MPRI Course 2-38-1: Algorithms and Combinatorics for Geometric Graphs
Lecture Notes

Arnaud de Mesmay

These are the lecture notes for the first half on the course 2-38-1.

Some practicalities:

- The course is on Thursdays, 12:45 to 15:45 in room 2035 of the Sophie Germain building.
- It will be graded with an exam, probably on November 26th.
- There will be an optional exercise sheet, with points contributing extra credits for the final grade.

This half of the course focuses on embedded graphs: we start with ***planar graphs*** and then move on to ***surface-embedded graphs***. These are graphs that can be drawn without crossings in the plane or in more complicated surfaces, see Figure 1. Therefore they form a ***combinatorial*** object with a ***topological*** constraint. The objective of the course is to ***explore how topology interacts with combinatorics and algorithms*** on this very natural class of objects. Some questions we will explore are:

- What are the combinatorial consequences of being planar? Are there combinatorial characterizations?
- How to test algorithmically whether a graph is planar? If yes, how to draw the graph?
- Can one exploit planarity to design better algorithms for planar graphs than for general graphs?
- What are the other topological two-dimensional spaces?
- How do the previous answers generalize to graphs embedded in these more complicated surfaces?
- How to solve certain topological questions algorithmically on surfaces?

While the focus of the course will stay very theoretical (e.g., mostly with a theorem, lemma, proof structure), embedded graphs are of great interest for the practically-oriented mind, as they appear everywhere, for example in road networks (where underpasses and bridges can be modeled using additional topological features), chip design or the meshes that are ubiquitous in computer graphics or computer aided design. In all these applications, there is a strong need for a theoretical understanding of embedded graphs, as well as algorithmic primitives related to their topological features. Additionally, embedded graphs are an important lens to study graphs in general, since any graph can be embedded on some surfaces. This is especially the

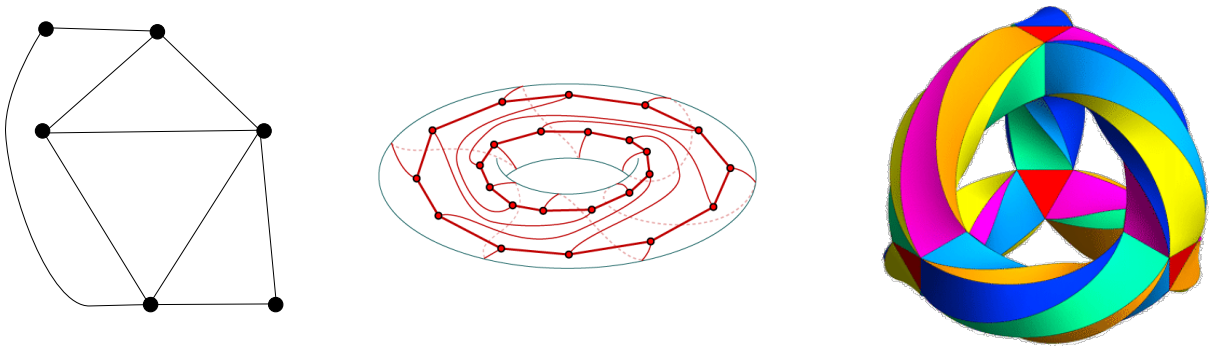


Figure 1: Graphs embedded in the plane, the torus, and the three-holed torus.

case in graph minor theory, where embedded graphs play an absolutely central role. We will barely touch on this topic and refer to the course 2-29-1.

These lecture notes cover (hopefully) closely the material taught in class. This is actually their point, as in my experience lecture notes with too much content can easily get overwhelming (especially when one misses a class). Thus we will refrain from (too many) digressions and heavy references. The tone will be (somewhat) conversational. These lecture notes are heavily inspired by (sometimes verbatim) the lecture notes of the previous iteration of the class(by Éric Colin de Verdière), and those for a course on Computational Topology I co-taught with Francis Lazarus in 2016-2018, and we refer to, and strongly recommend those for the missing digressions and references.

The second half of the course, by Vincent Pilaud, on Geometric Graphs, Triangulations, and Polytopes, has its own set of lectures notes.

1 Planar Graphs

A graph is planar if it can be drawn on the plane so that no two edges intersect, except at their common endpoints. This very simple property has a very strong impact on the ***combinatorics*** of the graph: for example they are sparse (their number of edges is linear in the number of vertices), they can be colored with few colors; as well as on the ***algorithms*** that one can design for these graphs.

1.1 Point-set topology, graphs and embeddings

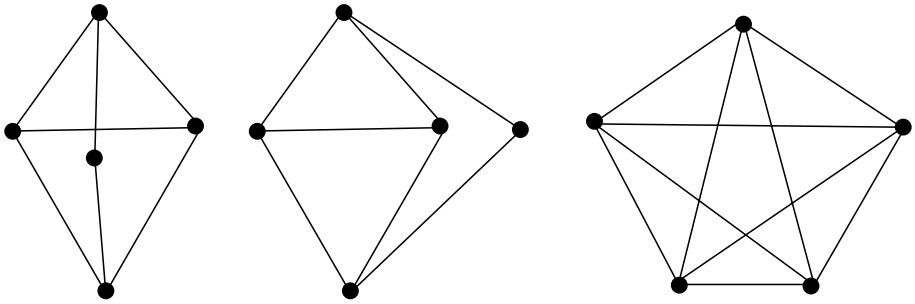
We will use basic notions of point set topology. For the unfamiliar reader, this is like metric topology, but without a metric (!). This is very relevant for because in most cases, we will not care about the Euclidean metric on the plane (or any other), and a vague notion of neighborhood will be enough. A ***topological space*** is a set X with a collection of subsets of X, called ***open sets***, such that:

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

Any metric space gives rise to a topological space by taking as open sets the smallest family of sets containing the open (in the metric sense) balls and satisfying the above axioms. A map f between two topological spaces is ***continuous*** if the inverse image of any open set by f is open. A map f is a ***homeomorphism*** if it is continuous, bijective, and if its inverse f^{-1} is also continuous. Homeomorphic spaces throughout these notes will generally be considered to be *the same*, and thus identified without precaution. Therefore, when we talk about *the* sphere, we talk about any space homeomorphic to the 2-sphere $\mathbb{S}^2 = \{x, y, z \mid x^2 + y^2 + z^2 = 1\}$ with its standard topology.

A ***graph*** $G = (V, E)$ is defined by a set $V = V(G)$ of vertices and a set $E = E(G)$ of edges where each edge is associated one or two vertices, called its ***endpoints***. A ***loop*** is an edge with a single endpoint. Edges sharing the same endpoints are said ***parallel*** and define a ***multiple edge***. A graph without loops or multiple edges is called ***simple***. In a simple graph every edge is identified unambiguously with the pair of its endpoints. A ***path*** is an alternating sequence of vertices and arcs such that every arc is preceded by its origin vertex and followed by the origin of its opposite arc. A path may have repeated vertices (beware that this is not standard, and usually called a ***walk*** in graph theory books). Two or more paths are ***independent*** if none contains an inner vertex of another. A ***circuit*** is a closed path, i.e., a path whose first and last vertex coincide. A ***cycle*** is a simple circuit (without repeated vertices). We will restrict to finite graphs for which V and E are finite sets.

The Euclidean distance in the plane $\mathbb{R}^2$ induces the usual topology where a subset $X \subset \mathbb{R}^2$ is open if every of its points is contained in a ball that is itself included in X. The ***closure*** $\bar{X}$ of X is the set of limit points of sequences of points of X. The ***interior*** $\mathring{X}$ of X is the union of the open balls contained in X. The boundary ∂X is the closure minus the interior. An ***embedding*** of a non-loop edge in the plane is a topological embedding (i.e., an injective map) of the segment $[0, 1]$ into $\mathbb{R}^2$. Likewise, an embedding of a loop-edge is an embedding of the circle $S^1 = \mathbb{R}/\mathbb{Z}$. Such an embedding of the circle is called a ***simple closed curve*** in the plane. An ***embedding*** of a finite graph $G = (V, E)$ in the plane is defined by a bijective map $V \to \mathbb{R}^2$ and, for each edge $e \in E$, by an embedding of e sending $\{0, 1\}$ to e's endpoints such that the relative interior of e (the image of $]0, 1[$) is disjoint from other edge embeddings and



a. b. c.

Figure 2: a. A planar graph (!). b. The same planar graph, this time embedded in the plane, i.e., a plane graph. c. A non-planar graph

vertices[1]. Note that this definition can be (and will be) generalized to other topological spaces than $\mathbb{R}^2$.

A graph is ***planar*** if it has an embedding into the plane, see Figure 2. The data of the planar graph and a specific embedding is called a ***plane*** graph, although it is customary to mix up everything and also call it a planar graph. We will do our best to stray away from this confusion. We consider two plane graphs to be the same if they are homeomorphic, i.e., if there exists a homeomorphism of the plane sending one to the other.

Exercise 1.1. 1. Any graph can be embedded in $\mathbb{R}^3$.

2. A graph can be embedded in the plane if and only if it can be embedded in the sphere. (Hint: prove that the plane is homeomorphic to the sphere with one arbitrary point removed).

The second exercise allows us to jump at will between embeddings in the plane and on the sphere, which we will constantly do.

1.2 The Jordan curve theorem

The main additional structure that a planar graph has, compared to an abstract one, is that it has faces. A ***face*** of a graph embedding is a connected component of the complement of the image (via the embedding) of the vertices and edges of the graph. Then the basic properties of planar graphs follow from the fact that a simple closed curve in the plane is the boundary of exactly two components, one of which is a disk:

Theorem 1.2 (Jordan curve theorem)**.** *Let C be a simple closed curve in the plane. Then its complement has two connected components, one of which is bounded, and each of which has C as a boundary.*

This obvious fact is famously painful to prove. One reason is that simple closed curves can be very pathological, like fractals, see for example Figure 3. In the simpler case that C is polygonal, i.e., a concatenation of segments, the proof is much simpler.

Proof in the polygonal case. : Since C is contained in a compact ball, its complement has exactly one unbounded component. Define the *horizontal rightward* direction $\vec{h}$ as some fixed direction transverse to the all the line segments of C. Rotate the plane so that the direction that we

[1]In other words, this is a topological embedding of the quotient space $(V \sqcup [0,1] \times E)/\sim$, where $\sim$ identifies edge extremities $(0, e)$ and $(1, e)$ with the corresponding vertices.

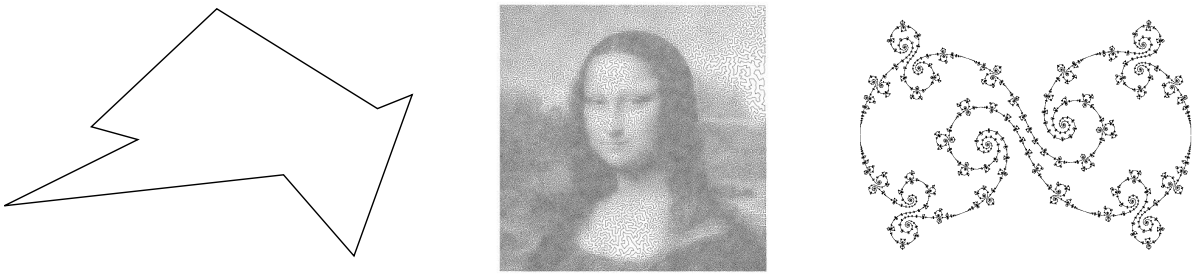


Figure 3: Three simple closed curves in the plane

have chosen is actually horizontal[2]. For every segment s of C we let $\underline{s}$ be the lower half-open segment obtained from s by removing its upper endpoint (rotate slightly the polygon if one of the segments is horizontal). We also denote by h_p the ray with direction $\vec{h}$ starting at a point $p \in \mathbb{R}^2$. We consider the *parity function* $\pi : \mathbb{R}^2 \setminus C \to \{$ even, odd $\}$ that counts the parity of the number of lower half-open segments of C intersected by a ray:

$$\pi(p) := \text{ parity of } |\{ \text{ segment } s \text{ of } C \mid h_p \cap \underline{s} \neq \emptyset \}|$$

Every $p \in \mathbb{R}^2 \setminus C$ is the center of small disk D_p over which π is constant. Indeed, let S_p be the set of segments (of C) that avoids h_p, let S'_p be the set of segments whose interior crosses h_p and let S''_p be the set of segments whose lower endpoint lies on h_p. If D_p is sufficiently small, then for every $q \in D_p$ we have $S_q = S_p$, $S'_q = S'_p$ and the parity of $|S''_q|$ and $|S''_p|$ is the same, see Figure 4 It follows that $\pi(q) = \pi(p)$.

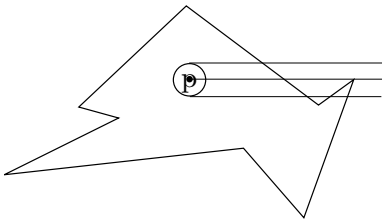


Figure 4: Polygonal Jordan curve theorem

Since π is locally constant, it must be constant over each connected component of $\mathbb{R}^2 \setminus C$. Moreover, the parity function must take distinct values on points close to C that lie on a same horizontal but on each side of C. It follows that $\mathbb{R}^2 \setminus C$ has at least two components. To see that $\mathbb{R}^2 \setminus C$ has at most two components consider a small disk D centered at a point interior to a segment s of C. Then $D \setminus C = D \setminus s$ has two components. Moreover, any point in $\mathbb{R}^2 \setminus C$ can be joined to one of these components by a polygonal path that avoids C: first come close to C with a straight line and then follow C in parallel until D is reached. Finally, it is easily seen by similar arguments as above that every point of C is in the closure of both components of $\mathbb{R}^2 \setminus C$. □

A stronger version of this theorem, which we will not prove either, shows that one of the two components is homeomorphic to an open disk.

Theorem 1.3 (Jordan-Schoenflies Theorem)**.** *Let C be a simple closed curve in the plane. Then its complement has two connected components, one of which is a topological disk, and each of which has C as a boundary.*

[2]This was not explicitly said during the lecture.

One (not that immediate) consequence of this theorem is that in an embedding of a connected graph, all of the faces except one are open disks (or equivalently, on a sphere, all of the faces are open disks). Such a plane graph is called a ***celullar embedding***. This definition is not very interesting for now but will get more important when embedding graphs on other surfaces. Each face can be described by a ***facial walk***, which is the circuit on the graph describing the boundary of the face. Note that a facial walk may have repeated vertices and edges (see Figure 5, left), but this does not happen when the graph is connected enough. A graph G is ***2-connected*** is for any vertex x, $G \setminus x$ is connected.

Lemma 1.4. *In a 2-connected graph, facial walks are cycles of the graphs (i.e., without repeated vertices and edges).*

Proof. Assume otherwise, let F be an open disk bounded by a facial walk that uses the same vertex x twice. Then $F \cup \{x\}$ contains a Jordan curve separating the graph, contradicting the 2-connectedness assumption. □

The ***dual graph*** of a cellular graph embedding $G = (V, E)$ on $\mathbb{S}^2$ is a plane graph G^* defined as follows: put one vertex f^* of G^* in the interior of each face of G, and for each edge e of G, create a dual edge in G^* crossing e and no other edge of G. The dual graph is also cellular, and is, up to homeomorphism, uniquely defined by the above. Note that the dual graph crucially depends on the embedding, and that duality does not preserve simplicity. As the name suggests, duality is an involution: $G^{**} = G$. See Figure 5, right).

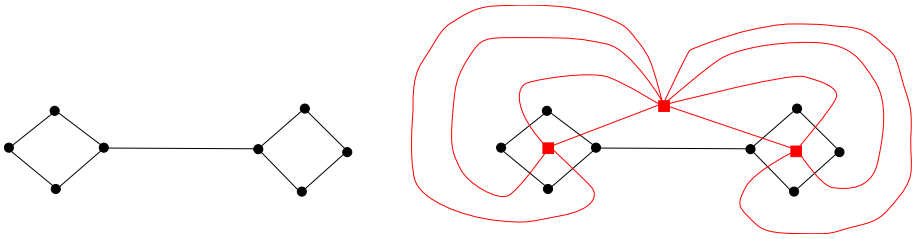


Figure 5: A cellularly embedded graph and its dual

This allows us to think of a plane graph as a collection of open disks glued together with a combinatorial data, which is convenient both for combinatorial and algorithmic purposes.

1.3 Actual planar graphs

When dealing with cellularly embedded graphs in an algorithmic fashion, we have to worry about how the embedding graph is specified to the algorithm. One approach could be to embed each edge as a polygonal curve (which is always doable (somewhat easy exercise)), and specify the coordinates of all the vertices and breaking points of the polygonal edges. This suffers from many issues: it is a very wasteful encoding since at least naively, there is no easy bound on the number of polygonal bends needed on each edge (we will come back to this question later). And it is not at all obvious when two encodings describe the same graph, i.e., two homeomorphic graphs.

Instead, one customary way to describe an embedding of a graph on a sphere is to use a ***combinatorial map***. A combinatorial map associated to a celullar graph embedding G is the set of closed walks in G obtained from walking around the boundary of each face of G. Note that these walks may have repeated vertices or edges. By the previous subsection, each face is a disk. Thus the combinatorial map contains enough information to build an embedding of G

on a sphere abstractly, by taking the abstract graph and gluing a disk on each face. For graphs embedded in the plane, it suffices to mark one face as the unbounded face.

Another approach is to use a ***rotation system***, which is to use as an encoding the cyclic permutation describing the order of the edges around each vertex in the embedding. See Figure 6 for examples of combinatorial maps and rotations systems?

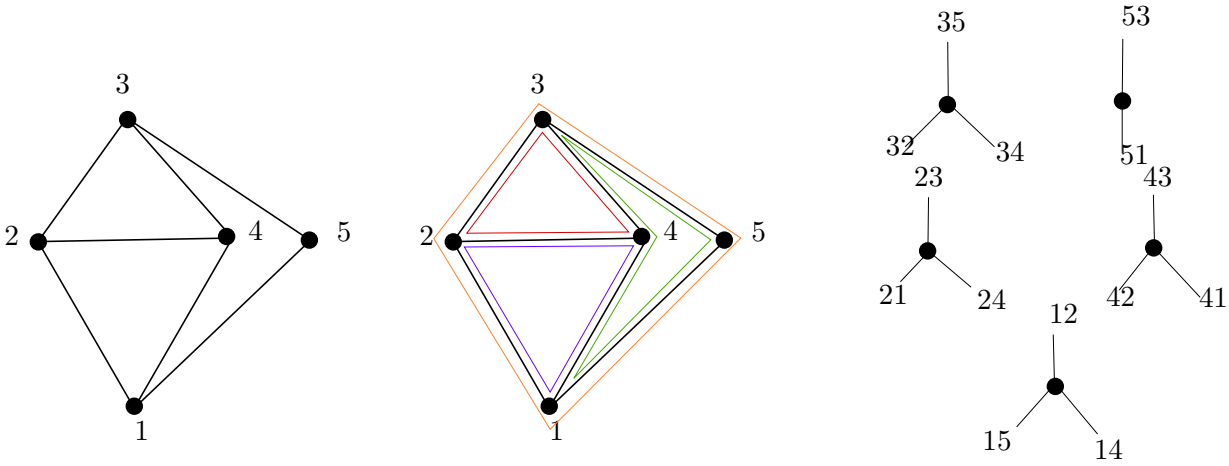


Figure 6: A cellularly embedded graph, its facial walks, and a rotation system describing it.

Exercise 1.5. Think about naively how to compute a rotation system from a combinatorial map and vice versa. Is the complexity any good?

Both approaches are a bit unwieldy for actual implementations, as basic operations like moving around a vertex or a face are overly costly. One more refined data structure is to use flags. A ***flag*** is an incidence between a vertex, an edge and a face of the embedding. Figure 7 shows a set of flags for a planar graph. Flags come with three involutions allowing us to move to a nearby flag:

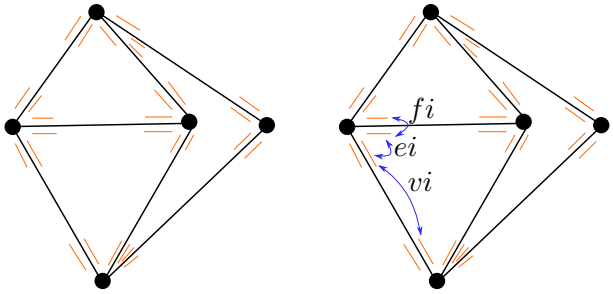


Figure 7: Flags, and the three involutions moving one of them.

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

Note that a flag is not uniquely defined by a triple (v, e, f) where v is incident to e and e is incident to f, as one can see on a graph with a single vertex and a single loop edge.

The ***complexity*** of a plane graph described by flags is the total number of flags involved, which is linear in the number of edges, and thus, as we will see very shortly, in the number of vertices and faces as well. Thus the complexity of a graph and of its embedding is linearly related.

Exercise 1.6. Think about naively how to compute a rotation system and a combinatorial map from a flag representation. Is it better?

Note that in this data structure, vertices, edges and faces are actually not stored. This can be unwieldy for some problems, and in this case, one can extend the data structure so that flags contain pointers to the underlying vertices, edges and faces. With this caveat, flags are versatile enough to handle any intuitive operation on a planar graph with the intuitive complexity. For example, navigating around a vertex v has complexity $O(degree(v))$. Similarly, the dual of a graph is completely straightforward and just amounts to replacing fi with vi and vice versa. Thus, throughout the rest of these notes, and as is common in the literature, the precise translation to switch between an informal description of a navigation in the graph, and the precise procedures with the flags and their involutions, will be swept under the rug.

1.4 Euler's formula

Euler's formula shows that the number of vertices, edges and faces of a graph are related by a linear equation. It has a rich history, of which we will only say that it was likely not first discovered by Euler.

Lemma 1.7. *Let G be a plane graph. If v is a vertex of degree one in G then $G - v$ and G have the same number of faces.*

Proof. We denote by e the edge incident to v in G. Every face of G is contained in a face of $G - v$. Moreover, the relative interior of (the embedding of) e is contained in a face f of $G - v$. Hence, every other face of $G - v$ is also a face of G. It remains to count the number of faces of G in f. Let p, p' be two points in $f \setminus e$. There is a path in f connecting p and p'. This path may intersect e, but we may avoid this intersection by considering a detour in a small neighborhood N_e of e in f (indeed, $N_e \setminus e$ is connected). It follows that p and p' belong to a same component of $f \setminus e$. We conclude that G has only one face in f, so that G and $G - v$ have the same number of faces. □

Theorem 1.8 (Euler's formula)**.** *Let $|V|, |E|$ and $|F|$ be the number of vertices, edges and faces of a connected plane graph G. Then,*

$$|V| - |E| + |F| = 2$$

Proof. We argue by induction on $|E|$. If G has no edges then it has a single vertex and the above formula is trivial. Otherwise, suppose that G has a vertex v of degree one. Then by Lemma 1.7, G has the same number of faces as $G - v$. Note that G has one vertex more and one edge more than $G - v$. By the induction hypothesis we can apply Euler's formula to $G - v$, from which we immediately infer the validity of Euler's formula for G. If every vertex of G has degree at least two, then G contains a cycle C. Let e be an edge of C. We claim that G has one face more than $G - e$. This will allow to conclude the theorem by applying Euler's formula to $G - e$, noting that G has the same number of vertices but one edge less than $G - e$. By the Jordan curve theorem 1.2, C cuts the plane into two components bounded by C. Since $G = C \cup (G - e)$, every face of G is included in the intersection of a face of C and a face of $G - e$. Let f be the face of $G - e$ containing the relative interior of e. Every other face of $G - e$ does not meet C, hence is also a face of G. Since f intersect the two faces of C (both bounded

by e), G has at least one face more than $G - e$. By considering a small tubular neighborhood of e in f, one shows by an already seen argument that $f \setminus e$ has at most two components. It follows that f contains exactly two faces of G, which concludes the claim. □

To illustrate the power of this formula, here is a standard application.

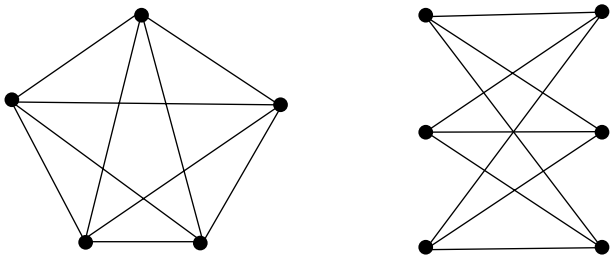


Figure 8: The graphs K_5 and $K_{3,3}$.

Theorem 1.9. *The graphs K_5 and $K_{3,3}$ (see Figure 8) are not planar.*

Proof. In any planar embedding of a graph, each edge is adjacent to two faces (perhaps twice the same). Since K_5 is simple, each face is adjacent to at least 3 edges. In any planar embedding of K_5, the Euler formula would give $|V| - |E| + |F| = 2$, and by the previous two observations, we have $3|F| \leq 2|E|$, therefore $|V| - |E|/3 \geq 2$. But plugging in the values gives $5/3 \geq 2$, a contradiction. For $K_{3,3}$, we observe that any facial walk in a planar embedding would have at least 4 edges because the graph is bipartite. This leads to $|V| - |E|/2 \geq 2$, e.g., $3/2 \geq 2$, another contradiction. □

The following two applications, very close to the proof we have just seen, are absolutely key.

Exercise 1.10. Every simple planar graph G with $n \geq 3$ vertices has at most $3n - 6$ edges and at most $2n - 4$ faces.

Exercise 1.11. Every simple planar graph has a vertex with degree less than 6.

The first one shows that simple planar graphs are sparse, i.e., verify $|V| = O(|E|)$. Therefore, when talking about the complexity of a simple planar graph, we can refer either to its vertices, edges or faces.

The second exercise has the following nice application. A k-coloring of a graph is a map $c : V \to \{1, \ldots, k\}$ so that two adjacent vertices are mapped to different colors.

Proposition 1.12. *Planar graphs are* 6*-colorable.*

Proof. We prove the result by induction on the number of vertices. For low values, this is immediate. For the induction step, pick a vertex z of degree less than 6, and color inductively $G \setminus z$. Then the five neighbors have at most 5 different colors, and we can color z with one of the remaining colors. □

Exercise 1.13. Planar graphs are 5-colorable. Hint: look at paths connecting non-adjacent neighbors of a degree 5 vertex.

As is well known, planar graphs are actually 4-colorable, and we will not prove this in the course.

To conclude, we prove a very strong variant of Exercise 1.10, which allows to quantify how non-planar dense graphs are. A ***drawing*** of a graph if just a continuous map $f : G \to \mathbb{R}^2$, that

is, a drawing of the graph on the plane where crossings *are* allowed. The ***crossing number*** $cr(G)$ of a graph is the minimal number of crossings over all the possible drawings of G. For instance, $cr(G) = 0$ if and only if G is planar. The ***crossing number inequality*** provides the following lower bound on the crossing number.

Theorem 1.14. $cr(G) \geq \frac{|E|^3}{64|V|^2}$ *if* $|E| \geq 4|V|$.

The proof is a surprising application of (basic) probabilistic tools.

Proof. Starting with a drawing of G with the minimal number of crossings, define a new graph G' obtained by removing one edge for each crossing. This graph is planar since we removed all the crossings, and it has $|E| - cr(G)$ edges, so we obtain that $|E| - cr(G) \leq 3|V|$ (Note that we removed the -6 to obtain an inequality valid for any number of vertices). This gives in turn

$$cr(G) \geq |E| - 3|V|.$$

This can be amplified in the following way. Starting from G, define another graph by removing vertices (and the edges adjacent to them) at random with some probability $1-p < 1$, and denote by G'' the obtained graph. Taking the previous inequality with expectations, we obtain $\mathbb{E}(cr(G'')) \geq \mathbb{E}(|E''|) - 3\mathbb{E}(|V''|)$. Since vertices are removed with probability $1-p$, we have $\mathbb{E}(|V''|) = p|V|$. An edge survives if and only if both its endpoints survives, and a crossing survives if an only if the four adjacent vertices survive (There may be less than four adjacent vertices in general, but not in the drawing minimizing the crossing number, we leave this as an exercise to check), so we get $\mathbb{E}(|E''|) = p^2|E|$ and $\mathbb{E}(cr(G'')) = p^4 cr(G)$. So we obtain

$$cr(G) \geq p^{-2}|E| - 3p^{-3}|V|,$$

and taking $p = 4|V|/|E|$ (which is less than 1 if $|E| \geq 4|V|$ gives the result. □

1.5 Kuratowski's Theorem

We say that H is a ***subdivision*** of G if H is obtained by replacing the edges of G by independent simple paths of one or more edges. Obviously, a subdivision of a non-planar graph is also non-planar. It follows from Theorem 1.9 that a planar graph cannot have a subdivision of K_5 or $K_{3,3}$ as a subgraph. It turns out that the converse is also true.

Theorem 1.15 (Kuratowski, 1929)**.** *A graph is planar if and only if it does not contain a subdivision of* K_5 *or* $K_{3,3}$ *as a subgraph.*

This is a striking instance of the interplay between the topology and the combinatorics: our topological constraint can be replaced by a purely combinatorial one, and vice versa. In the following K_5 and $K_{3,3}$ are called forbidden graphs.

Contracting an edge e in a graph G means removing it and identifying its two endpoints. When the graph is embedded, the contraction can be done while preserving the embedding. We denote this operation by G/e. A 3-connected graph is a graph that stays connected when one removes an arbitrary pair of vertices.

Exercise 1.16. Let G be a graph such that G/e contains a subdivision of K_5 or $K_{3,3}$. Then G also contains a such a subdivision. (Hint: this might not be the same forbidden graph!).

The proof of Kuratowski's theorem is very nice and clean in the 3-connected case, and requires somewhat tedious machinery to reduce to that case. This machinery is summarized in the following two lemmas, whose proof is skipped in class.

Lemma 1.17. *Any 3-connected graph G with at least five vertices contains an edge e such that G/e is 3-connected.*

Lemma 1.18. *Let G be a graph with at least four vertices, containing no subdivision of K_5 nor of $K_{3,3}$ and such that the addition of any edge between non-adjacent vertices creates such a subdivision. Then G is 3-connected.*

Proposition 1.19 (Kuratowski's theorem for 3-connected graphs)**.** *A 3-connected graph G without any subgraph isomorphic to a subdivision of a forbidden graph is a planar graph.*

Proof. We prove the contrapositive: any non-planar graph contains a subdivision of K_5 or $K_{3,3}$ as a subgraph. In order to do so, let us pick G a contraction-minimal non-planar graph, i.e., a graph such that for any edge, contracting the edge e yields a planar graph. We prove that G contains a subdivision of K_5 or $K_{3,3}$ as a subgraph, which proves the theorem, since any non-planar graph contains a contraction-minimal non-planar graph as a subgraph, and the Exercise 1.16 guarantees that this property will be verified after uncontracting.

Let us pick an edge $e = xy$ of G so that G/e is 3-connected, as guaranteed by Lemma 1.17. Then G/e is planar. In the graph G/e, e corresponds to a single vertex z, and $G/e \setminus z$ is 2-connected. Hence the face that contains z is a cycle C of the graph, by Lemma 1.4. Now we try to uncontract z while preserving planarity: this will fail either if x and y have at least three adjacent vertices in common, see Figure 9, left, or if x and y are adjacent to four vertices that *interleave*, that is, if there exist vertices x_1, x_2 adjacent to x, and y_1 and y_2 adjacent to y, that are cyclically ordered as x_1, y_1, x_2, y_2 on C, see Figure 9, right. In the first case, G contains a subdivision of K_5, and in the second case it contains a subdivision of $K_{3,3}$. □

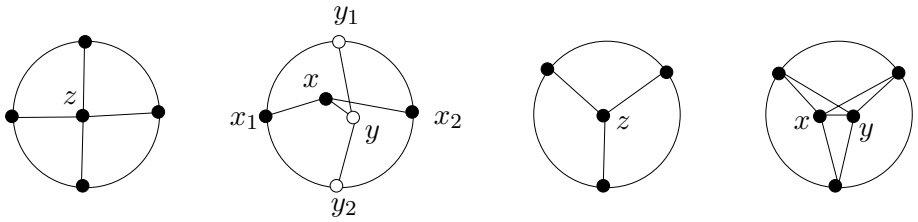


Figure 9: The two cases to find subdivisions of forbidden graphs.

Now for the missing proofs, in a quite hasty style:

Proof of Lemma 1.17. Let $e = xy$ be an edge of G. If G/e is not 3-connected, then G contains a vertex z such that $G\setminus\{x, y, z\}$ is disconnected. Let G_1 be the smallest component of $G\setminus\{x, y, z\}$. Then z is joined to G_1 by an edge $e_1 = zu$. If G/e_1 is not 3-connected, then there is a vertex v such that $G \setminus \{z, u, v\}$ is disconnected, and since $G \setminus z$ is connected, the smallest component is a subgraph of G_1. So we continue until we get the edge that we are looking for. □

Proof of Lemma 1.18. We pick a minimal counter-example: a smallest graph G with at least 4 vertices that is not 3-connected, containing no subdivision of a forbidden graph and such that the addition of any edge between non-adjacent vertices creates such a subdivision.

We claim that G is 2-connected. Otherwise, there is a cut vertex x, adjacent to vertices y_1 and y_2, each on one side. Adding an edge between y_1 and y_2 creates a subdivision K of a forbidden graph. Since K_5 and $K_{3,3}$ are both 3-connected, all the vertices of degree at least 3

of K are on the same side. Then the only way for K to use the other side is if a path uses the edge y_1y_2 and comes back via x, but it might as well use the edge y_ix, and so K was already present in G.

Now, if G is not 3-connected, there are two vertices x and y disconnecting it. We claim that the edge xy must be in G. Otherwise, adding it creates a subdivision K of a forbidden graph, and by the same previous argument, all of its vertices of degree at least 3 must be on the same side. Then we can reroute a path going through the other side with the edge xy, and so K was already present in G.

So now we are in the case where $\{x, y\}$ disconnects G into G_1 and G_2 and the edge xy is in G. Adding some edge in G_i creates a subdivision of a forbidden graph, which, with the same arguments as before, is contained in G_i. By minimality of G, G_i is 3-connected or has at most three vertices. By Proposition 1.19, they are both planar, pick one of their embeddings. Let $z_i \neq x, y$ be a vertex of a face F_i of G_i bounded by xy. Note that F_i must be a triangle, as otherwise one could add a diagonal within the face, preserving planarity and thus also the fact that it contains no subdivision of a forbidden graph. Then adding z_1z_2 creates a subdivision of a forbidden graph K.

At this stage, in the graph $G + z_1z_2$, there are now three vertices connecting G_1 and G_2, so it might be that the vertices of degree at least 3 of K are not all contained in the same side. Actually, this has to be the case, otherwise, some path uses z_1z_2 and either x and y, and thus can be rerouted to z_ix or z_iy. So K has at least one vertex on each side. If K is a subdivision of K_5, then there are four independent paths connecting those two vertices, which is not possible here. So K is a subdivision of $K_{3,3}$, and by a similar argument, five of its vertices are on one side, say G_1 and one on the other side. But then we could as well put that last vertex at some place p inside F_1 and connect it to x, y and z_1. This graph $G_1 + \{px, py, pz\}$ is planar by construction, but contains a subdivision of $K_{3,3}$. This last contradiction concludes the proof. □

A ***minor*** of a graph G is any graph obtained from a subgraph of G by contracting a subset of its edges. If G contains a subdivision of H then H is a minor of G, but the converse is not true. Nevertheless, a similar theorem to Kuratowski's hold for minors:

Theorem 1.20 (Wagner, 1937)**.** *A graph G is planar if and only if none of K_5 and $K_{3,3}$ is a minor of G.*

This theorem is actually implied by Kuratowski's theorem, by observing that a minor of a planar graph is planar, Theorem 1.9, and the above remark. Wagner's theorem admits generalizations to graphs embedded on other surfaces, and in what constitutes a jewel of structural graph theory, to any minor-closed family of graphs.

2 Planarity testing and planar graph algorithms

This section takes a more algorithmic turn than the previous one. We start with planarity testing: given an abstract graph G, how to decide whether it is planar or not? We then continue on this topic by proving the Tutte embedding theorem, providing a very intuitive way to construct a planar embedding. In a second part, we investigate how algorithmic problems become easier when the input graph is planar.

2.1 Planarity testing

2.1.1 A cubic time algorithm

From a naive perspective, it is not obvious at all how to test planarity in polynomial time: guessing a rotation system or the facial walks takes exponential time. Therefore the strength of the following result should come as a surprise.

Theorem 2.1 (Hopcroft-Tarjan)**.** *Given a graph G, one can, in linear time, determine whether G is planar, and, if so, compute a combinatorial map of G in the plane.*

As far as I know, linear-time algorithms are all significantly involved. In this course, we only provide a weaker version of this theorem, with cubic complexity.

The proof starts by reducing the problem to 2-connected graphs: a ***block*** of G is an inclusionwise maximal subgraph of G that is 2-connected.

Lemma 2.2. *A graph G is planar if and only if all its blocks are planar.*

Proof. If a graph is planar, all of its subgraphs, and thus all of its blocks, are planar. Reciprocally, suppose that all the blocks of a graph are planar. Then, we observe that the blocks of a graph are arranged in a tree-like fashion: formally, define the block graph H whose vertices are the blocks and the cut vertices and a block is adjacent to a cutvertex it contains. Then H has no cycle, as otherwise a cut vertex would not be a cut vertex. Therefore we can embed the graph G in the plane, by navigating the tree from the leaves and gluing together planar blocks as they come. □

Here is a very lazy algorithm to decompose a graph into its blocks.

Lemma 2.3. *Given a graph G, we can determine all its blocks in quadratic time.*

Proof. For each vertex x, test whether $G \setminus x$ is connected by a tree traversal. When it is not, split it into its connected components, and continue on each of them with the remaining vertices. Each search takes $O(n)$ time, and there are $O(n)$ vertices, hence the quadratic complexity. □

It is a classical exercise in graph algorithms to do this in linear time, with a single depth-first search, enhanced with some more bookkeeping.

These two lemmas imply that, for the proof of Theorem 2.1, we can assume that the input graph is biconnected.

The algorithm that we describe has its roots in the Jordan curve theorem. Each cycle C in a graph G is embedded as a cycle in the plane, and thus separates the plane into two parts. An edge in $E(G) \setminus E(C)$ connecting two vertices of C, or a subgraph of G induced by a connected component of $G \setminus C$ and the edges attaching this connected component to C, are called *segments*. Each segment needs to be put in one of the two parts. Furthermore, since we assume that the graph is 2-connected, each of these segments is connected to at least two vertices of C (hence the name), which we call their attachments. When these attachments interleave (as in the proof of the Kuratowski Theorem 1.19), they can not be put on the same side. This is formalized in a conflict graph, which we analyze to solve the planarity question.

In order to start doing anything, we first compute a cycle that has at least two pieces.

Lemma 2.4. *In linear time, we can either compute a cycle that has at least two segments, or certify that G is planar.*

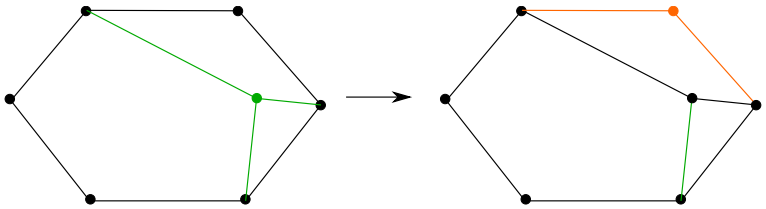


Figure 10: Changing the cycle so as to have at least two segments.

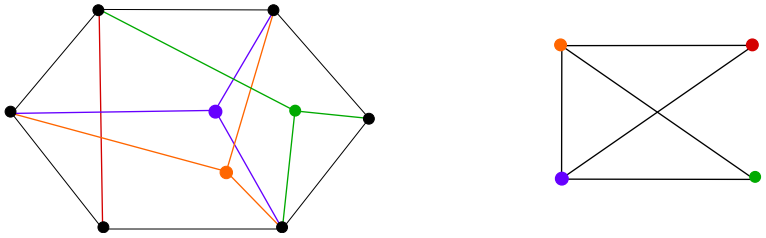


Figure 11: Four segments and their (non-bipartite) conflict graph.

Proof. We start with any cycle C of G, for example from a depth-first search. If $C = G$, G is planar. Otherwise, compute the segments of G. If there are at least two, we are done. If there is a single segment S and it is a path p, then $G \cup p$ is clearly planar. So the piece is not a path, and has at least two attachments $\{v_1, \dots, v_k\}$, where we order them according to their order on C. Then there is a path p in S connecting v_1 to v_2, and we look at the cycle C' made of this path and the subpath of C between v_1 and v_2 containing all the other attachments. This cycle C' has at least two segments: indeed there is at least the path between v_1 and v_2, and the complement $S \setminus p$. See Figure 10. □

Then, for such a cycle C, the conflict graph with respect to C is defined as follows. Each segment corresponds to a vertex, and two segments are in conflict if either they have three attachments in common, or one pair of attachments from each segment alternate on C, see Figure 11. Then if the conflict graph is not bipartite, the graph can not be embedded in the plane since two non-compatible segments will have to be put on the same side of a Jordan curve. The following lemma provides a recursive converse.

Lemma 2.5. *A biconnected graph G with a cycle C is planar if and only if the following two conditions hold:*

1. *The conflict graph with respect to C is bipartite.*
2. *For each segment S with respect to C, the graph $C \cup S$ is planar.*

Proof. The forward direction follows from the previous discussion.

For the opposite direction, the proof is constructive. We start with an arbitrary segment S and, by assumption we know that $S \cup C$ is planar. We order arbitrarily all of the segments $\{S_1, \dots, S_k\}$ which are in the same bipartition class as S, and add them one by one the the planar drawing. Imagine that we have already embedded G and the first i segments. Then we denote the attachments of S_{i+1} by $\{v_1, \dots, v_k\}$, ordered according to the cyclic order along C. By definition of the conflict graph, each of the pieces S_k for $k \leq i$ shares at most two consecutive attachments points with $\{v_1, \dots, v_k\}$, and the remaining attachments are inbetween those two. Say that for example some segment S is attached between v_1 and v_2 (possibly included). Then we can push the embedding of S past a path $p_{1,2}$ connecting v_1 and v_2, except possibly at the endpoints, as in Figure 12. Likewise, the planar embedding of $C \cup S_{i+1}$ given by the

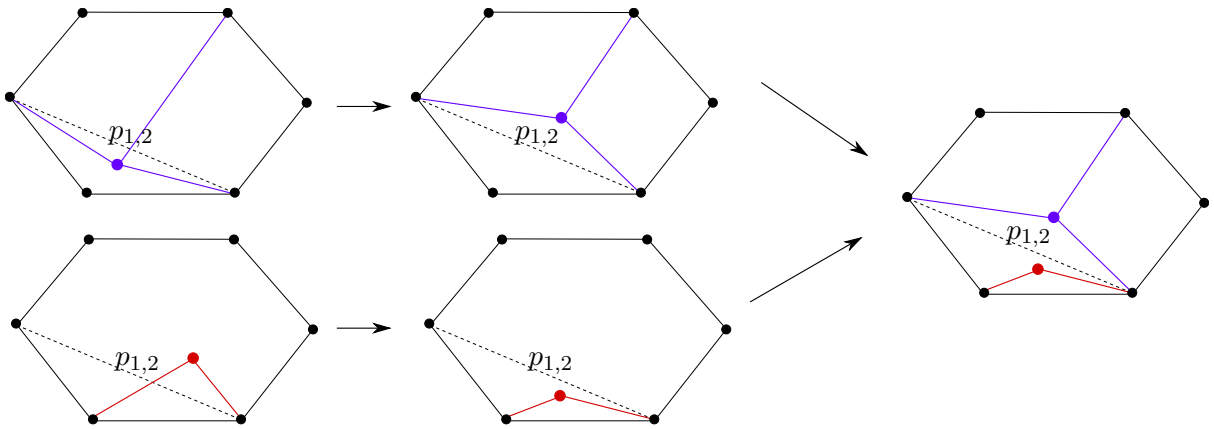


Figure 12: Pushing embedded pieces past a path so as to embed them simultaneously.

hypothesis can be pushed on the other side of all the paths $p_{i,i+1}$, except possibly at the endpoints. Therefore we can glue it next to the embedding of $G \cup \bigcup_{k \leq i} S_i$. The pieces on the other side are glued in the exact same way. □

From Lemma 2.5, we can naturally infer a recursive algorithm.

1. Use Lemma 2.4 to find either a separating cycle or conclude that the graph is planar, then
2. compute the segments with respect to C,
3. compute the conflict graph,
4. check that it is bipartite,
5. and then recursively check that $C \cup S$ is planar for each segment S.

The first step and second step are done in linear time. The third step can be done in quadratic time: for each pair of segments (S_1, S_2), we check whether they have three attachments in common, and if not, whether all of the attachments of S_2 are contained between two attachments of S_1 (possibly with identical endpoints). For each S_1, we can do all these tests in linear time, and thus the total is quadratic. The fourth step is linear in the complexity of the conflict graph, and thus quadratic.

Finally,

Lemma 2.6. *The number of recursive invocations is linear in the complexity of the input graph.*

Proof. Note that during a recursive call of the algorithm, the cycle that is used is never the same as in the parent call, since the cycle C in the parent call clearly cuts $C \cup S$ in a single piece. Therefore, during a recursive call, we can always choose a *witness edge* of C that is not an edge of the cycle in the parent call, actually that edge will always be in a segment S with respect to C. Therefore, two witness edges picked like that in sibling calls will also be different, since each edge is in a different segment of the parent cycle. Likewise, a witness edge never appears as a witness edge in a descendant call of a sibling. Finally, a witness edge e never appears again as a witness edge in a descendant call: walking down the tree, while e belongs to the cycle, by construction it can not be chosen again, and when e stops belonging to the cycle, then it belongs to a segment that is a path and therefore the next call will end the recursion at the application of Lemma 2.4 (it corresponds to the case cycle plus a path).

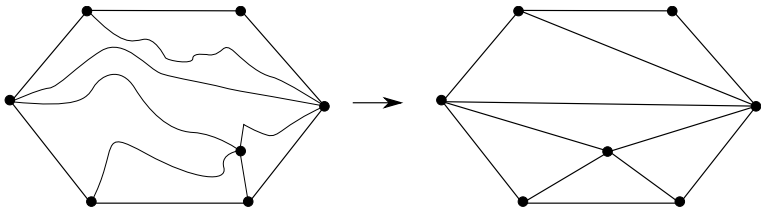


Figure 13: The Tutte magic

Therefore each recursive invocation can be charged to a different edge, and thus there is at most $O(|E|)$ of them. □

Plugging everything together, we get a cubic complexity. With some more work, we can compute a combinatorial representation of the embedding in the same complexity.

2.1.2 The Tutte barycentric embedding

The previous algorithm allows us to test whether a graph is planar, and with some more work, to compute a combinatorial representation of the planar embedding. In this section, we explain a very clean conceptual way to find a nice embedding of a planar graph, with just the knowledge of a facial cycle of the graph. We will assume that the graph is simple and ***triangulated***, that is, that each facial cycle, except possibly the unbounded one[3], is adjacent to three edges. Note that if one is given a planar embedding of a graph, one can add edges inside the faces of degree more than 3, subdividing those until the graph is triangulated. By Lemma 1.18, a triangulated graph with at least 4 vertices is 3-connected[4].

Then the Tutte embedding for a triangulated[5] graph proceeds as follows

1. Pick a facial cycle of G, which will bound the outer face.
2. Nail all its vertices in some strictly convex position on a wall.
3. Let the other vertices move freely, and put springs in the place of the edges, each with some arbitrary positive stiffness.
4. Let the system relax.
5. Enjoy the nice planar embedding of the graph at equilibrium.

This is illustrated in Figure 13.

More formally, denoting the vertices not on the outer face by V_I, we have

Theorem 2.7 (Tutte, 1963)**.** *Let G be a triangulated graph and C be the facial cycle bounding the outer face. Every strictly convex embedding of the vertices of C extends to a unique map $\tau : V \to \mathbb{R}^2$ such that for every internal vertex v, its image $\tau(v)$ is the convex combination of the image of its neighbors $N(v)$ with weights λ_{vw}, for $w \in N(v)$:*

$$\forall v \in V_I, \qquad \sum_{w \in N(v)} \lambda_{vw}(\tau(v) - \tau(w)) = 0. \tag{2.1}$$

[3]There is no consensus on this definition, sometimes the outer one is also forced to be triangular, sometimes not

[4]This lemma was not proved in the lecture. If you prefer, you can think of this proof as working with both the assumptions 3-connected and triangulated (but see also next footnote)

[5]It actually suffices for the graph to be 3-connected for this to work, we do the triangulated case for simplicity.

Moreover, τ induces an embedding of G by connecting the images of every pair of neighbor vertices with line segments.

When one takes symmetric weights $\lambda_{vw} = \lambda_{wv}$, the weights can be interpreted as the stiffness of springs and the system of equations of this theorem is the equilibrium condition of the physical system (obtained for example by differentiating the energy). Even more simply, taking all the weights equal to one puts each vertex that is not on the outer face at the barycenter of its neighbors. The theorem then stipulates that this very intuitive placement yields a planar embedding. In particular, one obtains:

Theorem 2.8 (Fàry-Stein-Wagner)**.** *Every planar graph can be drawn in the plane with straight line edges.*

There are easier proofs of this theorem, but I like this one because of how conceptual it is. The following notes will be alas pretty devoid of pictures because it is very hard to picture things that do not happen.

Throughout the proof, we number the internal vertices from 1 to k and the remaining ones from $k+1$ to n, and to homogenize notation, we consider the weight λ_{ij} for two non-adjacent vertices to be zero (this corresponds to a spring that does not pull). We first look at the system of equations 2.1:

Lemma 2.9. *The system of linear equations 2.1 has a unique solution.*

Proof. The unknowns are the positions of the vertices $\tau(i)$ for i from 1 to k.

(2.1) can be written

$$\Lambda \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_k \end{bmatrix} = \begin{bmatrix} \sum_{j>k} \lambda_{1j}\tau_j \\ \vdots \\ \sum_{j>k} \lambda_{kj}\tau_j \end{bmatrix}$$

where τ_i stands for $\tau(i)$ and

$$\Lambda = \begin{bmatrix} \sum_{j=1}^{n} \lambda_{1j} & -\lambda_{12} & \dots & -\lambda_{1k} \\ -\lambda_{21} & \sum_{j=1}^{n} \lambda_{2j} & \dots & -\lambda_{2k} \\ \vdots & \vdots & \ddots & \\ -\lambda_{k1} & -\lambda_{k2} & \dots & \sum_{j=1}^{n} \lambda_{kj} \end{bmatrix}$$

Beware that the notations used are a bit misleading, as each τ_i is two-dimensional. To conclude, it suffices to prove that Λ is invertible. It follows from the fact that it is diagonally dominant, i.e., each diagonal entry is bigger than the sum of the other entries on the same row. A proof goes as follows: if there exists x such that $\Lambda x = 0$, takes x_i to be one of its components of maximal absolute value. Then the only way for $(\Lambda x)_i = \sum_{j \in N(i)} \lambda_{ij}(x_i - x_j)$ to be zero is that for all j such that $\lambda_{ij} > 0$, we have $x_j = x_i = 0$. By connectivity of G , this extends to all the x_j, hence we conclude that Λ is invertible. □

Note that this theorem is readily algorithmic, as solving this system directly gives the coordinates of the vertices.

For an interior vertex v placed at $\tau(v)$, let $h : \mathbb{R}^2 \to \mathbb{R}$ denote an affine function vanishing on $h(v)$. We say that v is ***h-inactive*** if all its neighbors are on the line $h^{-1}(0)$. Otherwise, it is ***h-active***. If a vertex is active, since it is a barycenter of its neighbors, it can be connected to a vertex with strictly bigger h-value. In turn, that vertex has to be active, and thus we can find a ***rising path*** from v to an exterior vertex, i.e., a path where the h values strictly increase. Similarly, one can find a ***falling path***.

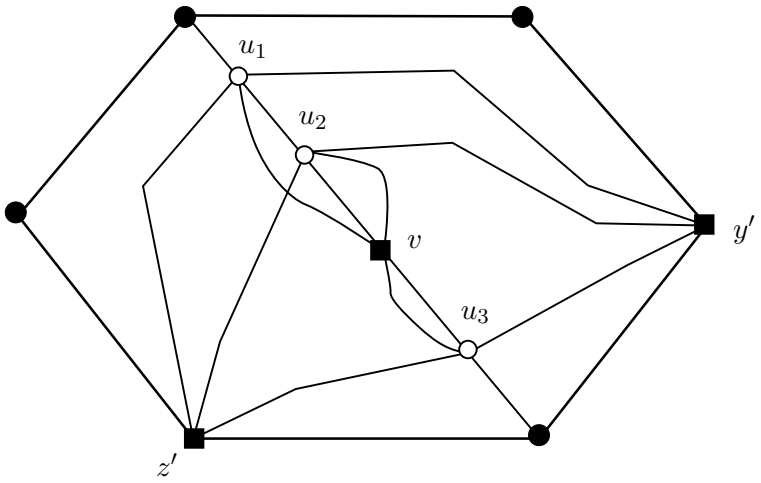


Figure 14: An inactive vertex leads to a $K_{3,3}$ subdivision.

Lemma 2.10. *For any vertex v, $\tau(v)$ is contained in the strict convex hull of the outer vertices.*

Proof. Assume otherwise, then there exists a vertex v and an affine form h so that $h(v) > 0$ but for all the internal vertices w, $h(w) \geq 0$. Pick such a v of maximal h-value. Then all of its neighbors must have the same value, i.e., v is inactive. We continue inductively: all their neighbors must have the same value, until by connectivity we reach an exterior vertex, and a contradiction. □

Lemma 2.11. *For any affine form h, all the interior vertices are h-active.*

Proof. Assume otherwise that some interior vertex v is inactive. Pick a vertex w on the outer face so that $h(\tau(w)) > h(\tau(v))$. Then, by 3-connectedness, there exists three independent paths P_i between v and w. Denote by u_1, u_2 and u_3 the first active vertices on P_i. Note that the three of them satisfy $h(\tau(u_i)) = 0$. Then we take three rising paths from u_i to a vertex y of maximal h-value. While the three rising paths may intersect, it is easy to see (and asked as an exercise just below this proof) that from them, we can always find a vertex y' such that there are three independent paths from u_i to y'. Similarly, we take three falling paths from u_i to a vertex z of minimal h-value, and out of them we can find three independent paths to a vertex z'. Now we look at the paths between v, u_1, u_2 and u_3, as well as the six other paths we have constructed, and recognize a subdivision of $K_{3,3}$, see Figure 14. This contradicts the planarity of the graph, and proves the lemma. □

Exercise 2.12. Let u_1, u_2, u_3 and y be distinct vertices of a graph. Assume that for each i, there exists a path P_i connecting u_i to y which avoids all the other u_j for $j \neq i$. Then there exist three independent paths P_i' connecting w_i to a common vertex y'.

Now that every vertex has been proved to be active, we can prove that τ is an embedding locally:

Lemma 2.13. *Let uvx and uvy be two facial triangles incident to an edge uv of G not in C, then $\tau(x)$ and $\tau(y)$ are each on one side of any line through $\tau(u)$ and $\tau(v)$.*

Note that the lemma does not assume that $\tau(u) \neq \tau(v)$.

Proof. As usual, pick h an affine form whose kernel goes through $\tau(u)$ and $\tau(v)$. Pick two rising paths from $\tau(u)$ and $\tau(v)$ to an exterior vertex. (A subset of) These two rising paths and the

edge uv, plus possibly some exterior edges, give rise to a cycle K for which all the h-values are positive. As in the proof of Lemma 2.10, all the vertices interior to K have positive h-value. In particular, one of x or y has positive h-value. Doing the same with falling paths, we get that one of x or y has negative h-value, which proves the lemma. □

Say that a facial triangle is non-degenerate if its embedding has positive area. From this lemma, we conclude that all the facial triangles are non-degenerate. Indeed, this is clear for the triangles adjacent to outer edges, and this lemma shows that a triangle adjacent to a non-degenerate triangle is also non-degenerate. Therefore the claim follows by connectivity.

We can now conclude the proof of Theorem 2.7:

Proof. We have now proved that all the facial triangles are non-degenerate. Let us prove that the embeddings of two different triangles have distinct interiors. If not, let p be a point contained in two triangles t and t'. Throw a ray from p that avoids all the vertices $\tau(v)$. This ray exits t at some edge, where it enters a different triangle t_2 (by Lemma 2.13)). Continuing like that until we hit the outer face, we have a sequence $t, t_2, \dots, t_n$ where t_n is adjacent to the outer face. Likewise, the ray crosses t' and induces a sequence of triangles $t', t'_2, \dots, t'_m$ where t'_m is adjacent to the outer face. But the ray crosses the outer face at a given point, where there is a unique triangle incident to the outer edge. So $t_n = t'_m$, and then, following the ray back, we see that all of the triangles in the sequence are equal, including t and t'. □

A nice corollary of this theorem is that we can always embed a planar 3-connected graph so that its faces are convex.

2.2 Efficient algorithms for planar graphs

Many algorithmic problems can be solved faster when the input graph is planar. This includes some NP-hard problems in general which can be solved in polynomial time in the planar case: for example Max-Cut, (uniform) Sparsest Cut, Feedback arc set, computing the Branch-width if a graph. We will not study any of those (choices have to be made). Instead, we will look at a few problems where planarity allows for faster and conceptually simpler algorithms than in the general case, but not that dramatically so. At the risk of overly simplifying things, there are in my opinion two main reasons as to why planar graphs tend to be easier to handle algorithmically. The first reason is *duality*: sometimes a problem that is not easy to solve in the primal graph becomes much easier to graph in the dual graph. Sometimes juggling between both the primal and the dual setting allows to make good progress. The second reason is the existence of *small separators*, which allows for efficient divide and conquer approaches for a lot of problems. Many proofs of the existence of these small separators (as ours does) rely on duality, so arguably there is only one reason here.

2.2.1 Minimum spanning trees

Let G be a graph with a weight function $w : E \to \mathbb{R}^+$. A ***minimum spanning tree*** of a graph $G = (V, E)$ is a tree $T = (V, E')$ with $E' \subseteq E$ that spans all the vertices of G (this was actually already implied by the notations of the vertices) and such that it has minimal total weight $\sum_{e \in E'} w(e)$ among all the spanning trees. Computing a minimum spanning tree is a fundamental primitive in algorithm design, and also an important practical problem in its own right: think about an electric company wanting to wire all the houses in a neighborhood at a minimal cost. For general graphs with n vertices and m edges, classical algorithms (e.g., Prim's or Kruskal's) run in time $O(m \log n)$ (note that in the sparse case, $m = O(n)$, but this is still

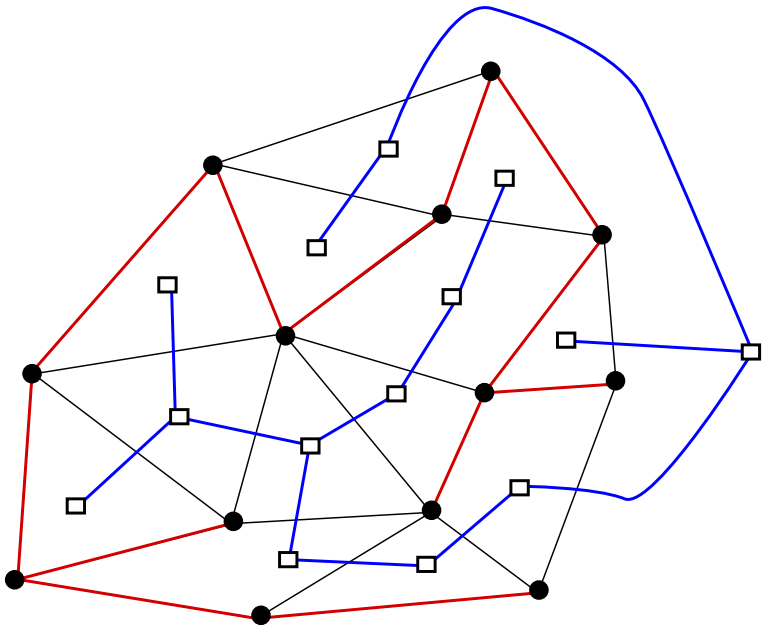


Figure 15: A spanning tree and the spanning co-tree. These are sometimes called ***interdigitating*** tree.

not linear). Using some very fancy data structures, one can do better, but there is no known deterministic algorithm running in linear time.

In contrast, for planar graphs, one can compute a minimum spanning tree very easily in linear time. The key is the use of duality:

Theorem 2.14. *If G is a planar graph with n vertices, one can compute a minimum spanning tree in time $O(n)$.*

We start with exploring how spanning trees interact with duality, as illustrated in Figure 15.

Lemma 2.15. *Let G be a planar graph and G^* be its dual graph. Then if $T = (V, E')$ and $E' \subseteq E$ is a spanning tree, then $T^* = (F^*, (E \setminus E')^*)$ is also a spanning tree, called the co-tree. If one is minimal, the other is maximal.*

Proof. If T is a spanning tree, it does not contain any cycle, which happens if and only if its complement $\mathbb{R}^2 \setminus E'$ is connected, by the Jordan curve theorem. But $\mathbb{R}^2 \setminus E'$ is connected if and only if T^* is connected. Indeed, any two points in $\mathbb{R}^2 \setminus E'$ are in faces of G, and thus can be connected without crossing edges of E' if and only if those faces can be connected in the dual graph where one removed the edges dual to those of E'. Since T and T' are both connected, they are both acyclic and thus are both trees. If one of them, say T, is minimal, then T^* is maximal, since otherwise one could increase the weight of T^* which would mechanically decrease the weight of T. □

Exercise 2.16. Prove that any tree with v vertices and e edges satisfies $v - e = 1$. Deduce from Lemma 2.15 a new proof of Euler's formula.

Observe that when we *contract* an edge in a graph, the corresponding edge in the dual graph gets *removed*, and vice-versa.

The following is an easy consequence of Euler's formula.

Lemma 2.17. *In a planar graph G, there is either a vertex of degree at most 3, or a face of degree at most 3.*

Proof. Assume otherwise. Then we can double count edges in two different ways and get $4v \leq 2e$, and $4f \leq 2e$. Then $v - e + f \leq 0$ contradicting Euler's formula. □

We now have all the tools to describe our algorithm.

Proof of Theorem 2.14. The algorithm is based on alternating actions between the primal and the dual graph. We initialize the set of edges E' that we pick at the empty set, and:

- Let v be a vertex of the graph G. If v is only surrounded by loops, then the solution is the trivial empty tree. Otherwise, at least one non-loop edge is adjacent to v. Among all these non-loop edges, one of minimal weight e necessarily belongs to the minimal spanning tree: otherwise, one could add it and remove another edge. So we can take it, add it to E' and recurse on G/e.
- Let f be a face of the graph G, and thus a vertex of the dual graph G^*. If all the edges adjacent to f in G^* are loops, then G^* has a single face, thus G is a tree, and the spanning tree is G itself. Otherwise, the dual of an edge e of maximal weight incident to f necessarily belongs to the maximal spanning co-tree, and thus e does not belong to the minimum spanning tree. So we can recurse on $G \setminus e$.

Each of these two actions removes an edge in one way or another from the graph that we consider, so the number of recursions is $O(n)$. For each of the two actions, the cost of finding which edge to contract or remove is of the order of the degree of the vertex or the face that we consider. So if we always pick a vertex or an edge of degree at most 3 (provided by Lemma 2.17), this will take constant time. Note that contracting (respectively removing) an edge adjacent to a vertex (respectively a face) of constant degree means updating $O(1)$ flags in the representation, so the recursive call can be made in constant time.

So there remains to explain how to find the vertex or face of low degree provided by Lemma 2.17 in constant time. We will *amortize* this search, i.e., we will prove that the total time spent looking for these is $O(n)$, and thus it is $O(1)$ in average per round. In order to do that, we first compute a list L containing all the vertices and the faces of degree 3 of the initial graph. This takes linear time by traversing both the primal and the dual. The list L will be updated throughout the algorithm so that it always contains the list of vertices or faces that *may* have degree at most 3. When contracting or removing an edge, the degree of the four adjacent vertices and faces can change, so we add them all the list L. Since there are $O(n)$ iterations, this process adds $O(n)$ elements to the list L throughout the algorithm.

Now, whenever we want to take an action, we look at the first element of the list L. If it does not exist anymore, or if it has degree more than 3, we remove it from the list, and continue. By Lemma 2.17, we always end up finding a vertex or a face. Since $O(n)$ vertices and faces were added to the list, we remove $O(n)$ of those throughout the algorithm. So in total, the search procedure takes $O(n)$ time, which proves our amortized complexity and finishes the proof of the algorithm. □

2.2.2 Planar separators

A ***(balanced) separator*** of a graph is a subset of the vertices $S \subseteq V$ so that each component of $V \setminus S$ contains at most two thirds of the vertices of the graph. The size of the separator is

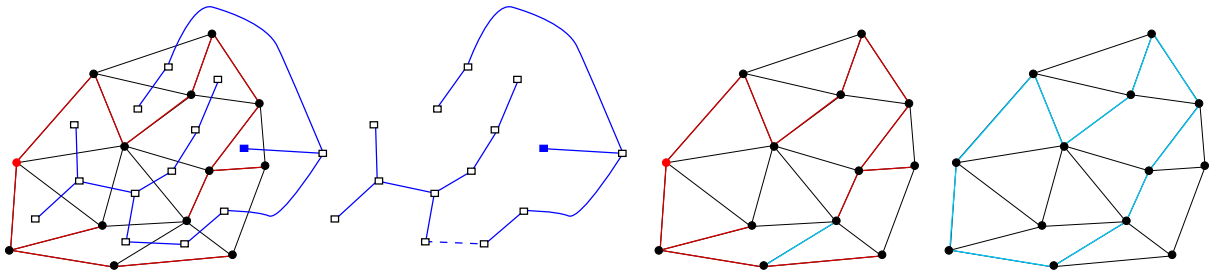


Figure 16: An edge separating the dual co-tree into balanced parts induces a balanced separator in the primal graph.

the number of vertices of S. Separators of small size are key to algorithm design, as they allow for very intuitive divide and conquer algorithms: cut the graph into two parts of roughly equal size, solve your problem recursively on both sides and glue back the solutions. Hopefully the gluing back is facilitated by the small size of the separators. We shall see a specific example of that approach.

Here we prove that planar graphs have (somewhat) small separators.

Theorem 2.18. *Let G be a planar graph with n vertices. Then one can compute in $O(n)$ time a separator for G of size $O(\sqrt{n})$.*

This is tight, as one can see by looking at a $\sqrt{n} \times \sqrt{n}$ grid.

Proof of Theorem 2.18. In the proof, we will be intentionally vague on the constants. We can safely assume that the graph G is triangulated: one can clearly add edges until the graph is triangulated, which takes linear time, and then a separator for the new graph is also a separator for the old graph.

We start from an arbitrary vertex r and compute a breadth-first search tree T rooted at r. The ***level*** of a vertex is the distance from that vertex to the root in T. Denote by L_i the set of vertices of level i. For any edge e that is not in T, denote by C_e the cycle induced by the edge e and the tree. We first prove that one of the levels L_i is a separator, and that one of the cycles C_e is a separator.

- At least one the L_i is a balanced separator. Indeed, if we sort all the vertices based on their level and look at the level L_m containing the $n/2$th vertex, it is necessarily a balanced separator.
- At least one of the C_e is a balanced separator. Indeed, look at the co-tree C corresponding to T. Root C at the leaf of your choice. Below that leaf, the tree C is a binary tree because G is triangulated. We walk down from the root, always picking the edge going towards the larger subtree. We stop when we reach the first vertex v for which the subtree has less than $2n/3$ dual vertices. Then the edge vw (where w the the parent of v) cuts the tree into two subtrees, each of which contains at most $2n/3$ dual vertices. Therefore, the cycle $C_{(vw)^*}$ separates G into two disks, each containing at most two thirds of the faces. From Euler's formula and the fact that the graph is triangulated it follows that this cycle is a balanced separator. See Figure 16 for an illustration.

Both of these balanced separators can be computed in time $O(n)$. If one of them is short, i.e., has size $O(\sqrt{n})$, we win. Otherwise, let us assume that both L_m and $C_{(vw)^*}$ are long. Then in particular the depth of the breadth-first search tree is bigger than $\sqrt{n}$. We look at at the

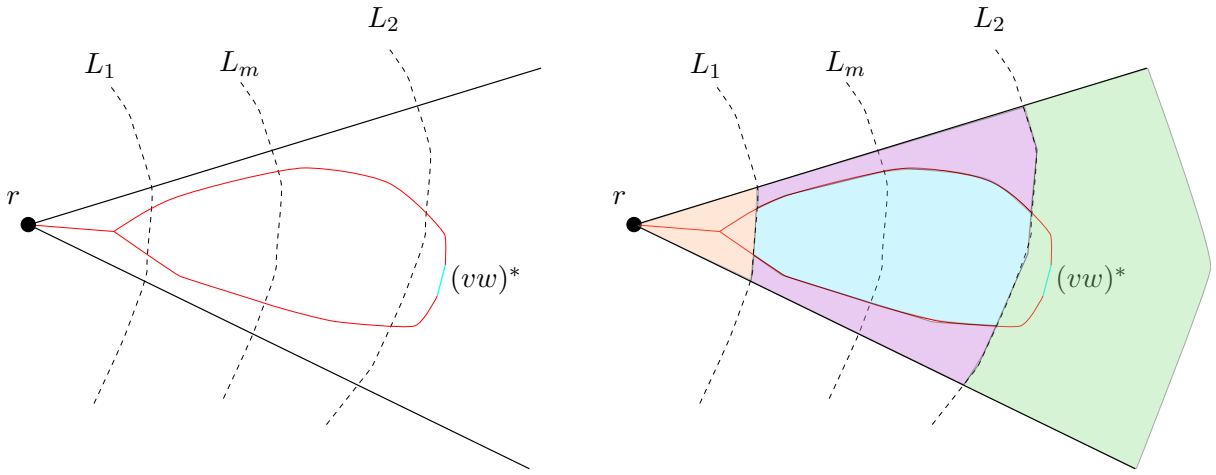


Figure 17: The separator is made of L_1, L_2, and the subpaths of the fundamental cycle between them. It cuts the graph into at most four small components.

levels L_i for i between m and $m+\sqrt{n}$. Since there are $\sqrt{n}$ levels, one of the levels has less than $\sqrt{n}$ elements, and we call it L_2. Likewise, looking at the levels L_i for i between m and $m-\sqrt{n}$, we find a level L_1 with at most $\sqrt{n}$ elements. The cycle $C_{(vw)^*}$ connects L_1 to L_2 with two paths α and β, which both have length $O(\sqrt{n})$. We claim that $L_1 \cup L_2 \cup \alpha \cup \beta$ is the balanced separator that we have been looking for. First, it is short. Second, it cuts G into at least four connected components (see Figure 17):

- The component between the root and L_1 has at most $n/2$ vertices since it is contained in one of the regions cut by L_m.
- The component after L_2 has at most $n/2$ vertices since it is contained in one of the regions cut by L_m.
- There are two components between L_1 and L_2, separated by α and β. Each of them has at most $2n/3$ vertices since they are contained in one of the components cut by $C_{(vw)^*}$.

All the steps in this proof are readily algorithmic and can be implemented in $O(n)$ time.

□

Here is a simple algorithmic applications of this theorem, using a divide and conquer framework.

Theorem 2.19. *In a planar directed*[6] *graph, one can find the shortest cycle in time* $O(n^{3/2})$.

The naive algorithm in general graphs consists in computing a shortest path tree at each vertex, which costs $O(n^2)$.

Proof. When one cuts the graph using planar separators, the shortest cycle is either on one side, or on the other side, or it crosses the separator. The first two cases can be handled recursively, and the last case can be handled with $O(\sqrt{n})$ computations of a shortest path tree. More precisely, the algorithm proceeds as follows:

1. Find a planar separator S cutting G into two balanced subsets A and B,

[6]The same problem for undirected graphs is made quite a bit easier because there is always a cycle of length at most 5.

2. Recursively search A and B,

3. For each s in S, compute the shortest cycle through s using a breadth-first search tree,

4. Return the shortest of the cycles output in steps 2. and 3.

Step 3 takes $O(n^{3/2})$ time, and this dominates the complexity of the algorithm, even in the recursive calls. □

Similarly, but with much more work, any problem involving shortest paths can be sped up on planar graphs using planar separators. In particular, a theorem that we will not prove is the following:

Theorem 2.20 (Henzinger, Klein, Rao, Subramanian 1997)**.** *On an edge-weighted planar graph, a shortest path tree from any given vertex can be computed in linear time.*

2.3 Minimum cut

Our last foray into algorithms for planar graphs concerns the computation of a minimum cut: given two vertices s and t, called ***terminals***, on a planar graph $G = (V, E)$, we want to compute the minimum set of edges X so that removing X from E separates s and t.

Theorem 2.21. *Let $G = (V, E)$ be a cellularly embedded edge-weighted planar graph, and s and t be two distinct vertices of G. Then the problem of computing a minimum $s - t$-cut of G can be solved in $O(n \log n)$ time.*

The basic idea of an efficient algorithm for min-cut on planar graphs is to look at it through the lens of duality: a cut on the primal graph separating s and t dualizes to a cycle in the dual graph separating the faces dual to s and t.

Proposition 2.22. *$X \subseteq E$ is an (s, t)-cut in G if and only if X^* contains the edge set of some cycle of G^* separating s and t.*

This proposition is considered obvious pretty much anywhere, but we will prove it, if only to emphasize that it does not hold on other surfaces (as usual, we will use the Jordan curve theorem).

Proof. The reverse direction is straightforward: if X^* contains a cycle of G^* separating s and t, then any path in G between s and t must cross this cycle, and thus X is an (s, t)-cut.

For the forward direction, we take X an (s, t)-cut in G, and choose C to be an inclusionwise minimal subset of X that is also an (s, t)-cut in G. We show that C^* is a cycle containing s and t. By minimality of C, each vertex of G can be connected to either s or t without taking edges of C, and the two cases are exclusive. We label vertices with "S" or "T" depending on which one they are connected to. Moreover, for any edge in C, its two endpoints cannot have the same label, and for any other edge, its endpoints have the same label. So if we look at a face f of G adjacent to an edge of C, the labels on the facial walk on the face alternate only when the edge is in C, and thus there is an even number of edges of C adjacent to f. So C^* is an ***Eulerian*** subgraph of G^*, that is, a subgraph where each vertex has even degree. Pick any cycle in that subgraph. By the Jordan curve theorem, it separates s and t. By minimality, C^* is that cycle. □

This proposition transforms a *combinatorial* problem into a *topological* one, namely, finding in the dual graph the shortest cycle enclosing a given face and not another given one. Even more topologically, if we *remove* the faces s^* and t^* from the dual graph, we obtain a surface homeomorphic to an annulus, and we look for the shortest cycle that goes around this annulus. However, this runs into an interesting technical issue: is s^* and t^* are adjacent, then removing s^* and t^* does not actually yield an annulus. Morally, this should not pose a problem: we just want to add an infinitesimally small buffer between s^* and t^* and we will be fine. One way to do this is to enlarge a tiny bit the curves that we look at. When working on the primal graph, we generally work with walks on the primal graph. When working on the dual graph, we work with walks on the dual graph, which correspond by duality to closed curves that are in ***general position*** with respect to G, i.e., they do not meet the vertices of G and cross the edges of G transversely. So our solution is to directly work in this setting of curves in general position with respect to G. The length of such a curve is defined to be the number of edges of G that it crosses. Note that this is a bit more general than just looking at walks on the dual graph: now we can define a pair of small curves in general position around s and t that are disjoint, even if s and t are adjacent in G. Yet from an algorithmic perspective, all the curves in general position can be pushed on the dual graph in a way that does not change the length, so any computation, for example shortest paths, can be made in the dual graph. In this new setting, Proposition 2.22 becomes:

Proposition 2.23. *Let γ be a simple closed curve in general position with respect to G, that separates s from t and that has minimal length among all such curves. Then the set of edges crossed by γ is a minimum (s,t)-cut in G.*

Recall that a simple closed curve on a sphere is an injective map $\gamma : \mathbb{S}^1 \to \mathbb{S}^2$.

Proof. Any path connecting s to t in G crosses γ, thus the set of edges crossed by γ is an (s,t)-cut. Conversely, a minimum (s,t)-cut dualizes to a cycle in G^* separating s and t, which corresponds to a simple closed curve γ in general position with respect to G, that separates s from t. □

Now, we remove a small disk around s and t, getting an annulus A, and a portion of the graph G embedded on this annulus as in Figure 18,. How do we compute a shortest closed curve that goes around the annulus A? We can fix one point on each boundary, and compute a shortest path between these two points, and call it p. Then we show that some shortest closed curve does not cross p more than once, and thus can be found by a shortest path computation.

Lemma 2.24. *Some shortest closed curve separating the two boundaries of A is simple and crosses p exactly once.*

Proof. This is illustrated in the left of Figure 19. Let γ be such a curve. If we cut A along p, we obtain a topological disk D, where the path p got cut into two paths p' and p'' on the boundary. The curve γ, once cut on D, contains a simple path q connecting p' to p'', otherwise γ would not separate s from t. Now we reglue D along p, and connect the two endpoints q_1 and q_2 of q by running parallel to p. We obtain a closed curve that is simple, crosses p exactly once and is no longer than γ since γ was connecting q_1 to q_2 as well. □

From this we get a naive quadratic algorithm. We compute the shortest path p, which has some length k. We pick points $v_0, \dots, v_k$ on this path such that the subpath $[v_i, v_{i+1}]$ has length one. Then, cutting along p, each vertex v_i gets duplicated into v_i' and v_i'', and we compute all shortest paths between each v_i' and v_i''. Following all the lemmas, one of them is the dual of a

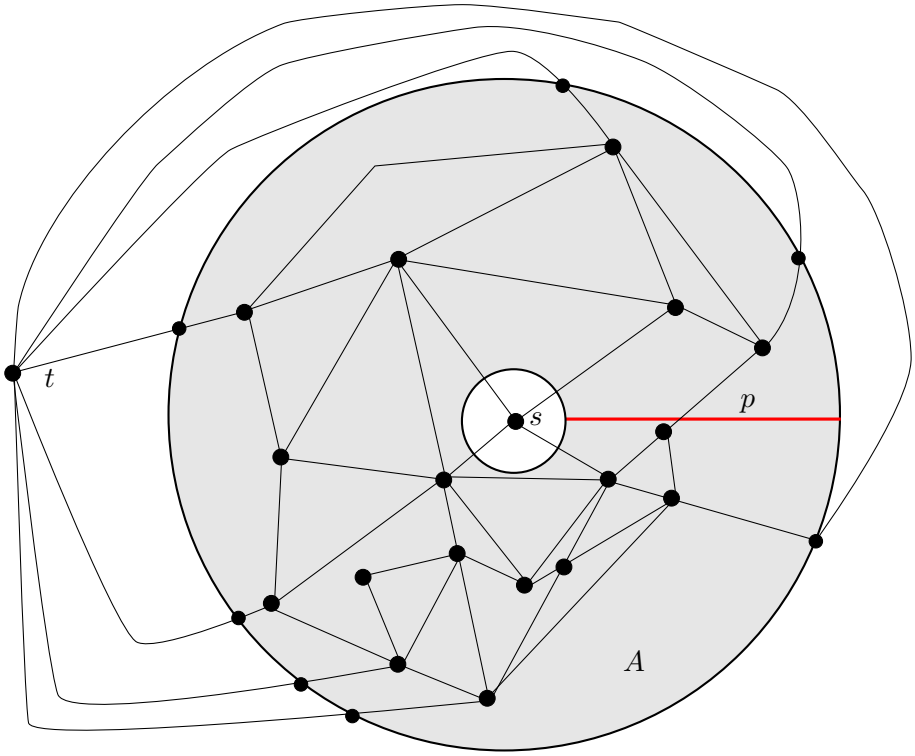


Figure 18: The annulus obtained after removing a small disk around s and t.

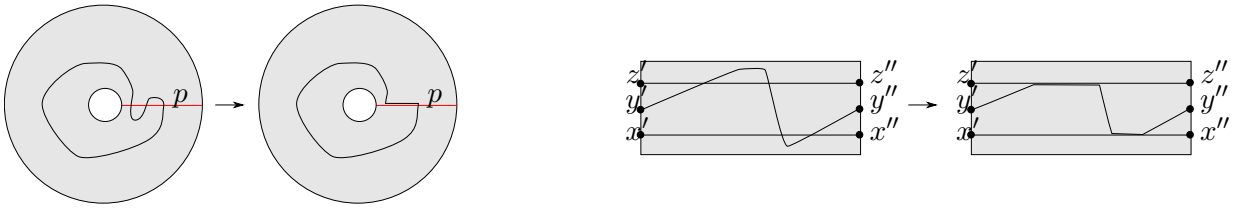


Figure 19: The two shortcutting arguments of Lemma 2.24 and Lemma 2.25.

minimal cut. Since shortest paths in the dual graph can be computed in linear time (cf. the unproved Theorem 2.20), this takes $O(n^2)$ time.

We can speed this up doing some divide-and-conquering.

Lemma 2.25. *Let (x, y, z) be points on p, appearing in this order. When cutting A along p, these get duplicated into (x', y', z') and (x'', y'', z''). If γ_x and γ_z are disjoint shortest paths between x' and x'', respectively z' and z'', then some shortest path between y and y' does not cross γ_x nor γ_z.*

Proof. This is illustrated in the right of Figure 19. Say that γ_y crosses γ_x, then it crosses it in at least two points. Let a and b be the first and the last crossing points when going from y' to y''. Then we can replace whatever γ_y was doing between a and b with the subpath of γ_x between a and b (or more precisely, some shortest paths infinitesimally close to it). Since γ_x is a shortest path, the new curve is at most as along as γ_y. Doing the same for the crossings with γ_z concludes the proof. □

This suggest the following recursive approach. Having computed our shortest path p of length k between the two boundaries of A, if $k > 2$,

1. we pick a vertex $v := v_{\lfloor k/2 \rfloor}$, cut along p and compute on D a shortest path p between the two vertices v' and v'' corresponding to v,

2. we reglue D into A, and the shortest path p is a simple closed curve γ. We cut A along γ and get two annuli A_1 and A_2.

3. we recurse on A_1 and A_2 and output the shortest of the two solutions.

There are two base cases for the recursion, when $k \leq 2$, we can compute the shortest closed curve by brute forcing the problem in $O(n)$ time as in the quadratic algorithm. Similarly, if there exists a face adjacent to both boundaries, we can compute a shortest cycle going through that face directly in time $O(n)$.

Here again, this algorithm would be quite a bit more annoying to describe purely in the dual graph, as the shortest path p might be following the boundary of an annulus, and thus when cutting along γ in step 2 we do not obtain annuli. This can be dealt with by appropriately subdividing in the correct places, which, when thinking about it, is exactly what this algorithm does – but I believe that the description using curves in general position is more transparent (this perspective is directly taken from Éric Colin de Verdière's notes).

To conclude the proof of the theorem, we establish the correctness and the complexity analysis:

Proof of Theorem 2.21. The algorithm terminates in $O(\log n)$ recursion levels since the length of the path p in the recursive calls shrinks by a half at each recursive call. The correctness follows from Proposition 2.23, Lemmas 2.24 and 2.25, since they prove that some minimal cut will be dual to the shortest cycle that our recursive calls will find.

The proof of correctness is not as immediate as one could expect, as recursive calls share a lot of structure with their parent: for example it looks like the same non-boundary edge of G might be cut several times by the shortest paths in the recursion, and thus appear in several of the annuli. But note that if an edge e is cut into subedges $e_1, \ldots, e_\ell$, then in all the recursive calls involving the annulus between e_i and e_{i+1}, the recursion stops directly, since there is a face adjacent to both sides of the annulus. Therefore, only e_1 and e_ℓ actually lead to recursive subcalls, and thus at each level of the recursion, throughout all branches of the recursion tree, each non-boundary edge only appears a constant number of times.

Similarly, the number of boundary edges on an annulus is at most twice the number of non-boundary edges, and thus the total number of boundary edges at a level of a recursion is $O(n)$.

Each computation in steps 1 and 2 of the algorithm takes time linear in the complexity of the annulus at this stage. There are $O(\log n)$ levels, and by the previous observations, each them costs $O(n)$ time in total, so the total complexity is $O(n \log n)$.

□

3 Surfaces

We now turn our attention to other topological spaces of dimension 2, and the graphs embedded thereon.

3.1 Definition and classification

A ***surface*** is a topological space locally homeomorphic to the plane, i.e., every point has an open neighborhood homeomorphic to $\mathbb{R}^2$. In this course, we restrict our attention to *compact* and *connected* surfaces. Behind this very wide-looking definition, there are actually only a few (yet infinitely many) different surfaces, and their classification is the topic of this subsection. Examples of surfaces are the sphere, the torus, and the projective plane, see Figure 20. As before, we consider spaces up to homeomorphism. It turns out that these are exhaustive in the sense that all the other surfaces can be built by gluing those together.

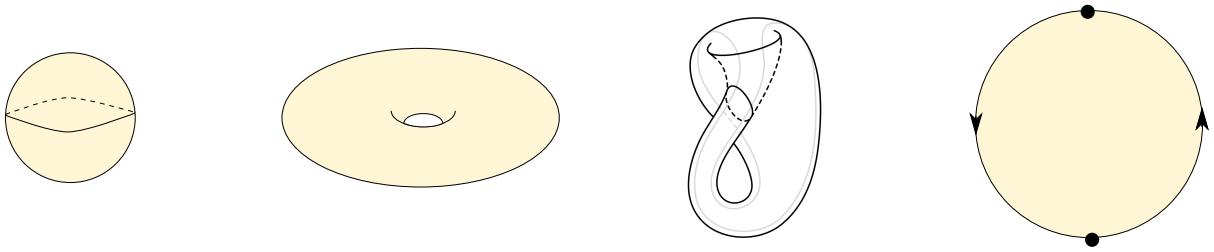


Figure 20: A sphere, a torus, a Klein bottle, and a projective plane (in the last one, the right boundary is identified to the left boundary in the direction indicated by the arrow).

In the first chapter, we saw how a connected graph embedded on the plane naturally cuts the plane into disks. Similarly, we will investigate and classify surfaces by the graphs embedded on them. The definition is the same as in the planar case: a graph $G = (V, E)$ is ***embeddable*** on a surface S if there exists an injective map $f : G \to S$, i.e., G can be drawn without crossings on S. A graph is ***cellularly embedded*** if every connected component of $S \setminus G$ is homeomorphic to an open disk.

Theorem 3.1 (Kerékjártó-Radó)**.** *On any compact connected surface, there exists a cellularly embedded graph.*

Start of a proof. By definition of a surface, every point has an open neighborhood homeomorphic to the plane, or an open two-dimensional disk. By compactness, we can extract a finite covering out of this collection of open sets. Their boundaries form a family of simple closed curves embedded on the surface. If they cross finitely many times, the interiors of their intersections can be seen to be bounded by simple closed curves, and thus, by the Jordan-Schoenflies theorem (which applies in the small neighborhoods since they are homeomorphic to the plane), they bound disks. Hence we have cut the surface into disks, and we win. If they cross infinitely many times, we can tinker with them to reduce to the case of finite crossings. □

This is more subtle than it appears: this sketch can be made correct in two and three dimensions, but not in higher dimensions, as there are example of higher dimensional manifolds that can not be *triangulated* (a higher dimensional analogue of our cutting into disks).

While we are not proving theorems, let us add the following one. A ***triangulation*** is a cellularly embedded graph where every face has degree 3. A refinement of a triangulation is obtained by either subdividing a face (adding a vertex in that face and edges adjacent to each vertex incident to that face), or an edge (adding a vertex on an edge and edges adjacent to each non-adjacent vertex in the two incident faces).

Theorem 3.2. *Any two triangulations of a surface have a common refinement.*

The same not-a-proof works: overlay the two triangulations, if they have a finite number of

intersections, we are done. Otherwise, we tinker things. Once again, this is quickly not true in higher dimensions.

For the reader disappointed in these two omissions (and I understand him), one alternate way to view this is that we are only looking at surfaces that are *defined* as disks glued together in a finite way, with two surfaces being isomorphic if they have a common refinement (this can be taken as a definition). Then the two omitted proofs show that we obtain the same surfaces as with the more topological definition, but if we are content within the purely combinatorial world, we do not need this equivalence.

We describe a graph cellularly embedded on a surface via ***polygonal schemata***, which encodes the way that the disks are glued together. Starting from a cellularly embedded graph, we first name the edges and orient them in an arbitrary way. Each facial walk induces a word (considered up to cyclic permutation), where an edge e taken in the reverse direction is denoted by $\bar{e}$ or e^{-1}. Each such facial walk is called a ***relation***. The polygonal scheme is the data of all these facial walks. Reciprocally, if one is given a collection of words where each letter of the alphabet appears exactly twice, one can interpret those as disks glued to each other, which together form a surface.

With Theorem 3.1 in hand, we are now ready to classify surfaces:

Theorem 3.3. *Every compact connected surface is homeomorphic to a surface given by one of the following polygonal schemata, each made of a single relation:*

1. $a\bar{a}$ *(the sphere),*
2. $a_1b_1\bar{a_1}\bar{b_1}\dots a_gb_g\bar{a_g}b_g$ *for some* $g \geq 1$,
3. $a_1a_1\dots a_ga_g$ *for some* $g \geq 1$.

The surfaces of the first and second category are called ***orientable***, those of the third category are ***non-orientable***. The integer g is the ***genus*** of the surfaces. The second case corresponds to g tori glued together (this is called a ***connected sum***), or equivalently, a sphere to which we have glued g handles. The third case corresponds to g projective planes glued together, or equivalently, to a sphere with g disks removed on which we have glued g ***Möbius bands***. See Figure 21, and Figure 22 for the polygonal schemata depicted on the surfaces. Many non-trivial homeomorphisms are hidden behind this apparently simple classification, and it almost equally simple proof: for example, the theorem implies that the connected sum of two projective planes is homeomorphic to a Klein bottle, and that the connected sum of a torus and a projective plane is homeomorphic to the connected sum of three projective planes.

Proof. Let S be a compact connected surface, and let G be a graph embedded on S, which exists by Theorem 3.1. We iteratively remove each edge adjacent to two different face, until there is just a single face. For each edge adjacent to two different vertices, we contract it, and keep the multiple edges or loops that might result, until there is just a single vertex. We now have a graph with a single vertex and a single face embedded on S. If there are no more edges, by uncontracting once we obtain a sphere as in case 1. of the theorem. Thus there is at least one edge.

The single face induces a polygonal scheme with a single relation. The rest of the proof aims at transforming this single relation into one of the two forms of Theorem 3.3 via ***cut-and-paste*** operations:

- If the polygonal scheme has the form $aPaQ$ where P and Q are possibly empty words, then we can transform it into $bbP\bar{Q}$ by adding a new edge and removing a, see Figure 23.

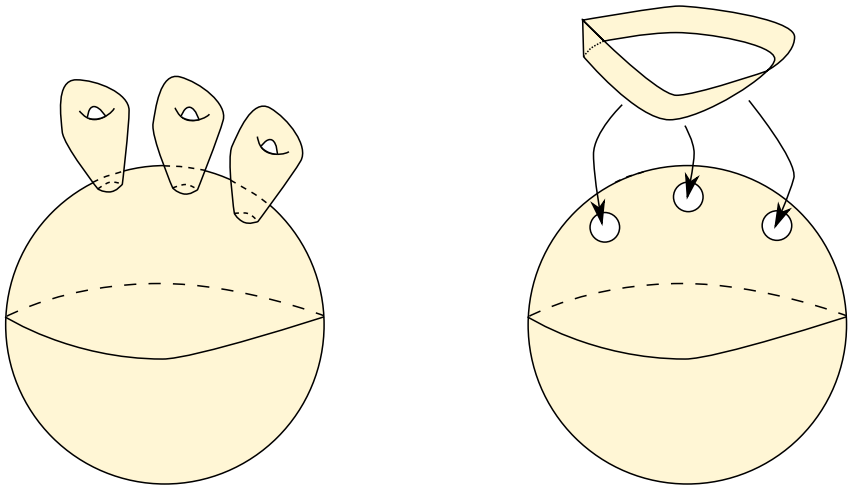


Figure 21: Attaching handles or Möbius bands to a sphere.

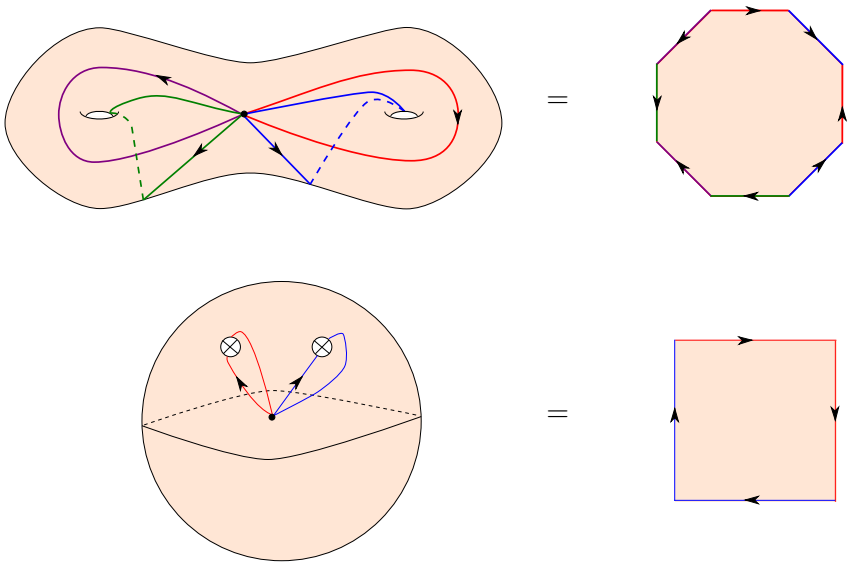


Figure 22: Polygonal schemata of the orientable and non-orientable surfaces. In the non-orientable cases, X denotes a disk on which a Möbius band has been glued.

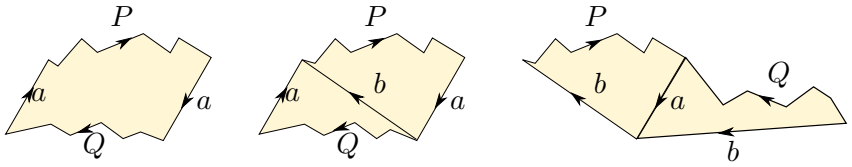


Figure 23: From $aPaQ$ to $bbP\bar{Q}$.

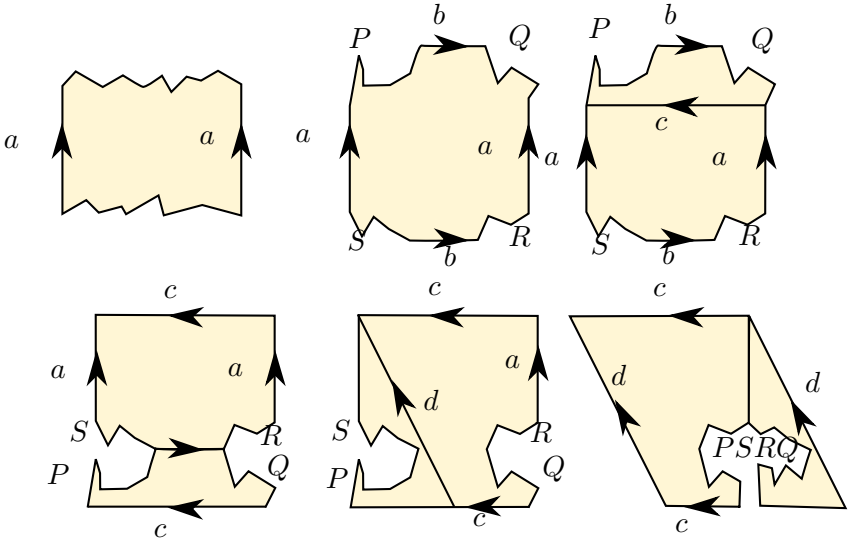


Figure 24: From $aPbQ\bar{a}R\bar{b}S$ to $cd\bar{c}\bar{d}PSRQ$.

Inductively, we conclude that each pair of symbols with the same orientation appears consecutively in the polygonal scheme.

- If the polygonal scheme has the form $aU\bar{a}V$, then U and V must share an edge b since otherwise G' would have more than one vertex. By the preceding step, b must appear in opposite orientations in U and V, so we have the form $aU\bar{a}V = aPbQ\bar{a}R\bar{b}S$. This can be transformed into $dcd\bar{c}PSRQ$, as pictured in Figure 24. Inductively, at the end of this step the relation is a concatenation of blocks of the form aa or $ab\bar{a}\bar{b}$. If all the blocks are of one of these types, we are in case 2 or 3 and we are done.

- Otherwise, the relation has a subword of the form $aabc\bar{b}\bar{c}$. This can be transformed into $\bar{d}\bar{c}\bar{b}d\bar{b}\bar{c}$, as in Figure 25, and then using the first step again this can be transformed into $eeffgg$. Inductively, we obtain a relation of the form 3.

This concludes the proof. □

The ***Euler characteristic*** of a surface S is defined to be $v - e + f$, where v, e and f are the numbers of vertices, edges and faces of a cellularly embedded graph on S. The fact that this actually does not depend on the graph is the object of the following proposition:

Proposition 3.4. *For any graph G cellularly embedded on S with v vertices, e edges and f faces, the value $v - e + f$ is the same.*

It is tempting to prove this using Theorem 3.3: any cellularly embedded graph can be transformed to one of the three graphs stipulated by the theorem, in a way that does not change the Euler characteristic. But to conclude, we need to prove that two different outputs of the theorem are not homeomorphic, which we have not done yet.

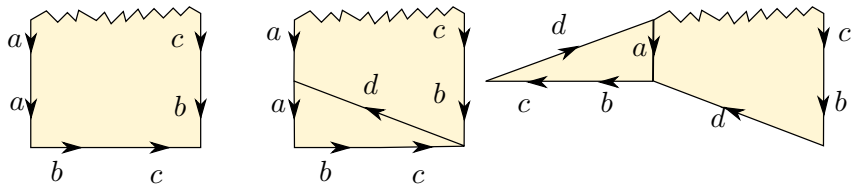


Figure 25: From $aabc\bar{b}\bar{c}$ to $\bar{d}\bar{c}\bar{b}d\bar{b}\bar{c}$.

Proof. We pick two graphs cellularly embedded on S. We add edges until they are both triangulations, which can easily be seen not to change the Euler characteristic. By Theorem 3.2, they have a common refinement. The proof follows from the fact that subdividing edges or faces does not changes the Euler characteristic. □

A look at the graphs corresponding to the cases $1, 2$ and 3 of Theorem 3.3 shows that the sphere has Euler characteristic 2 (which should not come as a surprise after the first chapter), the orientable surface of genus g has Euler characteristic $2-2g$, and the non-orientable surface of genus g has Euler characteristic $2-g$.

Proposition 3.5. *A surface is orientable if and only if for any cellularly embedded graph G, the boundaries of the faces can be oriented so that each edge appears in opposite directions in the two adjacent faces.*

Proof. First note that the property of the proposition is invariant under cut and pasting, as well as edge or face subdivision. Since the polygonal scheme of orientable surfaces satisfies the proposition, this proves the first implication. For the reverse implication, it suffices to observe that for non-orientable surfaces, such orientations of the faces are not possible. □

As a corollary, a non-orientable surface is never homeomorphic to an orientable surface. Since all the orientable surfaces (respectively non-orientable surfaces) have a different Euler characteristic and the Euler characteristic is a topological invariant, this shows that all the surfaces given by Theorem 3.3 are pairwise non-homeomorphic. Therefore we have found them all, and they are all different.

We conclude with a few remarks:

Surfaces with boundary: It is often very convenient to also consider surfaces where points can be homeomorphic either to $\mathbb{R}^2$ or to an open half-space $\mathbb{R}^2 \cap \{x \geq 0\}$. These are surfaces with boundary, where the ***boundary*** is the set of points homeomorphic to an open half-space. Examples are the disk, the annulus, the torus with a disk removed, etc. A similar classification theorem holds for those: surfaces are still classified by their genus, their orientability, but also by the number of connected components of their boundary. It can be proved by gluing disks on the boundary to reduce to the usual case of surfaces. For G a graph embedded on a surface S (in particular for G a simple closed curve), one can ***cut*** S along G and obtain a surface with boundary which we denote by $S \between G$.

Maps: As in the planar case, the actual description of a cellularly embedded graph can be made purely combinatorial. This is already the case with the polygonal schemes, and can be made more convenient algorithmically by using flags, in exactly the same way as in the planar case.

Duality: A graph cellularly embedded on a surface naturally induces a dual graph, as in the planar case.

Subdivisions and minors: There is no analogue of Kuratowski's theorem for surfaces, but there is an analogue for Wagner's theorem. However, the proofs are non-constructive, and the precise list of minors is unknown except for the projective plane.

Separators: An n-vertex graph embedded on a surface of genus g admits an $O(\sqrt{gn})$-sized balanced separator. A good reference for that is this paper of David Eppstein. In the first exercise sheet, you are invited to prove the weaker bound $O(g\sqrt{n}$. Algorithms for planar graphs that are exclusively leveraging the existence of small separators can therefore directly be generalized to surface-embedded graphs.

Testing Embeddability: The ***genus*** of a graph is the smallest genus of a surface it embeds on. It is NP-hard to compute the genus of a graph, but for a fixed surface, there exists a linear-time algorithm to test embeddability on that surface. This will not be covered here (the original algorithm of Mohar and technical and intricate, a more recent one by Kawarabayashi, Mohar and Reed is simpler but requires strong familiarity with graph minor theory techniques).

3.2 Some topological algorithms

The main difference between the plane and more complicated surfaces is that there is no Jordan curve theorem outside of the plane, and thus that some simple closed curves can be non-separating. Even among separating curves, there are differences, as is outlined in the following exercise:

Exercise 3.6. Show that on an orientable surface of genus $g > 0$, there are exactly $\lfloor g/2 \rfloor + 2$ non-homeomorphic[7] simple closed curves.

A standard criterion to differentiate closed curves on surfaces is ***homotopy***. Two closed curves γ_1 and γ_2 are ***homotopic*** if there is a homotopy between them[8]: a continuous map $h : [0,1] \times \mathbb{S}^1 \to S$ so that $h(0,\cdot) = \gamma$ and $h(1,\cdot) = \gamma_2$. The analogues of the nice curves in the planar case are the contractible curves: A simple closed curve γ on a surface S is ***contractible*** if it is homotopic to a trivial map at a point p. The following lemma, which could be taken as a definition, explains why:

Lemma 3.7. *A simple closed curve is contractible if and only if it bounds a disk.*

Half-proof: The reverse direction is immediate. For the forward direction, the subtlety comes from the fact the homotopy could introduce self-crossings. It turns out to be the case that such self-crossings are actually not needed, but we will not prove this. □

Conversely, non-contractible curves are the main object of interest on surfaces. Note that non-contractible curves can be separating. More generally,

We will address two basic problems related to non-trivial curves:

1. How to test whether a closed curve is contractible?
2. How to compute the shortest non-contractible, or non-separating curve?

These algorithmic questions will also serve as an opening for the end of the course, as we use them to illustrate how classical tools from algebraic topology (universal covers, homology) impact the design of algorithms for topological problems on surface-embedded graphs.

[7] Technical remark: any two simple closed curves are homeomorphic (since they are all homeomorphic to $\mathbb{S}^1$). Here, we mean "homeomorphism of the surface that sends one curve to the other one".

[8] This definition requires curves to be oriented, we will often disregard this by considering two curves to be homotopic if some orientation makes them homotopic.

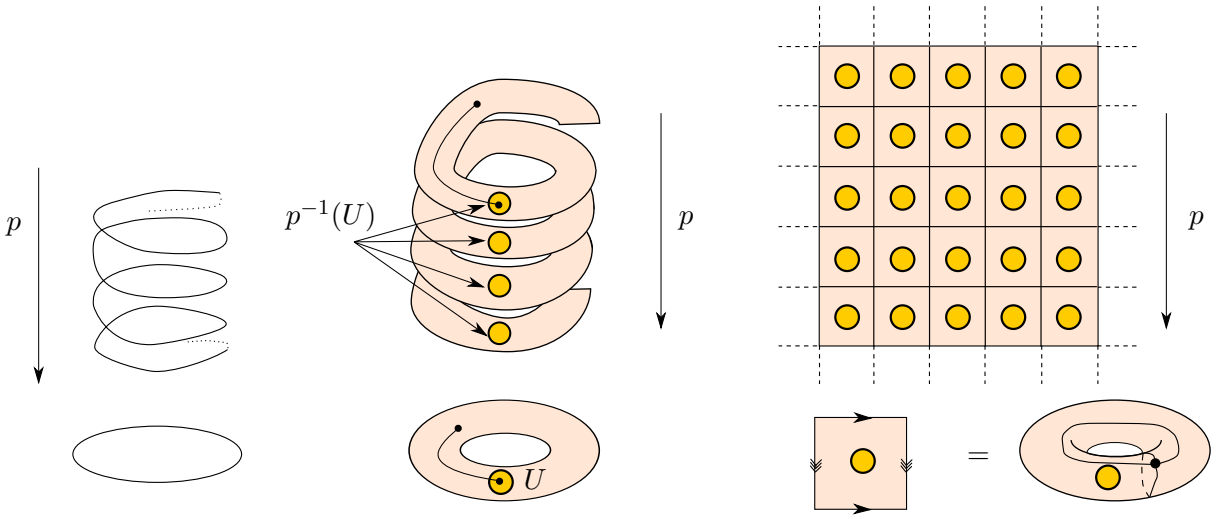


Figure 26: The universal covers of the circle, the annulus and the torus.

3.3 Homotopy testing

To test whether a closed curve on a surface S is contractible, if the curve is simple, we can simply cut along it and see whether one of the components is a disk. This works because of Lemma 3.7. But if the curve is not simple, the problem is significantly less obvious to tackle. This is a good excuse to do some more topology. We will focus on getting polynomial algorithms. With quite some work, everything can actually be made linear.

The main strategy that we use is to use an auxiliary space from S, called the *universal cover* (or at least a portion of it), which differentiates in a natural way curves that are not homotopic. A ***covering space*** $(\hat{S}, p)$ for a surface S is a topological space with a continuous map $p : \hat{S} \to S$ that is a local homeomorphism: any point x in S has an open neighborhood U so that $p^{-1}(U)$ is a disjoint union of open sets U_i which are all homeomorphic to U. The ***universal cover*** is a covering space where *every* simple closed curve is contractible – such a space is called *simply connected*. Such a universal cover is unique in some (natural) sense, though we will not prove this nor use it.

This abstract definition makes much more sense when looking at specific examples, see Figure 26 for some illustrations:

- The universal cover of a circle $\mathbb{S}^1$ is an infinite spiral over this circle.
- The universal cover of an annulus $\mathbb{A}$ is an infinite strip over this annulus.
- The universal cover of a sphere is the sphere itself.
- The universal cover of a torus is the plane $\mathbb{R}^2$ with a tiling into squares.

Exercise 3.8. What is the covering space of the projective plane?

Since the map p of a universal cover is a local homeomorphism, every path, or closed curve γ on S, once a specific preimage of one of its points has been chosen, can be traced on $\hat{S}$, yielding a ***lift*** $\hat{\gamma}$. This is pictured in Figure 26 in the annulus example. But this lift might not be a closed curve. The following lemma shows that it is a closed curve exactly when it is contractible:

Lemma 3.9. *A closed curve γ on a surface lifts in the universal cover to a closed curve if and only if it is contractible.*

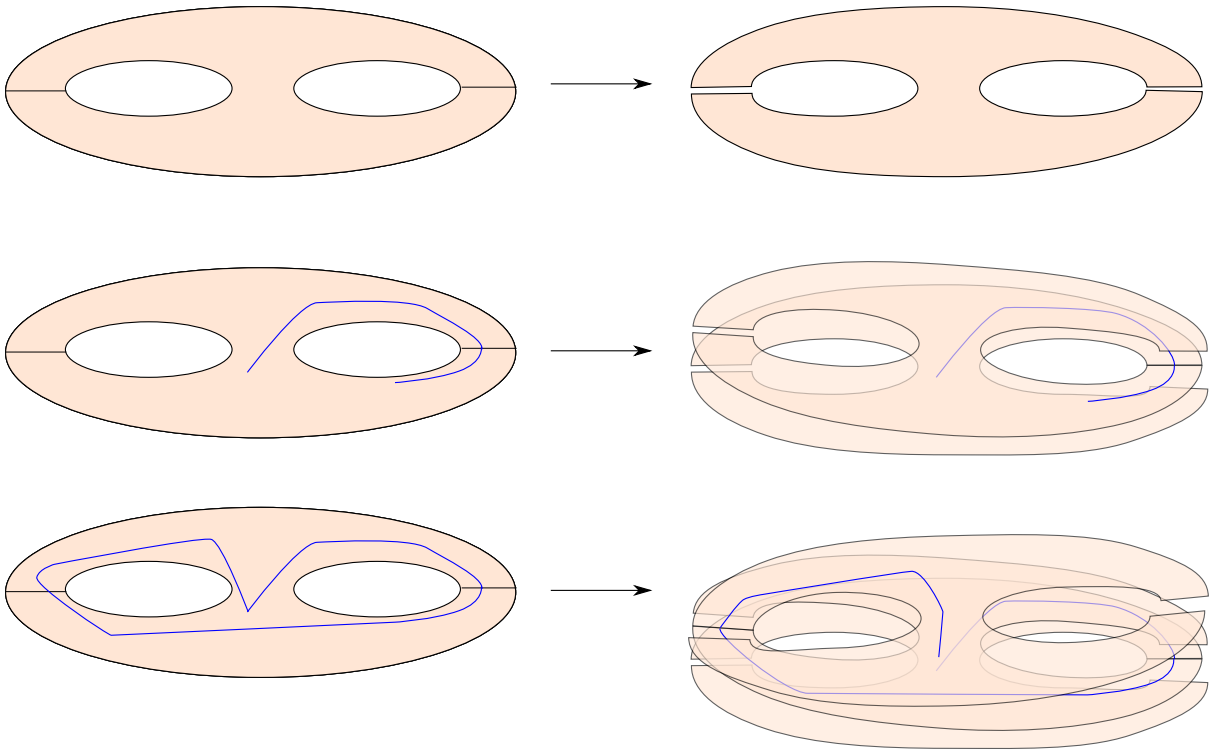


Figure 27: Lifting a closed curve on a disk with with two holes. Note that in the cover, the path starts on the first "floor" and ends on the third floor. Hence the curve is not contractible

Proof. The universal cover is simply connected, and thus each closed curve there can be contracted to a point. This contraction projects via p to a contraction on S, and thus a closed curve on S that lifts to a closed curve must be contractible. Conversely, the lift of a trivial curve is clearly a closed curve, and the contraction on S lifts to a homotopy on $\hat{S}$ □

Surfaces with boundary: For a surface with boundary, one can compute a portion of the universal cover in the following way. Starting from a graph G embedded on a surface with boundary, we first contract and remove edges until there is a single face and all the vertices are on the boundary. Denote by F the remaining edges. We cut along them, thus getting a disk D. The edges of F will act as ***fences***: whenever we cross such an edge, and nothing has been put yet on the other side, we glue a new copy of D. This process is naturally infinite, but is already useful for testing contractibility of a curve γ. First, observe that when an edge of G gets removed, one can reroute γ without changing its homotopy type, which might increase its length by an $O(g)$ factor. Then one can build the portion of the universal cover on which the lift of γ will live, trace the lift of γ on this portion and check whether the end point is the same as the starting point, see Figure 27. Thinking a bit more about this algorithm, it really only boils down to words: it amounts to writing the word formed by the edges of F that we intersect, with a $\cdot^{-1}$ if the orientation is reversed, and checking whether the word we obtain can be reduced to a trivial word by reducing subwords of the kind aa^{-1}. We clearly get a polynomial algorithm

Orientable surfaces without boundary: We first look at the easy example of the torus. In this case, the universal cover is $\mathbb{R}^2$ tiled by squares, and the boundaries of the square can be taken to be any pair of edges (a, b) in a single vertex, single face graph embedded on the torus. One can test the contractibility of a curve by walking on this tiling of $\mathbb{R}^2$ and checking whether we come back to the same vertex at the end of the walk, see Figure 28. We emphasize here

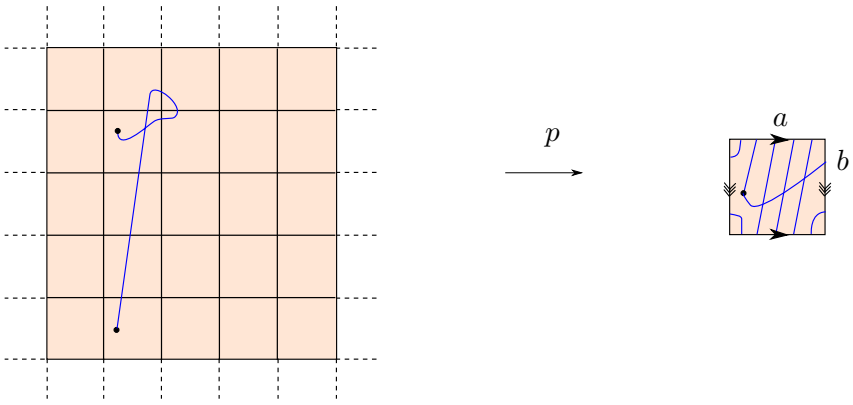


Figure 28: Lifting a closed curve on a torus. In the cover, it lifts to a path, hence the curve is not contractible

the main difference compared to the case with a boundary: after following the path for some time, we will sometimes come back to the same tile in $\mathbb{R}^2$ despite having never backtracked. Such a case happens in the example. Here again, in practice, this is a problem on words: this amounts to counting how many times the walk crosses a and b (counting the crossings positively or negatively depending on how we cross it) and checking whether at the end of the walk, those algebraic counts are zero.

For the more general case, as in the start of the classification of surfaces, up to removing and contracting edges, we can obtain from any graph a graph with a single vertex and a single face, which is often called a *system of loops*. By Euler's formula, this system of loops has $4g$ loops. Cutting along it yields a $4g$-gon. By analogy with the toroidal case, we want to look for the universal cover at a tiling of $\mathbb{R}^2$ where each tile is a $4g$-gon, and every vertex has degree $4g$. This is not possible in a Euclidean way, but there are such *hyperbolic* tilings, which are tilings in a space of negative curvature, see Figure 29 for an example with the genus 2 surface. (One model of) Hyperbolic geometry is a geometry where the ambient space is an open disk (thus homeomorphic to $\mathbb{R}^2$) and the straight lines are arcs of circle which are orthogonal to the boundary of the disk. The reader can check on the picture that we indeed get a tiling of the disk with octagons, eight of which meet at each vertex. Note that the definition of universal cover is purely topological, so the metric structure is just here at this stage to help us with intuition. One could try to trace on this hyperbolic tiling the walk and check whether we come back to a point, but now it is much less trivial to decide when we come back to the same tile. Instead, the following argument, dating back to Dehn, allows us to make progress locally until we the curve is trivial:

A closed curve on a graph G which is a system of loops can be seen as a word on $A \cup A^{-1}$, where the alphabet A is the set of loops. A spur is a subword of the form aa^{-1}.

Lemma 3.10. *Let γ be a closed curve on a graph G with is a system of loops, cellularly embedded as a system of loops on an orientable surface of genus at least 2. If γ has no spurs and is contractible, then it has a subpath consisting of more than half of a facial walk of G.*

Note that this is not true on a torus: indeed, this is a property that results from the hyperbolicity of the tiling. One can prove this using Euler's formula and a bit of sweat, but since we have introduced a bit of geometry, it is a good occasion to explore a geometric version of the Euler formula argument. It relies on a discrete notion of curvature, which quantifies how non-Euclidean a space is, which we first introduce. Let D be a disk on which a graph is embedded. Around a vertex, between each pair of consecutive edges, there is a ***corner***. For each corner c, let $\theta(c)$ be a positive number which we think of as the angle (in fractions of full

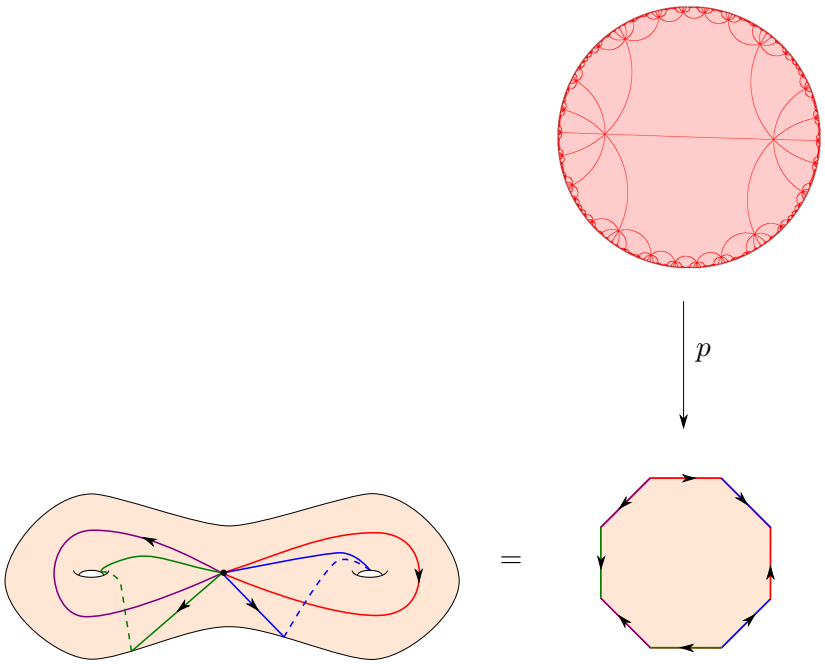


Figure 29: Covering a genus-two surface by a hyperbolic tiling of the open disk.

turns) at this corner. In a Euclidean polygon, the sum of angles on a polygon with d sides is always $(d-2)\pi$, so with our normalizations, this becomes $d/2-1$. Therefore, it makes sense to define the curvature of a face $\kappa(f)$ to be

$$\kappa(f) = \sum_{c \in f} \theta(c) - deg(f)/2 + 1.$$

Thus, a triangle where the sum of angles is less than π has negative curvature. Similarly, in usual space, the sum of angles around a vertex is 2π, or 1 when normalizing by full turns. Thus we define the curvature of an interior vertex $\kappa(v)$ is defined to be

$$\kappa(v) = 1 - \sum_{c \in v} \theta(c).$$

In these discrete terms, a vertex with negative curvature is a vertex with too much stuff around it. Finally, the curvature of a boundary vertex $\tau(v)$ is

$$\tau(v) = 1/2 - \sum c \in v\theta(c).$$

A magical formula called the Gauss-Bonnet formula stipulates that on a surface, the sum of the curvature equals the Euler characteristic (up to some constant normalizing factor). In our setting, the discrete version is the following:

Lemma 3.11. *For a graph G embedded on a disk D, and any choice of angles $\chi(c)$ on its corners, if we denote by V_i and V_∂ its interior and boundary vertices, we have:*

$$\sum_{v \in V_i} \kappa(v) + \sum_{v \in V_\partial} \tau(v) + \sum_{f \in F} \kappa(f) = \chi(D) = 1.$$

Proof. Observe that in the three summands, the terms involving $\theta(c)$ cancel each other. What remains is $|V_i| + 1/2|V_\partial| + |F| - \sum_f deg(f)/2$. The last sum counts the inner edges and half of the edges on the boundary, or equivalently, all the edges minus half the edges on the boundary. Since on the boundary of the disk, there are as many edges as vertices, this sums to $|V_i| + 1/2|V_\partial| + |F| - (|E| - |V_\partial|/2) = |V| - |E| + |F|$ which is exactly the Euler characteristic. □

Of course, this generalizes to higher genus surfaces, with an identical proof. With this tool in hand, we proceed to the proof of Dehn's lemma.

Proof of Dehn's Lemma. If γ is contractible, it lifts to a closed curve in the universal cover. This closed curve might self-intersect, and thus partitions our tiling of $\mathbb{R}^2$ into disks (and the outer face). Therefore it suffices to prove that any disk D bounded by a closed curve on a $(4g, 4g)$-tiling of $\mathbb{R}^2$ contains a subpath of length at least $2g + 1$ on the boundary of one tile. The idea of the proof is to choose the angles so that a lot of curvature has to happen on the boundary vertices, which will force the subpath that we are looking for.

Therefore, we set all the corners to have angle $1/4$. Then all the faces and all the interior vertices all have negative curvature. The vertices of the boundary have positive curvature if and only if they are adjacent to a single face, in which case they have curvature $1/4$. We call such a vertex *convex*. So by the combinatorial Gauss-Bonnet formula, there will be a lot of convex vertices:

$$\begin{aligned} \sum_{v \in V_i} \kappa(v) + \sum_{v \in V_\partial} \tau(v) + \sum_{f \in F} \kappa(f) &= 1 \\ |F|(1-g) + |V_i|(1-g) + |V_{convex}|/4 &\geq 1 \\ |V_{convex}| &\geq (4g-4)|F| + 4 \end{aligned}$$

So some face is adjacent to $4g-3$ convex vertices. These convex vertices must be consecutive, since otherwise there would be at least four of them. So some face has $4g-2$ consecutive edges on ∂D, which is strictly bigger than $2g$ for $g \geq 2$. □

With Lemma 3.10 in hand, we can describe a combinatorial algorithm to test contractibility: after having reduced to a system of loops, we look at the word formed the walk. We first inductively remove all the spurs. Then we scan it for subword consisting of more than half of the facial walk of the system of loops. Whenever there is one, we replace it by the complementary part of the facial walk: this is a homotopy, and thus does not change the contractibility of the walk. Each of these changes reduces the complexity of the word, and we induct. By Lemma 3.10, if the closed curve is contractible, we will reach the trivial word. The complexity is clearly polynomial. With quite a lot of care, it can be made linear.

Zooming out a bit: In both variants, we have seen that the problem boils down to a problem on words: deciding whether a given word reduces to a trivial word under spur reduction (case with boundary), or with spur reductions and a more complicated relation. The underlying reason behind this is that the problem can be phrased in terms of combinatorial group theory. Indeed, the homotopy classes of loops on a surface, with the concatenation law, forms a group called the *fundamental group* of the surface, and a presentation for this group can be readily computed from a graph embedded on the surface: in the case with boundary, it is a free group with the fences as generators, and in the case without boundary, it is the single-relator group obtained with the system of loops as generators and the facial walk as a relation. Then the contractibility test amounts to testing the triviality of an element of this group. This perspective

is easily misleading: in general testing triviality in a group defined by generators and relations is *undecidable*. The fact that this can be solved for the fundamental group of surface is therefore remarkable.

More generally, we can try to test whether two given curves are homotopic. This is more subtle, but can also be made in linear time, using linear tools.

3.4 Computing shortest interesting cycles

In this subsection, we provide algorithms to compute shortest non-contractible and non-separating curves. This is a natural and important topological primitive, and here are two applications for such algorithms:

- The first thing one often wants to do when given a surface is to cut it into something more planar. This is relevant for practical purposes (e.g., topological noise removal, texture mapping) and algorithm design, since then one can use the good old planar algorithms.
- In the planar case, as we saw, the min-cut problem reduces to computing a shortest cycle that "goes around" the annulus in the dual, i.e., the shortest non-contractible cycle. We are considering the direct generalization of this. This turns out to be relevant to solve min-cut on higher genus surfaces, although it is unlikely that we will have enough time to cover this in this course.

We will prove the following theorem:

Theorem 3.12. *For a graph G with n vertices embedded on a surface of genus g, we can compute a shortest non-contractible cycle, respectively a shortest non-separating cycle, in time $O(n^2 \log n)$.*

As we saw during the min-cut algorithm, when cutting surfaces, cutting along cycles in the primal or the dual graph can lead to annoying issues. For example when one cuts once along a cycle, and then along a second cycle sharing edges with the first one, this yields some degeneracies. In order to deal with these issues in a clean way, we formalize the approach that we used for min-cut, based on considering curves in general position with respect to a graph.

Recall that a curve is in ***general position*** with respect to an embedded graph if they cross transversely, away from the vertices, and a finite amount of times. A ***cross-metric surface*** (S, G^*) is a topological surface S with a (possibly edge-weighted) graph G^* embedded on it. A cross-metric surface assigns lengths to any curve γ in general position by simply counting the (possibly weighted) number of intersections with G^*. For a (possibly edge-weighted) graph G cellularly embedded on a surface S, measuring the length of a walk in G is the same as measuring its length in the cross-metric surface (S, G^*), where G^* is the dual (hence the notation). Similarly, shortest paths in the cross-metric surface (S, G^*) can be computed by the usual graph algorithms on G. The added value with the cross-metric surface, compared to just using duality, is that we are considering more curves than in the pure graph-theoretical world: for example there are often walks on a graph that are not really self-crossing, in the sense that one could clearly push them infinitesimally to make them simple. In the cross-metric setting, we can directly pick them to be simple, which makes proofs more streamlined. See Figure 30.

Algorithmically speaking, while we will be considering arbitrary topological curves in transverse position with the graph G^*, we do not need to encode the precise location of a curve γ, merely its crossing points with G^*, its self-crossing points if there are any, and what it does

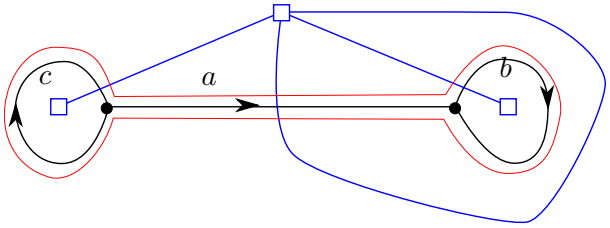


Figure 30: A graph and its dual. The walk $ab\bar{a}c$ is definitely not a cycle in the primal graph, but when we look at it in the cross-metric perspective and push it a bit, it is a nice simple closed curve (in red).

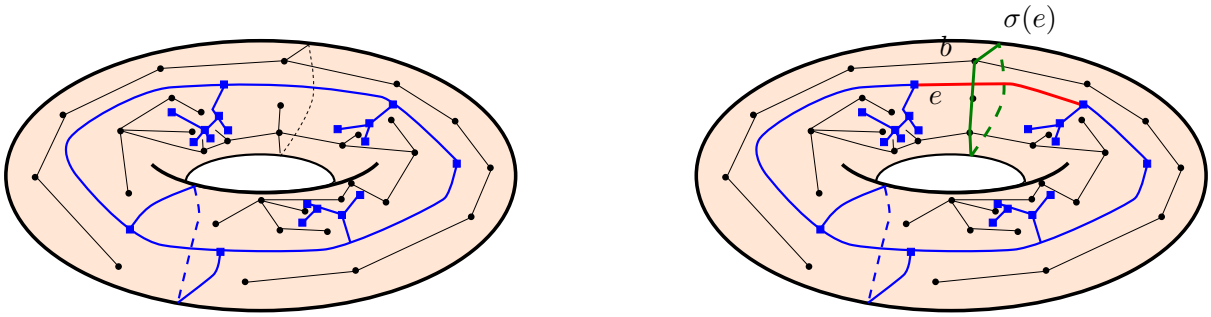


Figure 31: A shortest path tree, in black, defining a cut locus (in blue). Cutting the surface along the cut locus yields a disk. On the right, an edge of the cut locus corresponds to loop $\sigma(e)$ going through the basepoint b.

in between. Equivalently, we can encode the superposition of γ with G^*, which is also a cellularly embedded graph. Likewise, when there is more than one curve, we simply encode their superposition with the graph G^*.

Now that the setup is set up, we move forward. We first compute shortest *loops*: a ***loop*** ℓ is a closed curve $\ell : \mathbb{S}^1 \to S$ going through a fixed point b, called the ***basepoint*** of the loop.

For a point b in a face of (S, G^*), we denote by T a shortest path tree rooted at b, which can be computed by using Dijkstra's algorithm in the primal graph. The ***cut locus*** C of (S, G^*) with respect to b is the set of edges not crossed by T. Informally, we are blowing a balloon based at b, and the cut locus is the set of points where it self-intersects. This should be very reminiscent of the tree-cotree duality that we explored on planar graphs, and thus the following lemma should not come as a surprise. See Figure 31 for an illustration.

Lemma 3.13. *The cut locus cuts S into a disk.*

Proof. While growing the shortest path tree, the set of open faces visited by the tree union all the edges that it crosses is an open disk, and this is maintained until the end. At the end, the complement of this disk is exactly the set of of edges in the cut locus, which proves the lemma. □

For an edge e in the cut locus, we denote by $\sigma(e)$ the loop obtained by starting from b, taking a shortest path to one of the two faces of G^* adjacent to e, crossing e, and coming back to e via the shortest path on the other side of e. The weight of e is defined to be the length of $\sigma(e)$. The following key lemma shows that the shortest non-contractible loop can be found among the $\sigma(e)$:

Lemma 3.14. *Some shortest non-contractible loop has the form $\sigma(e)$.*

Proof. Let L be a shortest non-contractible loop crossing the cut locus C as few times as possible. If L crosses C at least twice, there is a point p between the two crossings, cutting L into L_1 and L_2. This point p is connected to the root via a path ρ on the shortest path tree T. Then either L_1 concatenated with ρ or L_2 concatenated with ρ is non-contractible, since otherwise the contraction of both would yield a contraction of L. This argument is sometimes called the 3-path condition, after Thomassen.

So L crosses C at most once. It has to cross it at least once, since otherwise it bounds a disk by the Jordan-Schoenflies theorem on the surface cut along the cut locus, and is thus contractible. So L crosses the cut locus at some edge e, and since L goes through the root, it has to have at least the length of $\sigma(e)$, which concludes the proof. □

This immediately suggests a brute-force algorithm to find the shortest non-contractible loop: try all the $\sigma(e)$, test their contractibility (which is easy since they are simple) and output the shortest one.

This is where the last lecture stopped. For completeness, we explain the rest of the argument and the non-separating case in what follows. This will not be part of the syllabus for the exam.

But one can be smarter, and figure out a combinatorial criterion to decide whether a $\sigma(e)$ is contractible:

Lemma 3.15. *Let e be an edge of C. Then $\sigma(e)$ is contractible if and only if some component of $C \setminus e$ is a tree.*

Proof. For the reverse direction, if some component of $C \setminus e$ is a tree, one can homotope $\sigma(e)$ into a trivial loop by following this tree. For the forward direction, if $\sigma(e)$ is contractible, it bounds a disk by Lemma 3.7. If no component of $C \setminus e$ is a tree, then in particular the component that is a disk is not a tree, and thus it contains a cycle. But this contradicts Lemma 3.13. □

We now have all the tools to prove the first half of Theorem 3.12:

Finding a shortest non-contractible cycle in $O(n^2 \log n)$ time. For each face of G, we fix a root r at G. Then we compute in $O(n \log n)$ time the cut locus based at r as well as the weight of each of its edges, i.e., the length of $\sigma(e)$. There remains to prune the cut locus, that is, to remove all of its useless arborescent parts. Since each tree has a degree-one vertex (its leaves), this can be done by removing all the degree one vertices, and then the new degree one vertices, etc. Our shortest non-contractible loop through r is then the smallest of the remaining $\sigma(e)$s. Looping through all the possible rs increases the complexity to $O(n^2 \log n)$, and we output the shortest of the resulting cycles. □

In order to find the shortest non-separating loop, we first want to establish that one of them is of the form $\sigma(e)$. For non-contractible loops, this relied on the three-path condition, and we need something similar here, something like "Let a and b be two points on S and p, q and r be three paths from a to b, oriented from a to b. If $p\bar{q}$ and $r\bar{q}$ are both separating, then so is $p\bar{r}$."

This poses an annoying issue, as it is not clear what it means for a curve that is not simple to be separating. So we first address this, and the convenient language for that is the language of (mod 2) *homology*.

Let G be a graph embedded on a surface S, with its set of vertices, edges and faces. We think of those as being 0-dimensional, 1-dimensional and 2-dimensional objects. A ***k-chain***, for $k = 0, 1$ or 2 is a subset respectively the set of vertices, edges or faces. We think of a chain

as being an element in a vector space over $\mathbb{Z}_2$, the set of integers mod 2. Therefore, chains can be added, using the rule $1 + 1 = 0$. These vector spaces are denoted by C_0, C_1 and C_2. We define the boundary of an edge to be the sum of its endpoints, and the boundary of a face to be the sum of its boundary edges. These two boundary maps extend by linearity on the whole spaces C_1 and C_2, defining linear maps $\partial_1 : C_1 \to C_0$ and $\partial_2 : C_2 \to C_1$. A 1-chain is a ***cycle*** if its boundary is trivial, and it is a ***boundary*** if it is the boundary of some 2-chain. Cycles are generally denoted by Z_i, and boundaries by B_i. Convince yourself that the boundary of a boundary is empty.

Now, a closed walk γ on G or a closed curve in general position with G can naturally be considered as a 1-chain (either for the graph G or in the graph that is the overlay of G and γ). The following lemma shows that being a homology boundary naturally generalizes being separating.

Lemma 3.16. *A non-trivial simple closed curve γ is a homology boundary if and only if it is separating.*

Proof. If γ is separating, then it is equal to the boundary of the sum of the faces of (either) one of the two connected components. If γ is a homology boundary, then it is the boundary of a sum of faces $\mathcal{F}$. Note that this set of faces cannot be all the faces, as the boundary of the sum would be empty. Then the faces in $\mathcal{F}$ are separated from the faces not in $\mathcal{F}$, since any path connecting them would cross the boundary γ. □

The homology group H_i is defined as the quotient of the space Z_i by the space B_i: it is the space of cycles which are not boundaries. So it directly generalizes the separating curve.

With this language, we have the two tools needed to prove our missing lemma: a notion of sum of cycles, and a notion of separating for non-simple curve:

Lemma 3.17. *Some shortest non-separating loop going through the root r has the form $\sigma(e)$.*

Proof. Since a non-separating loop is a non-trivial homology cycle, we can equivalently look for a shortest loop that is non-trivial in homology if we can prove (which we will) that one of them is simple. As in the non-contractible case, any shortest non-trivial homology loop L must cross C at least once. We pick one that crosses C a minimal number of times. If it crosses it more than once, we take p to be a point between two crossings, which is connected to the root via a path ρ. The point p cuts L into L_1 and L_2, and we look at the three cycles L, L_1 concatenated with ρ and L_2 concatenated with ρ. Note that the sum (as chains) of any two of these cycles forms the third cycle, and the set of homology boundaries is a vector space, and thus is closed under addition. So if both $L_1 + \rho$ or $L_2 + \rho$ were homology boundaries, then so would be L, which is a contradiction. So the shortest homology loop crosses C exactly once, and since it contains r it must be at least as long as $\sigma(e)$. Hence some shortest homology loop has the form $\sigma(e)$. Thus it is simple, and is thus non-separating. □

Now we can bruteforce and try all the $\sigma(e)$s to find a shortest non-separating one. Or we can try to be smarter, and prove that:

Lemma 3.18. *A loop $\sigma(e)$ is separating if and only if e separates C.*

Proof. If $\sigma(e)$ is separating, it separates C into at least two components. In the other direction, if e separates C, then any path on the surface connecting these two components and not crossing $\sigma(e)$ can be pushed back to C since $S \setminus C$ is a disk. □

The set of edges e separating C are called ***bridge edges***. Note that edges corresponding to a contractible $\sigma(e)$, as characterized by Lemma 3.15 are bridge edges, which makes sense since contractible curves are separating. One can determine in linear time all the bridge edges of C using depth-first search, this is very similar to the block decomposition alluded to in the planarity testing. Plugging everything together:

Finding a shortest non-separating cycle in $O(n^2 \log n)$ time. For each face of G, we fix a root r at G. Then we compute in $O(n \log n)$ time the cut locus based at r as well as the weight of each of its edges, i.e., the length of $\sigma(e)$. We compute all the bridge edges and compare the remaining ones to find the shortest non-separating loop based at r. Looping through all the possible rs increases the complexity to $O(n^2 \log n)$, and we output the shortest of the resulting cycles. □

Zooming out a bit: These two algorithms work because of the algebraic structure behind the curves: the concatenation of two contractible curves is contractible (i.e., the set of homotopy classes forms a group under concatenation), and the concatenation of two "separating" curves is "separating" (the homology classes form a group under addition). There are some other algebraic structures of interest, in particular, we can leverage *relative homology* to compute shortest systems of loops (see the lecture notes of Éric Colin de Verdière). Yet when there is no such known algebraic structure behind the problem that we consider, any optimization problem becomes much harder: how to compute the shortest polygonal scheme of the form $a_1 b_1 \bar{a_1} \bar{b_1} \dots a_g b_g \bar{a_g} b_g$ for an orientable surface? How to compute the shortest collection of closed curves cutting a surface into a collection of sphere with three holes (*a pants decomposition*)? No polynomial algorithm nor hardness proof is known for these two problems.