Graphs on Surfaces: Topological Algorithms

Éric Colin de Verdière

École de Printemps d'Informatique Théorique 2016

Foreword and introduction

Introduction

This is an introduction to some tools from computational topology of graphs on surfaces.

Chapters 1 and 2 serve as topological preliminaries for this spring school. The first chapter introduces planar graphs from the topological and combinatorial point of view. Then, we consider graphs on surfaces (planar graphs being an important special case). In Chapter 2, we introduce surfaces from the topological point of view.

Chapter 3 and 4 present basic concepts and algorithms in the field of topological algorithms for graphs on surfaces. In Chapter 3, we present algorithms using the cut locus to build short curves and decompositions of surfaces. In Chapter 4, we introduce two important topological concepts, homotopy and the universal cover, and describe an algorithm to decide whether a curve can be continuously deformed to a point on a surface.

The course will continue with applications of the aforementioned algorithms, presented using slides and thus not decribed in these notes:

- an algorithm to compute shortest curves up to homotopy [10];
- an efficient minimum cut algorithm for graphs on a fixed surface [6];
- depending on the time remaining, an algorithm to solve the minimum multicut problem for graphs on a fixed surface with a bounded number of terminals [9].

This document is mostly taken from more complete course notes available at `www.di.ens.fr/~colin/cours/all-algo-embedded-graphs.pdf`.

Date of this version: May 4, 2016.

Each exercise is labeled with one to three stars, supposed to be an indication of its *importance* (in particular, depending on whether the result or technique is used later), not of its difficulty.

Only a part of the material covered in this course appeared in textbooks. For further reading or different expositions, mostly on the topological aspects, recommended books are Mohar and Thomassen [32], Armstrong [1], and Stillwell [37]. For more examples on recent algorithms for topological problems on graphs on surfaces, see [8]. For a wider perspective in computational topology, see the recent course notes by Erickson [17].

Acknowledgments

I would like to thank several people who suggested some corrections in previous versions: Jeff Erickson, Francis Lazarus, Arthur Milchior, and Vincent Pilaud.

Chapter 1

Basic properties of planar graphs

1.1 Topology

1.1.1 Preliminaries on topology

We assume some familiarity with basic topology, but we recall some definitions nonetheless.

A ***topological space*** is a set X with a collection of subsets of X, called ***open sets***, satisfying the three following axioms:

- the empty set and X are open;
- any union of open sets is open;
- any finite intersection of open sets is open.

There is, in particular, no notion of metric (or distance, angle, area) in a topological space. The open sets give merely an information of ***neighborhood***: a neighborhood of $x \in X$ is a set containing an open set containing x. This is already a lot of information, allowing to define continuity, homeomorphisms, connectivity, boundary, isolated points, dimension.... Specifically, a map $f : X \to Y$ is ***continuous*** if the inverse image of any open set by f is an open set. If X and Y are two topological spaces, a map $f : X \to Y$ is a ***homeomorphism*** if it is continuous, bijective, and if its inverse f^{-1} is also continuous. A point of detail, ruling out pathological spaces: the topological spaces considered in these notes are assumed to be *Hausdorff*, which means that two distinct points have disjoint neighborhoods.

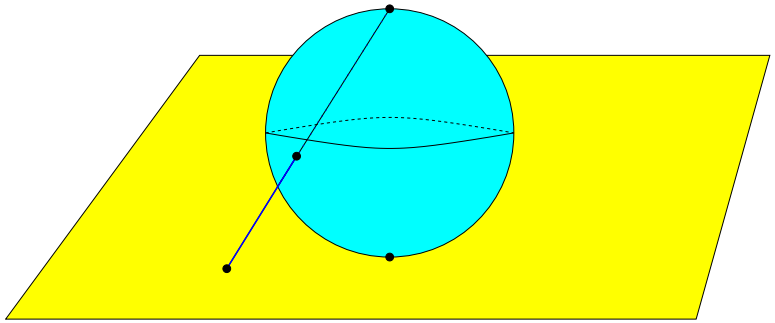


Figure 1.1. The stereographic projection.

Example 1.1. Most of the topological spaces here are endowed with a natural metric, which should be "forgotten", but define the topology:

- $\mathbb{R}^n$ $(n \geq 1)$;
- the n-dimensional sphere $\mathbb{S}^n$, i.e., the set of unit vectors of $\mathbb{R}^{n+1}$;
- the n-dimensional ball B^n, i.e., the set of vectors in $\mathbb{R}^n$ of norm at most 1; in particular $B^1 = [-1, 1]$ and $[0, 1]$ are homeomorphic;
- the set of lines in $\mathbb{R}^2$, or more generally the *Grassmannian*, the set of k-dimensional vector spaces in $\mathbb{R}^n$.

Exercise 1.2 (stereographic projection)**.** ☆☆ Prove that the plane is homeomorphic to $\mathbb{S}^2$ with an arbitrary point removed. (Indication: see Figure 1.1.)

A ***closed set*** in X is the complement of an open set. The ***closure*** of a subset of X is the (unique) smallest closed set containing it. The ***interior*** of a subset of X is the (unique) largest open set contained in it. The ***boundary*** of a subset of X equals its closure minus its interior. A topological space X is ***compact*** if any set of open sets whose union is X admits a finite subset whose union is still X.

A ***path*** in X is a continuous map $p : [0,1] \to X$; its ***endpoints*** are $p(0)$ and $p(1)$. Its ***relative interior*** is the image by p of the open interval $(0,1)$. A path is ***simple*** if it is one-to-one. A space X is ***connected***[1] if it is non-empty and, for any points a and b in X, there exists a path in X whose endpoints are a and b. The ***connected components*** of a topological space X are the classes of the equivalence relation on X defined by: a is equivalent to b if there exists a path between a and b. A topological space X is ***disconnected*** (or ***separated***) by $Y \subseteq X$ if and only if $X \setminus Y$ is not connected; points in different connected components of $X \setminus Y$ are ***separated*** by Y.

1.1.2 Graphs and embeddings

We will use standard terminology for graphs. Unless noted otherwise, all graphs are undirected and finite but may have loops and multiple edges. A ***circuit*** in a graph G is a closed walk without repeated vertices.[2]

A graph yields naturally a topological space:

- for each edge e, let X_e be a topological space homeomorphic to $[0,1]$; let X be the disjoint union of the X_e;
- for e, e', identify (quotient topology), in X, endpoints of X_e and $X_{e'}$ if these endpoints correspond to the same vertex in G.

An ***embedding*** of G in the plane $\mathbb{R}^2$ is a continuous one-to-one map from G (viewed as a topological space) to $\mathbb{R}^2$. Said differently, it is a "crossing-free drawing" of G on $\mathbb{R}^2$, being the data of two maps:

- Γ_V, which associates to each vertex of G a point of $\mathbb{R}^2$;
- Γ_E, which associates to each edge e of G a path in $\mathbb{R}^2$ between the images by Γ_V of the endpoints of e,

in such a way that:

- the map Γ_V is one-to-one (two distinct vertices are sent to distinct points of $\mathbb{R}^2$);
- for each edge e, the relative interior of $\Gamma_E(e)$ is one-to-one (the image of an edge is a simple path, except possibly at its endpoints);
- for all distinct edges e and e', the relative interiors of $\Gamma_E(e)$ and $\Gamma_E(e')$ are disjoint (two edges cannot cross);
- for each edge e and for each vertex v, the relative interior of $\Gamma_E(e)$ does not meet $\Gamma_V(v)$ (no edge passes through a vertex).

We can actually replace $\mathbb{R}^2$ above with *any* topological space Y and talk about an embedding of a graph in Y.

When we speak of embedded graphs, we sometimes implicitly identify the graph, its embedding, and the image of its embedding.

1.1.3 Planar graphs and the Jordan curve theorem

In the remaining part of this chapter, we only consider embeddings of graphs into the sphere $\mathbb{S}^2$ or the plane $\mathbb{R}^2$.

A graph is ***planar*** if it admits an embedding into the plane. By Exercise 1.2, this is equivalent to the existence of an embedding into the sphere $\mathbb{S}^2$.

The ***faces*** of a graph embedding are the connected components of the complement of the image of the vertices and edges of the graph.

Here are the most-often used results in the area.

Theorem 1.3 (Jordan curve theorem, reformulated; see [39])**.** *Let G be a graph embedded on $\mathbb{S}^2$ (or $\mathbb{R}^2$). Then G disconnects $\mathbb{S}^2$ if and only if it contains a circuit.*

Theorem 1.4 (Jordan–Schönflies theorem; see [39])**.** *Let $f : \mathbb{S}^1 \to \mathbb{S}^2$ be a one-to-one continuous map. Then $\mathbb{S}^2 \setminus f(\mathbb{S}^1)$ is homeomorphic to two disjoint copies of the open disk.*

These results are, perhaps surprisingly, difficult to prove: the difficulty comes from the generality of the hypotheses (only continuity is required). For example, if in the Jordan curve theorem one assumes that G is embedded in the plane with polygonal edges, the theorem is not hard to prove.

[1] In this course, the only type of connectivity considered is path connectivity.

[2] This is often called a *cycle*; however, in the context of these notes, this word is also used to mean a homology cycle, so it seems preferable to avoid overloading it.

A graph, embedded in $\mathbb{S}^2$, is ***cellularly embedded*** if its faces are (homeomorphic to) open disks. Henceforth, we only consider cellular embeddings. It turns out that a graph embedded on the sphere is cellularly embedded if and only if it is connected.[3]

1.2 Combinatorics

So far, we have considered curves and graph embeddings in the plane that are rather general.

1.2.1 Combinatorial maps for planar graph embeddings

We now focus on the combinatorial properties of cellular graph embeddings in the sphere. Since we are not interested in the geometric properties, and since each face is homeomorphic to a disk, it suffices to specify how the faces are "glued together", or alternatively the cyclic order of the edges around a vertex. Embeddings of graphs on the plane are treated similarly: just choose a distinguished face of the embedding into $\mathbb{S}^2$, representing the "infinite" face of the embedding in the plane.

An algorithmically sound way of representing combinatorially a cellular graph embedding in $\mathbb{S}^2$ is via ***combinatorial maps***. The combinatorial map associated to a cellular graph embedding G is the set of closed walks in G, obtained from walking around the boundary of each face of G. (In general, these walks may repeat edges and/or vertices). This information is enough to "reconstruct" the sphere combinatorially, by taking the abstract graph and attaching a disk to each facial walk.

However, in terms of data structures, these facial walks are not very easy to manipulate, so we now present a more elaborated data structure that contains the same information but is more convenient.

[3] Although this statement should be intuitively clear, it is not so obvious to prove. It may help to use the results of Chapter 2, especially the fact that every face of a graph embedding is a surface with boundary.

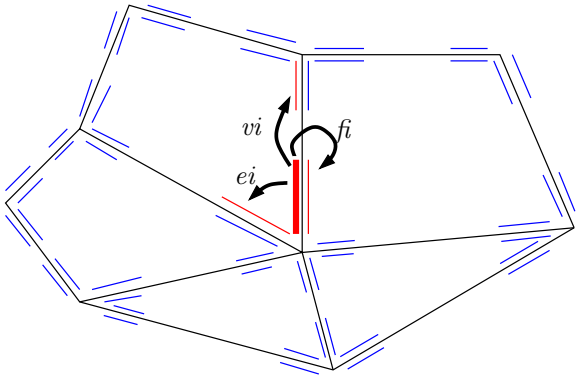


Figure 1.2. The flags are represented as line segments parallel to the edges; there are four flags per edge. The involutions vi, ei, and fi on the thick flag are also shown.

The basic notion is the ***flag***, which represents an incidence between a vertex, an edge, and a face of the embedding. Three involutions allow to move to a nearby flag, and, by iterating, to visit the whole graph embedding; see Figure 1.2:

- vi moves to the flag with the same edge-face incidence, but with a different vertex incidence;
- ei moves to the flag with the same vertex-face incidence, but with a different edge incidence;
- fi moves to the flag with the same vertex-edge incidence, but with a different face incidence.

Example 1.5. Figure 1.3, left, presents code to compute the degree of a vertex, i.e., the number of vertex-edge incidences of this vertex. The function takes as input a flag incident with that vertex. Note that a loop incident with the vertex makes a contribution of two to the degree. Dually, on the right, code to compute the degree of a face (the number of edge-face incidences of this face) is shown.

Note that a flag is not necessarily uniquely defined by its triple (vertex, edge, and face), as shows the example of a graph with a single vertex and

```
int vertex_degree(Flag fl) {
  int j=0;
  Flag fl2=fl;
  do {
    ++j;
    fl2=fl2->ei()->fi();
  } while (fl2!=fl);
  return j;
}
```

```
int face_degree(Flag fl) {
  int j=0;
  Flag fl2=fl;
  do {
    ++j;
    fl2=fl2->ei()->vi();
  } while (fl2!=fl);
  return j;
}
```

Figure 1.3. C++ code for degree computation.

a single (loop) edge.

The ***complexity*** of a graph $G = (V, E)$ is $|V| + |E|$. The ***complexity*** of a cellular graph embedding is the total number of flags involved, which is linear in the number of edges (every edge bears four flags), and also in the number of vertices, edges, and faces. Therefore the complexity of a graph cellularly embedded in the plane and of one of its embeddings are linearly related.

The data structure indicated above allows to "navigate" throughout the data structure, but does not store vertices, edges, and faces explicitly. In many cases, however, it is necessary to have one data structure ("object") per vertex, edge, or face. For example:

- if one has to be able to check in constant time whether an edge is a loop (incident twice to the same vertex), the data structure given above is not sufficient. On the other hand, if every flag has a pointer to the incident vertex, then testing whether an edge is a loop can be done by testing the equality of two pointers in constant time;
- in coloring problems, one need to store colors on the vertices of the graph. Such information can be stored in the data structure used for each vertex.

For such purposes, each flag can have a pointer to the underlying vertex, edge, and face (called respectively *vu*, *eu*, *fu*). Each such vertex, edge, or face contains no information on the incident elements, only information needed in the algorithms. If needed, one may similarly put some information in the vertex-edge, edge-face, vertex-face, and vertex-edge-face

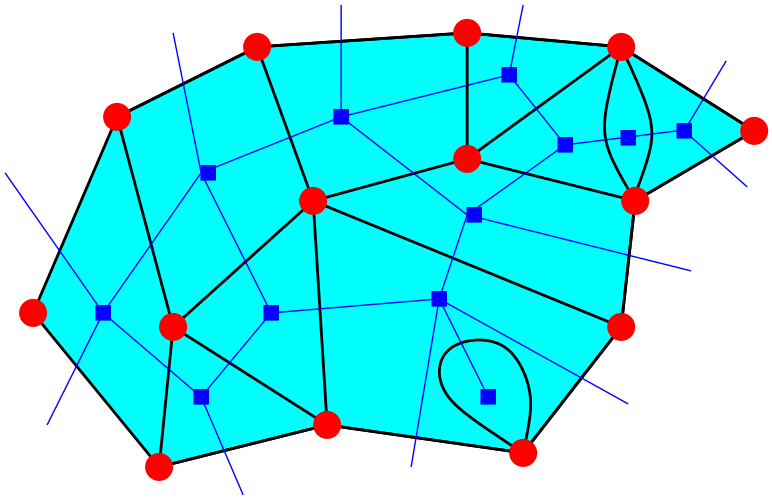


Figure 1.4. Duality.

incidences. Maintaining such informations, however, comes with a cost, which is not always desirable. For example, assume we want to be able to remove one edge incident to two different faces in constant time. If we keep the information *fu*, this must take time proportional to the smaller degree of the two faces (since the two faces are merged, the *fu* pointer has to be updated at least on one side of the edge). If we only keep *vu*, say, then such an update is not needed, and this edge removal can be done in constant time.

1.2.2 Duality and Euler's formula

A ***dual graph*** of a cellular graph embedding $G = (V, E)$ on $\mathbb{S}^2$ 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 1.4.

A dual graph embedding is also cellular. The combinatorial map of the

dual graph is unique. Actually, with the map representation, dualizing is easy: simply replace *fi* with *vi* and vice-versa. This in particular proves that duality is an involution: $G^{**} = G$.

Exercise 1.6 (easy)**.** ☆☆☆ Every tree (acyclic connected graph) with v vertices and e edges satisfies $v - e = 1$.

Lemma 1.7. *Let $G = (V, E)$ be a cellular graph embedding in $\mathbb{S}^2$, and let $G^* = (F^*, E^*)$ be its dual graph. Furthermore, let $E' \subseteq E$.*

Then (V, E') is acyclic if and only if $(F^*, (E \setminus E')^*)$ is connected. In particular, (V, E') is a spanning tree if and only if $(F^*, (E \setminus E')^*)$ is a spanning tree.

Proof. (V, E') is acyclic if and only if $\mathbb{S}^2 \setminus E'$ is connected, by the Jordan curve theorem 1.3. Furthermore, $\mathbb{S}^2 \setminus E'$ is connected if and only if $(F^*, (E \setminus E')^*)$ is connected: Two points x and x' in faces f and f' of G can be connected by a path avoiding E' and not entering any face other than f and f' if and only if f and f' are adjacent by some edge not in E', i.e. if and only if f^* and f'^* are adjacent in $(F^*, (E \setminus E')^*)$. □

Corollary 1.8 (Euler's formula for cellular graph embeddings in $\mathbb{S}^2$)**.** *For every cellular graph embedding in $\mathbb{S}^2$ with v vertices, e edges, and f faces, we have $v - e + f = 2$.*

Hence this formula also holds for every embedding of a connected graph in the plane.

Proof. Let T be the edge set of a spanning tree of G. The dual edges of its complement, $(E \setminus T)^*$, is also a spanning tree. The number of edges of G is $e = |T| + |(E \setminus T)^*|$, which, by Exercise 1.6, equals $(v - 1) + (f - 1)$. □

Exercise 1.9 (easy direction of Kuratowski's theorem)**.** ☆☆☆ Show that the complete graph with 5 vertices, K_5, is not planar. Indication: Use Euler's formula and double-counting on the number of vertex-edge and edge-face incidences. Also show that the bipartite graph $K_{3,3}$ (with 6 vertices $v_1, v_2, v_3, w_1, w_2, w_3$ and 9 edges, connecting every possible pair $\{v_i, w_j\}$) is not planar.

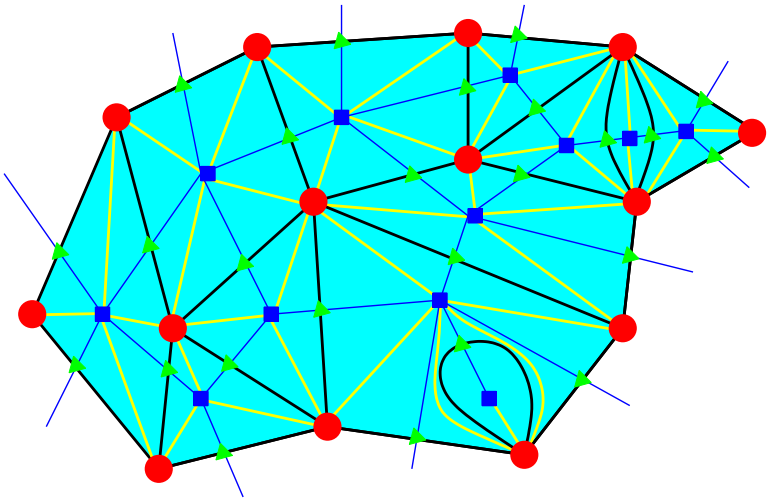


Figure 1.5. The barycentric subdivision of the part of the graph shown in Figure 1.4.

1.3 Notes

For more information on basic topology, see for example Armstrong [1] or Henle [22]; see also Stillwell [37]. For more informations on planar graphs, see (the next two chapters and) Mohar and Thomassen [32, Chapter 2].

There are many essentially equivalent ways of representing planar graph embeddings [14, 23]; the computational geometry library CGAL implements one of them[4]. We will see later that (most of) these data structures generalize to graphs embedded on surfaces. There are further generalizations to higher dimensions [2, 28, 29]; this is important especially in geometric modelling.

Eppstein provides many proofs of Euler's formula[5].

Exercise 1.9 shows that K_5 and $K_{3,3}$ are not planar. There is a converse statement: Kuratowski's theorem asserts that a graph G is planar if and only if it does not contain K_5 or $K_{3,3}$ as a subdivision; in other words, if and only if one cannot

[4] http://www.cgal.org/Manual/3.4/doc_html/cgal_manual/HalfedgeDS/Chapter_main.html.

[5] http://www.ics.uci.edu/~eppstein/junkyard/euler/.

obtain K_5 or $K_{3,3}$ from G by removing edges and isolated vertices and replacing every degree-two vertex and its two incident edges with a single edge [24, 30, 38].

Let G be a cellular embedding of a graph on $\mathbb{S}^2$. By overlaying G with its dual graph G^*, we obtain a *quadrangulation*: a cellular embedding of a graph G^+ where each face has degree four. See Figure 1.4. Every face of G^+ is incident with four vertices: one vertex v_G of G, one vertex v_{G^*} of G^*, and two vertices that are the intersection of an edge of G and an edge of G^*. If, within each face, we connect v_G with v_{G^*}, we obtain a triangulation, called the ***barycentric subdivision*** of G (Figure 1.5). Each triangle in the barycentric subdivision corresponds to a flag; its three neighbors are the flags reachable via the operations *vi*, *ei*, and *fi*.

Chapter 2

Topology of surfaces

2.1 Definition and examples

A ***surface*** is a topological space in which each point has a neighborhood homeomorphic to the unit open disk $\{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1\}$. We only consider *compact* surfaces in this chapter (and even later, unless specifically noted).

Examples of surfaces are the sphere, the torus, and the double torus: these are compact, connected, orientable (to be defined later) surfaces with zero, one, and two handles, respectively (see Figure 2.1). The classification of surfaces (Theorem 2.5) asserts that two compact, connected, and orientable surfaces are homeomorphic if and only if they have the same number of "handles".

Despite the figures, note that a surface is "abstract": the only knowledge we have of it is the neighborhoods of each point. A surface is not necessarily embedded in $\mathbb{R}^3$. Actually, the non-orientable surfaces cannot be

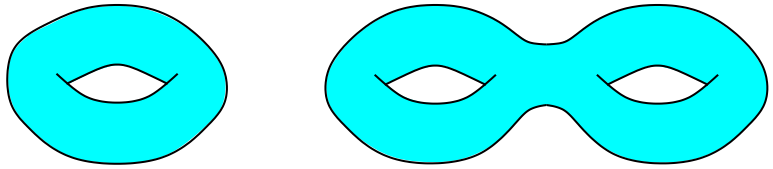


Figure 2.1. A torus and a double-torus.

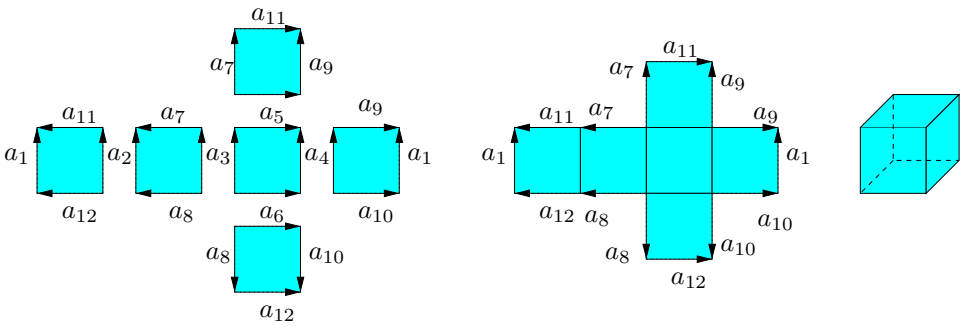


Figure 2.2. A polygonal schema of a graph embedded on a sphere (the graph of the cube) is: $a_2a_{11}\bar{a}_1\bar{a}_{12}$, $a_3a_7\bar{a}_2\bar{a}_8$, $a_4\bar{a}_5\bar{a}_3a_6$, $a_1\bar{a}_9\bar{a}_4a_{10}$, $a_9\bar{a}_{11}\bar{a}_7a_5$, and $a_{12}\bar{a}_{10}\bar{a}_6a_8$.

embedded in $\mathbb{R}^3$.

2.2 Surface (de)construction

2.2.1 Surface deconstruction

A graph embedded on a surface is ***cellularly embedded*** if all its faces are topological disks. As in the case of the plane, we may consider the combinatorial map of a graph cellularly embedded on a surface; the data structures are identical. The dual graph is defined similarly.

The ***polygonal schema*** associated with a cellular graph embedding is defined as follows: assign an arbitrary orientation to each edge; for each face, record the cyclic list of edges around the face, with a bar if and only if it appears in reverse orientation around the face. See Figure 2.2.

2.2.2 Surface construction

Conversely, the data of a polygonal schema allows to build up a surface and the cellular graph embedding. More precisely, let S be a finite set of *symbols* and let $\bar{S} = \{\bar{s} \mid s \in S\}$. Let R be a finite set of *relations*, each relation being a non-empty word in the alphabet $S \cup \bar{S}$, so that for every

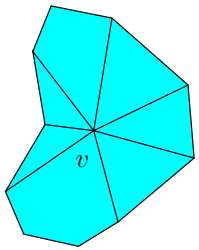


Figure 2.3. The "corners" incident to some vertex v can be ordered cyclically.

$s \in S$, the total number of occurrences of s plus the number of occurrences of $\bar{s}$ in R is exactly two.

For each relation of size n, build an n-gon; label its edges by the elements of R, in order, the presence of a bar indicating the orientation of the edge (see Figure 2.2). (Polygons with one or two sides are also allowed.) Now, identify the "twin" edges of the polygons corresponding to the same symbol in S, taking the orientation into account. (As a consequence, vertices get identified, too.)

Lemma 2.1. *The topological space obtained by the above process is a compact surface.*

Proof. Let X be the resulting topological space; X is certainly compact. We have to show that every point of X has a neighborhood homeomorphic to the unit disk. The only non-obvious case is that of a *vertex* v in X, that is, a point corresponding to a vertex of some polygons. But it is not hard to prove that a neighborhood of v is an *umbrella*: the "corners" (vertices) of the polygons corresponding to v can be arranged into a cyclic order; see Figure 2.3. □

We admit the following converse:

Theorem 2.2 (Kerékjártó-Radó; see Thomassen [39] or Doyle and Moran [13])**.** *Any compact surface is homeomorphic to a surface obtained by the gluing process above.*

This amounts to saying that, on any compact surface, there exists a cellular embedding of a graph. Equivalently, every surface can be triangulated.

(a) (b)

Figure 2.4. (a) The orientations of these two faces (triangles) are compatible. (b) Two non-compatible orientations of the faces. A surface is orientable if there exist orientations of all faces that are compatible.

2.3 Classification of surfaces

2.3.1 Euler characteristic and orientability character

Let G be a graph cellularly embedded on a compact surface $\mathscr{S}$. The ***Euler characteristic*** of G equals $v - e + f$, where v is the number of vertices, e is the number of edges, and f is the number of faces of the graph.

Proposition 2.3. *The Euler characteristic is a* **topological invariant***: it only depends on the surface $\mathscr{S}$, not on the cellular embedding.*

Sketch of proof. The Euler characteristic is easily seen to be invariant under Euler operations. The result is then implied by the following claim: any two cellular embeddings on a given surface can be transformed one into the other via a finite sequence of Euler operations. Proving this is not very difficult but requires some work; a key property is that one can assume both embeddings to be piecewise linear with respect to a given triangulation of the surface (using for example the method by Epstein [15, Appendix]). □

G is ***orientable*** if the boundary of its faces can be oriented so that each edge gets two opposite orientations by its incident faces (Figure 2.4). The orientability character is a topological invariant as well; the same proof as that of Proposition 2.3 works, but it can also be proven directly:

Exercise 2.4. ☆☆ G is orientable if and only if no subset of $\mathscr{S}$ is a Möbius strip.

2.3.2 Classification theorem

Theorem 2.5. *Every compact, connected surface $\mathscr{S}$ is homeomorphic to a surface given by the following polygonal schemata, called* **canonical** *(each made of a single relation):*

i. $a\bar{a}$ *(the sphere; Euler characteristic* 2*, orientable);*

ii. $a_1b_1\bar{a}_1\bar{b}_1 \dots a_gb_g\bar{a}_g\bar{b}_g$*, for* $g \geq 1$ *(Euler characteristic* $2-2g$*, orientable);*

iii. $a_1a_1 \dots a_ga_g$*, for* $g \geq 1$ *(Euler characteristic* $2-g$*, non-orientable).*

Furthermore, the surfaces having these polygonal schemata are pairwise non-homeomorphic. In particular, two connected surfaces are homeomorphic if and only if they have the same Euler characteristic and the same orientability character.

In the above theorem, g is called the ***genus*** of the surface; by convention $g = 0$ for the sphere. The orientable surface of genus g is obtained from the sphere by cutting g disks and attaching g "handles" in place of them. Similarly, the non-orientable surface of genus g is obtained from the sphere by cutting g disks and attaching g Möbius strips (since a Möbius strip has exactly one boundary component). See Figure 2.5. See also Figure 2.6 for a representation of a double-torus in canonical form.

Proof. Let $\mathscr{S}$ be a compact, connected surface, and G be a graph embedded on $\mathscr{S}$ (by Theorem 2.2). By iteratively removing edges incident with different faces, we may assume that G has only one face.[1] By iteratively contracting edges incident with different vertices, we may assume that G has only one vertex and one face[2] (unless this yields a sphere, so the polygonal schema is $a\bar{a}$ — actually, we could say that the polygonal schema made of the empty relation is a degenerate polygonal schema for the sphere). The surface $\mathscr{S}$ cut along G is therefore a topological disk; we use cut-and-paste operations on this polygonal schema to obtain a standard form.

[1] This amounts to removing all primal edges of a spanning tree in the dual graph.
[2] This amounts to contracting the edges of a spanning tree in the primal graph.

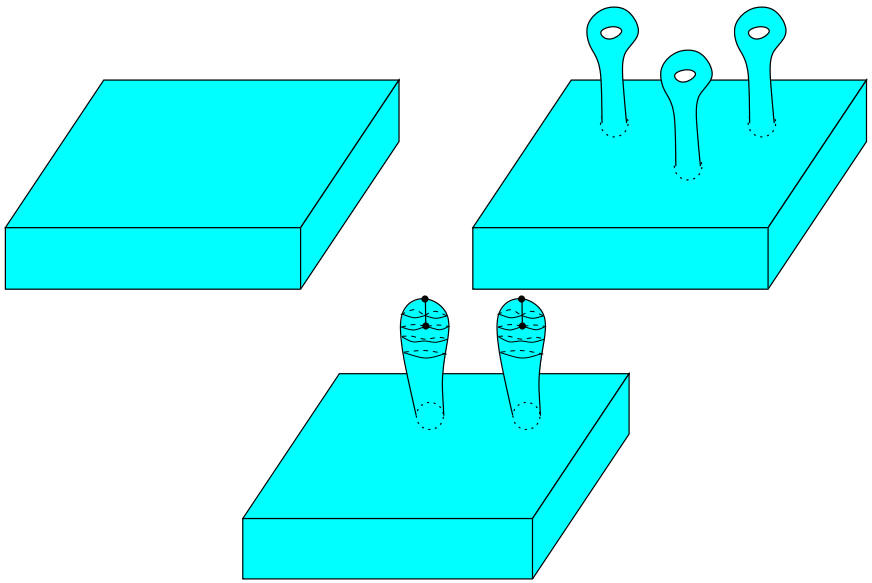


Figure 2.5. Every compact, connected surface is obtained from a sphere by removing disjoint disks and attaching handles (orientable case) or Möbius strips (non-orientable case). However, the non-orientable surfaces are not embeddable in $\mathbb{R}^3$.

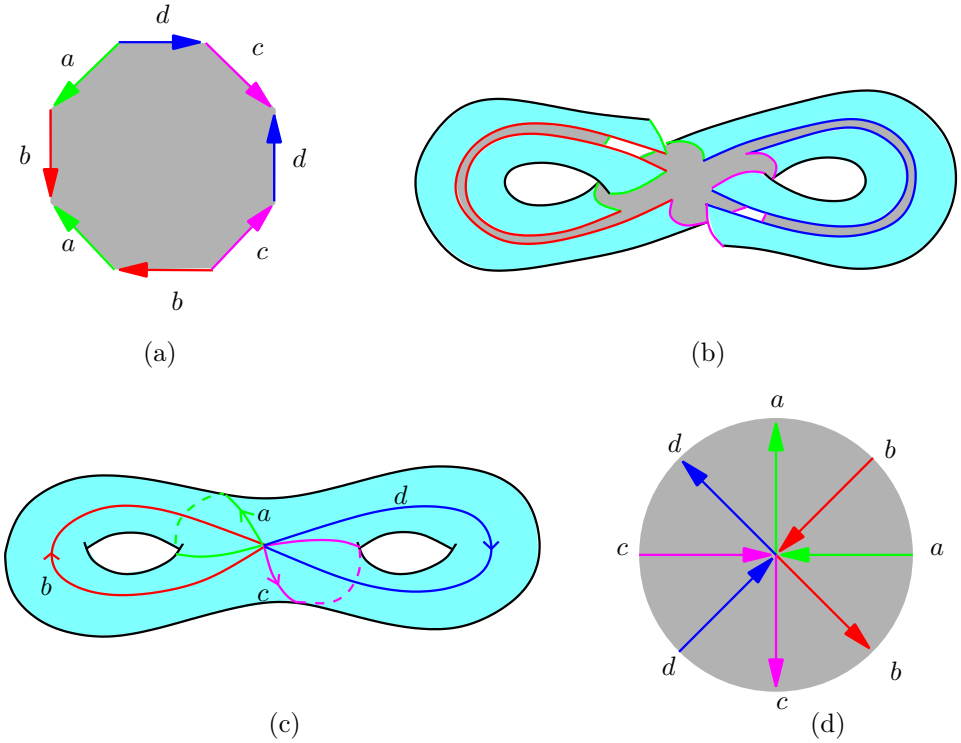


Figure 2.6. (a) A canonical polygonal schema of the double torus. (b) The identification of the edges of the schema. (c) The actual graph embedded on the double torus. (d) Closeup on the order of the loops around the basepoint of the surface, as seen from below; it can be derived directly from (a).

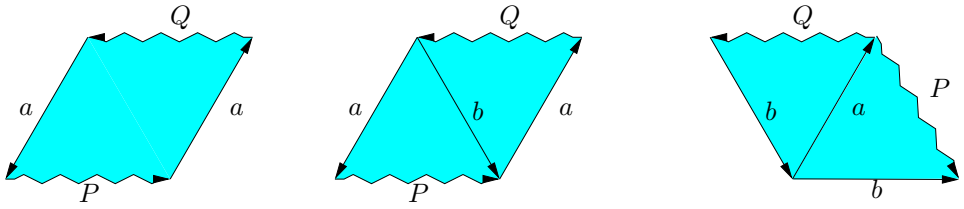


Figure 2.7. The classification of surfaces: grouping the twin edges appearing with the same orientation.

If the polygonal schema has the form $aPaQ$ (where P and Q are possibly empty sequences of symbols), then we can transform it into $bb\bar{P}Q$ (Figure 2.7)—$\bar{Q}$ denotes the symbols of Q in reverse order, inverting also the presence or absence of a bar above each letter. So inductively, we may assume that each pair of symbols appearing in the polygonal schema with the same orientation is made of two consecutive symbols. We still have one face and one vertex.

Assume some edge appears twice in the polygonal schema with opposite orientations: $aP\bar{a}Q$. Then P and Q must share an edge b, because otherwise the endpoints of a would not be identified on the surface. By the preceding step, b must appear in opposite orientations in P and Q, so we may assume that the polygonal schema has the form $aPbQ\bar{a}R\bar{b}S$. Then, by further cut-and-paste operations, we may transform the polygonal schema into $dc\bar{d}\bar{c}RQPS$ (Figure 2.8). We still have one face and one vertex, and can iterate the process. After this stage, the polygonal schema is the concatenation of blocks of the form aa and $ab\bar{a}\bar{b}$.

If there are no blocks of the form aa, or no blocks of the form $ab\bar{a}\bar{b}$, then we are in form (ii) or (iii), respectively. Otherwise, onea part of the boundary of the polygonal schema has the form $aabc\bar{b}\bar{c}$. We may transform it to $\bar{d}\bar{c}\bar{b}d\bar{b}\bar{c}$ (Figure 2.9), and, applying the method of Figure 2.7 to b, c, and d in order, we obtain that we replaced the part of the boundary we considered into $eefffgg$; the other part of the boundary is unchanged. So iterating, we may convert the polygonal schema into form (iii).

The Euler characteristics and orientability characters of the surfaces are readily computed, since the canonical polygonal schemata have exactly one

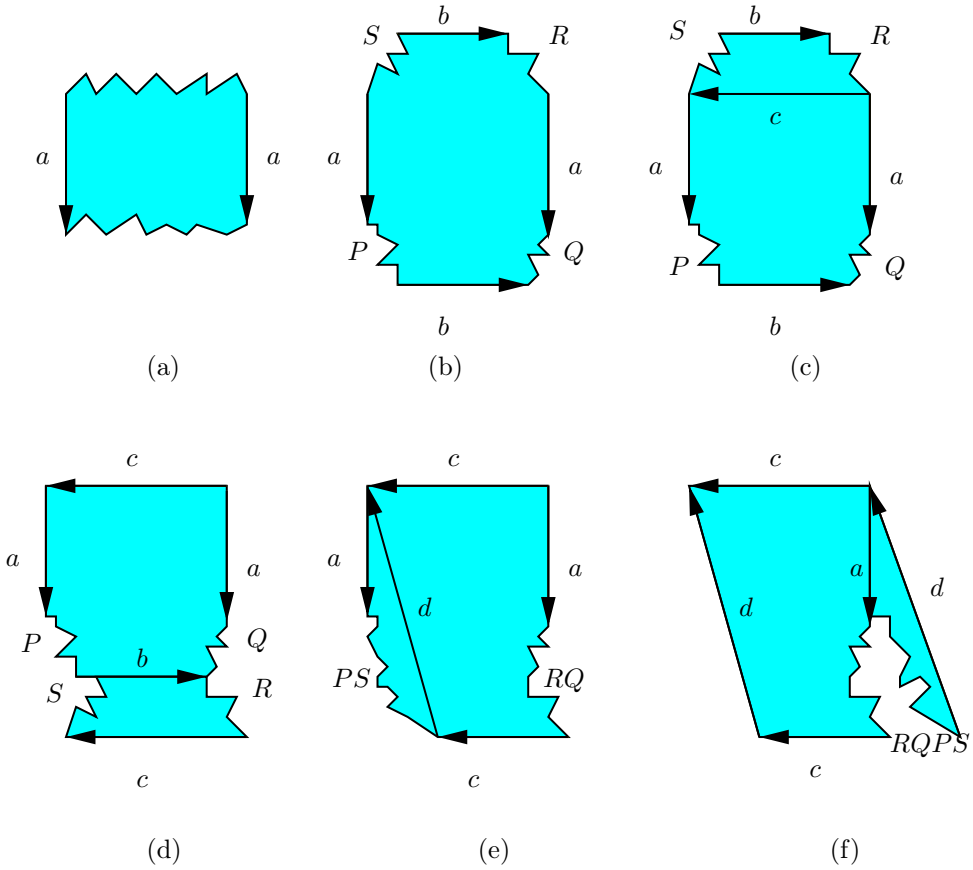


Figure 2.8. The classification of surfaces: grouping pairs of twin edges appearing with different orientations.

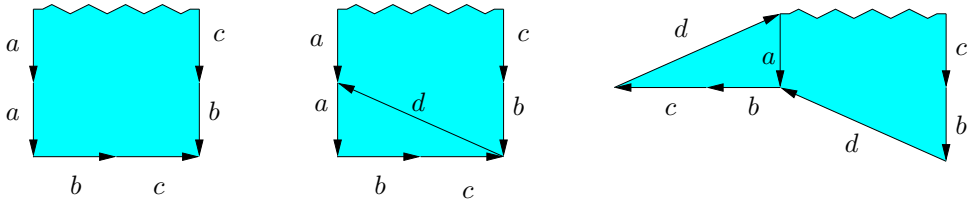


Figure 2.9. The classification of surfaces: transforming one form into the other.

vertex and one face. Since two distinct canonical polygonal schemata do not have the same Euler characteristic and the same orientability character, they cannot be homeomorphic, by Proposition 2.3 and Exercise 2.4. □

Example 2.6.

- The orientable surface with genus 1 is a ***torus***; the orientable surface with genus 2 is the ***double torus***; and so on.
- The non-orientable surface with genus 1 is a ***projective plane***; with genus 2 it is the ***Klein bottle***.

Exercise 2.7. ☆☆ Identify the surfaces with the following schemata:

1. $a\bar{a}b\bar{b}$;
2. $abab$;
3. $ab a\bar{b}$;
4. $a_1a_2\dots a_n\bar{a}_1\bar{a}_2\dots\bar{a}_n$;
5. $a_1a_2\dots a_{n-1}a_n\bar{a}_1\bar{a}_2\dots\bar{a}_{n-1}a_n$.

2.4 Surfaces with boundary

A ***surface (possibly) with boundary*** $\mathscr{S}$ is a topological space in which each point has a neighborhood homeomorphic to the unit open disk $\{(x,y)\in\mathbb{R}^2 \mid x^2+y^2<1\}$ or to the unit half disk $\{(x,y)\in\mathbb{R}^2 \mid x^2+y^2<1 \text{ and } x\geq 0\}$.

The ***boundary*** of $\mathscr{S}$, denoted by $\partial\mathscr{S}$, comprise the points of this surface that have no neighborhood homeomorphic to the unit disk. The ***interior*** of $\mathscr{S}$ is the complementary part of its boundary.

A ***cellular embedding*** on a surface with boundary is defined as in the case of surfaces without boundary. In particular, since each face must be an open disk, the boundary of the surface must be the union of some edges of the graph. The classification theorem (Theorem 2.5) can be extended for surfaces with boundary: Given a surface with boundary $\mathscr{S}$, we may attach a disk to each of its boundary components, obtaining a surface without boundary $\bar{\mathscr{S}}$, and apply the previous classification theorem. Furthermore, the number of boundary components is a topological invariant.

The Euler characteristic and the orientability character of a cellular embedding on a surface with boundary $\mathscr{S}$ are defined as in the case of surfaces without boundary; they are also topological invariants. The Euler characteristic of $\mathscr{S}$ equals that of $\bar{\mathscr{S}}$ minus the number of boundary components of $\mathscr{S}$. So two surfaces with boundary $\mathscr{S}$ and $\mathscr{S}'$ are homeomorphic if and only if they have the same Euler characteristic, orientability character, and number of boundary components.

If we have a graph embedding G without isolated vertex on a surface $\mathscr{S}$, then ***cutting*** $\mathscr{S}$ along G is a well-defined operation that yields a surface with boundary, denoted by $\mathscr{S}\backslash\backslash G$.[3] This fact is not trivial, and follows from the fact that every graph embedding on a surface $\mathscr{S}$ can be mapped by a homeomorphism of $\mathscr{S}$ (actually, an isotopy) to a piecewise-linear embedding with respect to a fixed triangulation of $\mathscr{S}$, using, e.g., the method by Epstein [15, Appendix].

2.5 Notes

The classification theorem is due to Brahana, Dehn, and Heegaard; the present proof is inspired from Stillwell [37]. For another, more visual proof, see Francis and Weeks [21].

The proofs of the classification theorem usually involve two steps, the first one being topological (Theorem 2.2, Proposition 2.3, Exercise 2.4), the second one being combinatorial. In the same vein, the *Hauptvermutung* ("main conjecture") says that any two embeddings of a graph on a surface are subdivisions of graph embeddings that are combinatorially identical. This is true, but some higher-dimensional analogs do not hold.

Let G and M be simple graphs (that is, without loops or multiple edges). M is a ***minor*** of G if M can be obtained from G by iteratively contracting edges, deleting edges, and deleting isolated vertices (at each step, the graph should be made simple by removing loops and identifying multiple edges). Let $\mathscr{S}$ be a fixed surface. Clearly, if G is embeddable on $\mathscr{S}$, then every minor of G is also

[3] This notation is not standard (yet).

embeddable on $\mathscr{S}$. Let $\mathcal{F}$ be the set of minor-minimal graphs *not* embeddable on $\mathscr{S}$; thus G is embeddable on $\mathscr{S}$ if and only if no graph in $\mathcal{F}$ is a minor of G. Kuratowski's theorem asserts that G is planar if and only if it does not have K_5 or $K_{3,3}$ as a minor; in other words, if $\mathscr{S}$ is the sphere, the family $\mathcal{F}$ is finite. This actually holds for every surface $\mathscr{S}$; however, no algorithm is known to enumerate the family $\mathcal{F}$.

More generally, this property is implied by a deep result by Robertson and Seymour [36] (whose proof needed no less than 20 papers and several hundreds of pages): In any infinite family of graphs, at least one is a minor of another.

Chapter 3

Computing shortest graphs with cut loci

In this chapter, we describe algorithms to compute shortest curves and graphs that "cut" a given surface into simpler pieces.

3.1 Combinatorial and cross-metric surfaces

We aim at computing "short" graphs and curves on surfaces. For this, we need to define a metric on a surface that is both accurate in the applications and simple enough so as to be handled algorithmically. We shall introduce two ways of doing this, which are dual of each other. Depending on the context, some results and algorithms are more easily described using one setting or the other.

In this chapter, all surfaces are compact, connected, and orientable. They do not have boundaries.

3.1.1 More types of curves

We already defined paths on surfaces; we need to introduce more types of curves.

A ***loop*** ℓ is a path with the same endpoints; $\ell(0) = \ell(1)$ is called the ***basepoint*** of the loop. A path is ***simple*** if it is one-to-one. A loop

is ***simple*** if its restriction to $[0,1)$ is one-to-one (of course, due to the identified endpoints, it cannot be one-to-one on $[0,1]$).

The ***concatenation*** of p and q, denoted by $p \cdot q$, is the path defined by:

- $(p \cdot q)(t) = p(2t)$, if $0 \leq t \leq 1/2$;
- $(p \cdot q)(t) = q(2t-1)$, if $1/2 \leq t \leq 1$.

A ***reparameterization*** of a path p is a path of the form $p \circ \varphi$, where $\varphi : [0,1] \to [0,1]$ is bijective and increasing. If the paths are considered up to reparameterization, the concatenation is associative. The ***inverse*** of a path p, denoted by $\bar{p}$, is the map $t \mapsto p(1-t)$.

3.1.2 Combinatorial surfaces

A ***combinatorial surface*** $(\mathcal{S}, M)$ is the data of a surface $\mathcal{S}$ (possibly with boundary), together with a cellular embedding M of a weighted graph. The weights must be non-negative. In this model, the only allowed curves are walks in M; the length of a curve is the sum of the weights of the edges traversed by the curve, counted with multiplicity.

3.1.3 Cross-metric surfaces

We will, however, use a dual formulation of this model, which allows to define *crossings* between curves: this turns out to be helpful both for stating the results and as intermediate steps. A ***cross-metric surface*** $(\mathcal{S}, M^*)$ is a surface $\mathcal{S}$ together with a cellular embedding of a weighted graph M^*. We consider only *regular* paths on $\mathcal{S}$, which intersect the edges of M^* only transversely and away from the vertices. The ***length*** $\mathrm{length}(\gamma)$ of a regular 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*.

To any combinatorial surface $(\mathcal{S}, M)$ without boundary, we associate by duality a cross-metric surface $(\mathcal{S}, M^*)$, where M^* is (as notation suggests) the dual graph of M. 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. So far, the notions of combinatorial and of cross-metric surfaces (without boundary) are thus essentially the same, up to duality. We can easily construct shortest paths on a cross-metric surface by restating the usual algorithms (for example, Dijkstra's algorithm) on M in terms of the dual graph M^*.

3.1.4 Curves on cross-metric surfaces, algorithmically

We can represent an arbitrary set of possibly (self-)intersecting curves on a cross-metric surface $(\mathcal{S}, M^*)$ by maintaining the *arrangement* of M^* and of the curves, i.e., the combinatorial embedding associated with the union of the curves (assuming this union forms a cellular embedding, which will always be the case). Contrary to combinatorial surfaces, this data structure also encodes the crossings between curves. The initial arrangement is just the graph M^*, without any additional curve. We embed each new curve *regularly*: 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 M^* 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. However, it is sometimes desirable to assign a non-zero weight to the edges of a curve. For example, the cross-metric surface $\mathcal{S} \setminus\!\setminus \alpha$ obtained by cutting $\mathcal{S}$ along an embedded curve α can be represented simply by assigning infinite crossing weights to the edges that comprise α, indicating that these edges cannot be crossed by other curves.

3.1.5 Complexity

The ***complexity*** of a combinatorial surface $(\mathcal{S}, M)$ is the total number of vertices, edges, and faces of M; similarly, the ***complexity*** of a cross-metric surface $(\mathcal{S}, M^*)$ is the total number of vertices, edges, and faces of M^*.

The ***complexity*** of a set of curves is the number of times it crosses edges of M^*.

3.2 Cut loci

Let us fix the notations for the remaining part of this chapter. Unless otherwise noted, $(\mathscr{S}, M^*)$ is a cross-metric surface (connected, compact, orientable, without boundary) of genus g and complexity n. Furthermore, b is a point inside a face of M^* and is the basepoint of all loops considered in this chapter (we omit the precision that the basepoint is b in the sequel).

Let T be the shortest path tree from b to a point in each face of M^*.[1] The ***cut locus*** C of $(\mathscr{S}, M^*)$ with respect to b is the subgraph of M^* obtained by removing all edges of M^* crossed by T. It is therefore a graph embedded on $\mathscr{S}$. See Figure 3.1.

Lemma 3.1. *$\mathscr{S} \setminus\setminus C$ is a disk.*

Proof. At some stage of the growth of the shortest path tree T, consider the union of all open faces of M^* visited by T, and of all edges of M^* crossed by T. This is an open disk; at the end, it contains all faces of M^*, and its complement is C. In particular, $\mathscr{S} \setminus\setminus C$ is a disk. □

Intuitively, we are inflating a disk around b progressively, without allowing self-intersections, until it occupies the whole surface; the cut locus C is the set of points of the surface where the boundary of the disk touches itself.

Dijkstra's algorithm implies that we can compute C in $O(n \log n)$ time.

Exercise 3.2 (Complexity of the reduced cut locus)**.** ☆ Let C' be the graph obtained from the cut locus C' by repeatedly removing every degree-one vertex, together with its incident edge, and replacing every degree-two vertex v and its incident edges with an edge connecting the two neighbors of v. Prove that C' has complexity $O(g)$.

[1] Strictly speaking, the shortest path tree is not always unique: there may be several shortest paths between two given points. However, uniqueness holds for generic choices of the weights; in other words, it can be enforced using an arbitrarily small perturbation of the lengths. By a slight abuse of language, we will therefore use the article "the" in such cases, since it does not harm (and may help the reader) to think that uniqueness holds. Nevertheless, no algorithm or result presented here requires uniqueness of shortest paths.

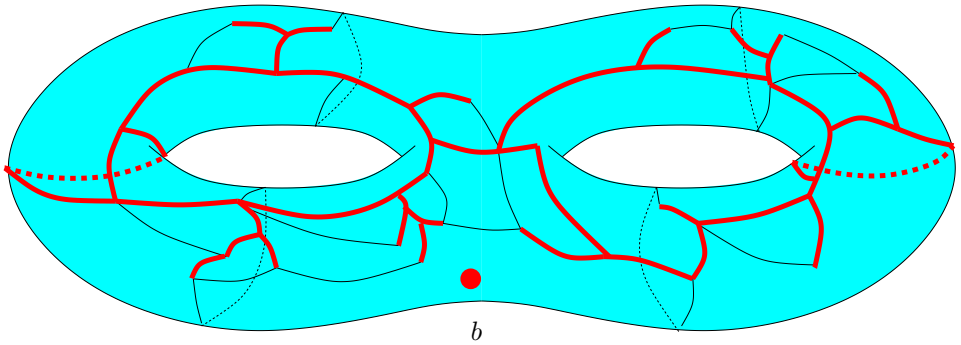


Figure 3.1. The cut locus C of a double torus (in bold lines) and the remaining edges of M^* (in thin lines).

Given an edge $e \in C$, the loop $\sigma(e)$ is defined as a loop with basepoint b that follows the shortest path tree to go from its root b to a face incident with e, crosses e, and goes back from the other face incident with e to the root. This can be done so that all the loops $\sigma(e)$ are simple and disjoint (except, of course, at their basepoint b—we shall omit this triviality in the sequel). See Figure 3.2.

Define the *weight* of an edge e of C to be the length of the corresponding loop $\sigma(e)$ (this is not the same as the crossing weight, defined for every edge of M^*!); these weights can be computed with no time overhead during the cut locus computation.

3.3 Shortest non-contractible loop

A (possibly non-simple) loop is ***contractible*** if it can be continuously deformed into a point.

Exercise 3.3. ☆☆☆ Prove that, on a disk or a sphere, every loop is contractible.

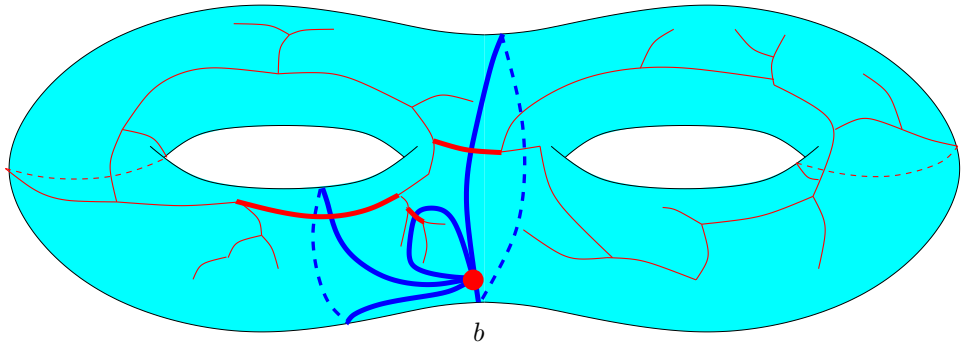


Figure 3.2. The loops $\sigma(e)$, for three edges $e \in C$.

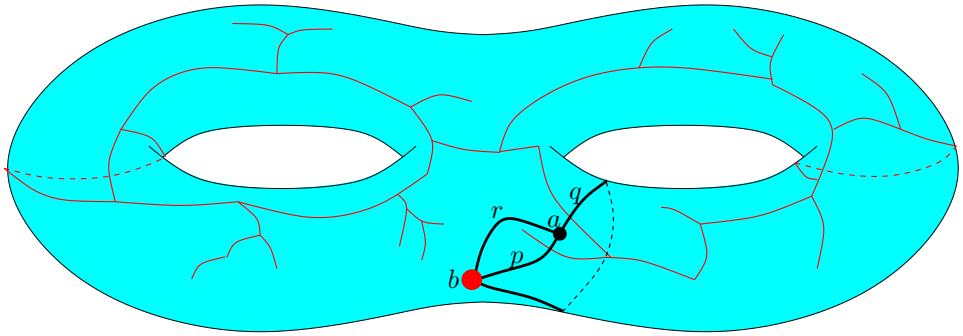


Figure 3.3. Illustration of Lemma 3.6.

Lemma 3.4. *A simple loop is contractible if and only if it bounds a disk.*

Proof. If a loop bounds a disk, it is certainly contractible. The proof of the converse is more difficult, and we admit it. □

Our goal now is to give an algorithm to compute the shortest non-contractible loop.

3.3.1 3-path condition

A set L of loops satisfies the ***3-path condition*** if, for any point $a \neq b$ and any three paths p, q, and r from b to a, if $p \cdot \bar{q}$ and $q \cdot \bar{r}$ belong to L, then $p \cdot \bar{r}$ belongs to L.

Lemma 3.5. *The set of contractible loops satisfies the 3-path condition.*

Proof. If $p \cdot \bar{q}$ and $q \cdot \bar{r}$ are contractible, then so is their concatenation, $(p \cdot \bar{q}) \cdot (q \cdot \bar{r})$, which deforms continuously to $p \cdot \bar{r}$. □

Lemma 3.6. *Let L be a set of loops satisfying the 3-path condition. Some shortest loop not in L crosses the cut locus C at most once.*

Proof. See Figure 3.3 for an illustration of the proof. Let ℓ be a shortest loop not in L; without loss of generality, we can choose ℓ such that it crosses C as few times as possible. Assume, for the sake of a contradiction, that ℓ crosses C at least twice; let a be a point on ℓ not on M^* between its first and last crossing with C. This point a splits ℓ into two paths p and q, both from b to a, and we have $\ell = p \cdot \bar{q}$. Furthermore, let r be the shortest path from b to a; this path does not cross C.

The 3-path condition applied to p, q, and r implies that $p \cdot \bar{r}$ or $q \cdot \bar{r}$ does not belong to L. Both paths are no longer than $\ell = p \cdot \bar{q}$ and cross C fewer times than ℓ, implying the desired contradiction. □

3.3.2 Structural lemmas

Lemma 3.7. *Some shortest non-contractible loop has the form $\sigma(e)$.*

Proof. Let ℓ be a shortest non-contractible loop. By Lemmas 3.5 and 3.6, some shortest non-contractible loop crosses the cut locus at most once. On the other hand, every non-contractible loop has to cross C at least once (since $\mathscr{S} \setminus\setminus C$ is a disk). Hence some shortest non-contractible loop crosses the cut locus exactly once, at some edge e. This loop deforms continuously to $\sigma(e)$, which cannot be longer. The result follows. □

Lemma 3.8. *Let e be an edge of C. Then $\sigma(e)$ is contractible if and only if some component of $C - e$ is a tree.*

Proof. Assume first that one component of $C - e$ is a tree. One can them move $\sigma(e)$ continuously over the tree to make it disjoint from C; the resulting loop is contractible.

Conversely, if $\sigma(e)$ is contractible, it bounds a disk D by Lemma 3.4. We want to prove that the part of C inside D is a tree. But if it is not the case, this part contains a circuit, which further bounds a disk $D' \subset D$, and therefore C cuts $\mathscr{S}$ into at least two pieces, one of which is D'; this is impossible (Lemma 3.1). □

3.3.3 Algorithm

Theorem 3.9. *Finding a shortest non-contractible loop can be done in $O(n \log n)$ time. The loop computed is simple.*

Proof of Theorem 3.9. We first compute the cut locus C, and assign to every edge e of C a weight that is the length of $\sigma(e)$, in $O(n \log n)$ time. We show how to eliminate the edges e such that at least one component of $C - e$ is a tree. This concludes, since it then suffices to select the minimum-weight remaining edge of C (by Lemmas 3.7 and 3.8).

This graph pruning can be done in $O(n)$ time: put all edges incident with a degree-one vertex in a list. Then, while the list is non-empty, remove an edge e from it; remove it from C (unless it was already removed); if one or both of its endpoints have now degree one in C, put the corresponding edge(s) in the list. Clearly, this removes only edges e such that no component of $C - e$ is a tree. All them must eventually be removed, because a tree has a degree-one vertex (a leaf). □

Corollary 3.10. *Finding a shortest non-contractible loop without specified basepoint can be done in $O(n^2 \log n)$ time.*

Proof. For every face of M^*, run the algorithm in Theorem 3.9 with the basepoint in that face, and return the shortest loop. □

3.4 Shortest non-separating loop

3.4.1 Types of simple loops

A simple loop ℓ is ***separating*** if $\mathscr{S} \setminus\!\setminus \ell$ is not connected. A simple contractible loop bounds a disk, hence is separating; the converse is false. So there are (essentially) three kinds of simple loops: contractible, separating but not contractible, and non-separating. These three types are illustrated in Figure 3.2.

Exercise 3.11. ☆

1. Give an algorithm that determines whether a given simple loop is separating.
2. Give an algorithm that determines whether a given simple loop is contractible. Indication: use Lemma 3.4.

Our present goal is to compute the shortest non-separating (simple) loop. We need first to define the notion of homology boundary, which generalizes the notion of separating loop to possibly non-simple loops. To anticipate, we introduce a bit more technicalities than those needed for this sole purpose.

3.4.2 Preliminaries on homology

We introduce ***1-dimensional homology for graphs embedded on surfaces, over*** $\mathbb{Z}/2\mathbb{Z}$.

To simplify matters, we assume here (and in Section 3.5) that all curves considered are drawn on a very dense graph $G = (V, E)$ embedded on $\mathscr{S}$, transversely to M^*.[2] We consider ***chains***: subsets of E. It is a natural

[2] This would not be needed if we introduced *singular homology*, but it seems preferable to avoid doing so. The assumption above is actually not needed: we only require G to be dense enough so that the loops $\sigma(e)$ are disjoint walks on G and so that G contains some shortest non-separating loop (or some shortest system of loops, in Section 3.5). The existence of such a graph G is clear, and it is never used in the algorithms, only in proofs, so its complexity does not matter.

$\mathbb{Z}/2\mathbb{Z}$-*vector space*: the addition of two subsets of E is the symmetric difference, multiplication by 0 gives the empty subset of E, and multiplication by 1 is the identity.

A chain $E' \subseteq E$ is a ***homology cycle*** if every vertex of V is incident with an even number of edges of E'. A chain $E' \subseteq E$ is a ***homology boundary*** if the faces of G can be colored black and white so that E' is the set of edges of E with exactly one black and one white incident face. Equivalently, if we consider the "dual" graph of (V, E'), which has one vertex inside each face of (V, E') and one edge crossing each edge of (V, E'), then E' is a homology boundary if and only if this dual graph is bipartite.

Exercise 3.12. ☆☆☆

1. Prove that the set of homology cycles (resp. homology boundaries) forms a vector space, and that every homology boundary is a homology cycle.
2. Assume $\mathscr{S}$ is a sphere. Prove that every homology cycle is a homology boundary.

Lemma 3.13. *A simple loop ℓ in G disconnects $\mathscr{S}$ if and only if its edge set forms a homology boundary.*

Proof. Let E' be the set of edges of ℓ. Either the graph (V, E') has one face, in which case the only boundary is the empty set, or it has two faces, in which case coloring one face in black and the other one in white yields a non-zero boundary formed by the edge set of ℓ. □

So the notion of homology boundary extends the notion of being separating.

As shown in Exercise 3.12, the set of homology boundaries, B, is included in the set of homology cycles, Z. The reverse inclusion does not hold in general. Homology measures the "difference" between Z and B; formally, it is Z/B, the $\mathbb{Z}/2\mathbb{Z}$-vector space that is the quotient of the two $\mathbb{Z}/2\mathbb{Z}$-vector spaces Z and B.

Given a loop ℓ in G, its ***mod 2 image*** is the set of edges of G used an odd number of times by ℓ. (We sometimes identify a loop with its mod 2 image.)

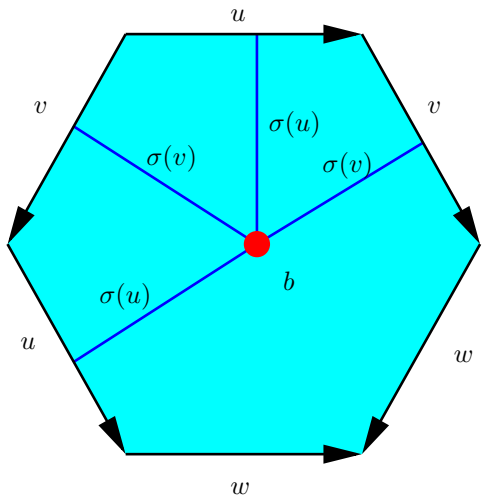


Figure 3.4. A view of the disk $\mathscr{S}\backslash\backslash C$, whose polygonal schema is $uvw\bar{w}\bar{u}\bar{v}$. The loops $\sigma(u)$ and $\sigma(v)$ are cut into two paths connecting the basepoint to twin points.

3.4.3 Algorithm

We prove here:

Theorem 3.14. *Finding a shortest loop that is not a homology boundary can be done in $O(n \log n)$ time. The loop computed is simple, and is (therefore) also a shortest simple non-separating loop.*

Corollary 3.15. *Finding a shortest loop without specified basepoint that is not a homology boundary (or a shortest simple non-separating closed curve) can be done in $O(n^2 \log n)$ time.*

Lemma 3.16. *A subset A of the edges of C disconnects C if and only if the set of loops $\sigma(A)$ disconnects $\mathscr{S}$.*

Proof. We may certainly assume $A \neq \emptyset$. Let D be the disk $\mathscr{S}\backslash\backslash C$; the basepoint b belongs to the interior of D. Each loop $\sigma(e)$ in $\sigma(A)$ corresponds, in D, to two paths from b to the boundary of D, connecting twins of e. See Figure 3.4.

Therefore, if we let $\tau(e)$ be the intersection of e with $\sigma(e)$, any path in $\mathscr{S} \setminus \sigma(A)$ continuously retracts to a path in $C \setminus \tau(A)$, without moving the endpoints if they already belong to C. This implies that $\mathscr{S} \setminus \sigma(A)$ is connected if and only if $C \setminus \tau(A)$ is connected; this is in turn equivalent to having $C - A$ connected. □

Proof of Theorem 3.14. The general strategy is very similar to the proof of Theorem 3.9. The set of all loops in G whose mod 2 images are homology boundaries satisfies the 3-path condition. Hence, by Lemma 3.6, some shortest loop in G whose mod 2 image is *not* a homology boundary crosses the cut locus at most once, hence exactly once, at some edge e, by Exercise 3.12. A slight extension of that exercise implies that $\sigma(e)$ is in the same homology class, and it is no longer. Hence some shortest loop whose mod 2 image is not a homology boundary has the form $\sigma(e)$.

In particular, it is simple, and is therefore a non-separating loop (Lemma 3.13). It must be a shortest non-separating loop in G because every separating loop is a homology boundary. It is therefore a shortest non-separating loop, because we can (retroactively) assume that G contains some shortest non-separating loop.

By Lemma 3.16, we are thus looking for a minimum-weight edge e of C such that $C - e$ is connected; such edges are called *non-bridge* edges. By Lemma 3.17 below, we can determine all non-bridge edges in linear time. Alternatively, note that any minimum-weight edge not in a maximum spanning tree of C is such an edge. □

Lemma 3.17. *Let G be a graph of complexity n. One can in $O(n)$ time determine all the bridge edges of G.*

Proof. Run a depth-first search on the graph G, starting from an arbitrary root vertex. Recall that this partitions the edges of G into *link edges*, which belong to the rooted search tree T, and *back edges*, which connect a vertex v with an ascendent of v in T. Clearly, no back edge is a bridge. A link edge e with endpoints u and v, where u is visited before v, is a non-bridge edge if and only if there exists a back edge from a descendent of v (maybe v itself) to an ascendent of u (maybe u itself). The algorithm will consider each back edge (uv) in turn and mark as non-bridge the edges on the path from u to v in T; the remaining edges are exactly the bridge edges.

To achieve this in linear time, take all back edges $(x_1, y_1), \dots, (x_k, y_k)$ (where y_k is an ascendent of x_k), ordered such that $y_1, \dots, y_k$ are discovered in this order during the depth-first search (such an ordering can easily be found in $O(n)$ time). Starting from x_1, and walking towards the root of T, mark every edge as being a non-bridge edge until reaching y_1. Start from x_2, and walk towards the root of the tree, marking every edge as non-bridge, until either reaching y_2 or reaching an edge e that is already marked as non-bridge. If the latter possibility occurs, y_1 must be an ancester of y_2 in T by the choice of the ordering, so all edges between e and y_2 must be already marked. Continue similarly with the other back edges. This process clearly takes linear time in total. □

3.5 Shortest system of loops

In this section, we describe an algorithm to compute a shortest topological decomposition of the surface. Namely, a ***system of loops*** L is a set of simple loops meeting pairwise only at their common basepoint b, such that $\mathscr{S} \setminus\setminus L$ is a disk (refer to Figure 2.6(c) for an example). We give an algorithm to compute the shortest system of loops of a given surface.

3.5.1 Algorithm

Define a ***homology basis of loops*** to be a set of loops whose homology classes (of their mod 2 images) form a basis of the homology vector space. There exist homology bases of loops:

Exercise 3.18. ☆☆☆ Prove that every homology cycle is the mod 2 image of a loop.

Recall that a *system of loops* L is a set of simple loops meeting pairwise only at their common basepoint, such that $\mathscr{S} \setminus\setminus L$ is a disk. Denote by $[\ell]$

the homology class of a loop ℓ, and by $[L]$ the set of homology classes of a set of loops L.

Lemma 3.19. *Some shortest homology basis is made of loops of the form $\sigma(e)$. In particular, the loops in that basis are simple and disjoint.*

Proof. Let ℓ be a loop in the shortest homology basis. Let $e_1, \dots, e_k$ be the edges of the cut locus crossed by ℓ. Then it is not too hard, using Exercise 3.12(2), to prove that $[\ell] = [\sigma(e_1)] + \dots + [\sigma(e_k)]$.

In particular, ℓ crosses at least one edge of the cut locus. Furthermore, since $[\ell]$ is linearly independent from the homology classes of the other loops in the basis, one of the $[\sigma(e_i)]$ must be linearly independent from the homology classes of the other loops in the basis. Replacing ℓ with $\sigma(e_i)$ still yields a homology basis, which is no longer than the original one because $\sigma(e_i)$ is a shortest loop with basepoint b among the loops that cross e_i, and ℓ indeed crosses e_i. Iterating, we obtain that some shortest homology basis is made of loops of the form $\sigma(e)$. □

Exercise 3.20. ☆☆☆ Let L be a set of simple, disjoint loops in G. Prove that L disconnects $\mathscr{S}$ if and only if the homology classes of the loops in L are linearly dependent.

Theorem 3.21. *We can compute a a shortest homology basis of loops in $O(gn + n \log n)$ time. Furthermore, there are $2g$ loops, each of the form $\sigma(e)$.*

Proof. By Lemma 3.19, computing a shortest homology basis of loops boils down to computing a shortest inclusionwise maximal set of loops $\sigma(e_1), \dots, \sigma(e_k)$ with linearly independent homology classes, or, equivalently, that does not disconnect $\mathscr{S}$ (Exercise 3.20). This is equivalent to computing an inclusionwise maximal set S of edges of C such that $C - S$ is connected, with minimal sum of weights (Lemma 3.16). This precisely means computing the complement of a maximum-weight spanning tree of C.

Recall that C is cellularly embedded on $\mathscr{S}$ with one face (Lemma 3.1). Therefore, by Euler's formula, the number of vertices, v, and edges, e, of C satisfy $v - e = 2 - 2g - 1 = 1 - 2g$. A spanning tree always contains $v - 1$ edges (Lemma 1.6), so the complement of a spanning tree of C has exactly $2g$ edges; we conclude that there are $2g$ loops in L.

Computing the cut locus C takes $O(n \log n)$ time. A maximum spanning tree can be computed in $O(n \log n)$ time using any textbook algorithm. The actual loops may each have $O(n)$ size, and there are $2g$ of these. □

Proposition 3.22. *The shortest homology basis of loops L computed in Theorem 3.21 is actually a shortest system of loops.*

Proof. Every system of loops is made of $2g$ loops by Euler's formula. The homology classes of a system of loops are linearly independent (Exercise 3.20), and there are $2g$ of these, so they form a basis. So any system of loops is a homology basis. It therefore suffices to prove that L is a system of loops.

L is a set of $2g$ simple, disjoint loops that does not disconnect $\mathscr{S}$. Cutting along it yields a (connected) surface of Euler characteristic 1 (because cutting along the first loop keeps the Euler characteristic unchanged and cutting along each subsequent loop increases it by one), hence a disk. □

3.6 Notes

3.6.1 Discrete vs. continuous setting

Most of the combinatorial and cross-metric surface model is taken from Colin de Verdière and Erickson [10]. Several tools of this section were described in a combinatorial setting for simplicity of exposition, but they have well-studied continuous counterparts.

In general, the cut locus of a point x in a metric space S is the set of points in S for which there exist at least two distinct shortest paths to x. It is closely related to the notion of the *medial axis* of a compact set $K \subset S$: it is the set of points of $S \setminus K$ whose distance to K is realized by at least two points of K. If K is finite, the medial axis contains in particular the Voronoi diagram of K.

The main topological property of a cut locus we have used (in Lemmas 3.8 and 3.16) can be stated as follows for a surface with boundary: for any subset A of the edges of C, $\mathscr{S} \setminus \sigma(A)$ deformation retracts to $C - A$. In particular,

they have the same number of connected components, and one of the components of $\mathscr{S} \setminus \sigma(A)$ is a disk if and only if the corresponding component of $C - A$ (is connected and) contains no non-contractible loop, i.e., is a tree.

As mentioned earlier, homology can be defined in a continuous setting (*singular homology*), which vastly generalizes the ad-hoc route we took. Let S be any topological space. Let Δ_n be the n-dimensional simplex. The set of *n-chains* C_n is the vector space (say over $\mathbb{Z}/2\mathbb{Z}$, but this generalizes to arbitrary fields, and even rings) generated by all continuous maps $\Delta_n \to S$. There is a *boundary operator* ∂_n taking C_n to C_{n-1}: the boundary of $\Delta_n \to S$ is a sum of $n+1$ maps $\Delta_{n-1} \to S$, one for each face of Δ_n. One checks the important property that $\partial_{n-1} \circ \partial_n = 0$, so $\operatorname{Im} \partial_n \subseteq \operatorname{Ker} \partial_{n-1}$. The set of *homology cycles* is $Z_n := \operatorname{Ker} \partial_{n-1}$ and the set of *homology boundaries* is $B_n := \operatorname{Im} \partial_n$. These vector spaces have infinite dimension (except in trivial cases), but their quotient $H_n := Z_n/B_n$, the *homology vector space*, is usually of finite dimension; it is non-trivial to prove that, under reasonable conditions, H_1 is isomorphic to the homology vector space as introduced in Section 3.4.2.

3.6.2 Algorithms

Erickson and Har-Peled [18] gave the first algorithms to compute the shortest non-contractible or non-separating loop, relying on the idea of "wavefront propagation". The method presented here is different; the idea of considering the edges of the cut locus is borrowed from Erickson and Whittlesey [19]. The 3-path condition is a variation on Mohar and Thomassen [32, p. 110].

If the genus is small, then our $O(n^2 \log n)$ algorithm is not very efficient; after successive improvements [4, 25], the best algorithm up to date has running-time $O(g^3 n \log n)$ [3]. In contrast, computing the shortest separating but non-contractible simple loop (without specified basepoint) is NP-hard [5].

Erickson and Whittlesey [19] described the algorithm of Section 3.5; the algorithm was further generalized, and the proof was simplified, by Colin de Verdière [7], which was in turn simplified by Erickson [16].

Note that there are systems of loops whose polygonal schema is not in canonical form (for example $abcd\bar{a}\bar{b}\bar{c}\bar{d}$). The shortest system of loops is not necessarily in canonical form. There is an $O(gn)$ time algorithm to compute a system of loops in canonical form [26, 40], but computing the shortest such system is likely to be NP-hard. There are other kinds of topological decompositions of surfaces, such as *pants decompositions*: sets of disjoint simple closed curves that cut the surface into spheres with three boundary components. The status of computing the shortest pants decomposition is open [35].

Chapter 4

Testing homotopy via the universal cover

In this chapter, we introduce two important tools related to surfaces. The notion of *homotopy* captures the intuitive notion of deformation. The *universal cover* of a surface provides a way to compute paths restricted to a given homotopy class, i.e., up to deformation.

In this section, $\mathcal{S}$ is a compact, connected, orientable surface, although the definition would apply to almost arbitrary topological spaces.

4.1 Homotopy

4.1.1 Definition

Two paths p and q on $\mathcal{S}$, having both u and v as endpoints, are ***homotopic*** if there exists a continuous family of paths whose endpoints are u and v between p and q. More formally, a ***homotopy*** between p and q is a continuous map $h : [0,1] \times [0,1] \to \mathcal{S}$ such that $h(0,\cdot) = p$, $h(1,\cdot) = q$, $h(\cdot,0) = u$, and $h(\cdot,1) = v$. This definition applies in particular to the case of loops ($u = v$).

4.1.2 Fundamental group

Let b be a point of $\mathcal{S}$. The relation "is homotopic to" partitions the set of loops with basepoint b into ***homotopy classes***. Let us denote by $[\![\ell]\!]$ the homotopy class of a loop ℓ. The set of homotopy classes can be equipped with the law "$\cdot$" defined by $[\![\ell]\!] \cdot [\![\ell']\!] = [\![\ell \cdot \ell']\!]$, and, with this law, the set of homotopy classes of loops with basepoint b is a group, called the ***fundamental group*** of $(\mathcal{S}, b)$ and denoted by $\pi_1(\mathcal{S}, b)$ or more concisely $\pi_1(\mathcal{S})$, whose unit element is the class of contractible loops.

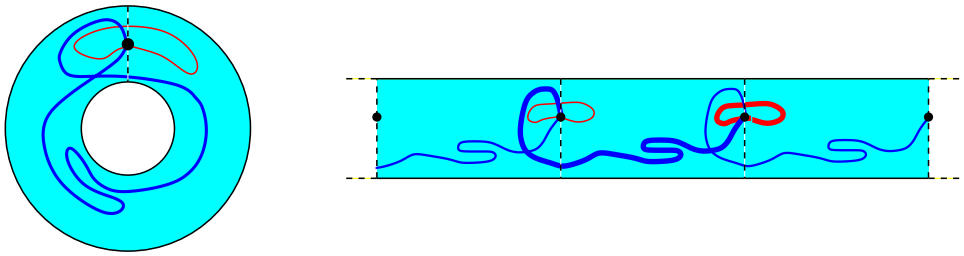
In particular, the fundamental group of the disk or the sphere is trivial: two paths having the same endpoints are homotopic. The fundamental group of the annulus is $\mathbb{Z}$ (the homotopy class of a loop is the same as the signed number of times it "winds around the hole"), and the fundamental group of the torus is $\mathbb{Z}^2$.

4.2 Universal cover

Informally, the ***universal cover*** $\widetilde{\mathcal{S}}$ of a surface $\mathcal{S}$ is a surface $\widetilde{\mathcal{S}}$ which "locally looks like $\mathcal{S}$", but is "much larger than $\mathcal{S}$": it is not compact (except in trivial cases) and every point in $\mathcal{S}$ generally corresponds ("lifts") to infinitely many points in $\widetilde{\mathcal{S}}$; two paths are homotopic in $\mathcal{S}$ if and only if these paths can be lifted to paths which have the same endpoints in $\widetilde{\mathcal{S}}$. The universal cover is thus a tool to compute homotopy.

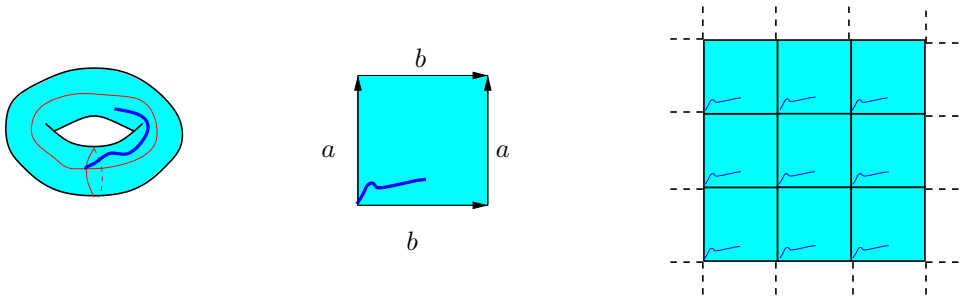
4.2.1 Examples

Let $\mathcal{S}$ be the annulus depicted on Figure 4.1(a). If this annulus is cut along the dashed line segment, we obtain a rectangle; if we glue together infinitely many copies of this rectangle, we obtain an "infinite strip", depicted on Figure 4.1(b), which will be denoted by $\widetilde{\mathcal{S}}$. There is a natural "projection" π from $\widetilde{\mathcal{S}}$ onto $\mathcal{S}$, such that a path in $\mathcal{S}$ can be ***lifted*** to a path (in fact, infinitely many paths) in $\widetilde{\mathcal{S}}$. We see that two paths p and p' are homotopic in $\mathcal{S}$ if two lifts of p and p' starting at the same point of $\widetilde{\mathcal{S}}$



(a) (b)

Figure 4.1. (a): An annulus $\mathscr{S}$ and two loops with the same basepoint (in black). (b): Its universal cover $\widetilde{\mathscr{S}}$, with lifts of these loops. The vertices of $\widetilde{\mathscr{S}}$ in black are the lifts of the basepoint.



(a) (b) (c)

Figure 4.2. (a): A torus. (b): A polygonal schema of the torus. (c): The universal cover of the torus.

have the same targets. The two loops represented on the figure are not homotopic, because one of them is contractible (its lifts in $\widetilde{\mathscr{S}}$ are loops), and the other one is non-contractible (its lifts are not closed).

The same kind of figure can be drawn for the torus (Figure 4.2(a)). If this torus $\mathscr{S}$ is viewed as a polygonal schema in canonical form (Figure 4.2(b)), a square whose opposite sides will be identified to obtain $\mathscr{S}$, its universal cover consists of infinitely many copies of this copy organized in a grid-like fashion: hence, it is the plane (Figure 4.2(c)).

4.2.2 Definition and properties

Precisely, a ***universal cover*** of a connected surface $\mathscr{S}$ is the data of a pair $(\widetilde{\mathscr{S}}, \pi)$, where:

- $\widetilde{\mathscr{S}}$ is a (possibly non-compact) surface which is *simply connected*, i.e., every loop in $\widetilde{\mathscr{S}}$ is contractible;
- π is a continuous map from $\widetilde{\mathscr{S}}$ onto $\mathscr{S}$, called *projection*, which is a *local homeomorphism*: any point x of $\mathscr{S}$ has an open, connected neighborhood U such that $\pi^{-1}(U)$ is a disjoint union of open sets $(U_i)_{i\in I}$ and $\pi|_{U_i} : U_i \to U$ is a homeomorphism.

Every connected surface (possibly with boundary) has a universal cover (we will provide constructions in the following sections). On the other hand, two universal covers are isomorphic (that is, there is a homeomorphism between them that "projects" to the identity map). This allows to speak without ambiguity of *the* universal cover of a surface $\mathscr{S}$.

A ***lift*** of a path p is a path $\widetilde{p}$ in $\widetilde{\mathscr{S}}$ such that $\pi \circ \widetilde{p} = p$.

The main properties of $(\widetilde{\mathscr{S}}, \pi)$ that we will use are:

- the *lift property*: let p be a path in $\mathscr{S}$ whose source is y; let $x \in \pi^{-1}(y)$. Then there exists a unique path $\widetilde{p}$ in $\widetilde{\mathscr{S}}$, whose source is x, such that $\pi \circ \widetilde{p} = p$;
- the *homotopy property*: two paths p_1 and p_2 with the same endpoints are homotopic in $\mathscr{S}$ if and only if they have two lifts $\widetilde{p}_1$ and $\widetilde{p}_2$ sharing the same endpoints in $\widetilde{\mathscr{S}}$;
- the *intersection property*: a path p in $\mathscr{S}$ self-intersects if and only if either a lift of p self-intersects, or two lifts of p intersect.

4.2.3 General construction for surfaces without boundary

Let $\mathscr{S}$ be an orientable surface without boundary, with genus g. If $g = 0$, $\mathscr{S}$ is the sphere, and the universal cover is $\mathscr{S}$ itself, as every loop in $\mathscr{S}$ is contractible. If $g = 1$, $\mathscr{S}$ is the torus, and the universal cover is described in Figure 4.2. We now explain how to build the universal cover of $\mathscr{S}$, assuming $g \geq 2$.

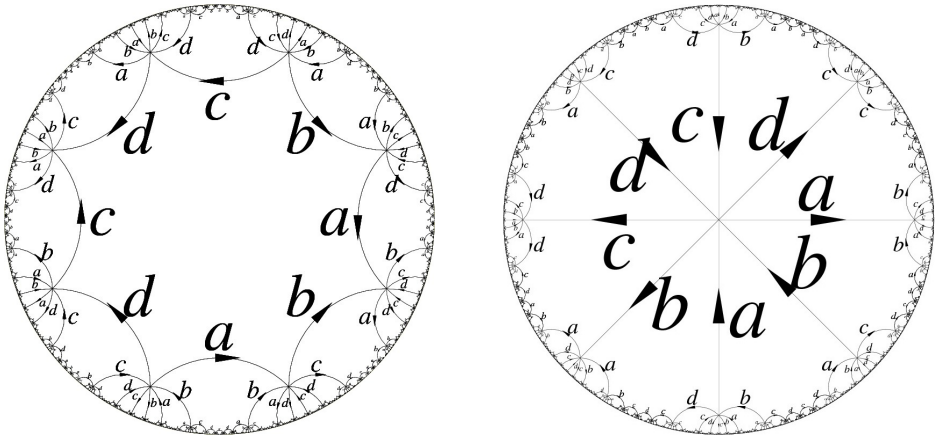


Figure 4.3. Two views of the universal cover of the double torus (images taken from http://topologygeometry.blogspot.fr/2010/06/notes-from-062310.html).

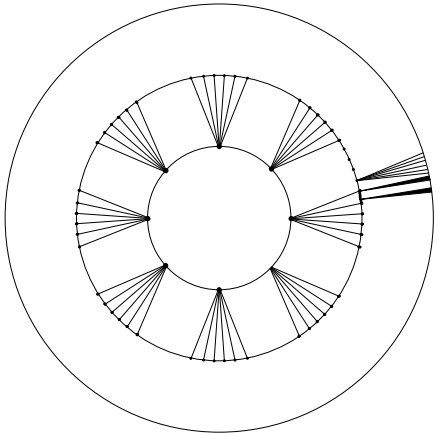


Figure 4.4. The combinatorial construction of the universal cover of the double torus.

$\mathscr{S}$ has a polygonal schema of the form $a_1b_1\bar{a}_1\bar{b}_1 \dots a_gb_g\bar{a}_g\bar{b}_g$; namely, a $4g$-gon with sides identified by pairs. Moreover, the unique vertex of the corresponding graph on $\mathscr{S}$ has a single vertex, of degree $4g$. By analogy with the case $g = 1$, the universal cover of $\mathscr{S}$ can be built by gluing together $4g$-gons in the plane in a way that each vertex has degree $4g$, see Figure 4.3 for an example for $g = 2$. Of course the $4g$-gons become quickly distorted, but there is no obstruction in designing this construction combinatorially by induction as follows (see Figure 4.4).

For each positive integer i, let C_i be the circle centered at the origin with radius i. We place $4g$ points on the first circle C_1, whose interior forms the first $4g$-gon. Now, each vertex of C_1 must have degree $4g$, so needs to be connected with $4g - 2$ new vertices, which we place on C_2. Each arc between consecutive vertices on C_2 is now subdivided with the appropriate number of vertices ($4g - 2$ or $4g - 3$) so that each face between C_1 and C_2 is a $4g$-gon. Now, each vertex on C_2 is linked to $4g - 2$ new vertices on C_3. And so on.

Moreover, if we choose the labels of the sides of the initial polygon as prescribed by the polygonal schema $a_1b_1\bar{a}_1\bar{b}_1 \dots$, one sees that, by induction, one can label the edges of the polygons in a way consistent with the polygonal schema. This defines the projection from our space to $\mathscr{S}$. We have thus built the universal cover of $\mathscr{S}$.

4.3 Testing homotopy

The *contractibility problem* is defined as follows: Given a loop ℓ in a graph G cellularly embedded on a surface with genus g, determine whether ℓ is contractible. This is an instance of the *word* problem in combinatorial group theory (given a group specified in terms of generators and relations, and a word in the generators and their inverses, decide whether the element represented by the word is the unit element).

Deciding whether ℓ is contractible can be done in time *linear* in the input size (namely, the complexity of G and the number of edges of ℓ) [12,20,27]. We provide a simpler algorithm with worse running time, but still linear if the genus g is fixed. We start with a special case.

Lemma 4.1. *If G is a system of loops, then one can determine whether a loop ℓ with k edges is contractible in time $O(k \operatorname{poly}(g))$.*

The proof is essentially an argument due to Dehn [11] more than one century ago. In the proof below, we make no attempt to optimize the dependence on g, because more complicated linear-time algorithms exist.

Proof. The case $g = 0$ is obvious, as every loop is contractible. The case $g = 1$ is easy; as can be seen from the universal cover of the torus, if G is made of two loops a and b, then ℓ is contractible if the algebraic numbers of occurrences of both a and b in ℓ are zero. So we now assume $g \geq 2$.

If ℓ has a *spur*, an edge of G used twice consecutively in opposite directions, we can remove that spur. Removing iteratively all spurs takes $O(k)$ time; so we can assume that ℓ has no spur.

Assume that ℓ is contractible but not reduced to a single point. We claim that a subpath of ℓ consists of strictly more than half of the facial walk of G. To see this, look at a lift $\tilde{\ell}$ of ℓ in the universal cover, defined as above. Let C_k be the outermost circle used by $\tilde{\ell}$, and let $\tilde{\ell}'$ be a maximal subpath of $\tilde{\ell}$ on C_k. Since ℓ has no spur, $\tilde{\ell}'$ is made of at least $4g-2$ edges on C_k, because $\tilde{\ell}$ arrives and leaves C_k by an edge going to C_{k-1}. Thus the first $4g-2 > 2g$ edges of $\tilde{\ell}'$ project to a subpath of ℓ that is more than half of the facial walk of G. This proves the claim.

Accordingly, here is the algorithm to test contractibility. While some subpath of ℓ consists of strictly more than half of the facial walk of G, we replace that subpath with the complementary part of the facial walk of G; this strictly decreases the length of ℓ and does not change its homotopy class. When no such subpath exists, the loop ℓ is contractible if and only if it is reduced to a single vertex.

Encoding ℓ with the word of the oriented edges used by ℓ, finding an appropriate subpath of ℓ boils down to combinatorial pattern matching. Each time we find an appropriate subpath, we replace it with the complementary part, decreasing the length of ℓ. We need to go back along ℓ by $O(g)$ edges, because the replacement may have created a new appropriate subpath starting $O(g)$ edges earlier. So each step either moves forward along ℓ, or decreases its length and goes back by $O(g)$ edges. Each such step takes $\text{poly}(g)$ time, and there are at most k steps. □

More generally:

Theorem 4.2. *Let ℓ be a loop with k edges in a graph G with complexity n cellularly embedded on a surface with genus g. In $O(n+k)\,\text{poly}(g)$ time, we can determine whether ℓ is contractible.*

Proof. We iteratively contract edges of G until we get a single vertex, removing the occurrences of the corresponding edges in ℓ. Each time we have a face with degree one, we remove the incident edge in G, and all its occurrences in ℓ (since the face is a disk, the edge is contractible). Each time we have a face with degree two, we remove one of the two incident edges in G, and replace every occurrence of that edge in ℓ with the other edge incident to the face.

Euler's formula with double counting now implies that G has $O(g)$ edges. We choose a subset of edges that form a system of loops G', as in the beginning of the proof of Theorem 2.5. Each edge not in G' used by ℓ can be replaced with a homotopic subpath of $O(g)$ edges in G'. The new loop ℓ has $O(gk)$ edges. We have thus reduced the problem to the case where G is a system of loops, for which we can apply Lemma 4.1. □

4.4 Notes

Homotopy is a very natural and "geometric" notion (compared with homology, for example). However, homology has more algebraic structure and is therefore more tractable. Homotopy problems are generally hard: determining whether a loop is contractible is *undecidable* in innocent-looking spaces such as two-dimensional simplicial complexes and four-dimensional manifolds. The case of manifolds of dimension three is related to the Poincaré conjecture, solved only recently [33,34].

Massey [31, Chapter 5] contains details on the construction of the universal cover of more general topological spaces. The description of the universal cover for surfaces without boundary is also described by Stillwell [37, Sect. 6.1.3].

The aforementioned algorithms [20, 27] provide linear-time algorithms not only for the contractibility problem, but also for the *free homotopy problem* (corresponding, in group theory, to the conjugacy problem): Given two loops ℓ_1 and ℓ_2, can one transform one into the other by a *free homotopy*, a deformation that allows moving the basepoint during the deformation?

Bibliography

[1] Mark Anthony Armstrong. *Basic topology*. Undergraduate Texts in Mathematics. Springer-Verlag, 1983. [pp. 2 and 7]

[2] Erik Brisson. Representing geometric structures in d dimensions: topology and order. *Discrete & Computational Geometry*, 9:387–426, 1993. [p. 7]

[3] 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. [p. 22]

[4] 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. [p. 22]

[5] 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. [p. 22]

[6] Erin W. 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. [p. 2]

[7] 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*, number 6347 in Lecture Notes in Computer Science, pages 100–111, 2010. [p. 22]

[8] Éric Colin de Verdière. *Topological algorithms for graphs on surfaces*. Habilitation thesis, École normale supérieure, 2012. Available at `http://www.di.ens.fr/~colin/textes/12hdr.pdf`. [p. 2]

[9] Éric Colin de Verdière. Multicuts in planar and bounded-genus graphs with bounded number of terminals. In *Proceedings of the 23rd European Symposium on Algorithms (ESA)*, number 9294 in Lecture Notes in Computer Science, pages 373–385, 2015. [p. 2]

[10] Éric Colin de Verdière and Jeff Erickson. Tightening nonsimple paths and cycles on surfaces. *SIAM Journal on Computing*, 39(8):3784–3813, 2010. [pp. 2 and 21]

[11] Max Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72:413–421, 1912. [p. 26]

[12] Tamal K. Dey and Sumanta Guha. Transforming curves on surfaces. *Journal of Computer and System Sciences*, 58:297–325, 1999. [p. 25]

[13] P. H. Doyle and D. A. Moran. A short proof that compact 2-manifolds can be triangulated. *Inventiones Mathematicae*, 5:160–162, 1968. [p. 9]

[14] 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. [p. 7]

[15] David B. A. Epstein. Curves on 2-manifolds and isotopies. *Acta Mathematica*, 115:83–107, 1966. [pp. 10 and 13]

[16] Jeff Erickson. Combinatorial optimization of cycles and bases. In Afra Zomorodian, editor, *Computational topology*, Proceedings of Symposia in Applied Mathematics. AMS, 2012. [p. 22]

[17] Jeff Erickson. Computational topology, 2013. Course notes available at `http://compgeom.cs.uiuc.edu/~jeffe/teaching/comptop/`. [p. 2]

[18] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004. [p. 22]

[19] 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. [p. 22]

[20] Jeff Erickson and Kim Whittlesey. Transforming curves on surfaces redux. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1646–1655, 2013. [pp. 25 and 26]

[21] George K. Francis and Jeffrey R. Weeks. Conway's ZIP proof. *American Mathematical Monthly*, 106(5):393–399, 1999. [p. 13]

[22] Michael Henle. *A combinatorial introduction to topology*. Dover Publications, 1994. [p. 7]

[23] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry: Theory and Applications*, 13:65–90, 1999. [p. 7]

[24] Casimir Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930. [p. 8]

[25] 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. [p. 22]

[26] 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. 22]

[27] Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 440–449, 2012. [pp. 25 and 26]

[28] Bruno Lévy. *Topologie algorithmique: combinatoire et plongement*. PhD thesis, Institut National Polytechnique de Lorraine, 1999. [p. 7]

[29] Pascal Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry & Applications*, 4(3):275–324, 1994. [p. 7]

[30] Yuri Makarychev. A short proof of Kuratowski's graph planarity criterion. *Journal of Graph Theory*, 25:129–131, 1997. [p. 8]

[31] William S. Massey. *Algebraic topology: an introduction*, volume 56 of *Graduate Texts in Mathematics*. Springer-Verlag, 1977. [p. 26]

[32] Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001. [pp. 2, 7, and 22]

[33] Grisha Perelman. The entropy formula for the Ricci flow and its geometric application. arXiv:math/0211159, 2002. [p. 26]

[34] Grisha Perelman. Ricci flow with surgery on three-manifolds. arXiv:math/0303109, 2003. [p. 26]

[35] Sheung-Hung Poon and Shripad Thite. Pants decomposition of the punctured plane. arXiv:cs.CG/0602080, 2006. [p. 22]

[36] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004. [p. 14]

[37] John Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, second edition, 1993. [pp. 2, 7, 13, and 26]

[38] Carsten Thomassen. Kuratowski's theorem. *Journal of Graph Theory*, 5(3):225–241, 1981. [p. 8]

[39] Carsten Thomassen. The Jordan-Schönflies theorem and the classification of surfaces. *American Mathematical Monthly*, 99(2):116–130, 1992. [pp. 4 and 9]

[40] 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. 22]

Contents