FINDING SHORTEST NON-TRIVIAL CYCLES IN DIRECTED GRAPHS ON SURFACES*

Sergio Cabello,† *Éric Colin de Verdière,*‡ *and Francis Lazarus*§

Abstract. Let D be a weighted directed graph cellularly embedded in a surface of genus g, orientable or not, possibly with boundary. We describe algorithms to compute shortest non-contractible and shortest surface non-separating cycles in D, generalizing previous results that dealt with *undirected* graphs.

Our first algorithm computes such cycles in $O(n^2 \log n)$ time, where n is the total number of vertices and edges of D, thus matching the complexity of the best general algorithm in the undirected case. It revisits and extends Thomassen's *3-path condition*; the technique applies to other families of cycles as well.

We also provide more efficient algorithms in special cases, such as graphs with small genus or bounded treewidth, using a divide-and-conquer technique that simplifies the graph while preserving the topological properties of its cycles. Finally, we give an efficient output-sensitive algorithm, whose running time depends on the length of the shortest non-contractible or non-separating cycle.

1 Introduction

Graphs embedded in the plane have been studied in depth. For several computational problems, we know algorithms for planar graphs that are faster than a generic algorithm for arbitrary graphs: shortest paths [30], minimum spanning trees [39], (sub)graph isomorphism [18, 19], maximum flow, minimum cut [31, 43], etc. For some NP-hard problems, one can obtain approximation schemes when restricting to planar graphs [1]. Most of, if not all, these algorithms ultimately rely on topological properties of the plane and on the Jordan curve theorem.

Graphs embedded on surfaces are a natural generalization of plane graphs that have attracted much attention. From a theoretical perspective, they play a prominent role in the graph minor theory developed by Robertson and Seymour (see e.g. Mohar and Thomassen [40]). From an applied perspective, efficient algorithms are needed to process

*A preliminary version appeared in Proc. ACM Symp. on Computational Geometry, 2010 [7]. Research partially supported by the Slovenian Research Agency, program P1-0297 and project BI-FR/09-10-PROTEUS-014, funded by the French Ministry of Foreign and European Affairs.

†*Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia*, sergio.cabello@fmf.uni-lj.si

‡*Département d'informatique, École normale supérieure, CNRS, Paris, France*, Eric.Colin.de.Verdiere@ens.fr

§*GIPSA-Lab, CNRS, Grenoble, France*, Francis.Lazarus@gipsa-lab.grenoble-inp.fr

non-planar surfaces: texture mapping, topological denoising, remeshing, and visualization (see e.g. the discussion by Erickson and Har-Peled [23]); in these scenarios, graphs are often weighted (each edge has a non-negative *length*). In particular, the computation of shortest cycles of a certain topological type has been largely considered, also because it serves as a basic tool for developing algorithms on the class of graphs embeddable on a fixed surface [16,33,34,36]. Among the basic topological types are the non-contractible cycles (which cannot be continuously deformed to a point on the surface) and the non-separating cycles (which do not split the surface into two pieces).

Most of these results, however, are restricted to undirected graphs. In this paper, we study the problem of finding shortest non-contractible or shortest non-separating cycles in weighted directed graphs embedded on surfaces. We will generically refer to such cycles as shortest *non-trivial* cycles.

Shortest non-trivial cycles in undirected graphs on surfaces. Quite a few algorithms have been described to compute shortest non-trivial cycles. In most cases, the graph is undirected and weighted. Furthermore, the weights are always assumed to be non-negative, as in the present paper. It is also assumed that the graph is cellularly embedded on the surface, i.e., that each face of the graph embedding is a disk. Here and in the sequel, n is the total number of vertices and edges of the graph.

Thomassen [46] proves that shortest non-trivial cycles in unweighted, undirected graphs can be found in near-cubic time (see also Mohar and Thomassen [40, Chapter 4]). His approach works more generally for certain families of cycles that satisfy the so-called *3-path condition* which we revisit in this paper.

Erickson and Har-Peled [23] give an improved algorithm that runs in $O(n^2 \log n)$ time, for weighted undirected graphs with n edges; this is the best current result when the measure of complexity is the input size. Cabello and Mohar [9] propose considering the problem parameterized by the genus. Kutz [37] proposes an algorithm for orientable surfaces of genus g with running time $O(g^{O(g)} n \log n)$, subsequently improved to $O(g^2 n \log n)$ by Cabello et al. [6]. Italiano et al. [31] provide a $g^{O(g)} n \log\log n$-time algorithm, finally improved to $2^{O(g)} n \log\log n$ by Fox [27], which is the best current result if the genus is small. In another article [8] we consider the restricted scenario of *unweighted*, undirected graphs, and show that a shortest non-trivial cycle can be found in $O(gnk)$ time, where k is the length of the output cycle.

Several authors also considered computing shortest cycles with different topological properties on a surface, such as *splitting cycles* [11], which split the surface into non-trivial components, and *essential cycles* [25], which cannot be deformed to lie in a boundary component. The techniques of the present paper do not apply to these problems.

Directed graphs. This paper provides several algorithms for computing shortest non-trivial cycles in weighted directed graphs on surfaces. This is the same as looking for shortest non-trivial closed walks, since (as is easy to prove; see Lemma 4.1 below) shortest non-trivial closed walks are (simple) cycles; that is, they do not repeat any vertex. However, shortest non-trivial walks through a given vertex may have self-crossings, as on Figure 1, even if

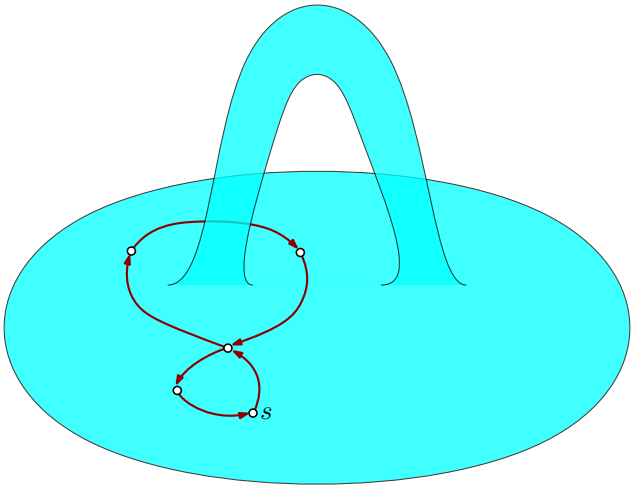

Figure 1: The shortest non-trivial closed walk through s has self-crossings. We can add arbitrarily many vertices and edges of large weight to make the embedding cellular.

	non-separating	non-contractible
This paper	$O(g^{1/2}n^{3/2}\log n)$	$O(g^{1/2}n^{3/2}\log n)$
Erickson and Nayyeri [24]	$2^{O(g)}n\log n$	
Erickson [22]	$O(g^2 n\log n)$	$g^{O(g)}n\log n$
Fox [27]		$O(g^3 n\log n)$

Table 1: Summary of various algorithms for computing shortest non-separating or non-contractible cycles in directed graphs on surfaces without boundary.

shortest paths are unique.

Our main result is an $O(g^{1/2}n^{3/2}\log n)$-time algorithm to compute a shortest non-contractible or non-separating cycle in a weighted directed graph on an arbitrary surface (Section 6). Since the preliminary version of our work [7] was published, three papers have given improved algorithms for graphs on surfaces of small genus; see Table 1 for a summary. Erickson and Nayyeri [24] have provided a $2^{O(g)}n\log n$-time algorithm for computing a shortest non-separating cycle. This has been improved to $O(g^2 n\log n)$ by Erickson [22], who also provides a $g^{O(g)}n\log n$-time algorithm for computing a shortest non-contractible cycle on surfaces without boundary. More recently, Fox [27] gives a better algorithm for the non-contractible case, with running time $O((g^3+gb)n\log n)$, where b is the number of boundary components of the surface.

In particular, our algorithms are the fastest known ones if $g=\Omega(n^{1/5})$ (non-contractible case) and $g=\Omega(n^{1/3})$ (non-separating case) for surfaces without boundary. If g is smaller compared to n, then the recent works [22, 24, 27] remain competitive.

Other works on algorithms for directed graphs on surfaces. Very few results are available concerning algorithms for directed graphs on general surfaces. Here are the only instances of which we are aware. The data structure of Cabello et al. [6] represents distances from

the vertices of a face in an embedded directed graph. Chambers et al. give an algorithm to compute maximum flows in directed surface-embedded graphs [12]. Erickson shows that the best known algorithm to compute maximum s-t flows in planar directed graphs [4] does not extend to directed graphs on the torus [21, Section 3].

Overview of our results. In Section 3, we first revisit the concept of the 3-path condition given by Thomassen [46]. Specifically, we extend this notion to a family $\mathbb{L}$ of closed loops based at a given vertex s in a directed graph. Note that the loops need not be simple and that the graph is directed and needs not be embedded. It is convenient to think of the elements of $\mathbb{L}$ as "trivial" loops. This concept captures and extends the families of topologically trivial loops we want to study for embedded graphs. We provide a generic algorithm to find a shortest loop based at s *outside* the family $\mathbb{L}$. The algorithm basically produces a small family of loops that should contain a shortest desired loop, and calls an oracle for each loop to decide membership in $\mathbb{L}$. This algorithm is extended in Section 4 to the computation of a shortest closed walk (without fixing a basepoint).

We then move, in Section 5, to the particular case of topologically non-trivial loops on surfaces. Here we use techniques from our other paper [8] to determine in amortized constant time whether a given candidate loop based at s belongs to $\mathbb{L}$. This leads to an algorithm that finds the shortest non-trivial loop through s in $O(n \log n)$ time, and (repeating the procedure for each vertex) a shortest non-trivial cycle in $O(n^2 \log n)$ time, thus matching the complexity of the best general algorithm for solving this problem in the undirected case [23].

In Section 6, we study the case where $g = o(n)$ or the treewidth is bounded. We develop an algorithm based on a somehow *topologically preserving divide-and-conquer*. The key point is to use small separators in the graph to divide the problem into balanced subproblems, but each subproblem is kept in the original surface to avoid changing the topological character of the closed walks. This is reminiscent of the numerous divide-and-conquer algorithms for planar graphs using cycle separators; see for example [10, 26, 35, 43]. However, in these cases, the preservation of the topology is implicit. We believe that our new approach could be useful for other problems as well. In our case, this gives a subquadratic algorithm for bounded-genus graphs (see also the recent works [22, 24, 27]), and a near-linear-time algorithm for bounded-treewidth graphs.

In Section 6.3, we also give an $O(bn \log n)$-time algorithm for the case where the surface is the plane with b boundaries.

Finally, we provide an efficient output-sensitive algorithm for computing shortest non-trivial cycles on surfaces: The algorithm runs in $O(gnk)$ time, where k is the length of the output cycle. This matches the running time of the best algorithm in the undirected case [8].

2 Preliminaries

We introduce the background material used in this paper.

2.1 Graph Theory Definitions and Notations

Graphs considered in this paper may have parallel edges and loop edges. We recall some usual definitions related to walks in graphs; we want to emphasize that all walks considered here are *oriented*. Specifically, let G be a (directed or undirected) graph. We denote the vertex set of G by $V(G)$ and the edge set of G by $E(G)$. Given two vertices x and y of G, we denote by xy the edge oriented from x to y in G. In presence of parallel edges or loop edges, this notation is ambiguous, but it will always be clear from the context which edge xy is meant.

A *walk* is either *constant*, i.e., reduced to a vertex, or is a sequence of edges $e_1, \ldots, e_m$ such that the target of e_i is the source of e_{i+1}, for $i = 1, \ldots, m-1$. The walk is *closed* when it is constant or when the target vertex of e_m coincides with the source vertex of e_1. A *cycle* in G is an oriented closed walk without repeated vertices; a cycle has no distinguished vertex. In contrast, a *loop* based at a vertex s of G is an oriented closed walk with a distinguished occurrence of s. If every oriented edge xy of G has a non-negative *length* (or *weight*) $\ell(xy)$, the *length* of a walk α is the sum of the weights of the corresponding oriented edges (counted with multiplicity). It is denoted by $\ell(\alpha)$. Given walks α and β, we denote by α^{-1} the reversal of α, and by $\alpha \cdot \beta$ the concatenation of α and β.

2.2 Topological Background

We review some basics of the topology of surfaces as needed for Section 5 and thereafter. See Hatcher [29], Massey [38], or Stillwell [44] for a comprehensive treatment.

A *surface* (or 2-manifold) Σ possibly with boundary is a compact, connected, topological space where each point has a neighborhood homeomorphic either to the plane or to the closed half-plane; the points without neighborhood homeomorphic to the plane comprise the *boundary* of Σ. A surface is *non-orientable* if it contains a subset homeomorphic to the Möbius band, and *orientable* otherwise. Here and in the sequel, surfaces are considered up to homeomorphism; in particular, a *disk* is just a surface homeomorphic to the standard unit disk in $\mathbb{R}^2$.

An orientable surface is homeomorphic to a sphere where g disjoint disks are removed, a handle (a torus with one boundary component) is attached to each of the remaining g boundary circles, and then b disjoint disks are removed, for unique integers $g, b \geq 0$. A non-orientable surface is homeomorphic to a sphere where g disjoint disks are removed, a Möbius band is attached to each of the remaining g circles, and then b disjoint disks are removed, for unique integers $g \geq 1$ and $b \geq 0$. In both cases, g is called the *genus* of the surface. $\overline{\Sigma}$ denotes the surface without boundary obtained by attaching a disk to each boundary component of Σ.

An *embedding* of a graph G (viewed as an undirected graph) in a surface Σ is a drawing of G on Σ without crossings. More formally, the vertices of G are mapped to distinct points of the interior of Σ; each edge is mapped to a path in the interior of Σ, such that the endpoints of the path agree with the points assigned to the vertices of that edge. Moreover, all the paths must be without intersection or self-intersection except, of course,

at common endpoints. We sometimes identify a graph G with its embedding on Σ. The *faces* of G are the connected components of the complement of the image of G.

We only consider *cellular embeddings* of a graph. By this, as in [8], we mean that $\overline{\Sigma}$ minus the image of the graph is a union of disjoint disks. In particular, each face of a cellular embedding on Σ is homeomorphic to an open disk with zero, one, or more disjoint open disks removed; the boundaries of these open disks belong to the boundary of Σ. Note that this notion of cellular embedding is slightly more general than the usual one, and we exploit this difference in Section 6. Also note that every graph that is cellularly embedded must be connected.

Let G be a graph cellularly embedded on Σ, with V vertices, E edges, and F faces; then *Euler's formula* states that $V-E+F=2-2g$ if Σ is orientable, and $V-E+F=2-g$ if Σ is non-orientable. In particular, $E=O(V+F+g)$. To simplify our complexity results, we sometimes introduce $n=V+E$; by Euler's formula, we have $g=O(n)$ and $F=O(n)$.

We assume that the embedding is represented in a suitable way, like for example the *gem representation*, using the incidence graph of *flags* (vertex-edge-face incidences) discussed by Eppstein [20], or rotation systems [40]. For orientable surfaces, one can also use the doubly connected edge list (DCEL) that is customary in computational geometry [15]. In order to deal with surfaces with boundary, we store the embedding of G on $\overline{\Sigma}$, and mark within each face of the embedding the number of boundary components of Σ it contains.

We use the standard topological notions of homotopy and homology (say with $\mathbb{Z}/2\mathbb{Z}$ coefficients), which will not be formally defined here. Intuitively, two paths or loops are *homotopic* if one can be deformed continuously on the surface to the other one, keeping the endpoints fixed during the deformation. A loop is *contractible*, or *homotopically trivial*, if it is homotopic to the constant loop. If α and β are contractible loops with the same basepoint, then so are α^{-1} and $\alpha\cdot\beta$. Similarly, the reverse and the concatenation (formally, the sum) of *null-homologous*, or *homologically trivial*, closed walks is null-homologous. Recall that a closed walk in G is homologically trivial if is it the boundary of a union of faces of G. Let us only mention that homotopically trivial implies homologically trivial. Moreover, a closed walk or loop without repeated vertices is null-homologous in $\overline{\Sigma}$ if and only if its image is *separating* in Σ (or, equivalently, $\overline{\Sigma}$) that is, its complement in Σ is not connected.

Every cellular embedding contains a non-contractible loop, except if Σ is the sphere or the disk. Every cellular embedding has a non-separating closed walk, except if $\overline{\Sigma}$ is the sphere.

2.3 Reduction to an Undirected Graph with Asymmetric Weights

Let D be a directed graph (not necessarily embedded), where each arc has a non-negative weight (length). Our goal is to compute shortest loops or cycles in D satisfying some given property. We first transform D into an *undirected* graph G *with asymmetric weights*. Specifically, G is the undirected version of D; each oriented edge of G corresponds:

- either to an arc of D with some length ℓ. In this case, the length of that oriented edge is ℓ;

- or to the reversal of an arc of D. In this case, the length of that oriented edge is a "very large" (symbolically infinite) number M.[1]

Without loss of generality, in all the problems that we consider, we can assume that all the edges have strictly positive weights. Indeed, we can add an infinitesimal positive weight to edges of zero weight[2], noting that a shortest path for these perturbed weights is a shortest path for the original weights.

If there is a loop or cycle satisfying the desired property in D, one such loop or cycle has length strictly smaller than M. As a consequence, to compute shortest loops or cycles in D, it suffices to compute them in G. If the length of the resulting loop or cycle is at least M, then no such loop or cycle with that desired property exists in D; otherwise, that loop or cycle is a shortest desired loop or cycle in D.

3 Computing Shortest Non-Trivial Loops

In this section and in Section 4, we consider the problem of computing a shortest loop or cycle in an undirected graph with asymmetric weights in the complement of a given family satisfying the *3-path condition*. This property can be expressed without reference to an embedding of the graph, so here and in the following section, our graph needs not be embedded.

Our primary goal is to compute shortest non-trivial cycles. This will be done by taking each vertex s of the input graph, computing a shortest non-trivial loop based at s, and returning the shortest such loop. So, in this section, we focus on computing shortest non-trivial *loops*.

3.1 The 3-Path Condition

Let G be an undirected graph. A family $\mathbb{L}$ of *loops* of G based at some vertex s satisfies the *3-path condition* when the following two conditions are satisfied:

invariant under reversal: if $\alpha \in \mathbb{L}$, then $\alpha^{-1} \in \mathbb{L}$;

sum of zeros is zero: if α, β, γ are s-to-x walks in G (for some vertex x) such that $\alpha \cdot \beta^{-1}$ and $\beta \cdot \gamma^{-1}$ are in $\mathbb{L}$, then $\alpha \cdot \gamma^{-1}$ is also in $\mathbb{L}$.

Figure 2, left, illustrates the condition "sums of zero is zero". In conjunction with the invariance under reversal, this condition can be restated as: if two of the three closed

[1] Alternatively and more formally, lengths could be now considered as vectors in $\mathbb{R}^2_+$, which are added componentwise and compared lexicographically; every oriented edge of G corresponding to an arc of D of length k would have length $(0, k)$, and the reversal of such an edge would have length $(1, 0)$, that is, "infinitely larger" than the weight of the arcs of D.

[2] More formally, we can consider weights as ordered pairs in the semigroup product $\mathbb{R}_+ \times \mathbb{Z}_+$; an edge of weight w is given the weight pair $(0, 1)$ if $w = 0$ and $(w, 0)$ otherwise. As previously noted, lengths are added componentwise and compared lexicographically.

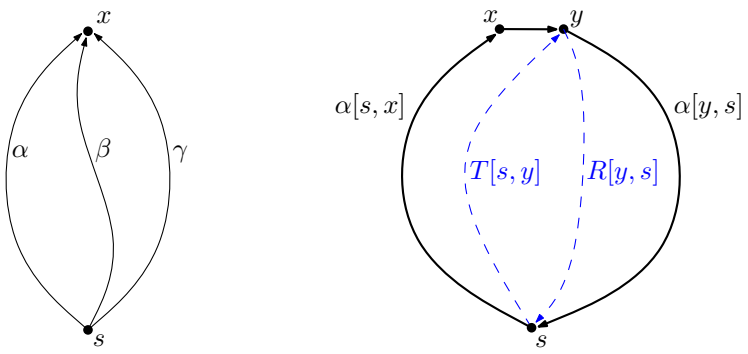

Figure 2: Left: Illustration of the 3-path condition. Right: Scenario in the proof of Lemma 3.1.

walks constructed from α, β, γ are in $\mathbb{L}$, no matter their orientations, so is the third closed walk. In practice, we will use this version of the "sums of zero is zero" condition.

For the reader familiar with the work of Thomassen [46], we remark that although the spirit of our definition is similar to that of Thomassen, we will exchange the role of the family $\mathbb{L}$ and its complement. Our presentation is motivated by the following: the loops of $\mathbb{L}$ behave like the "trivial elements in a certain group", while the loops outside $\mathbb{L}$ behave like the "non-trivial elements". Our 3-path condition basically asks for the sum of two zero elements to be zero. Our aim is to find a shortest loop outside $\mathbb{L}$, that is, a shortest "non-trivial element". Here are some families of loops that satisfy the 3-path condition: the family of contractible loops, the family of null-homologous loops, the family of loops with an even number of edges, and the family of two-sided loops [40, Section 4.3]. Again, we will be discussing the problem of finding a shortest loop in the *complement* of such families.

3.2 Properties Assuming Uniqueness of Shortest Paths

Now, let us assume that G is an undirected graph with asymmetric weights. In general, the condition that any pair of vertices of G is connected by a *unique* shortest path does not hold; just consider the case when all edges have unit length. However, Erickson and Har-Peled [23] point out that this condition can be enforced with high probability via the Isolation Lemma [41, Chapter 12]: if we add to the length of each oriented edge xy a value $r_{xy} \cdot \varepsilon$, where ε is a formal infinitesimal and r_{xy} is chosen uniformly at random from a large enough set, then all shortest paths are unique with high probability.[3] In fact, it is possible to get rid of this probabilistic assumption using a modified definition of "length" that makes the discussion more dense. In Section 3.3, we introduce this alternative definition and explain in detail how to get rid of the assumption of unique shortest paths. To keep the exposition simple, we assume in this section the uniqueness of shortest paths.

Let $\overline{\mathbb{L}}$ be the family of loops based at s that are *not* in $\mathbb{L}$. Our objective is to find a shortest element in $\overline{\mathbb{L}}$.

[3] Alternatively and more formally, the lengths are now vectors in $\mathbb{R}^3_+$, added componentwise and compared lexicographically; the first two components are as above, and the third one contains the number r_{xy}.

Let T be the shortest path tree in G from the source s. Hence, for any vertex x of G the path in T from s to x is a shortest path. Similarly, let R be the reversed shortest path tree in G to the sink s, so that for any vertex x of G the path in R from x to s is a shortest path.

For a vertex x, we will use $T[s,x]$ to denote the unique s-to-x path in T, and $T[x,s]$ for the reversal of that path. Similarly, we will use $R[x,s]$ to denote the unique x-to-s path in R, and $R[s,x]$ for the reversal of that path. (So, the only relations among $\ell(T[s,x])$, $\ell(T[x,s])$, $\ell(R[s,x])$, and $\ell(R[x,s])$ that hold in general are $\ell(T[s,x]) \leq \ell(R[s,x])$ and $\ell(R[x,s]) \leq \ell(T[x,s])$.)

For any (oriented) loop α based at s and passing through vertex x, let $\alpha[s,x]$ denote the subwalk of α that goes from s to x, and $\alpha[x,s]$ be the complementary part of α. Formally, as α may pass several times through a vertex, x should be an *occurrence* of a vertex along α. We omit this precision in the sequel and the occurrence we are considering will always be clear from the context.

For any oriented edge $xy \in E(G) \setminus E(T)$, let $\mathrm{loop}_{T,R}(xy)$ denote the loop $T[s,x] \cdot xy \cdot R[y,s]$ [4]. Let $\mathbb{M}$ be the family of loops of the form $\mathrm{loop}_{T,R}(xy)$ that are in $\overline{\mathbb{L}}$:

$$\mathbb{M} = \{\mathrm{loop}_{T,R}(xy) \mid xy \in E(G) \setminus E(T)\} \cap \overline{\mathbb{L}}.$$

Lemma 3.1. *If $\mathbb{L}$ satisfies the 3-path condition, then every shortest element of $\mathbb{M}$ is a shortest element of $\overline{\mathbb{L}}$.*

Proof. Assume that $\overline{\mathbb{L}}$ is non-empty and let α be a shortest loop in $\overline{\mathbb{L}}$. Let x be the last vertex in the loop α as we start walking from s that satisfies $\alpha[s,x] = T[s,x]$. Let y be the next vertex in α after x. See Figure 2, right. The loop $\mathrm{loop}_{T,R}(xy) = \alpha[s,y] \cdot R[y,s]$ is no longer than α. To prove the lemma, it is thus enough to show that $\mathrm{loop}_{T,R}(xy)$ is in $\overline{\mathbb{L}}$, hence in $\mathbb{M}$.

By uniqueness of shortest paths, $T[s,y]$ is strictly shorter than $\alpha[s,y]$, which implies that $T[s,y] \cdot \alpha[y,s]$ is strictly shorter than α, hence in $\mathbb{L}$. Similarly, $R[y,s]$ is no longer than $\alpha[y,s]$, whence $T[s,y] \cdot R[y,s]$ is strictly shorter than α and thus in $\mathbb{L}$. We can thus apply the 3-path condition to the s-to-y walks $R[y,s]^{-1}, T[s,y]$, and $\alpha[y,s]^{-1}$ to deduce that $R[y,s]^{-1} \cdot \alpha[y,s]$ is in $\mathbb{L}$.

If $\mathrm{loop}_{T,R}(xy)$ was also in $\mathbb{L}$, we would conclude, applying the 3-path condition to $\alpha[s,y], R[y,s]^{-1}$, and $\alpha[y,s]^{-1}$, that $\alpha = \alpha[s,y] \cdot \alpha[y,s]$ is also in $\mathbb{L}$, and thus we would reach a contradiction. □

For each oriented edge $xy \in E(G) \setminus E(T)$, let $\mathrm{loop}_T(xy)$ denote the loop $T[s,x] \cdot xy \cdot T[y,s]$. For the topological problems that we will study below, it is convenient to introduce the following family of loops based at s:

$$\mathbb{N} = \{\mathrm{loop}_{T,R}(xy) \mid xy \in E(G) \setminus E(T), \mathrm{loop}_T(xy) \in \overline{\mathbb{L}}\}. \quad (1)$$

[4] Since we assume unique shortest paths, the loop $\mathrm{loop}_{T,R}(xy)$ depends only on the source s. However, we keep this notation that depends on the trees T and R for compatibility with the general case discussed in Section 3.3.

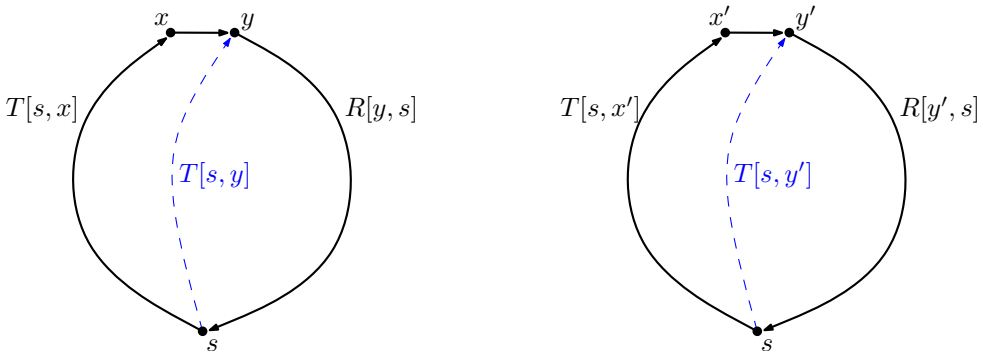

Figure 3: Scenarios in the proof of Proposition 3.2.

Note that we decide to include a loop in $\mathbb{N}$ by checking membership in $\overline{\mathbb{L}}$ of a different loop. The reason for this becomes apparent in the proof of the following lemma. A motivation to consider $\mathbb{N}$ is that, for certain topological properties to be considered below, we can manipulate $\mathbb{N}$ faster than $\mathbb{M}$ (see Section 5).

Proposition 3.2. *If $\mathbb{L}$ satisfies the 3-path condition, then every shortest element of $\mathbb{N}$ is a shortest element of $\overline{\mathbb{L}}$.*

Proof. Assume that $\overline{\mathbb{L}}$ is non-empty. By Lemma 3.1 the family $\overline{\mathbb{L}}$ has a shortest element of the form $\alpha = \text{loop}_{T,R}(xy)$ for some edge $xy \in E(G) \setminus E(T)$. We first prove that α belongs to $\mathbb{N}$, or equivalently, that $\text{loop}_T(xy) \in \overline{\mathbb{L}}$. See Figure 3, left. Since $xy \notin E(T)$ and shortest paths are unique, $T[s, y]$ is strictly shorter than $\alpha[s, y] = T[s, x] \cdot xy$. It follows that $T[s, y] \cdot R[y, s]$ is strictly shorter than α and thus it belongs to $\mathbb{L}$. If $\text{loop}_T(xy)$ was also in $\mathbb{L}$, then we could apply the 3-path condition to the s-to-y walks $T[s, x] \cdot xy$, $T[s, y]$, and $R[y, s]^{-1}$, and we would conclude that α is also in $\mathbb{L}$, which contradicts our choice of α. Thus $\text{loop}_T(xy) \in \overline{\mathbb{L}}$ and the loop α is in $\mathbb{N}$.

Let β be any shortest element in $\mathbb{N}$; we can write β as $\text{loop}_{T,R}(x'y')$ for some edge $x'y' \in E(G) \setminus E(T)$ such that $\text{loop}_T(x'y') \in \overline{\mathbb{L}}$. There remains to prove that β belongs to $\overline{\mathbb{L}}$. See Figure 3, right. By the preceding paragraph β is no longer than α. Since $T[s, y'] \cdot R[y', s]$ is strictly shorter than β, hence than α, it must be in $\mathbb{L}$. If β was also in $\mathbb{L}$, we could apply the 3-path condition to the s-to-y' walks $T[s, x'] \cdot x'y'$, $R[y', s]^{-1}$, and $T[s, y']$ to deduce that $\text{loop}_T(x'y')$ is in $\mathbb{L}$, and reach a contradiction. Thus β is in $\overline{\mathbb{L}}$, and since it is no longer than α, it is a shortest element of $\overline{\mathbb{L}}$. □

Proposition 3.2 shows that finding a shortest element in $\mathbb{M}$ or in $\mathbb{N}$ is enough to find a shortest element in $\overline{\mathbb{L}}$. However, it should be pointed out that the role of $\mathbb{N}$ is surprising. In particular, $\mathbb{N}$ can contain loops from $\mathbb{L}$. We only know that the shortest ones are in $\overline{\mathbb{L}}$. In general, it does not hold that $\mathbb{N} \subseteq \mathbb{M}$ or $\mathbb{M} \subseteq \mathbb{N}$. They just agree in having the same shortest elements.

3.3 Without Uniqueness of Shortest Paths

For simplicity of exposition, the proposition of the previous section was proved assuming that there is a unique shortest path between any pair of vertices. In this section, we prove that similar properties hold in general. While, in the present context, it is reasonable to assume uniqueness of shortest paths, the techniques that we describe now will be crucial for the unweighted case (Section 7). In our applications, we can remove the assumption on unique shortest paths without increasing the asymptotic running time. It should be noted that in some cases, like for example in [5,6], it is not known how to remove the assumption on unique shortest paths without increasing the running time.

We first introduce a composite definition of length, denoted by λ-length. Let s be the vertex for which we want to compute a shortest non-trivial loop. Fix a shortest path tree T in G from the source s and also a reversed shortest path tree R in G to the sink s. For any loop α through s we define $\lambda_T(\alpha)$ to be the number of edges of the longest segment of α starting from s and contained in T. Finally, let us define the λ-length of α to be the couple $(\ell(\alpha), -\lambda_T(\alpha))$. We compare λ-lengths using the lexicographic order: for two loops α and β, we write $\alpha \preceq \beta$ if the λ-length of α is lexicographically smaller than that of β. This means that to obtain a λ-shortest loop through s we want first to minimize the length of the loop, then to maximize the length of the segment starting from s and contained in T. The following lemma extends Lemma 3.1 without assumptions.

Lemma 3.3. *If $\mathbb{L}$ satisfies the 3-path condition, then every λ-shortest element of $\mathbb{M}$ is a λ-shortest element of $\overline{\mathbb{L}}$.*

Proof. Assume that $\overline{\mathbb{L}}$ is non-empty, and let α be a λ-shortest loop in $\overline{\mathbb{L}}$. Let x be the last vertex in α as we start walking from s that satisfies $\alpha[s,x] = T[s,x]$. Let y be the next vertex in α after x. Note that $\text{loop}_{T,R}(xy) = \alpha[s,y]\cdot R[y,s]$. By the definition of λ-length, we have $\text{loop}_{T,R}(xy) \preceq \alpha$. Because $\mathbb{M} \subseteq \overline{\mathbb{L}}$, the lemma will follow if we show that $\text{loop}_{T,R}(xy)$ is in $\overline{\mathbb{L}}$ since by definition it will also be in $\mathbb{M}$. We next note the following.

- The loop $\beta = T[s,y] \cdot R[y,s]$ is in $\mathbb{L}$. Indeed, either $\ell(\beta) < \ell(\alpha)$ or $\ell(\beta) = \ell(\alpha)$ and $\lambda_T(\alpha) < \lambda_T(\beta)$. In both cases $\beta \prec \alpha$, and β cannot be in $\overline{\mathbb{L}}$.
- The loop $T[s,y] \cdot \alpha[y,s]$ is in $\mathbb{L}$. The same argument as above applies.

By the 3-path condition we conclude, exactly as in the proof of Lemma 3.1, that $R[y,s]^{-1} \cdot \alpha[y,s]$ is in $\mathbb{L}$. If $\text{loop}_{T,R}(xy)$ was in $\mathbb{L}$ we would also conclude, applying the 3-path condition to $\alpha[s,y], R[y,s]^{-1}$ and $\alpha[y,s]^{-1}$, that $\alpha = \alpha[s,y]\cdot\alpha[y,s]$ is in $\mathbb{L}$, reaching a contradiction. □

Similarly, we replace Proposition 3.2 with the following version:

Proposition 3.4. *If $\mathbb{L}$ satisfies the 3-path condition, then every λ-shortest element of $\mathbb{N}$ is a λ-shortest element of $\overline{\mathbb{L}}$.*

Proof. Assume $\overline{\mathbb{L}}$ is non-empty; by Lemma 3.3, $\overline{\mathbb{L}}$ has a λ-shortest element of the form $\alpha = \text{loop}_{T,R}(xy)$ for some edge $xy \in E(G) \setminus E(T)$. The loop $T[s,y] \cdot R[y,s]$ is in $\mathbb{L}$: indeed,

as in the proof of Lemma 3.3, we have $T[s,y] \cdot R[y,s] \prec \alpha$ while α is λ-minimal in $\overline{\mathbb{L}}$. If $\mathrm{loop}_T(xy)$ was also in $\mathbb{L}$, then we could apply the 3-path condition to the s-to-y walks $T[s,x] \cdot xy$, $T[s,y]$, and $R[y,s]^{-1}$, and conclude that α is also in $\mathbb{L}$, a contradiction. We thus have $\mathrm{loop}_T(xy) \in \overline{\mathbb{L}}$; hence, $\alpha \in \mathbb{N}$.

Let β be a λ-shortest element in $\mathbb{N}$; we can write β as $\mathrm{loop}_{T,R}(x'y')$ for some edge $x'y' \in E(G) \setminus E(T)$ such that $\mathrm{loop}_T(x'y') \in \overline{\mathbb{L}}$. There remains to prove that β belongs to $\overline{\mathbb{L}}$. Since $\alpha \in \mathbb{N}$, we have $\beta \preceq \alpha$. By the same reasoning as above $T[s,y'] \cdot R[y',s] \prec \beta$. Hence, $T[s,y'] \cdot R[y',s]$ is strictly λ-shorter than α and must be in $\mathbb{L}$. If β was also in $\mathbb{L}$, we could apply the 3-path condition to the s-to-y' walks $T[s,x'] \cdot x'y'$, $R[y',s]^{-1}$, and $T[s,y']$ to deduce that $\mathrm{loop}_T(x'y')$ is in $\mathbb{L}$, again a contradiction. □

3.4 Algorithm

Let G be an undirected graph with asymmetric weights, and let s be one of its vertices. Let $\mathbb{L}$ be a family of loops in G that satisfies the 3-path condition. We now describe a generic algorithm to compute a shortest loop based at s that is not in $\mathbb{L}$. In the following algorithm, the λ-length should be used if it is not assumed that shortest paths are unique.

1. Construct a shortest path tree T and a reversed shortest path tree R for vertex s.
2. Determine the set A of edges $xy \in E(G) \setminus E(T)$ such that $\mathrm{loop}_T(xy) \notin \mathbb{L}$.
3. If A is empty, report that $\overline{\mathbb{L}}$ is empty. Otherwise, return a shortest loop of the form $\mathrm{loop}_{T,R}(xy)$, over all $xy \in A$.

Theorem 3.5. *The aforementioned algorithm computes a shortest loop based at s not in $\mathbb{L}$, or correctly report that no such loop exists.*

Proof. If we assume uniqueness of shortest paths, then correctness is clear by Proposition 3.2. If we do not make this assumption, then the result follows from Proposition 3.4 and from the observation that a shortest loop in $\overline{\mathbb{L}}$ with respect to the λ-length is also a shortest loop in $\overline{\mathbb{L}}$ with respect to the usual length. □

We remark that, in the second step of our algorithm, we could replace $\mathrm{loop}_T(xy)$ with $\mathrm{loop}_{T,R}(xy)$ (this is valid by Lemma 3.1 or 3.3). However, in our applications, determining membership in $\mathbb{L}$ happens to be much easier for $\mathrm{loop}_T(xy)$ than for $\mathrm{loop}_{T,R}(xy)$.

4 Computing Shortest Non-Trivial Cycles

Let G be an undirected graph. A family of *closed walks* $\mathbb{W}$ in G satisfies the *3-path condition* when the following two conditions are satisfied (recall that, by definition, closed walks have no distinguished basepoint, in contrast to loops):

invariant under reversal: if $\alpha \in \mathbb{W}$, then $\alpha^{-1} \in \mathbb{W}$;

sum of zeros is zero: if α, β, γ are x-to-y walks in G (for some arbitrary vertices x and y) such that $\alpha \cdot \beta^{-1}$ and $\beta \cdot \gamma^{-1}$ are in $\mathbb{W}$, then $\alpha \cdot \gamma^{-1}$ is also in $\mathbb{W}$.

In particular, if G is embedded on a surface, the families of contractible, null-homologous, even, and two-sided closed walks satisfy this condition [40, Section 4.3]. In the sequel we assume that at least one closed walk is not in $\mathbb{W}$.

Lemma 4.1. *Every shortest closed walk not in $\mathbb{W}$ is a cycle.*

Proof. Let α be a shortest element not in $\mathbb{W}$ and assume, for the sake of contradiction, that α is not a cycle. Let x be a vertex that is visited twice by the walk α. Let β_1 be a closed subwalk of α between the first and the second appearance of x in α. Let β_2 be the closed walk between the second and the first appearance of x in α. By construction it holds that $\alpha = \beta_1 \cdot \beta_2$. Since β_1 and β_2 are strictly shorter than α, they must belong to $\mathbb{W}$. However, by the 3-path condition applied to the paths β_1, the constant walk consisting of the single vertex x, and β_2^{-1}, it follows that the path $\beta_1 \cdot (\beta_2^{-1})^{-1} = \alpha$ is also in $\mathbb{W}$. This is a contradiction with the choice of α. □

Now, let us assume that G is an undirected graph with asymmetric weights, and let $\mathbb{W}$ be a family of closed walks in G that satisfies the 3-path condition. For every vertex s of G we let $\mathbb{W}_s$ be the possibly empty subfamily of loops in $\mathbb{W}$ with basepoint s. Because $\mathbb{W}_s$ satisfies the 3-path condition for loops, we can run the generic algorithm of Section 3.4 on each $\mathbb{W}_s$ to return the overall shortest loop, regardless of the basepoint. We conclude that:

Theorem 4.2. *The above procedure returns a shortest closed walk not in $\mathbb{W}$, or correctly report that no such walk exists.*

Note that the returned closed walk is necessarily a cycle by the previous lemma.

5 Computing Shortest Non-Trivial Loops and Cycles on Surfaces

For the rest of the paper, let Σ be a compact surface, orientable or not, with genus g and b boundaries. Let G be a graph cellularly embedded on Σ. In this section, we present improved algorithms to compute a shortest non-contractible or non-separating loop or closed walk in G. Let $\mathbb{L}$ denote either the family of contractible loops or the family of non-separating loops based at some vertex s of G; this family satisfies the 3-path condition.

5.1 Computing Shortest Non-Trivial Loops on Surfaces

Theorem 5.1. *Let G be an undirected graph with asymmetric weights, V vertices, and E edges, cellularly embedded on Σ. Let s be one of its vertices. In $O(E + V \log V)$ time, we can find a shortest non-contractible, respectively non-separating, loop based at s. The running time improves to $O(E)$ in each of the following cases:*

- *if the genus is bounded;*

- *if G has bounded treewidth;*
- *if all the weights are one (or symbolically infinite).*

Proof. We apply Theorem 3.5.

Computing the shortest path trees T and R takes $O(E + V \log V)$ time, using Dijkstra's algorithm [17] sped up with Fibonacci heaps [28]. However, in some cases T and R can be computed faster. First note that $V = O(E)$ since G is connected. Shortest path trees can be computed in $O(E)$ time for proper minor-closed families of graphs [45], and the graphs with bounded genus or bounded treewidth are such families. (For graphs with bounded treewidth there are easier linear-time algorithms [13], modulo obtaining a tree-decomposition.) If all the weights are one or symbolically infinite, we can use breadth-first search, also obtaining a shortest path tree in $O(E)$ time. In the same amount of time we can also compute all the lengths $\ell(T[s, x])$ and $\ell(R[x, s])$ for any vertex x from and to the base vertex s.

Then we need to determine the set A of edges $xy \in E(G) \setminus E(T)$ such that $\mathrm{loop}_T(xy)$ is non-contractible (resp. non-separating); this is possible in $O(E)$ time using an algorithm from the authors [8, Lemma 4]. Moreover, for any such xy, the λ-length of $\mathrm{loop}_{T,R}(xy)$ is $(\ell(T[s, x]) + \ell(xy) + \ell(R[x, s]), -\ell(T[s, x]))$ and can thus be computed in constant time. The result follows. □

5.2 Computing Shortest Non-Trivial Cycles on Surfaces

Theorem 5.2. *Let G be an undirected graph with asymmetric weights, V vertices, and E edges, that is cellularly embedded on Σ. In $O(E+Vg+V^2 \log V)$ time, we can find a shortest non-contractible (resp. non-separating) closed walk; this is also a shortest non-contractible (resp. non-separating) cycle. The $\log V$ factor can be removed from the running time in each of the following cases:*

- *if the genus is bounded;*
- *if G has bounded treewidth;*
- *if all the weights are one (or symbolically infinite).*

Proof. The algorithm consists essentially in applying V times Theorem 5.1, choosing each vertex in turn to be the basepoint. However, this gives a running-time of $O(VE+V^2 \log V)$. To obtain the complexity in the theorem, we use Proposition 5.3 below. This proposition says that, after $O(E)$ pre- and post-processing, we may assume $E = O(V + g)$. This leads to the running time $O(E + V(V + g) + V^2 \log V)$, as claimed. (Again, the $\log V$ factor disappears if one of the specified conditions is met, by Theorem 5.1.)

The output of the algorithm is a cycle by Lemma 4.1. □

Proposition 5.3. *Assume we want to compute a shortest non-contractible or non-separating cycle in G. With $O(E)$ time pre-processing and post-processing, this reduces to computing*

the shortest such cycle on another graph G' with V vertices and $E' = \Theta(V + g)$ edges, cellularly embedded on a surface Σ' with the same genus and orientability as Σ and no more boundary components than Σ.

Proof. We focus on the non-separating case first. In this case, we may assume $b = 0$, since attaching a disk to each boundary component does not change the separability character of a cycle. We will choose $\Sigma' = \Sigma$.

Whenever a face has degree one, we may remove the incident loop edge, which is contractible. Whenever a face has degree two, we may merge the two homotopic edges e_1 and e_2, with endpoints x and y, into a single edge e with the same endpoints. The weight of e, oriented from x to y, is the minimum of the weights of e_1 and e_2 from x to y, and similarly for the weight from y to x.

This way, we can remove in linear time all the faces with degree one or two to obtain a graph G' cellularly embedded on Σ. In each homotopy class the shortest cycles in G and G' have the same length. So, a shortest non-separating cycle in G' corresponds to a shortest non-separating cycle in G; recovering the corresponding cycle in G takes $O(E)$ time.

Let E' and F' be the number of edges and faces of G'. Since G and G' have the same (number of) vertices, we get $E' = V + F' + \Theta(g)$ from Euler's formula. By double counting of the edge-face incidences we also get $F' \leq 2E'/3$, whence $E' = \Theta(V + g)$.

For the non-contractible case, the same simplification algorithm works, except that we cannot eliminate faces of degree two that are punctured, so this only gives us $E' = O(V+g+b)$. We next show how to modify the algorithm in order to still get $E' = \Theta(V+g)$. Recall that the embedding of G is given by an embedding in $\overline{\Sigma}$ and by further specifying a number of perforations in each face of this embedding. As far as contractibility of closed walks is concerned, this exact number is unimportant and we only need to know if a face is perforated or not. When $\overline{\Sigma}$ is a sphere, we may assume that at least two faces of G are perforated since otherwise every cycle in G would be contractible.

The strategy is as in the non-separating case, except that we may remove some non-contractible cycles during the simplification from G to G'. Our algorithm maintains (1) an embedded graph G', such that every non-contractible cycle in G' corresponds to a non-contractible cycle in G of the same length, and (2) some non-contractible cycle α in G. The invariant is that the shortest non-contractible cycle in G either is α or corresponds to the shortest non-contractible cycle in G'. Initially, $G' = G$ and α is undefined.

If a face (of the embedded graph G on $\overline{\Sigma}$) has degree one or two, and is not perforated in Σ, we proceed exactly as in the non-separating case to simplify the graph. We now assume that every face of degree one or two is perforated.

If a (perforated) face has degree one, its incident loop edge, say β, is non-contractible; we remove it and merge the two incident faces into a single perforated face. Also, we let α be the shortest cycle among β, β^{-1} and the current cycle α.

Assume now that a face of degree two is surrounded by two other faces of degree two. We thus have four parallel edges e_1, e_2, e_3, and e_4 such that $e_i e_{i+1}^{-1}$ $(i = 1, 2, 3)$ bounds a perforated face of degree two. The shortest non-contractible cycle in G might be contained

in $e_2 \cup e_3$, so we let α be the shortest cycle among $e_2 \cdot e_3^{-1}$, $e_2^{-1} \cdot e_3$, and the current α. Otherwise, since it has no repeated vertices, it uses at most one of e_2 and e_3, and at most once, so we may remove the face between e_2 and e_3, merging e_2 and e_3 into a single edge e with appropriate weights. This is valid since *any* cycle without repeated vertices using e_2 or e_3 must be non-contractible (for otherwise it would bound a disk, contradicting the fact that e_2 and e_3 are incident to perforated faces on both of their sides), so the shortest one uses whichever is cheaper.

The pre-processing and post-processing steps can be implemented to run in $O(E)$ time. The new graph G' has V vertices; let E' and F' be its number of edges and faces. It remains to prove that $E' = \Theta(V + g)$. Euler's formula implies that $E' = \Omega(V + g)$, and we now prove $E' = O(V + g)$. Let G'' be the graph G' where we iteratively remove every edge incident to a face of degree two; it has V vertices; let E'' be its number of edges. Then G'' contains no face of degree one or two, so we have $E'' = O(V + g)$ as before. Furthermore, transforming G' into G'' removes at most two thirds of the edges, by the preceding paragraph. So $E' = O(E'')$, which concludes the proof. □

6 Improved Algorithm for Sublinear Genus or Constant Treewidth

Let Σ be a surface (orientable or not) with genus g and b boundaries. In this section, we improve the result of the previous section in the case where $g = o(V)$ or the treewidth is bounded.

6.1 Sublinear Genus

Theorem 6.1. *Let G be an undirected graph with asymmetric weights, V vertices, and E edges, that is cellularly embedded on Σ. We can compute a shortest non-contractible or non-separating cycle in G in $O(E + Vg \log g + g^{1/2}V^{3/2} \log V)$ time, or $O(E + V^{3/2})$ time if $g = O(1)$.*

The proof uses *topology-preserving divide-and-conquer*: we compute a separator of the embedded graph, and recurse on two instances in which different parts of the graph are simplified while the topology of the surface is preserved. Figure 4 illustrates this strategy.

Lemma 6.2. *Let H be a non-empty subgraph of G. In $O(E)$ time, we can contract some edges of $E(G) \setminus E(H)$ such that the resulting graph is still cellularly embedded in Σ but contains only vertices of H.*

Proof. The strategy is to compute an initially empty set F of edges of $E(G) \setminus E(H)$ such that contracting all of them does not change the topology of the surface and removes all vertices of $G \setminus H$.

For every component K of the graph induced by the vertices in $G \setminus H$, we compute a spanning tree T of K. This tree must be incident to at least one edge with a vertex in H because G is connected. Let e be one such edge; we add the edges of T and the edge e to the set F.

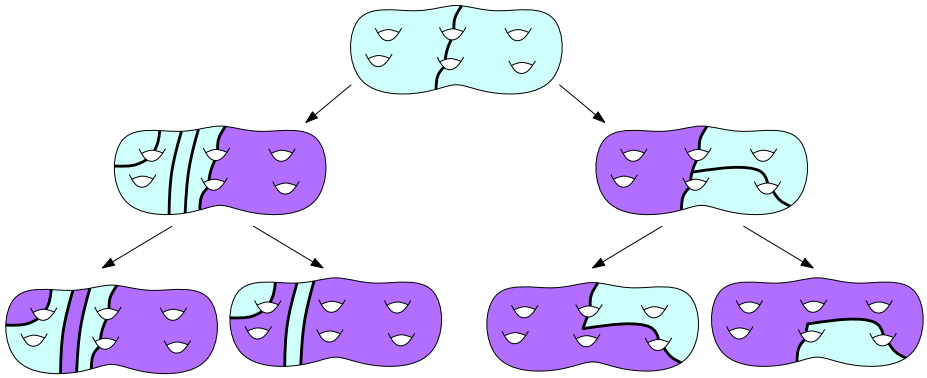

Figure 4: Topologically preserving divide and conquer. The light part of the surface represents the graph. The darker part represents faces without vertices. The bold curves represent the separator of the graph.

The union of the edges in F is a forest (any cycle using only these edges would enter and leave at least one tree T, which is connected to the remaining part of F by at most one edge). So we can perform the contraction of the edges of F topologically, on the embedding G. After this operation, the new graph contains only vertices of H.

It remains to check that the contraction of F can be performed in $O(E)$ time while preserving the embedding information. This essentially amounts to maintaining the cyclic ordering of the edges around each vertex of H. Note that an edge of G incident to a vertex of H either is not in F or is an edge e connecting a tree T as above. The edges in $E(G) \setminus F$ incident to T form a fan that replaces e in the contracted graph. This set of edges as well as its ordering in the fan is easily deduced from an Euler tour of T, starting from e. This takes time proportional to the size of T and the fan, and thus takes $O(E)$ time in total. □

Proof of Theorem 6.1. We describe the algorithm for non-separating cycles; the same argument applies to non-contractible cycles. By Proposition 5.3, using $O(E)$ pre-processing and post-processing time, we may assume that $E = \Theta(V+g)$. If $V = O(g)$, we just compute the optimal solution using Theorem 5.2. Thus, it only remains to consider the case $V = \Omega(g)$.

In $O(V)$ time, we find a graph separator X for G of size $O(\sqrt{gV})$ [20, Theorem 5.1]; that is, X is a set of $O(\sqrt{gV})$ vertices such that $G-X$ is the disjoint union of two subgraphs H_1 and H_2, each with at most $2V/3$ vertices. If $V = \Theta(g)$, we simply set $X = V(G)$. Now, by Theorem 5.1, we can compute the shortest non-separating loop α_0 passing through any of the vertices in X, in $O(\sqrt{gV}\,(V+g+V\log V)) = O(g^{1/2}V^{3/2}\log V)$ time.[5] There remains to compute the shortest non-separating loop avoiding X. If $X = V(G)$, there is nothing to do. Otherwise, in $O(V)$ time, we simplify the embedding G to a cellular embedding G_1 that contains H_1 and has the same vertex set as H_1 (Lemma 6.2), and put infinite weights on the edges not in H_1. We can recursively compute a shortest non-separating cycle α_1 in G_1 (the pre-processing and post-processing step of Proposition 5.3 takes $O(V)$ time). Similarly, we

[5] An additional property of the separator X is that the graph $G - X$ is the union of connected planar graphs. However, the embedding of such connected components in Σ is not contained in a disk, but in a sphere with (possibly multiple) punctures, and thus may contain non-trivial cycles.

simplify G to a cellular embedding G_2 that contains H_2 and has the same vertex set as H_2, and compute a shortest non-separating cycle α_2 in G_2. The shortest of α_0, α_1, and α_2 is a shortest non-separating cycle in G.

Let $T(V)$ be the time spent in the recursive calls for an input with V vertices, without taking into account the time needed at the bottom of the recursion, that is, when the subproblem has $V = O(g)$. We thus have $T(V) = 0$ when $V = O(g)$. When $V = \Omega(g)$ we have

$$T(V) \leq O(g^{1/2}V^{3/2}\log V) + \max\{T(V_1) + T(V_2)\},$$

where the maximum is taken over values V_1, V_2 that satisfy

$$V_1 + V_2 \leq V \text{ and } 0 \leq V_1, V_2 \leq 2V/3.$$

This solves to $T(V) = O(g^{1/2}V^{3/2}\log V)$.

Note that at the bottom of the recursion we get subproblems with pairwise disjoint subsets of vertices. Therefore, at the bottom of the recursion, we have $\Theta(V/g)$ subproblems, each of them with $O(g)$ vertices. Including the $O(E)$ pre- and post-processing time, and the time to solve the subproblems at the bottom of the recursion, we obtain a total running time of $O(E + Vg\log g + g^{1/2}V^{3/2}\log V)$. The log factors disappear if $g = O(1)$, because of Theorems 5.1 and 5.2. □

6.2 Constant Treewidth

Similarly, we get an improvement when the treewidth is constant:

Theorem 6.3. *Let G be an undirected graph with asymmetric weights and V vertices that is cellularly embedded on Σ. Assume that G has bounded treewidth. We can compute the shortest non-contractible or non-separating cycle in G in $O(E + Vg + V\log V)$ time.*

Proof. The proof is basically the same as that of Theorem 6.1. The main difference is that some operations can be handled faster on graphs of constant treewidth. We can compute in linear time a separator of constant size either using an approximate algorithm, like [42], or computing a tree-decomposition of minimum width [2] and extracting the separator from it. See [3] for a careful discussion connecting separators and treewidth. Recall also that shortest non-contractible or non-separating loops (Theorem 5.1) can be computed in linear time.

We note that the operations we performed in the proof of Theorem 6.1 are edge contractions and removals, which cannot increase the treewidth, so all graphs considered in the recursion have bounded treewidth. The recurrence for $V = \Omega(g)$ is now

$$T(V) \leq O(V) + \max\{T(V_1) + T(V_2)\},$$

where the maximum is taken over values V_1, V_2 that satisfy

$$V_1 + V_2 \leq V \text{ and } 0 \leq V_1, V_2 \leq 2V/3.$$

We stop the recursion when the number V' of vertices is $\Theta(g)$ (hence the number of edges is $\Theta(g)$), in which case we compute the shortest non-separating cycle using Theorem 5.2. This costs $O(V'^2) = O(V'g)$. Because the leaves of the recursion tree correspond to disjoint vertex sets, the total cost for those leaves is $O(Vg)$. The above recurrence solves to $O(V \log V)$ for the rest of the recursion tree. Adding the $O(E)$-term for the pre- and post-processing yields the result. □

6.3 Improved Algorithm for Genus Zero

In the case of a sphere with boundaries (orientable, $g = 0$), if b is fixed, we can get an almost linear dependence on the complexity of the input graph. In this case, since every cycle is separating, only the non-contractible case is relevant. Essentially, we exploit the fact that a minimum (s,t)-cut in the dual graph of a planar graph G gives a shortest cycle in G separating s and t, which is standard at least in the undirected case, and was used in the directed case e.g. by Janiga and Koubek [32]. The following theorem and its proof can be considered as folklore, but we include the result for the sake of completeness.

Theorem 6.4. *Assume Σ is a sphere with b boundaries. Let G be an undirected graph with asymmetric weights and E edges that is cellularly embedded on Σ. We can compute the shortest non-contractible cycle in G in $O(bE \log E)$ time.*

Proof. Let $B_1, \ldots, B_b$ be the boundary components of Σ. A cycle on Σ is non-contractible if and only if it separates B_1 from another boundary component of Σ. Let $i \in \{2, \ldots, b\}$. We now prove that computing the shortest cycle that has B_1 on its left and B_i on its right is possible in $O(E \log E)$ time, which concludes (because the same reasoning allows to compute the shortest cycle that has B_1 on its right and B_i on its left).

We view G as an undirected plane graph. Let G^* be its dual graph with respect to the sphere without boundary obtained after attaching a disk to every boundary component; for each k, let B_k^* be the vertex of G^* corresponding to B_k. If e is an *oriented* edge of G, then e^* is the dual edge of G^* oriented so that e^* crosses e from left to right. Let X be a set of oriented edges of G. Then X contains the oriented edges of some cycle with B_1 on its left and B_i on its right if and only if every path from B_1^* to B_i^* in G^* uses an oriented edge in X^* [14, Lemma 7.2].

In $O(E \log E)$, we compute a minimum cut on G^* from B_1^* to B_i^*, where the capacity of a dual oriented edge is the length of its primal oriented edge; this can be done, for example, by computing a maximum flow [4, 21] and using the max-flow min-cut theorem. If all lengths are non-zero (which can be assumed by putting infinitesimally small lengths if necessary), this gives an inclusionwise minimum cut of smallest capacity, hence, by the preceding paragraph, a shortest cycle having B_1 on its left and B_i on its right. □

7 The Unweighted Case: an Output-Sensitive Algorithm

In this section, we consider the case where our input directed graph is unweighted, and we describe an output-sensitive algorithm for computing shortest non-trivial cycles. The case

of unweighted, undirected graphs was covered in our paper [8]. Here we combine techniques from that paper [8] with our previous characterizations for directed graphs. Recall that the *complexity*, n, of an embedded graph G denotes the total number of vertices and edges of G.

Theorem 7.1. *Let G be an unweighted directed graph of complexity n cellularly embedded on a surface Σ. Given an integer k, in $O((g+b)nk)$ time (resp. $O(gnk)$ time) we can decide whether there exists a non-contractible (resp. non-separating) cycle of length at most k, and, if this is the case, we can compute a shortest such cycle in G.*

The main result is an immediate corollary of the previous theorem:

Corollary 7.2. *Let G be an unweighted directed graph of complexity n cellularly embedded on a surface Σ. Let k_0 be the length of a shortest non-contractible (resp. non-separating) cycle on G. We can compute a shortest non-contractible (resp. non-separating) cycle on G in $O((g+b)nk_0)$ time (resp. $O(gnk_0)$ time).*

Proof. Apply Theorem 7.1 with $k = 2^0, 2^1, 2^2, \ldots$ until a non-trivial cycle is found, namely, when $k \geq k_0$. □

We can view G as an undirected graph with asymmetric coefficients: for each edge, one direction has length 1 and the other has length ∞. We focus on the non-separating case (thus $b = 0$), the non-contractible case being entirely analogous (with g replaced by $g + b$). Let G_1 be the same graph as G, except that all weights in all directions equal 1. Note that the length of a walk in G is at least its length in G_1.

Lemma 7.3. *In $O(gn)$ time, we can compute a set K of vertices of G of such that:*

- *every non-separating cycle has to intersect K;*
- *K can be split in $O(gk)$ batches, such that within each batch, the distance in G_1 between any two vertices is at least $2k + 2$.*

Proof. The proof is a variation on a technique introduced in our previous paper [8, Theorem 10]. Here, distances are measured in G_1, not in G. Let s be an arbitrary vertex of G_1. We first compute a set K of vertices of G_1 that intersects every non-separating closed walk in G_1 and is the union of $O(g)$ shortest paths from s. We can take for K the vertices in the union of $2g$ loops $\{\mathrm{loop}_{s,E}(e_i)\}_{1 \leq i \leq 2g}$, where E is a breadth-first search tree rooted at s and the edges e_i are chosen such that the union of the loops cuts the surface Σ into a disk. By a result in our previous paper [8, Lemma 8], such a K can be computed in $O(n)$ time. For each i, $0 \leq i \leq n$, let S_i be the set of vertices of K at distance exactly i from s in G_1. Each S_i has cardinality $O(g)$. For each j, $0 \leq j \leq 2k + 1$, we put the elements of $S_j, S_{(2k+2)+j}, S_{2(2k+2)+j}, \ldots$ into $O(g)$ batches, each containing at most one element from each of these S_i. In total, we have a partition of K into $O(gk)$ batches such that any two vertices in the same batch are at distance at least $2k + 2$ from each other in G_1 (because an element in S_i and an element in S_j are at distance at least $|i - j|$ from each other). □

Henceforth, lengths are measured in G unless noted otherwise. Consider a given batch B. Note that the length of any path between any two vertices in B is at least $2k+2$. Build shortest path trees in G, starting from every vertex of B simultaneously, but stopping the growth of the trees when the distance from the sources strictly exceeds k; since the weights of G are either one or infinity, this is easy to do in $O(n)$ time. These shortest path trees are disjoint by definition of the batches. Extend the union of these shortest path trees arbitrarily to a spanning tree T of G. Similarly, build a tree of reversed shortest paths in G, starting from every vertex of B simultaneously, stopping the growth of the trees when the distance to the sources strictly exceeds k. These trees are disjoint by definition of the batches. Extend this forest to a spanning tree R of G.

We introduce notation for some loops that are slight variations on those introduced in Section 3; we just indicate the base point of the loops considered as an additional index. Let $s \in B$. For any oriented edge $xy \in E(G) \setminus E(T)$, let $\mathrm{loop}_{s,T,R}(xy)$ denote the loop $T[s,x] \cdot xy \cdot R[y,s]$. Similarly, let $\mathrm{loop}_{s,T}(xy)$ denote the loop $T[s,x] \cdot xy \cdot T[y,s]$. Let $\mathbb{L}_s$ be the set of separating loops with basepoint s. Let $\overline{\mathbb{L}}_s$ be the set of non-separating loops with basepoint s. Finally, let $\mathbb{N}_s$ be the set $\{\mathrm{loop}_{s,T,R}(xy) \,|\, \mathrm{loop}_{s,T}(xy) \in \overline{\mathbb{L}}_s\}$.

We define the λ-length as in Section 3.3: intuitively, if two loops based at the same vertex s have the same length in G, we break some ties by declaring that the shorter is the one with the longest prefix in T (in terms of number of edges). We have the following property:

Proposition 7.4. *If a λ-shortest element in $\mathbb{N}_s$ has length at most k, then it is a shortest element in $\overline{\mathbb{L}}_s$. Otherwise, every element in $\overline{\mathbb{L}}_s$ has length at least $k+1$.*

Proof. We first claim: If a λ-shortest loop in $\overline{\mathbb{L}}_s$ has length at most k, then every λ-shortest element of $\mathbb{N}_s$ is a shortest element of $\overline{\mathbb{L}}_s$. To prove the claim we consider the restriction of T to the vertices whose distance from s in G is at most k. By definition of T, this restriction can be extended to a shortest path spanning tree T' in G. We define R' similarly by extending the restriction of R to the vertices whose distance to s is at most k. We denote by $\mathbb{N}'_s$ the corresponding set of loops (see Equation (1)):

$$\mathbb{N}'_s = \{\mathrm{loop}_{s,T',R'}(xy) \mid xy \in E(G) \setminus E(T'), \mathrm{loop}_{s,T'}(xy) \in \overline{\mathbb{L}}_s\}.$$

We also refer to the λ'-length $(\ell(\alpha), -\lambda_{T'}(\alpha))$ of a loop α based at s exactly as in Section 3.3, using a prime to emphasize the dependency on T' and R'. By Proposition 3.4, every λ'-shortest loop of $\mathbb{N}'_s$ is a λ'-shortest element of $\overline{\mathbb{L}}_s$.

For loops with length at most k, the λ- and λ'-lengths coincide. We assume that a λ-shortest loop in $\overline{\mathbb{L}}_s$ has length at most k; so the shortest elements in $\overline{\mathbb{L}}_s$, with respect to the λ- or the λ'-length, are the same. It now follows from the statement ending the previous paragraph that the λ'-shortest loops of $\mathbb{N}'_s$ have length at most k; therefore, the shortest loops in $\mathbb{N}'_s$, with respect to the λ- or the λ'-length, are the same. Now the statement ending the previous paragraph proves the claim.

By contraposition, this claim implies that, if every λ-shortest element in $\mathbb{N}_s$ has length at least $k+1$, then every element in $\overline{\mathbb{L}}_s$ has length at least $k+1$. This is the second part of the proposition.

Now, assume that a λ-shortest element α in $\mathbb{N}_s$ has length at most k. There remains to prove that it is a shortest element in $\overline{\mathbb{L}}_s$. By the claim, it suffices to prove that some loop in $\overline{\mathbb{L}}_s$ has length at most k. This is trivial if $\alpha \in \overline{\mathbb{L}}_s$, so let us assume that $\alpha \in \mathbb{L}_s$. Since $\alpha \in \mathbb{N}_s$, then $\alpha = T[s,x] \cdot xy \cdot R[y,s]$ for some edge $xy \in E(G) \setminus E(T)$. We have $T[s,x] \cdot xy \cdot T[y,s] \in \overline{\mathbb{L}}_s$ because $\alpha \in \mathbb{N}_s$. Thus, the 3-path condition implies that $T[s,y] \cdot R[y,s] \in \overline{\mathbb{L}}_s$, and this loop is no longer than α (since the distance from s to y is at most k), which proves the result. □

We can now conclude:

Proof of Theorem 7.1. Let B be one of the $O(gk)$ batches obtained in Lemma 7.3. We will, in $O(n)$ time, determine a shortest non-separating loop based at any vertex of B, or correctly report that every such loop has length at least $k+1$. Since there are $O(gk)$ batches, this concludes the proof of the theorem.

We first build the trees T and R corresponding to B, as described above. Moreover, when building T, we label every vertex v of G at distance at most k *from* some vertex s in B with that vertex s and with the λ-length from s to v in T. By definition of the batches, every vertex receives at most one label. Similarly, we label the vertices of G at distance at most k *to* some vertex s in B with that vertex s and the value of that distance.

For every edge $xy \in E(G) \setminus E(T)$, whether $\mathrm{loop}_{s,T}(xy)$ is separating does not depend on s. In $O(n)$ time, we can determine the set U_T of edges xy such that $\mathrm{loop}_{s,T}(xy)$ is non-separating [8, Lemma 4].

By Proposition 7.4, it suffices, in $O(n)$ time, to compute, among all loops in $\mathbb{N} := \bigcup_{s \in B} \mathbb{N}_s$, one with the shortest λ-length, or to correctly report that every such loop has length at least $k+1$. Note that $\mathbb{N} = \{\mathrm{loop}_{s,T,R}(xy) \mid xy \in U_T, s \in B\}$. Thus, it suffices to be able, in $O(1)$ time, to compute, for a given edge $xy \in U_T$, a λ-shortest loop in $\mathbb{N}_{xy} := \{\mathrm{loop}_{s,T,R}(xy) \mid s \in B\}$ (represented by edge xy and vertex s), or to correctly report that every loop in $\mathbb{N}_{xy}$ has length at least $k+1$.

Vertex x is at distance at most $k+1$ from at most one vertex s of B, and we can determine whether there is such an s using the labeling described above in $O(1)$ time. If there is no such s, then $\mathbb{N}_{xy}$ contains no loop of length at most k. Otherwise, again using the labeling, we can compute the λ-length from s to x in T. We can similarly determine if there is a path of length at most k from y to some vertex $t \in B$, and if so we can determine its length. If there is no such t, then $\mathbb{N}_{xy}$ contains no loop of length at most k. Otherwise, similarly as before, t is unique; if $t \neq s$, then they are at distance at least $2k+2$ apart in G_1, and thus $\mathbb{N}_{xy}$ contains no loop of length at most k. Finally, if $s = t$, using the labeling we can compute the λ-length of $\mathrm{loop}_{s,T,R}(xy)$ in constant time. If this length is at most k, we return that loop, and otherwise we report that every loop in $\mathbb{N}_{xy}$ has length at least $k+1$. □

8 Acknowledgments

We would like to thank the referees for their helpful comments.

References

[1] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[2] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[3] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[4] Glencora Borradaile and Philip Klein. An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *Journal of the ACM*, 56(2), 2009.

[5] Glencora Borradaile, Piotr Sankowski, and Christian Wulff-Nilsen. Min st-cut oracle for planar graphs with near-linear preprocessing time. *ACM Transactions on Algorithms*, 11(3):16:1–16:29, 2015.

[6] Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-source shortest paths in embedded graphs. *SIAM Journal on Computing*, 42(4):1542–1571, 2013.

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

[8] Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Algorithms for the edge-width of an embedded graph. *Computational Geometry: Theory and Applications*, 45:215–224, 2012.

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

[10] Parinya Chalermsook, Jittat Fakcharoenphol, and Danupon Nanongkai. A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 828–829, 2004.

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

[12] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Homology flows, cohomology cuts. *SIAM Journal on Computing*, 41(6):1605–1634, 2012.

[13] Shiva Chaudhuri and Christos D. Zaroliagis. Shortest paths in digraphs of small treewidth. part I: sequential algorithms. *Algorithmica*, 27(3):212–226, 2000.

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

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

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

[17] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[18] Frederic Dorn. Planar Subgraph Isomorphism Revisited. In *27th International Symposium on Theoretical Aspects of Computer Science*, pages 263–274, 2010.

[19] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(1-3), 1999.

[20] David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 599–608, 2003.

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

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

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

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

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

[26] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006.

[27] Kyle Fox. Shortest non-trivial cycles in directed and undirected surface graphs. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 352–364, 2013.

[28] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

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

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

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

[32] Ladislav Janiga and Václav Koubek. Minimum cut in directed planar networks. *Kybernetica*, 28(1):37–49, 1992.

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

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

[35] Philip N. Klein, Shay Mozes, and Oren Weimann. Shortest paths in directed planar graphs with negative lengths: a linear-space $O(n \log^2 n)$-time algorithm. *ACM Transactions on Algorithms*, 6(2), 2010.

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

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

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

[39] Tomomi Matsui. The minimum spanning tree problem on a planar graph. *Discrete Applied Mathematics*, 58(1):91–94, 1995.

[40] Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001.

[41] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Press Syndicate of the University of Cambridge, 1995.

[42] Bruce A Reed. Finding approximate separators and computing tree width quickly. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 221–228. ACM, 1992.

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

[44] John Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, 1980.

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

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