies in a graph on a compact surface. *Journal of Combinatorial Theory, Series B*, 51(1):127–159, 1991. [p. 23]

[226] Alexander Schrijver. Free partially commutative groups, cohomology, and paths and circuits in directed graphs on surfaces. Preprint available on the author's webpage, 2008. [pp. 23 and 30]

[227] Anastasios Sidiropoulos. Optimal stochastic planarization. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 163–170, 2010. [p. 23]

[228] Ernst Steinitz and Hans Rademacher. *Vorlesungen über die Theorie der Polyeder*. Springer-Verlag, 1934. [p. 79]

[229] John Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, second edition, 1993. [pp. 7, 19, and 53]

[230] Thom Sulanke. Generating irreducible triangulations of surfaces. arXiv:math/0606687, 2006. [pp. 79 and 81]

[231] Thom Sulanke. Irreducible triangulations of low genus surfaces. arXiv:math/0606690, 2006. [pp. 79 and 80]

[232] Thom Sulanke. Note on the irreducible triangulations of the Klein bottle. *Journal of Combinatorial Theory, Series B*, 96:964–972, 2006. [p. 79]

[233] Jun-ya Takahashi, Hitoshi Suzuki, and Takao Nishizeki. Shortest noncrossing paths in plane graphs. *Algorithmica*, 16:339–357, 1996. [p. 63]

[234] Itaru Takarajima. A combinatorial representation of curves using train tracks. *Topology and its Applications*, 106:169–198, 2000. [p. 69]

[235] Ser Peow Tan. Self-intersections of curves on surfaces. *Geometriae Dedicata*, 62(2):209–225, 1996. [p. 22]

[236] Siamak Tazari and Matthias Müller-Hannemann. Shortest paths in linear time on minor-closed graph classes, with an application to Steiner tree approximation. *Discrete Applied Mathematics*, 157(4):673–684, 2009. [p. 23]

[237] Carsten Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989. [p. 21]

[238] Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *Journal of Combinatorial Theory, Series B*, 48(2):155–177, 1990. [pp. 42 and 50]

[239] Carsten Thomassen. Five-coloring maps on surfaces. *Journal of Combinatorial Theory, Series B*, 59(1):89–105, 1993. [pp. 18 and 45]

[240] Dylan P. Thurston. On geometric intersection of curves on surfaces. Available at http://www.math.columbia.edu/~dpt/DehnCoordinates.ps, 2008. [p. 19]

[241] William T. Tutte. A census of planar maps. *Canadian Journal of Mathematics*, 15:249–271, 1963. [p. 20]

[242] Helge Tverberg. Proof of Grünbaum's conjecture on common transversals for translates. *Discrete & Computational Geometry*, 4:191–203, 1989. [p. 91]

[243] René van Oostrum and Remco C. Veltkamp. Parametric search made practical. *Computational Geometry: Theory and Applications*, 28:75–88, 2004. [p. 89]

[244] Gert Vegter. Computational topology. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 32, pages 517–536. CRC Press LLC, Boca Raton, FL, second edition, 2004. [pp. 1 and 21]

[245] Gert Vegter and Chee K. Yap. Computational complexity of combinatorial surfaces. In *Proceedings of the 6th Annual Symposium on Computational Geometry (SOCG)*, pages 102–111. ACM, 1990. [p. 41]

[246] Yusu Wang. Measuring similarity between curves on 2-manifolds via minimum deformation area. Manuscript, 2008. [p. 96]

[247] Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, 2004. [pp. 17 and 96]

[248] Xiaotian Yin, Miao Jin, and Xianfeng Gu. Computing shortest cycles using universal covering space. *The visual computer*, 23:999–1004, 2007. [p. 52]

[249] Xingxing Yu. Disjoint paths, planarizing cycles, and spanning walks. *Transactions of the American Mathematical Society*, 349:1333–1358, 1997. [p. 45]

[250] Wei Zeng, Miao Jin, Feng Luo, and Xianfeng David Gu. Canonical homotopy class representative using hyperbolic structure. In *Proceedings of the International Conference on Shape Modelling and Applications (SMI)*, pages 171–178, 2009. [p. 96]

[251] Afra Zomorodian. Computational topology. In Mikhail J. Atallah, editor, *Algorithms and Theory of Computation Handbook*. Chapman & Hall, 2009. [pp. 1 and 21]

[252] Afra Zomorodian. Topological data analysis. In Afra Zomorodian, editor, *Computational topology*, Proceedings of Symposia in Applied Mathematics. AMS, 2012. [p. 17]

[253] Alexander Zvonkin. Matrix integrals and map enumeration: an accessible introduction. *Computers and Mathematics with Applications: Mathematical and Computer Modelling*, 26(8–10):281–304, 1997. [p. 21]

Abstract

The common idea underlying most of the works presented in this habilitation thesis is the study of algorithms for topological problems regarding curves and graphs on surfaces. These results belong to the field of computational topology, with tight connections to topological graph theory and graph algorithms.

A motivation for these works is the recent development of graph algorithms that are efficient in the case where the input graph is drawn without crossings on a fixed surface. Very often, these algorithms need to make the graph planar by cutting or removing vertices and edges; our results are relevant for this purpose. Moreover, the data of a graph with a drawing of it on a surface gives rise to new natural problems, such as the computation of shortest paths in the graph among those that are homotopic (deform continuously on the surface) to a given path. Finally, forgetting about the graph, our algorithms can also be viewed as a way of topologically simplifying or decomposing a surface, a problem that arises in particular in computer graphics.

After a survey on graphs on surfaces in various areas, we revisit in a unified way our results for computing shortest curves or graphs with prescribed topological properties on surfaces. We move on with other algorithmic and combinatorial results on graphs on surfaces that are more related to topological graph theory. Finally, we present more succinctly our other recent works on algorithms in the plane or on results in combinatorial geometry, which also use topological tools at various levels of sophistication.

Résumé

La démarche commune à la plupart des travaux présentés dans ce mémoire d'habilitation est l'étude d'algorithmes pour des problèmes topologiques sur les courbes et graphes tracés sur les surfaces. Ces résultats s'inscrivent dans le domaine de la topologie algorithmique, avec des liens forts en théorie topologique des graphes et algorithmique des graphes.

Une motivation pour ces travaux est le foisonnement récent d'algorithmes de graphes efficaces dans le cas où le graphe donné en entrée est dessiné sans croisement sur une surface fixée. Ces algorithmes ont très souvent besoin de transformer le graphe en un graphe planaire par découpage ou suppression de sommets et d'arêtes ; nos résultats s'inscrivent dans cette problématique. De plus, la donnée d'un graphe avec un dessin de celui-ci sur une surface donne naissance à des problèmes naturels comme le calcul de plus courts chemins dans ce graphe parmi ceux qui s'obtiennent à partir d'un chemin donné par déformation (homotopie) sur la surface. On peut aussi, en faisant abstraction du graphe, voir les algorithmes que nous proposons comme permettant de simplifier ou de décomposer topologiquement une surface, un problème qui se pose notamment en infographie.

Après un panorama sur les graphes sur les surfaces dans diverses disciplines, nous revisitons, de façon unifiée, nos résultats sur le calcul de plus courtes courbes ou graphes possédant des propriétés topologiques données sur les surfaces. Nous poursuivons avec d'autres résultats algorithmiques et combinatoires sur les graphes sur les surfaces, plus reliés à la théorie topologique des graphes. Enfin, nous présentons de façon plus succincte nos autres travaux récents d'algorithmique dans le plan ou de géométrie combinatoire, qui utilisent également des outils topologiques plus ou moins sophistiqués.