GEOMGRAPHS: Algorithms and Combinatorics of Geometric Graphs
MPRI 2025-2026

Arnaud de Mesmay

These are the lecture notes for my half of the course GEOMGRAPHS. The other half is taught by Luca Castelli Aleardi and the slides and exercise sheets for his half are available on the course's webpage.

Some practicalities:

- The course is on Thursdays, from 8:45 to 11:45, in Sophie Germain room 1002.
- There is an exercise session in the middle of each lecture, after the half-time break.
- The class will be graded with a final written exam.
- There will be two optional exercise sheets, with points contributing extra credits for the final grade.

This aim of this course is to provide an overview of the combiantorial, geometric and algorithmic properties of 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 on 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:

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

While the focus of the course stays 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

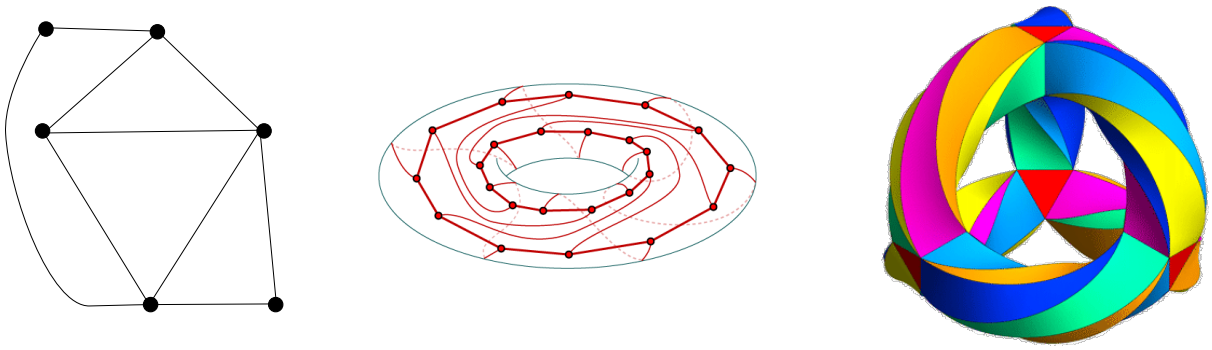

Figure 1: Graphs embedded in the plane, the torus, and the three-holed torus.

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 surface. This is especially the case in graph minor theory, where embedded graphs play an absolutely central role (but we barely touch on this topic).

The class is taught alternatively by Luca and myself: his class focuses on combinatorial aspects and graph-drawing questions, while my half covers some algorithmic questions and topological aspects of surface-embedded graphs.

These lecture notes follow quite closely the material taught in my half of the 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 refrain from (too many) digressions and heavy references. The tone aims to be conversational. I will do my best however to add each week the missing details for the proofs which may have been a bit handwavy during the lectures. The content has a strong overlap with the lecture notes of a previous iteration of the class (by Éric Colin de Verdière), and with 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.

Instead of the standard list of references at the end, in these notes I directly provide hyperlinks to some relevant papers, but alas, this does not work too well if you print the notes.

1 Planar Graphs

1.1 A partial recap of the first lecture

A ***planar graph*** is a graph that can be embedded, i.e., drawn without crossings, in the plane, or equivalently the sphere. Throughout this course, it is quite important that we allow graphs to have multiple edges and loops, which is handled seamlessly by the previous definition. A graph without multiple edges nor loops is called a ***simple graph***. A ***plane graph*** is an embedding of a planar graph, and two plane graphs are considered equivalent if there is a homeomorphism of the plane sending one to the other one (or intuitively, if one can continuously deform one into the other without adding crossings). Note that a planar graph can correspond to multiple different plane graphs: think for example of a graph with a single vertex and multiple loops.

Compared to just a general graph, a plane graph has an additional combinatorial structure: there are now ***faces***, which are the connected components of the complement $\mathbb{R}^2 \setminus G$. This additional structure interacts with the initial graph in multiple ways:

- The ***Euler characteristic*** stipulates that for any connected plane graph, we have $v - e + f = 2$, where v, e and f are respectively the number of vertices, edges and faces.
- This implies that planar graphs are ***sparse***: for $v \geq 3$, $e \leq 3v - 6$ (and in general $e \leq 3v$. So planar graphs are very particular compared to general graphs.
- To any plane graph G we can associate a ***dual graph*** G^*, whose vertices are the faces of G and whose edges connect adjacent faces of G (with multiplicity and loops if needed). Note that the dual of a simple graph is in general not simple, and that the dual of graph depends on the embedding.
- The combinatorial data of the faces is actually all that is needed to encode a plane graph. There are various data structures one can use to do that. In these notes, we will simply consider that we have such a data structure that allows us to do "intuitive" operations in the natural time: for example moving from an edge incident to a vertex to the next edge in the circular order in constant time, or listing the k edges adjacent to a face in time $O(k)$, etc.

All of the properties of planar and plane graphs boil down to "intuitive" (but hard to prove) topological properties of the plane: the ***Jordan curve theorem*** shows that any simple (i.e., non self-intersecting) closed curve in the plane separates it into two components, one bounded and one unbounded. The ***Jordan Schoenflies theorem*** shows that the bounded component is homeomorphic to a disk. These theorems are hard to prove because simple closed curves in the plane can be very complicated, as pictured in Figure 2. If one restricts our attention, say, to polygonal curves, then the proofs become much easier. This leaves us with two alternatives for the class: either we define all our graphs to have edges as polygonal segments, and then all the intuitive facts are easy to prove (there is such a proof for the Jordan curve theorem in my older lecture notes), or we take the general definition and then trust these hard theorems that everything works. I leave you to choose your preferred option.

In particular, all the bounded faces of a connected plane graph are (homeomorphic to) a disk. We say that such a graph is ***cellularly embedded*** (this definition is quite useless for plane graphs but will become useful on other surfaces). We will see later on that the failure of such nice topological properties for more complicated surfaces leads to interesting topological questions.

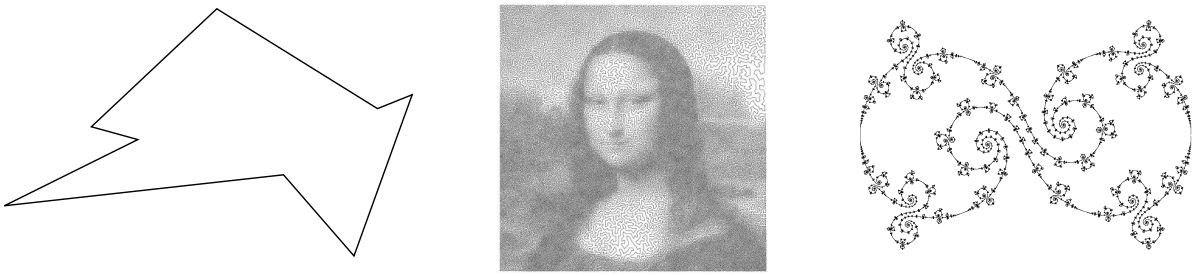

Figure 2: Three simple closed curves in the plane

We also recall the famous theorem of Kuratowski, which will not be proved in this class (nor in Luca's) (you can find a proof in my older lecture notes).

Theorem 1.1 (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.*

Finally, the Fàry theorem shows that every plane graph can be realized with edges drawn as straight lines. This is proved in Luca's half as a corollary of the Tutte embedding theorem (there are also easier direct proofs).

1.2 Coloring

The sparsity has the following nice implication. A k-(vertex) ***coloring*** of a simple graph is an assignment of k colors to the vertices so that no two adjacent vertices share a common color.

Proposition 1.2. *Simple 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 which exists because of the sparsity, 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. □

The following improvement is on the exercise sheet, and relies on more than just the Euler characteristic and the sparsity.

Exercise 1.3. Prove that simple 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.

1.3 The crossing lemma

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. We will only consider drawings ***in general position***: that means that vertices must be mapped to distinct points, edges do not intersect vertices except at their endpoints, edges only intersect transversely and at most two edges intersect at each point. Note that such a drawing in general position induces a plane graph, where every crossing has been replaced by a vertex of degree 4. As we said earlier, via Fàry's theorem, such a plane graph can be assumed to have its edges realized as straight lines. So without loss of

generality, we can and we will assume that this is the case in all our drawings: this implies that all the edges are drawn as polygonal segments (which may bend at crossings).

The ***crossing number*** $cr(G)$ of a graph is the minimal number of crossings over all the possible drawings in general position 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 1.4. $cr(G) \geq \frac{|E|^3}{64|V|^2}$ *if* $|E| \geq 4|V|$.

The proof is a surprising application of (basic) probabilistic tools. There is a nice and much more indepth discussion of this proof available on Terry Tao's blog.

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 at least $|E| - cr(G)$ 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 resulting 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, so $\mathbb{E}(|E''|) = p^2|E|$. Finally a crossing survives if an only if the four adjacent vertices survive[1]. The resulting drawing might not be crossing-minimal but it does imply that $\mathbb{E}(cr(G'') \leq p^4 cr(G)$, which is the inequality we will need. 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. □

In particular, applying this inequality to dense graphs, and in particular to complete graphs K_n shows that any drawing of the complete graph K_n has $\Omega(n^4)$ crossings. Finding the correct constants is notoriously difficult though, even for that very specific-looking case of complete graphs: the Hill conjecture posits that

$$cr(K_n) = \frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor \sim \frac{1}{64}n^4,$$

but the best known lower bound is quite far off[2], at about $cr(K_n) \geq 0.985\frac{1}{64}n^4$ for sufficiently large n. Likewise, the correct constant is unknown for complete bipartite graphs, and in general there are tons of simple-looking open problems on crossing numbers (see this exhaustive survey by Marcus Schaefer).

[1] There may be less than four adjacent vertices in general, but not in the drawing minimizing the crossing number, since drawings where there is a crossing forming an α shape can be simplified by removing one crossing. So in a crossing-minimal drawing, they do not happen

[2] Note that the fact that 1/64 also appears in Theorem 1.4 is a red herring: the number of edges in a complete graph is $\binom{n}{2}$, so a blind application of the crossing lemma only gives $cr(K_n) \geq \frac{1}{64\cdot 2^3}n^4$.

1.4 The Hanani-Tutte theorem

Another interesting result of planar drawings is the following surprising (to me at least) theorem. Two edges of a graph are ***independent*** if they are not incident to a common vertex.

Theorem 1.5 (Strong Hanani-Tutte theorem, 1934)**.** *Any drawing in general position of a non-planar graph contains two independent edges that cross an odd number of times.*

Conversely, if one can draw a graph so that all independent edges cross an even number of times, then the graph is planar (!)

Proof. We first claim that it suffices to prove the theorem for K_5 and $K_{3,3}$. Indeed, let G be a non-planar graph. By Kuratowski's theorem, it contains a subdivision of K_5 or $K_{3,3}$. If the theorem is proved for K_5 and $K_{3,3}$, then we can find a pair of disjoint paths in our graph G that cross an odd number of times. This means that there exists a pair of independent edges that cross an odd number of times. Indeed, otherwise, summing all the even number of crossings among all pairs of edges in the pair of paths would give an even number of crossings for the pair of paths.

In order to prove the theorem for K_5 and $K_{3,3}$, we prove the stronger result that in any drawing of these two graphs, the sum of the number of crossings over all independent pairs of edges is odd. The reason is that this quantity mod 2 does not actually depend on the drawing:

Claim 1.6. *For $G = K_5$ or $K_{3,3}$ and any two drawings G_1 and G_2 in general position of the graph G, the quantity*

$$\sum_{\substack{e,e' \\ \text{independent edges}}} cr(e,e') \mod 2$$

is the same.

Proof. We first describe a general way to deform any G_1 into any G_2, and then prove that the target quantity is invariant throughout this deformation.

As we said earlier, we can assume that in the drawings G_1 and G_2, the edges are polygonal segments. We first move the vertices of G_1 one by one so that they coincide with the respectives vertices of G_2. This can be done by choosing, for every vertex v in G, a polyogonal path p_v between its position v_1 in G_1 and its position v_2 in G_2. This path can clearly be chosen so that it avoids vertices of G_1 and G_2 and so that it intersects their edges transversely and outside of existing crossings. Then we move v_1 along p, dragging all the incident edges along the way, as pictured in Figure 3, top.

In a second step, we inductively straigthen the edges in G_1 and in G_2: we focus on G_1, the case of G_2 being identical. Each edge of G_1 is a polygonal path $e = (q_1, \dots, q_k)$, where the points q_i and q_{i+1} are connected with straight segments. We can straighten such a polygonal path by replacing the triangle q_1, q_2, q_3 by the straight segment q_1q_3: this is done by moving one tip of the triangle to the opposite edge. Here again, one can do so along a path that avoids all the vertices of G_1 and G_2 and intersects their edges transversely and outside of existing crossings, see Figure 3, middle.. We then go on inductively so that every edge in G_1 is a straight line between its endpoints, and likewise in G_2. Since the vertices coincide, the drawings are now identical.

Let us now analyze how the crossings evolve as we do these deformations. By our choice of deformation paths, such evolutions only happen at discrete moments: in the first step this will be when a vertex passes through an edge (Figure 3, (a)), and in the second step this will be when a point q_i crosses an edge (Figure 3, (b)), when one of the two incident straight lines

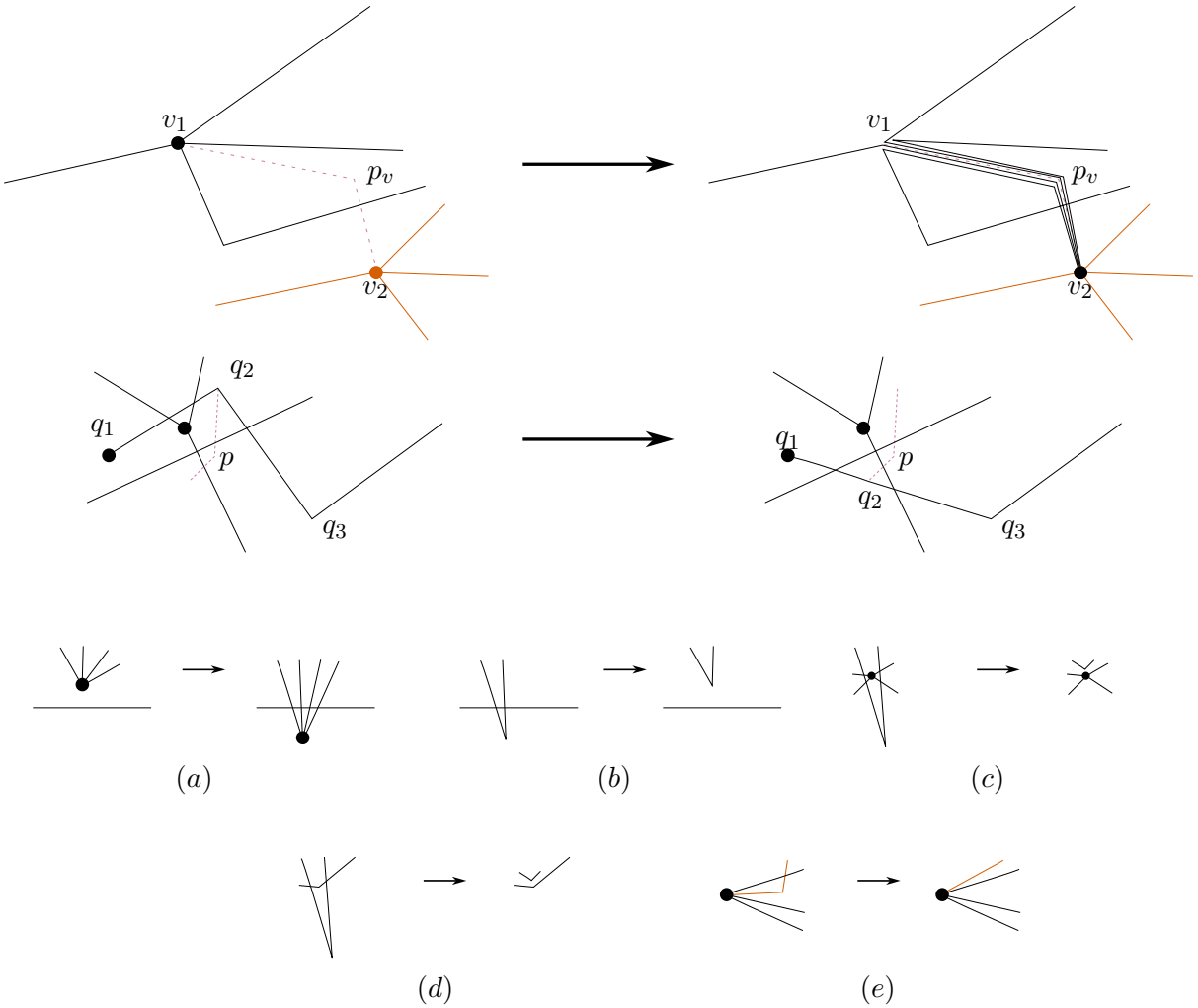

Figure 3: Deforming drawings.

passes through a vertex or a crossing (Figure 3, (c) and (d)), or when two edges switch their order around a vertex (Figure 3, (e)). Note that outside of these discrete events, the crossings do not change: actually the homeomorphism class of the drawing (viewed as a plane graph) does not change. Also note that the events of type (b), (d) and (e) do change the drawing but do not change the parity of crossings of independent edges.

There remains events (a) and (c) which both feature, in slightly different ways, a vertex v passing through an edge e. If v is incident to e , the changes in crossings only involve non-independent edges and are thus irrelevant to our count. If $G = K_{3,3}$, the degree of every vertex is 3, but for any vertex v not incident to e, exactly two of the edges incident to v are independent with e (look at Figure 4). So the count of independent crossings changes by 2, an even amount. If $G = K_5$, every vertex v in K_5 has degree exactly four, and for any fixed e not incident to v, exactly two of the edges incident to v are independent with e (look again at Figure 4). So here again the count of independent crossings changes by 2, an even amount. ⌟

We conclude the proof by exhbiting drawings of K_5 and $K_{3,3}$ where this sum is odd, as done in Figure 4 (note that the proof implies that *any* drawing will work). □

This theorem and its proof provide a polynomial-time algorithm to test whether a graph is planar. It is not as efficient as other more standard approaches, but is simple conceptually and, with a lot of work, can be generalized to higher dimensions. Let G be a graph of which

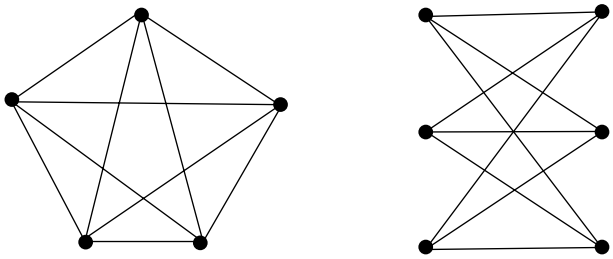

Figure 4: The graphs K_5 and $K_{3,3}$.

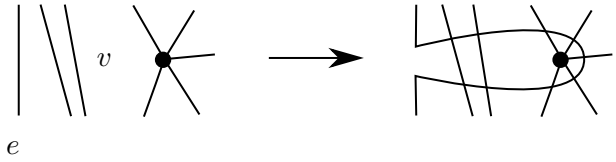

Figure 5: A finger move for a pair $p = (e, v)$.

we want to test planarity. Let us consider the vector space S over the field with 2 elements $\mathbb{F}_2$ whose basis consists of pairs of independent edges (e, f) in G.

1. Start with *any* drawing of the graph G. Write down the vector $x(D) \in S$ of the number of crossings mod 2 of independent edges.
2. For every pair $p = (v, e)$ in the graph G, define a vector $u(p) = \sum_{f \text{ incident to } v}(f, e) \in S$: it contains a 1 for each pair of edges (f, e) such that f is incident to e, and 0 otherwise.
3. Denote by U the subspace spanned by the family $u(p)$. Test whether $x(D)$ belongs to U. If yes, output that the graph is planar, otherwise, output that it is not planar.

Lemma 1.7. *This algorithm is correct and has polynomial complexity.*

Proof. The reason why this algorithm is correct follows from (the proof of) Theorem 1.5. Indeed, starting from any drawing D of G, one can change its vector $x(D)$ by exactly $u(p)$ by applying a ***finger move*** (see Figure 5) from the edge e around the vertex p. If $x(D)$ belongs to U, there exists a sequence of finger moves that brings the vector of crossings $x(D)$ to zero mod 2. Then by Theorem 1.5, the graph is planar. Conversely, if the graph is planar, starting from any drawing we can deform it similarly to the proof of Theorem 1.5 until there are no crossings remaining, and this morphing gives a sequence of finger moves, and thus a family of vectors $u(p)$ showing that $x(D)$ belongs to U.

The third step can be solved by Gaussian elimination in cubic time (or there are faster algorithms, even more so since the underlying field is $\mathbb{F}_2$). Since S has size $O(|E|^2) = O(|V|^2)$ (if the graph is not sparse, we can reject it straight away), this yields a complexity $O(|V|^6)$. □

For the algebraic-minded reader, this algorithm actually amounts to computing an obstruction class in the equivariant cohomology of the deleted product (ask me in class if you are interested about what these words mean).

1.5 Efficient algorithms for planar graphs

Many algorithmic problems can be solved faster when the input graph is planar. This includes some problems which are NP-hard in general but can be solved in polynomial time in the planar

case: for example MAX-CUT, (uniform) SPARSEST CUT, FEEDBACK ARC SET, or computing the BRANCH-WIDTH of a graph. Similarly, testing for GRAPH ISOMORPHISM is (probably) not NP-hard but no polynomial-time algorithm is known in general, while it can be solved efficiently on planar graphs (take a look at Exercise 3 in the Exercise Sheet accompanying Lecture 4). 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. At the risk of overly simplifying things, there are in my opinion three main reasons as to why planar graphs tend to be easier to handle algorithmically than general graphs. The first reason is *sparsity*, which has strong combinatorial and algorithmic consequences, as we have seen earlier for colouring problems. The second reason is *duality*: sometimes a problem that is not easy to solve in the primal graph becomes much easier to solve in the dual graph. Sometimes juggling between both the primal and the dual setting allows to make good progress. The third reason is the existence of *small separators*, which will be proved in Lucas's half, and which allows for efficient divide and conquer approaches for a lot of problems. Famously, a clever use of (iterated) small separators yields the following theorem, showing that shortest paths, which are arguably the most important algorithmic primitive, can be computed in linear time on planar graphs. This is to be compared with Dijkstra's algorithm which runs in time $O(n \log n)$ in sparse graphs. (But note that for unweighted graphs, a breadth-first search tree from a vertex is a shortest path tree, so in that case, we can also compute shortest paths in time $O(n)$).

Theorem 1.8 (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.*

1.5.1 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 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 1.9. *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 6.

Lemma 1.10. *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

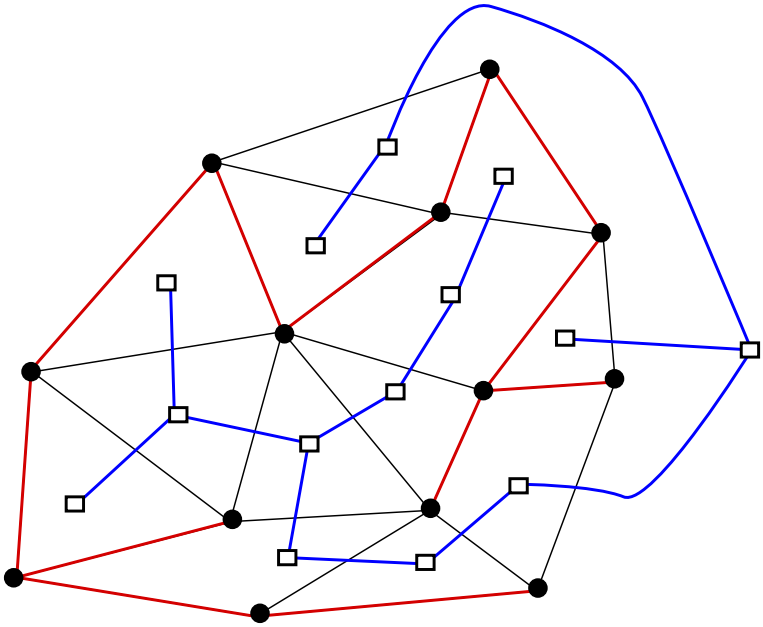

Figure 6: A spanning tree and the spanning co-tree. These are sometimes called ***interdigitating*** trees.

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. □

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 1.11. *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 1.9. 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 1.11), 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 1.11 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 to 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 1.11, 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. □

Note that this algorithm crucially relies on the fact that we work with non-simple graphs (even if the initial graph is simple, it becomes non-simple after contractions). This is why we rely on Lemma 1.11 and duality and not merely sparsity.

1.6 Minimum cut

Our second 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 1.12. *Let $G = (V, E)$ be a connected 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 $O(n \log^2 n)$ time, and even $O(n \log n)$ time if one uses Theorem 1.8.*

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 1.13. *$X \subseteq E$ 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 separating 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 is separating, and since the faces on each side of each edge are labelled "S" and "T", 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 set of 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 1.13 becomes:

Proposition 1.14. *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 7. 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.

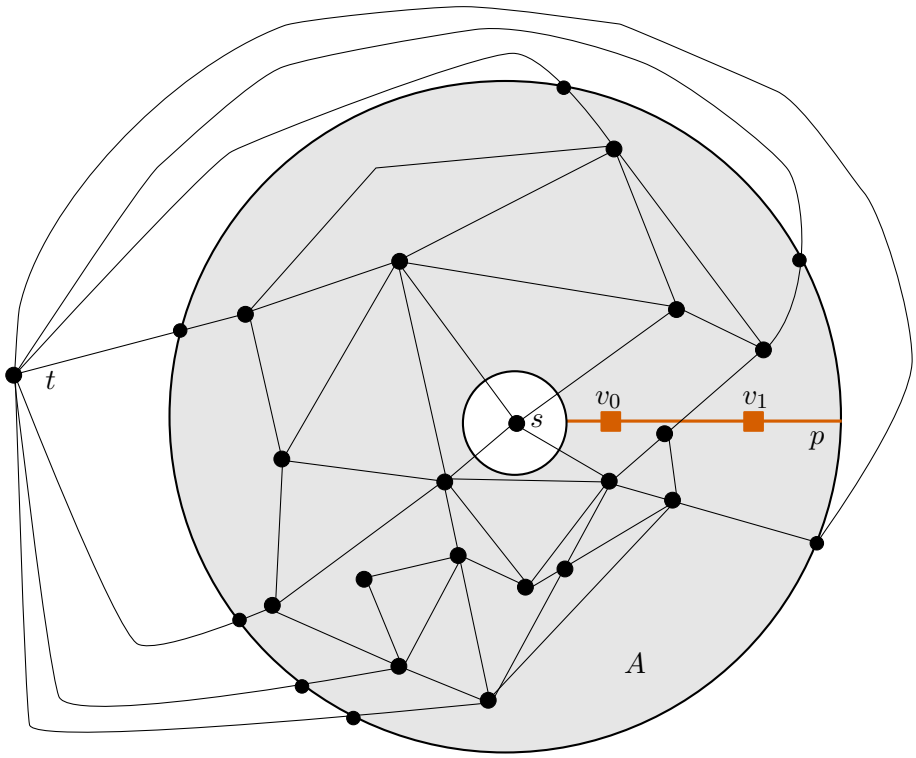

Figure 7: The annulus obtained after removing a small disk around s and t.

Lemma 1.15. *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 8. Let γ be such a curve. If we cut A along p, we obtain a topological disk B, where the path p got cut into two paths p' and p'' on the boundary. The curve γ, once cut on B, contains a simple path q connecting p' to p'', otherwise γ would not separate s from t. Now we reglue B 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 minimal cut. Since shortest paths in the dual graph can be computed in time $O(n \log n)$, this takes $O(n^2 \log n)$ time.

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

Lemma 1.16. *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 8. 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

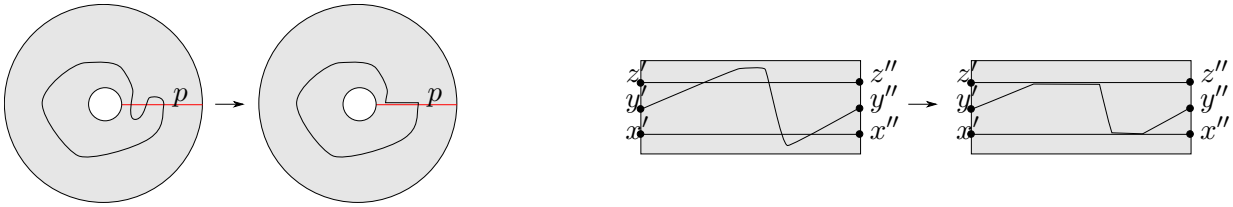

Figure 8: The two shortcutting arguments of Lemma 1.15 and Lemma 1.16.

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 B a shortest path p between the two vertices v' and v'' corresponding to v,
2. we reglue B 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.

We stop the recursion when we reach one of the following two base cases for the recursion: (1) 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, and (2) if there exists a face adjacent to both boundaries, we can compute a shortest cycle going through that face directly in time $O(n \log n)$.

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 "cross-metric" 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 1.12. 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 1.14, Lemmas 1.15 and 1.16, 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 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 edge only appears a constant number of times.

Therefore, at each level of the recursion, the annulus A is cut into k subannuli $A_1 \cup \ldots \cup A_k$ of total complexity $O(n)$, and thus solving the shortest path computations in all of them takes $O(n \log n)$ time (because the map $x \mapsto x \log x$ is concave).

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 $O(n \log n)$ time in total, so the total complexity is $O(n \log^2 n)$. Using linear-time shortest paths, this improve to $O(n \log n)$. □

2 Surfaces

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

2.1 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 9. 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.

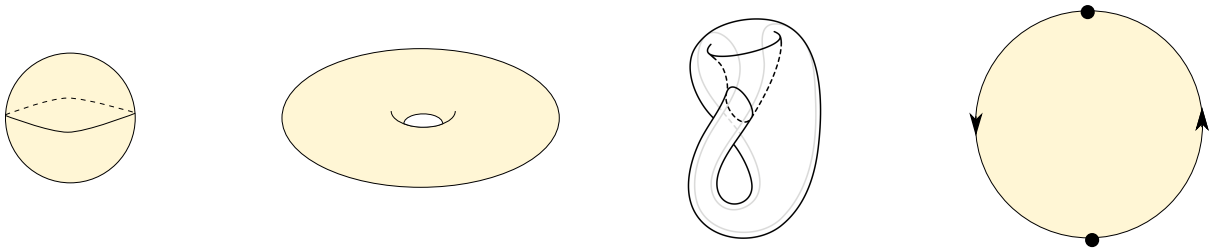

Figure 9: A sphere, a torus, a Klein bottle, and a projective plane (in the last one, the right boundary is identified to the left boundary in the direction indicated by the arrow).

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 2.1 (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 (see for example this mathoverflow thread), 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 2.2. *Any two triangulations of a surface have a common refinement.*

The same not-a-proof (does not) work(s): 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 them), 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 2.1 in hand, we are now ready to classify surfaces:

Theorem 2.3. *Every compact connected surface is homeomorphic to one of the surfaces 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 10, and Figure 11 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.

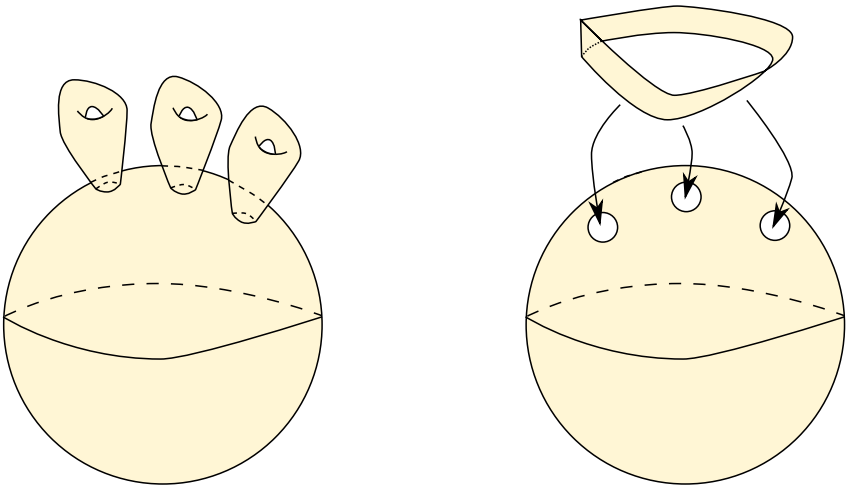

Figure 10: Attaching handles or Möbius bands to a sphere.

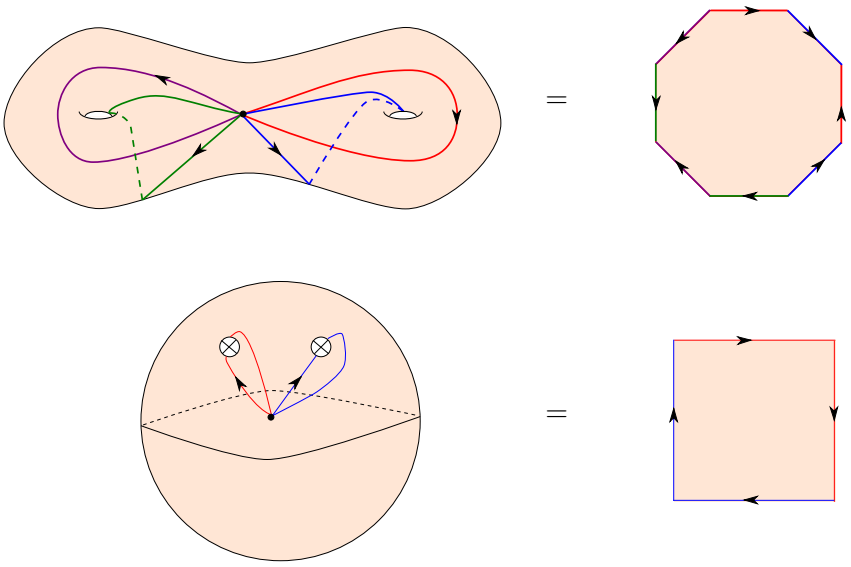

Figure 11: Polygonal schemata of the orientable and non-orientable surfaces. In the non-orientable cases, X denotes a disk on which a Möbius band has been glued.

Proof. Let S be a compact connected surface, and let G be a graph embedded on S, which exists by Theorem 2.1. 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 2.3 via ***cut-and-paste*** operations:

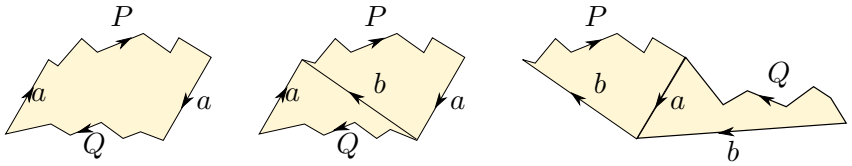

Figure 12: From $aPaQ$ to $bbP\bar{Q}$.

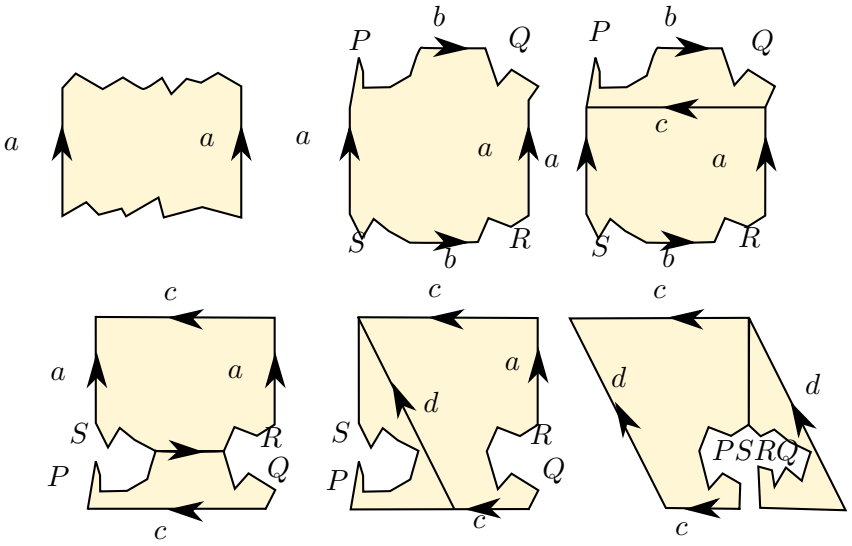

Figure 13: From $aPbQ\bar{a}R\bar{b}S$ to $cd\bar{c}\bar{d}PSRQ$.

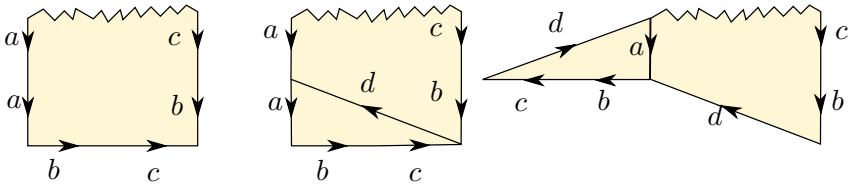

Figure 14: From $aabc\bar{b}\bar{c}$ to $\bar{d}\bar{c}\bar{b}d\bar{b}\bar{c}$.

- 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 12. 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 $dc\bar{d}\bar{c}PSRQ$, as pictured in Figure 13. 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 14, and then using the first step again this can be transformed into $eefffgg$. Inductively, we obtain a relation of the form 3.

This concludes the proof. □

During the class I had written in the statement of the theorem that the possibilities were exclusive (i.e., that these surfaces are all pairwise non-homeomorphic) but here I prefer to prove it separately, because the Euler characteristic deserves better than being hidden inside a 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 2.4. *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 2.3: 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.

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 2.2, 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 2.3 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 2.5. *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 2.3 are pairwise non-homeomorphic. Therefore we have found them all, and they are all different.

We conclude with a few remarks exploring how the topics we saw earlier generalize to surfaces:

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. If b is the number of connected components of the boundary, the Euler characteristic becomes $2-2g-b$ (orientable) or $2-g-b$ (non-orientable). 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 $S \setminus\!\!\!\!\succ G$.

Maps and data structures: 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 the same data structures used in planar graphs can be generalized to surfaces. It is important to understand the following subtlety though: when dealing with orientable surfaces, one can describe a cellularly embedded graph by just giving the ordered list of edges belonging to each face: for example if I know that the surface is orientable, I can describe a torus by the one-face graph $abab$. Indeed, there is a single way to identify the two a edges or the two b edges while preserving orientability. But with non-orientable surfaces, it is necessary to add an orientation information. This is for example the case with our polygonal schemes, in which aa is very much not the same surface as $a\bar{a}$. This orientation information has multiple avatars depending on the precise data structure in use (for example signatures within a rotation system, or quad-edges instead of half-edges).

Sparsity: Everything based on the Euler formula directly generalizes by using the Euler characteristic instead. This includes the crossing lemma, but take a look at this paper of mine if you care about the asymptotics with respect to g.

Duality: A graph cellularly embedded on a surface naturally induces a dual graph, as in the planar case. Non-cellularly embedded graphs do have faces, but lead to infinitely many possible choices for edges connecting these faces, even when identifying "similar" edges, so one should avoid using duality on those...

Subdivisions and minors: There is no analogue of Kuratowski's theorem for surfaces, but there is an analogue for Wagner's theorem[3]. However, the proofs are non-constructive, and the precise list of minors is unknown except for the projective plane.

Hanani-Tutte theorem: The Hanani-Tutte theorem generalizes to the plane and the torus but not in general. There is a known counter-example in genus 4. A weaker form of the theorem does hold though, see for example here.

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). Note that the finite list of excluded minors could be used to test embeddability (even if that list is unknown, it still proves the existence of an algorithm), but this leads to the delicate question of how to test minors efficiently. A very recent breakthrough allows do to that in almost-linear time.

Schnyder Woods: Wait for Luca's next class!

[3]Wagner's theorem is a sibling of Kuratowski's theorem stating that a graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor. Here, H is a minor of G if H can be obtained from G with a sequence of edge contractions, edge deletions and vertex deletions.

2.2 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 display more interesting topology. 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[4]: 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 2.6. *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. □

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) and geometry (hyperbolic geometry, curvature, Gauss-Bonnet theorem) impact the design of algorithms for topological problems on surface-embedded graphs.

2.3 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 2.6. 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, and 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 15 for some illustrations:

- The universal cover of a circle $\mathbb{S}^1$ is an infinite spiral over this circle.

[4]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.

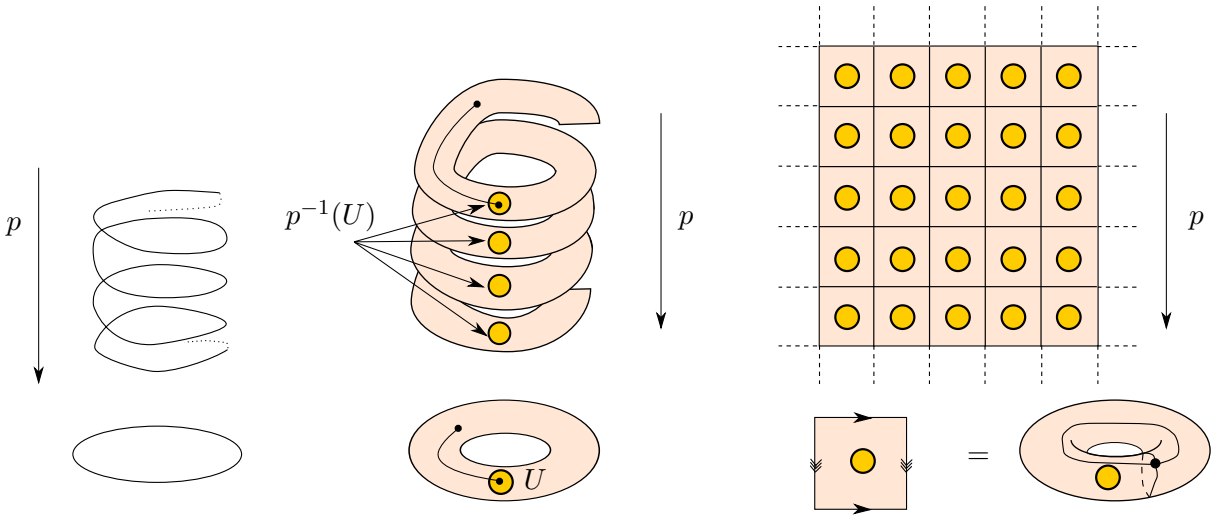

Figure 15: The universal covers of the circle, the annulus and the torus.

- 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.

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 15 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 2.7. *A closed curve γ on a surface lifts in the universal cover to a closed curve if and only if it is 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 a trivial curve, and the contraction on S lifts to a contraction in $\hat{S}$. □

The toroidal case: We first look at the easy example of the torus. Given a graph G cellularly embedded on the torus, and a closed walk w on that graph, we first contract edges in G until there is a single vertex and remove edges in G until there is a single face. This is the same is in the proof of the classification theorem of surfaces, except that we must take care to update the walk while doing these operations. This is straightforward for contractions, and only slightly less straightforward for deletions: by construction, we only delete edges adjacent to two faces, and thus any time the walk w uses that edge, it can be rerouted using along one of the adjacent faces. After these operations, the graph now consists of a single vertex, two edges and a square corresponding to the polygonal scheme $ab\bar{a}\bar{b}$.

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 16. This is a problem on words: this amounts to counting how many times the walk uses (or dually, crosses)

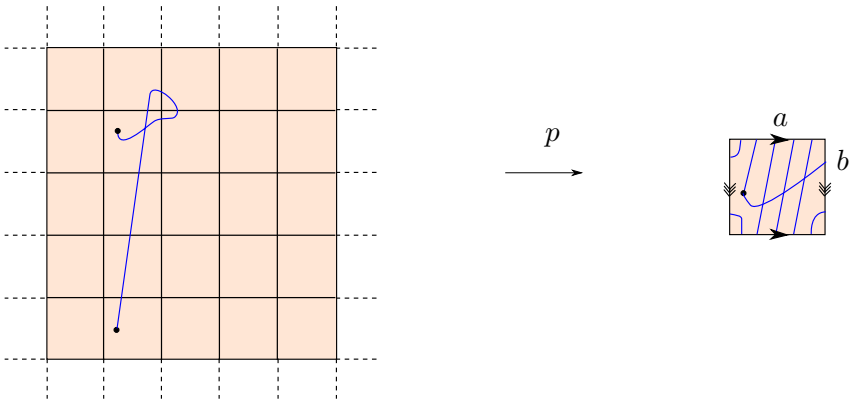

Figure 16: Lifting a closed curve on a torus. For illustration purposes, we dualized the tiling so that the curve crosses a and b instead of following them. In the universal cover, the curve lifts to a path, hence the curve is not contractible

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.

The hyperbolic case: 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 17 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 the walk on this hyperbolic tiling, 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 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 2.8. *Let γ be a closed curve on a graph G which is a cellularly embedded 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 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

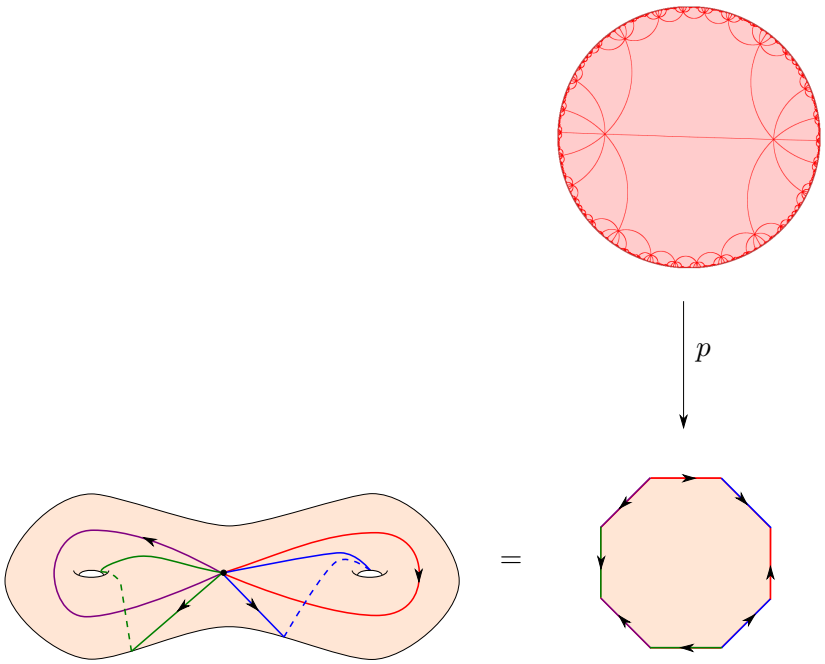

Figure 17: Covering a genus-two surface by a hyperbolic tiling of the open disk.

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

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

$$\tau(v) = 1/2 - \sum_{c \in v} \theta(c).$$

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:

Theorem 2.9. *For a graph G embedded on a surface S, 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:*

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

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.

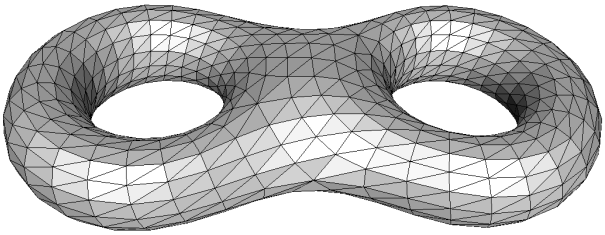

Figure 18: A double torus realized geometrically with Euclidean triangles in $\mathbb{R}^3$

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. $\square$

Let us comment a bit on this formula before using it. In order to make sense of the notions of angles and curvature, it makes sense to look at specific geometric instances of surfaces.

- One such instance is the case of a surface realized in $\mathbb{R}^3$ by Euclidean triangles pasted along their boundaries, as pictured in Figure 18. Such a surface is not purely topological and is also endowed with a piecewise-Euclidean structure inherited from the triangles. This structure endows each corners with angles. Since each triangle is actually Euclidean, the angles in each triangle sum up to π (which is 1 in our normalization), so the curvature in each triangle is zero. Then the Euler formula connects the curvature at the vertices with the topology of the surfaces.
- A different instance occurs if we consider a disk drawn on a hyperbolic tiling such as the one in Figure 17. There, the geometry also yields a notion of angle at each corner, and by construction the surface is planar so the angles sum to 2π (1 in our normalization) around each internal vertex, so there is no curvature there. However, the curvy edges lead to an angular defect in each face, and the curvature appears there. The curvature also occurs at the boundary vertices, depending on how many tiles such a boundary sees. The Gauss-Bonnet formula connects the sum of these facial and boundary curvatures to the topology of the surface.

The confusing and magical thing about the discrete Gauss-Bonnet formula is that these are just inspiring examples but the formula holds irrespective of any geometry: we get to choose the value of the angles $\theta(c)$ in the corners however we want. This is the case for the following proof where the angles are *not* taken to be their natural geometric values in the hyperbolic tiling.

Proof of Dehn's Lemma 2.8. 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 positive 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. Since edges adjacent to these convex vertices are on the boudary of the disk, $4g-3$ of these convex vertices must be consecutive. So some face has $4g-2$ consecutive edges on ∂D, which is strictly bigger than $2g$ for $g \geq 2$. □

With Lemma 2.8 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 by the walk. We first inductively remove all the spurs. Then we scan it for a 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 2.8, 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: Both in the toroidal and the hyperbolic variants, the problem boils down to a problem on words: deciding whether a given word reduces to a trivial word under spur reductions and a more complicated relation defining the surface. 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: 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 similar tools, see for example here.

2.4 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, but we will not have enough time to cover this in this course.

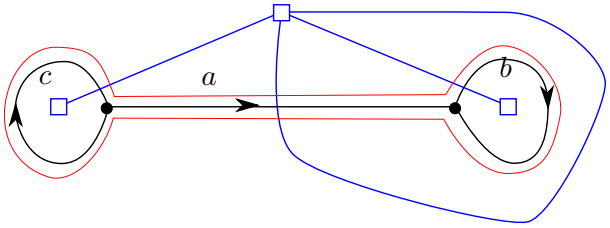

Figure 19: A graph and its dual. The walk $ab\bar{a}c$ is definitely not a cycle in the primal graph, but when we look at it in the cross-metric perspective and push it a bit, it is a nice simple closed curve (in red).

We will prove the following theorem:

Theorem 2.10. *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 19.

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 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 20 for an illustration.

Lemma 2.11. *The cut locus cuts S into a disk.*

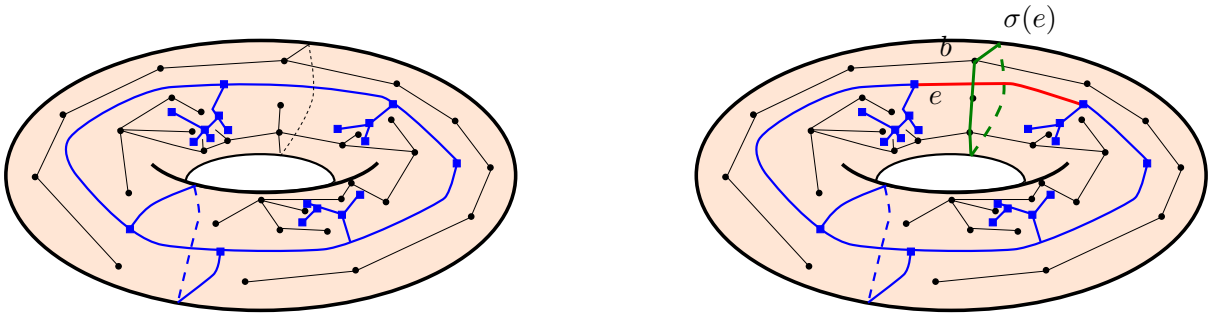

Figure 20: A shortest path tree, in black, defining a cut locus (in blue). Cutting the surface along the cut locus yields a disk. On the right, an edge of the cut locus corresponds to loop $\sigma(e)$ going through the basepoint b.

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 2.12. *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 2.13. *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 2.6. 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 2.11. □

We now have all the tools to prove the first half of Theorem 2.10:

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 2.14. *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 2.15. *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 2.16. *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 2.13 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.