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

Arnaud de Mesmay

These are the lecture notes for the last two classes of the course 2-38-1.

Some practicalities:

- The course is on Fridays, 12:45 to 15:45 in room 1004 of the Sophie Germain building.
- It will be graded with an exam.
- There will be an optional exercise sheet given on November 4, with points contributing extra credits for the final grade.

The last set of two lectures focuses on surfaces, which generalize the plane, and graphs embedded thereon. We will introduce surfaces, and use embedded graphs to prove the classification theorem, showing that, up to homeomorphism, they are classified by their number of handles (***genus***) and ***orientability***. We will then look into algorithms to deal with specific topological problems that arise with graphs embedded 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 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.

1 Surfaces

1.1 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 1. 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.

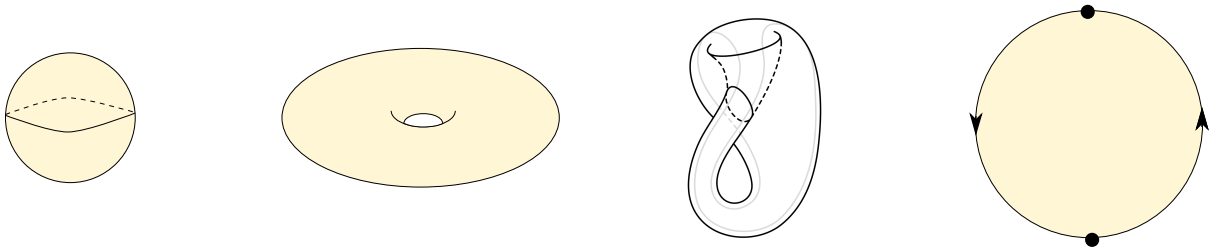

Figure 1: A sphere, a torus, a Klein bottle, and a projective plane (in the last one, the right boundary is identified to the left boundary in the direction indicated by the arrow).

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. For planar graphs, many properties rely in a more or less explicit way on Jordan's theorem (or rather the Jordan-Schoenflies theorem), which states that any simple closed curve bounds a disk. It implies that any connected plane graph cuts the plane into disks. This is easily seen not to be the case on surfaces, hence the following definition: a graph is ***cellularly embedded*** if every connected component of $S \setminus G$ is homeomorphic to an open disk. The following theorem shows that this is a non-vacuous definition on any compact connected surface.

Theorem 1.1 (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). A proof of Theorem 1.1 can be found here.

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 1.2. *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 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 a ***polygonal scheme***, which encodes the way that the disks are glued together (this is similar to ***combinatorial maps*** introduced in Luca Castelli Aleardi's lecture). 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 1.1 in hand, we are now ready to classify surfaces:

Theorem 1.3. *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. $a_1 b_1 \bar{a_1} \bar{b_1} \dots a_g b_g \bar{a_g} b_g$ *for some* $g \geq 1$,
3. $a_1 a_1 \dots a_g a_g$ *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 2, and Figure 3 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 1.1. 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.

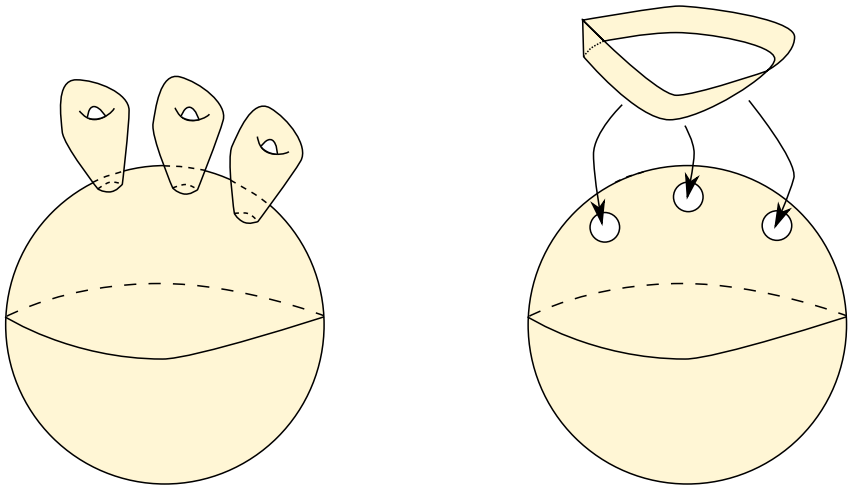

Figure 2: Attaching handles or Möbius bands to a sphere.

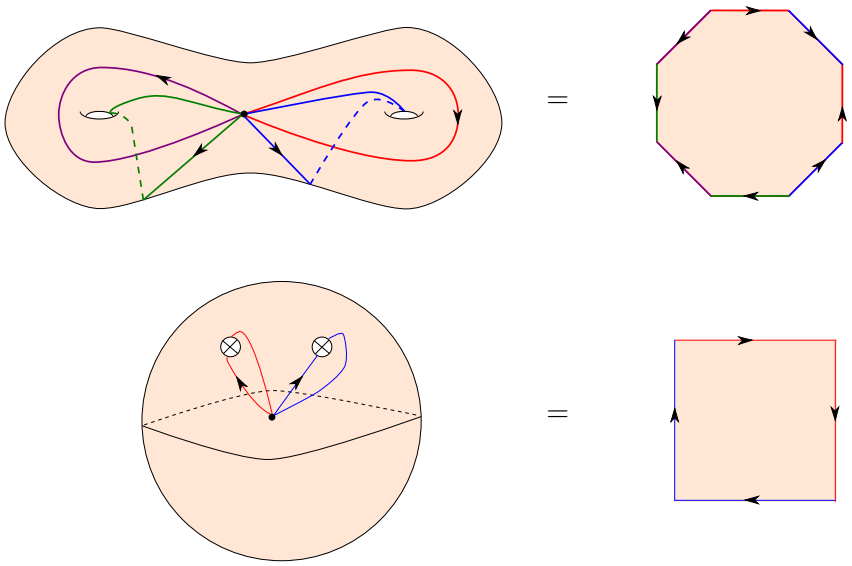

Figure 3: Polygonal schemata of the orientable and non-orientable surfaces. In the non-orientable cases, X denotes a disk on which a Möbius band has been glued.

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 1.3 via ***cut-and-paste*** operations:

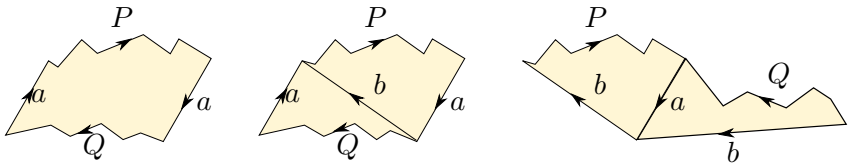

Figure 4: From $aPaQ$ to $bbP\bar{Q}$.

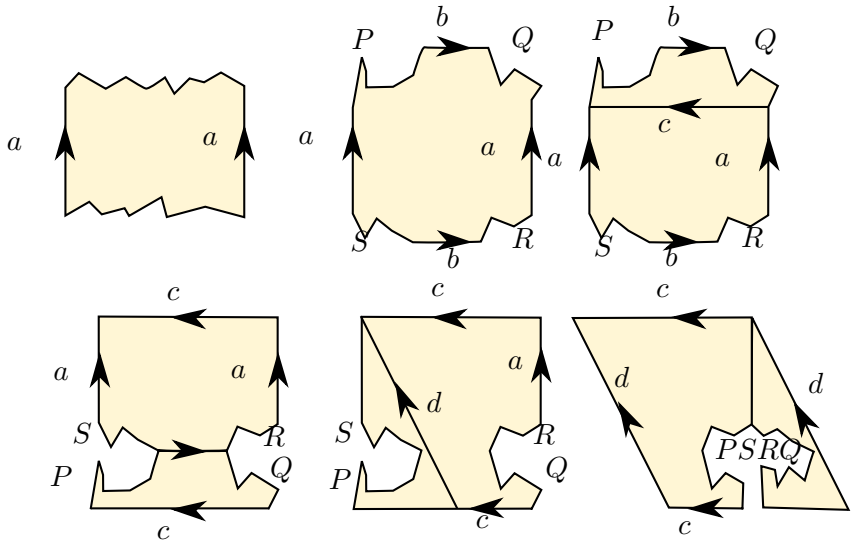

Figure 5: From $aPbQ\bar{a}R\bar{b}S$ to $cd\bar{c}\bar{d}PSRQ$.

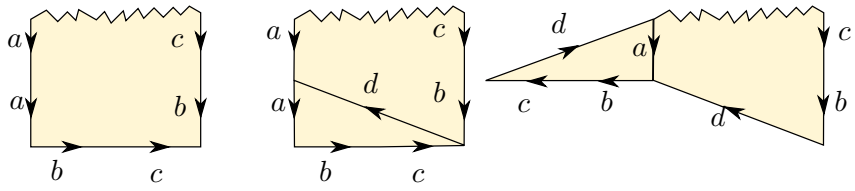

Figure 6: From $aabc\bar{b}\bar{c}$ to $d\bar{c}\bar{b}d\bar{b}\bar{c}$.

- 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 4. 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 5. 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 $d\bar{c}\bar{b}d\bar{b}\bar{c}$, as in Figure 6, 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. □

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 1.4. *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 1.3: 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 1.2, 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 1.3 shows that the sphere has Euler characteristic 2 (therefore recovering the Euler formula for planar graphs), 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 1.5. *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 1.3 are pairwise non-homeomorphic. Therefore we have found them all, and they are all different.

Exercise 1.6.

Prove that every graph can be cellularly embedded on some surface.

Provide an example of two cellular embeddings of the same graph G on two different surfaces.

What is the surface described by the polygonal schemes $abc\overline{abc}$? $abcd\overline{abcd}$?

What is the computational complexity of recognizing a surface from the polygonal scheme of a cellularly embedded graph?

We conclude with a few remarks:

Duality: A graph cellularly embedded on a surface naturally induces a dual graph, as in the planar case, but be careful that the dual graph is not well-defined for graphs that are not cellularly embedded.

Subdivisions and minors: There is no analogue of Kuratowski's theorem for surfaces, but there is an analogue for Wagner's theorem, stating that the family of graphs embedded on a fixed surface is characterized by a fainite family of excluded minors[1]. However, the proofs are non-constructive, and the precise list of minors is unknown except for the projective plane.

Schnyder Woods: Generalizing Schnyder Woods to surfaces of genus at least 2 is an open problem.

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 is technical and intricate, a more recent one by Kawarabayashi, Mohar and Reed is simpler but requires strong familiarity with graph minor theory techniques).

1.2 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 1.7. Show that on an orientable surface of genus $g > 0$, there are exactly $\lfloor g/2 \rfloor + 2$ non-homeomorphic[2] 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[3]: 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 1.8. *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. We will address two basic algorithmic problems related to non-trivial curves:

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

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.

[1] A graph H is a minor of G is H can be obtained from G by removing vertices, removing edges or contracting edges

[2] 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".

[3] 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.

1.2.1 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, since 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.

Shortest curve in an annulus: min-cut in planar graphs. We first look at the problem of computing a shortest non-contractible curve in the simplest non-trivial surface: the annulus. It turns out that in that case, the problem is equivalent to computing a minimum-cut in a planar graph: 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 1.9. *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 1.10. *$X \subseteq E$ 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 (this relies on 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 separates s and t. By minimality, C^* is that cycle. □

This proposition transforms a *combinatorial* problem into a *topological* one and vice-versa: if we *remove* the faces s^* and t^* from the dual graph, we obtain a surface homeomorphic to an annulus, finding the minimum-cut between s and t amounts to finding the shortest cycle that goes around this annulus, i.e., the shortest non-contractible cycle. 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

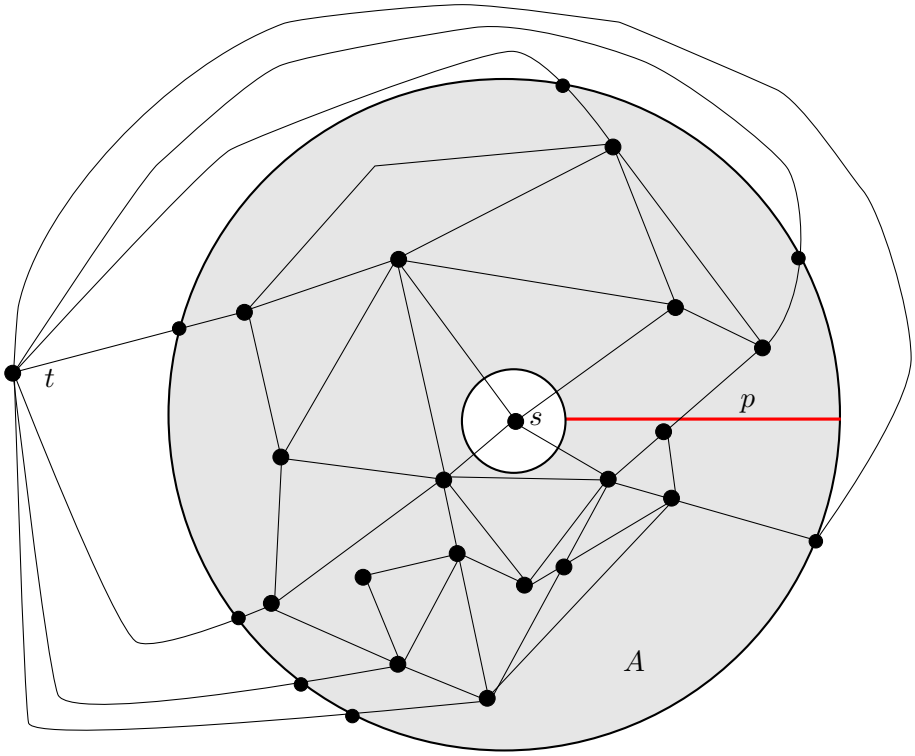

Figure 7: The annulus obtained after removing a small disk around s and t.

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 1.10 becomes:

Proposition 1.11. *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.

Lemma 1.12. *Some shortest closed curve separating the two boundaries of A is simple and crosses p exactly once.*

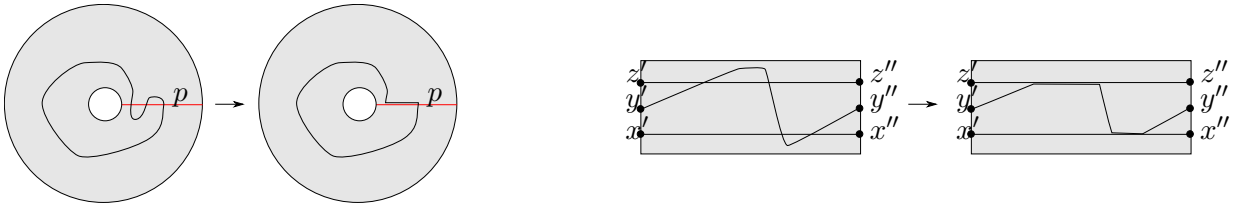

Figure 8: The two shortcutting arguments of Lemma 1.12 and Lemma 1.13.

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 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 minimal cut. Since shortest paths in the dual graph can be computed in linear time (this is non-trivial, but one can use Dijkstra's algorithm to compute them in near-linear time instead), this takes $O(n^2)$ time.

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

Lemma 1.13. *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 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 suggests 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.

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 $O(n)$.

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 1.9. 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 1.11, Lemmas 1.12 and 1.13, 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 $e_1, \dots, e_\ell$, 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 $O(n \log n)$.

□

Computing shortest non-contractible/non-separating cycles on surfaces of genus g
We will prove the following theorem:

Theorem 1.14. *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

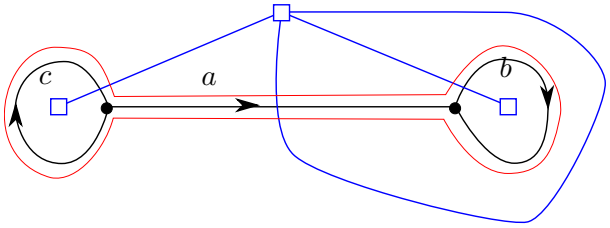

Figure 9: A graph and its dual. The walk $ab\bar{a}c$ is definitely not a cycle in the primal graph, but when we look at it in the cross-metric perspective and push it a bit, it is a nice simple closed curve (in red).

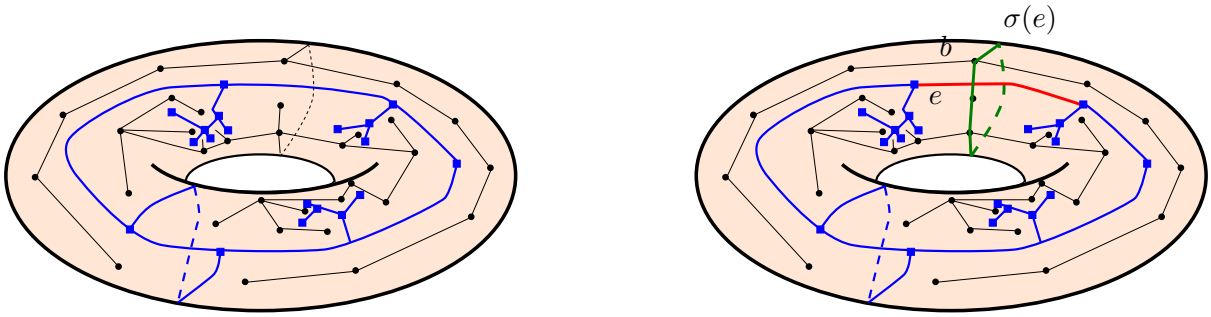

Figure 10: A shortest path tree, in black, defining a cut locus (in blue). Cutting the surface along the cut locus yields a disk. On the right, an edge of the cut locus corresponds to loop $\sigma(e)$ going through the basepoint b.

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 9.

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. See Figure 10 for an illustration.

Lemma 1.15. *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 1.16. *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.

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

Lemma 1.17. *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 1.8. 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 1.15. □

We now have all the tools to prove the first half of Theorem 1.14:

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 1.18. *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 1.19. *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 1.20. *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 1.17 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.

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.

1.2.2 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 1.8. 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 11 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.

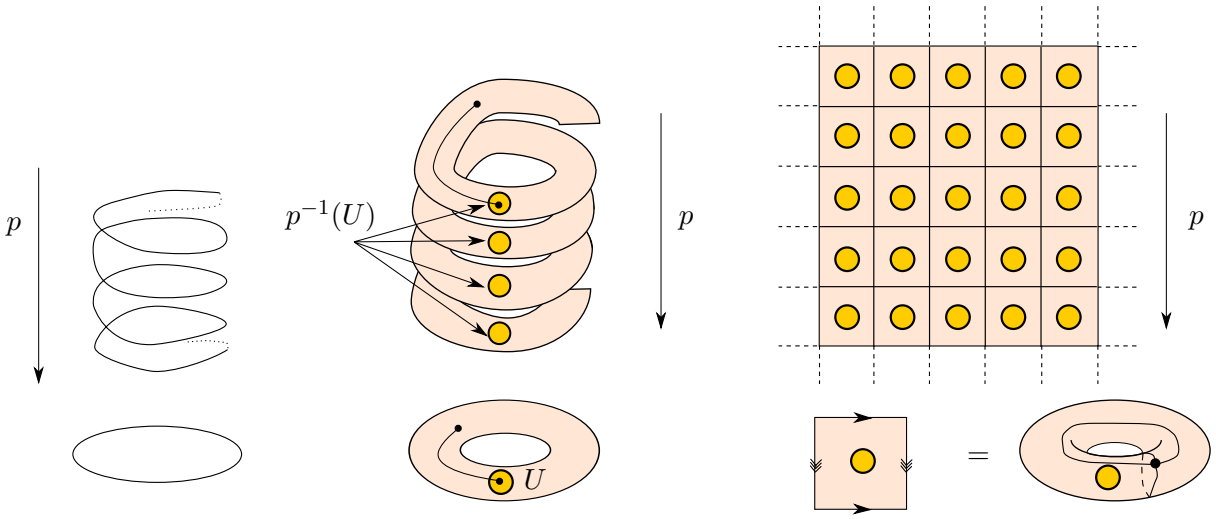

Figure 11: The universal covers of the circle, the annulus and the torus.

- 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 1.21. 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 11 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 1.22. *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 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 12. 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

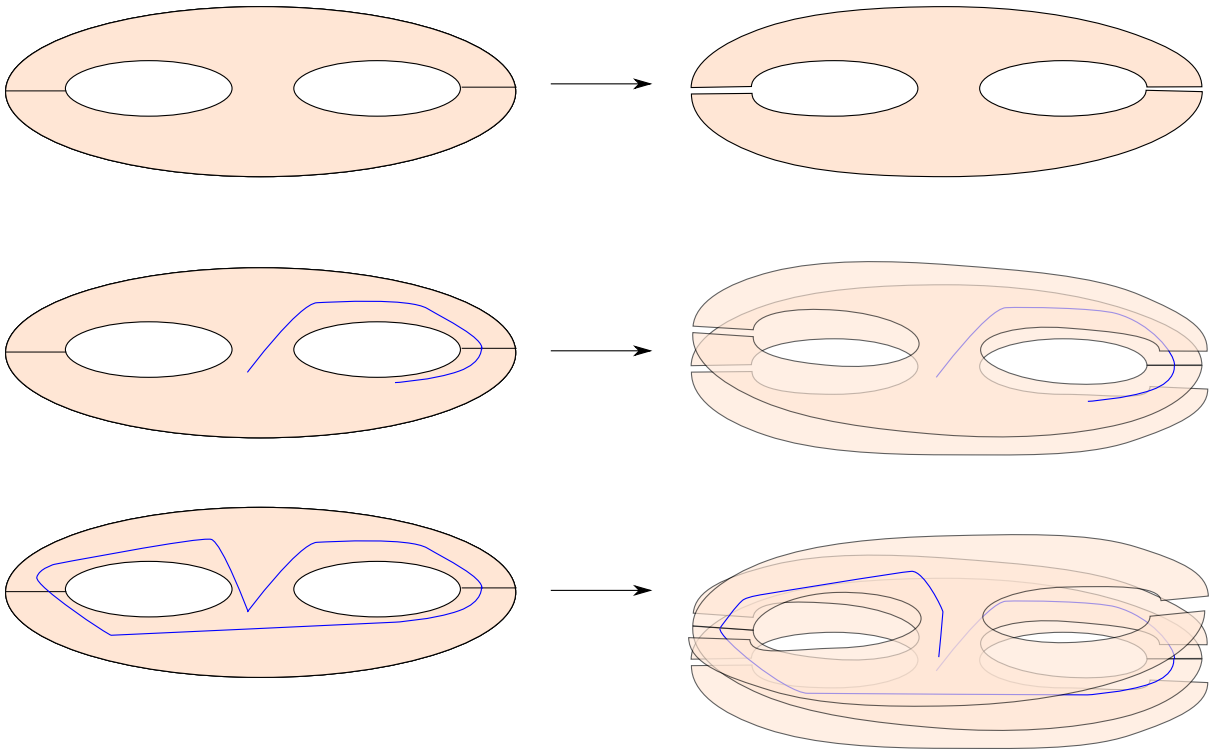

Figure 12: Lifting a closed curve on a disk with with two holes. Note that in the cover, the path starts on the first "floor" and ends on the third floor. Hence the curve is not contractible

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 13. We emphasize here 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 14 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

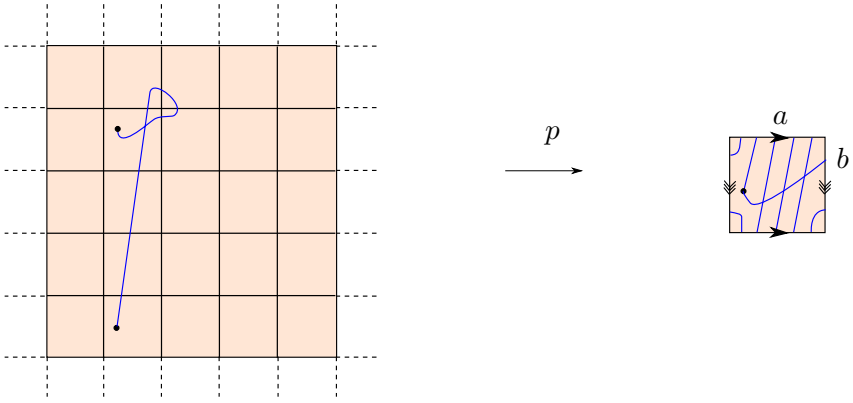

Figure 13: Lifting a closed curve on a torus. In the cover, it lifts to a path, hence the curve is not contractible

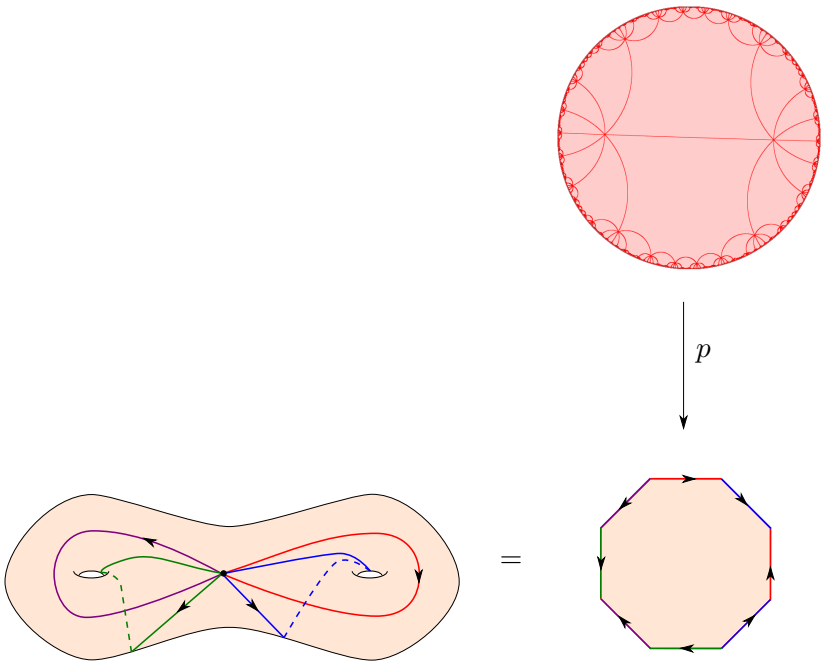

Figure 14: Covering a genus-two surface by a hyperbolic tiling of the open disk.

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 1.23. *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 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 1.24. *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 1.23 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 1.23, 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.

1.3 Separators in embedded graphs

If time allows, we will also talk a bit about small separators for embedded graphs, which is an absolutely central feature in algorithm design. 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 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 1.25. *Let G be a planar graph with n vertices. Then one can compute in $O(n)$ time a separator for G of size $O(\sqrt{n})$.*

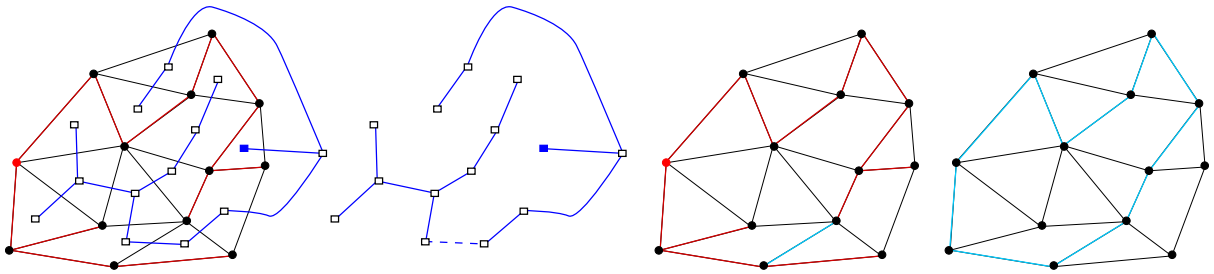

Figure 15: An edge separating the dual co-tree into balanced parts induces a balanced separator in the primal graph.

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

Proof of Theorem 1.25. 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 tree in the dual of G that is made of the dual of edges not in T (this is sometimes called a ***cotree***). 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 is 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 15 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 the 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 16):

- 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.

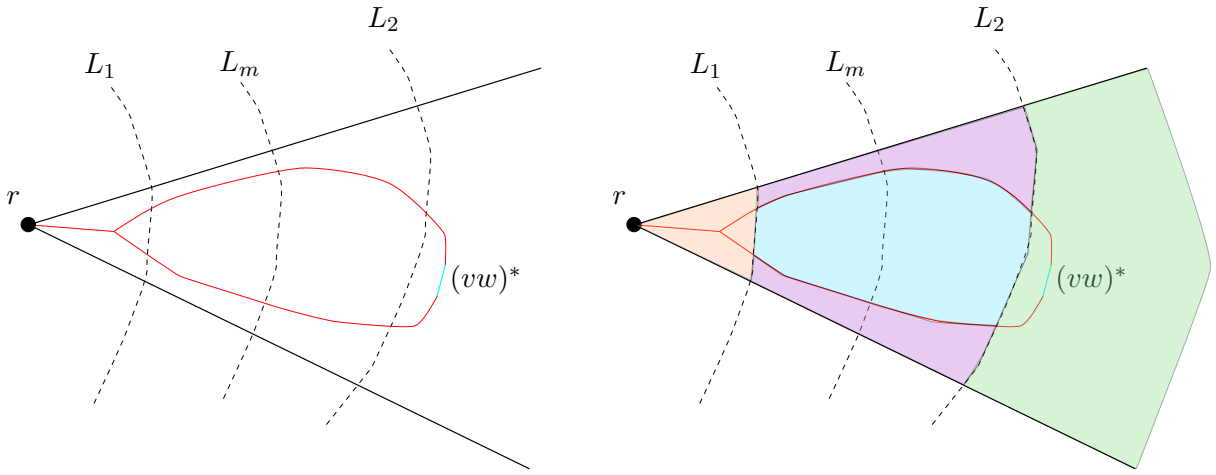

Figure 16: The separator is made of L_1, L_2, and the subpaths of the fundamental cycle between them. It cuts the graph into at most four small components.

- 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 application of this theorem, using a divide and conquer framework.

Theorem 1.26. *In a planar directed*[4] *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,
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. □

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

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 1.27 (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.*

More generally, many problems for which the best algorithm runs in time $2^{O(n)}$ or $n^{O(n)}$ can be sped up on planar graphs to obtain algorithms running in time $2^{O(\sqrt{n})}$ or $n^{O(\sqrt{n})}$. This is sometimes called the Square root phenomenon. A good framework to prove such upper bounds is to introduce the notion of treewidth and use planar separators to prove that planar graphs have treewidth $O(\sqrt{n})$.

The generalization of the planar separator theorem for graphs embedded on surfaces is as follows.

Theorem 1.28. *An n-vertex graph embedded on a surface of genus g admits an $O(\sqrt{gn})$-sized balanced separator.*

The following proof works by planarizing the graph and applying the separator theorem. It can be turner into a linear-time algorithm.

Proof. We can assume by adding edges if necessary that the graph is connected and cellularly embedded. Then we start as in the planar case by computing a BFS tree from any vertex. As in that case, if the L_m level, which is the one containing the $n/2$th vertex, has size $O(\sqrt{gn})$, then we are done. Otherwise, as before we denote by L_1 and L_2 the first level below (respectively) above L_m having $O(\sqrt{gn})$ vertices. Note that there are $O(\sqrt{n/g})$ levels between L_1 and L_2, since otherwise there would be more than n vertices. Now, we contract all the vertices in L_1 and L_2 into a single vertex r, and look at the resulting graph H. Therefore, up to considering this graph H instead of the input graph G (note that the surface on which H is embedded has genus at most g), we can assume that the input graph has a breadth-first search tree of depth $O(\sqrt{n/g})$ rooted at r.

Now, as in the proof of classification of surfaces, let us remove edges in the graph G until there is a single face. Furthermore, let us remove all the vertices of degree one and their incident edges. Let us denote the resulting graph by G', and by e and v its edges and vertices. By the Euler formula, we have $e = v - 1 + 2g$. The BFS tree partitions these edges into the tree and the non-tree edges, and there are thus $v - 1$ edges of the first kind and $2g$ edges of the second kind. Now, any vertex of H is in some path of that tree, and any path in the tree contains at most $O(\sqrt{n/g})$ vertices and connects the root to a non-tree edge. Thus there are at most $4g$ of those paths, and thus there are $O(\sqrt{gn})$ vertices in the graph G'. Removing all these vertices cuts the surface into a disk. Then we can use the planar separator theorem (Theorem 1.26) to find a balanced separator with $O(\sqrt{n})$ vertices. All in all, we have used $O(\sqrt{gn})$ vertices for the first reduction, $O(\sqrt{gn})$ vertices for the planarization and $O(\sqrt{n})$ vertices for the final planar separator, and thus we have found a balanced separator of size $O(\sqrt{gn})$. □

By applying Theorem 1.28, we can similarly improve the running times of many algorithms using divide-and-conquer approaches.