Finding Shortest Non-Trivial Cycles in Directed Graphs on Surfaces*

Sergio Cabello
Department of Mathematics, IMFM
Department of Mathematics, FMF
University of Ljubljana, Slovenia
sergio.cabello@fmf.uni-lj.si

Éric Colin de Verdière
Laboratoire d'informatique
École normale supérieure
CNRS, Paris, France
Eric.Colin.de.Verdiere@ens.fr

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

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 a shortest non-contractible and a shortest surface non-separating cycle in D. This generalizes previous results that only 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 known 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 give an algorithm with subquadratic complexity in the complexity of the input graph, if g is fixed. Specifically, we can solve the problem in $O(\sqrt{g}\, n^{3/2} \log n)$ time, using a divide-and-conquer technique that simplifies the graph while preserving the topological properties of its cycles. A variant runs in $O(ng \log g + n \log^2 n)$ for graphs of bounded treewidth.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical algorithms and problems—*Computations on discrete structures; Geometric problems and computations*; G.2.2 [Discrete Mathematics]: Graph theory—*Graph algorithms; path and circuit problems*; I.3.5 [Computer Graphics]: Computational geometry and object modeling—*Geometric algorithms, languages, and systems*

General Terms: Algorithms, Performance, Theory

*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.

SCG'10, June 13–16, 2010, Snowbird, Utah, USA.

Keywords: Computational topology, topological graph theory, surface, embedded graph, non-contractible cycle, non-separating cycle, directed graph

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, minimum spanning trees, (sub)graph isomorphism, maximum flow, minimum cut, etc. For some NP-hard problems we can obtain approximation schemes when restricted to planar graphs. 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 planar graphs that have attracted much attention. From a theoretical perspective, they play a prominent role in particular in the graph minor theory developed by Robertson and Seymour (see e.g. Mohar and Thomassen [27]). From an applied perspective, efficient algorithms are needed to process non-planar surfaces: texture mapping, topological denoising, remeshing, and visualization (see e.g. the discussion in Erickson and Har-Peled [16]). In particular, when the embedded graph is weighted, the computation of shortest cycles satisfying some prescribed topological properties has been studied in a series of papers (see references below).

Considering digraphs (directed graphs) embedded in a surface is a natural next step in this area. This setting remains largely unexplored; except for the case of planar graphs, we are only aware of the following results. The data structures of Cabello and Chambers [3] to represent distances from the vertices of a face in an embedded graph can be easily extended to digraphs. Chambers et al. give an algorithm to compute maximum flows in undirected surface-embedded graphs, and claim their approach extends to the directed case [9, Conclusion]. Erickson has recently shown that the best known algorithm to compute maximum s-t flows in planar directed graphs [2] does not extend to directed graphs on the torus [15, Section 3].

In this paper, we study the problem of finding shortest non-contractible or non-separating cycles in weighted digraphs embedded on surfaces. Like in previous works, we assume that the weights of a graph or digraph are always non-negative.

This is the same as looking for shortest non-trivial closed walks (where *trivial* means either contractible or separating), since (as we will see shortly) shortest non-trivial closed walks are simple cycles; that is, they do not repeat any vertex. However, in contrast to the undirected case, shortest non-trivial walks through a given vertex may have several self-crossings.

Background on Shortest Non-Trivial Cycles. In the directed setting, the only case where the problem of computing shortest non-trivial walks is solved is the annulus [23]. In contrast, in the undirected setting, the problem for general surfaces has been largely explored; one reason is that cutting along such a cycle is the simplest way of simplifying the topology of a surface. Additionally, this is the basic ingredient for several more complex algorithms.

Thomassen [31] shows that shortest non-trivial cycles in undirected graphs can be found in near-cubic time (see also Mohar and Thomassen [27, Chapter 4]). The approach actually works for certain families of cycles that satisfy the so-called *3-path condition*, which we revisit later in this paper. Erickson and Har-Peled [16] provide an algorithm to find shortest non-trivial cycles in near-quadratic time; this is the best current result. Cabello and Mohar [5] propose considering the problem parameterized by the genus. Kutz [25] propose an algorithm for orientable surfaces with running time $O(g^{O(g)} n \log n)$, subsequently improved to $O(g^3 n \log n)$ by Cabello and Chambers [3]. In our companion paper [4] 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. Here and in the sequel, n is the number of vertices and edges of the graph, assumed to be cellularly embedded on the surface (i.e., each face of the graph is a disk).

Many other more complex algorithms need to cut the surface along shortest non-trivial cycles (possibly through a given vertex) as a subroutine [17, 10, 8, 7].

Several authors also considered computing shortest cycles with different topological properties, such as *splitting cycles* [8] and *essential cycles* [18]. The techniques of the present paper do not apply to these problems.

Overview of our Results. In Section 3, we first revisit the concept of 3-path condition given by Thomassen [31]. Specifically, we extend this notion to a family $\mathbb{L}$ of closed loops based at a given vertex s in a digraph. Note that the loops need not be simple and that the graph is directed and needs not be embedded. In an imprecise sense, 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 candidate 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. Here we use techniques from our companion paper [4] 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 known algorithm solving this problem in the undirected case.

In Section 6, we next turn our attention to the case where the genus is small, and give an algorithm that finds a shortest non-trivial cycle in $O(\sqrt{g}\, n^{3/2} \log n)$ time. The algorithm is based on a technique we could name *topologically preserving divide-and-conquer*. The idea 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. The idea is reminiscent of the numerous algorithms for planar graphs that divide-and-conquer using a cycle separator; see for example [29, 6, 19, 24]. However, in these cases, the preservation of the topology is implicit. We believe that our new approach is very natural and could be useful for other problems as well. For graphs with bounded treewidth, which admit constant-size separators, this strategy also leads to an $O(ng \log g + n \log^2 n)$-time algorithm. We also give an $O(bn \log n)$-time algorithm for the special case where $g = 0$.

2. PRELIMINARIES

We introduce the background material used in this paper. A large part of it was also presented in our companion paper [4], but we repeat it for completeness.

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 $E(G)$.

A *walk* is a sequence of edges $e_1, \ldots, e_m$ with the property that the target of e_i is the source of e_{i+1}, for $i = 1, \ldots, m-1$. 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 of G has a non-negative *weight*, the *weight* of a closed walk or loop is the sum of the lengths of the corresponding oriented edges (counted with multiplicity).

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. The weight of the oriented edge xy is denoted by $\ell(xy)$.

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. Furthermore, $\ell(\alpha)$ denotes the length of α. For any vertices x, y of G, we use $d_G(x, y)$ to denote the minimum length of the walks from x to y in G. We emphasize that $d_G(\cdot, \cdot)$ is not a distance in general (it does not necessarily satisfy symmetry); nevertheless, we still use the words "length" (as a synonym for "weight") and "shortest".

2.2 Topological Background

We review some basic topology of surfaces, which will be needed only after Section 5 on. See any of the books by

Hatcher [21], Massey [26], or Stillwell [30] 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 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, we mean that $\overline{\Sigma}$ minus the image of the graph is a union of 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.

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 $g = O(n)$.

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 [14], or rotation systems [27]. For orientable surfaces, one can also use the DCEL that is customary in Computational Geometry (see, e.g., de Berg et al. [12]). More precisely, 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.

Homotopy and homology are standard topological notions, which will not be formally defined here. Intuitively, two paths or loops are *homotopic* if one can be deformed continuously on the surface to become the other, keeping the endpoints fixed during the deformation. A loop is *contractible*, or *homotopically trivial*, if it is homotopic to the constant loop with the same basepoint. If α and β are contractible loops with the same basepoint, then so are α^{-1} and $\alpha \cdot \beta$. Similar properties hold for *null-homologous*, or *homologically trivial*, loops[1]. Let us only mention that homotopically trivial implies homologically trivial. Moreover, a closed walk or loop without repeated vertices is null-homologous on $\overline{\Sigma}$ if and only if its image is *separating*, that is, it splits Σ into two connected components.

In every cellular embedding there is a non-contractible loop (based at an arbitrary vertex), except if Σ is the sphere or the disk. Every cellular embedding has a non-separating loop (based at an arbitrary vertex), 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 *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* lengths. 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.[2]

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.

For simplicity, we will assume throughout the paper that there is a *unique shortest path* between any pair of vertices of G. Clearly, this assumption does not hold in general; just consider the case when all edges have unit length. However, Erickson and Har-Peled [16] point out that this condition can be enforced with high probability via the Isolation Lemma [28, 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] It is possible to get rid of this probabilistic assumption using a more technical definition of "length" that makes the discussion more dense. In Appendix A, we introduce this alternative definition of length and explain in detail how to get rid of the assumption on unique shortest paths.

[1] In this paper, homology is considered with respect to any ring or field.

[2] 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.

[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}.

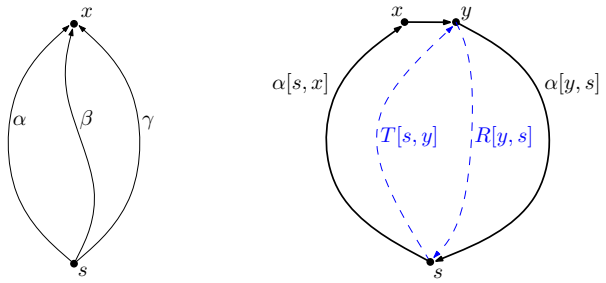

Figure 1. Left: Illustration of the 3-path condition. Right: Scenario in the proof of Lemma 1.

3. COMPUTING SHORTEST NON-TRIVIAL LOOPS

In this section and in Section 4, we consider the problem of computing the shortest loop or cycle in an undirected graph with asymmetric weights among a given family satisfying a certain property, 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 of *loops* $\mathbb{L}$ in 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 1, left, illustrates the condition "sums of zero is zero". Some remarks on the definition may be convenient. Firstly, a clarification for the reader familiar with the work of Thomassen [31]: although the spirit of our definition is similar as that of Thomassen, we have exchanged the role of the family $\mathbb{L}$ and its complement. We believe that our presentation is more convenient, and 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". The main property of 3-path condition basically tells that the sum of two zero elements must be zero. Our aim is to find a shortest loop outside $\mathbb{L}$, that is, a shortest "non-trivial element".

Secondly, we may mention some families of loops based at s 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 [27, Section 4.3]. We will be discussing the problem of finding a shortest loop in the *complement* of such families.

3.2 Properties

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 $\overline{\mathbb{L}}$.

Let T be the shortest path tree in G from the source s. (Recall that we assume that shortest paths are unique.) Similarly, let R be the reversed shortest path tree in G to the sink s; that is, R corresponds to a shortest path tree from s in the graph G' obtained by exchanging the lengths $\ell(xy)$ and $\ell(yx)$ of each edge xy. Hence, for any vertex x of G, we have $d_G(s,x) = d_T(s,x)$ and $d_G(x,s) = d_R(x,s)$.

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 $\ell(T[s,x]) < \ell(R[s,x])$ and $\ell(R[x,s]) < \ell(T[x,s])$.)

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.[4]

For any oriented edge $xy \in E(G) \setminus E(T)$, let $\mathrm{loop}_{T,R}(xy)$ denote the loop $T[s,x] \cdot xy \cdot R[y,s]$. 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 1. *If $\overline{\mathbb{L}}$ is non-empty, then a 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 1, 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 $\alpha[s,y]$. It follows that the loops $T[s,y] \cdot R[y,s]$ and $T[s,y] \cdot \alpha[y,s]$ are strictly shorter than α, hence in $\mathbb{L}$. By invariance under reversal $R[y,s]^{-1} \cdot T[s,y]^{-1}$ is also in $\mathbb{L}$, and we can 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:

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

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).

[4]More formally, x should be an *occurrence* of a vertex in the loop α, if α passes several times through that vertex; otherwise the definition is ambiguous. We omit this precision in the sequel, since the occurrence we are considering will always be clear from the context.

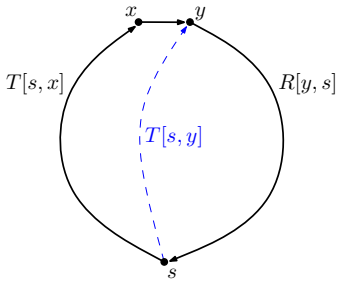

Figure 2. Scenario in the proof of Lemma 2.

LEMMA 2. *If $\overline{\mathbb{L}}$ is non-empty, then a shortest element of $\mathbb{N}$ is a shortest element of $\overline{\mathbb{L}}$.*

PROOF. Assume that $\overline{\mathbb{L}}$ is non-empty. By Lemma 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 2. Since $T[s,y] \cdot R[y,s]$ is strictly shorter than α, 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 the loop α is in $\mathbb{N}$.

Let β be a shortest element in $\mathbb{N}$; we can write β as $\text{loop}_{T,R}(xy)$ for some edge $xy \in E(G) \setminus E(T)$ such that $\text{loop}_T(xy) \in \overline{\mathbb{L}}$. There remains to prove that β belongs to $\overline{\mathbb{L}}$. See Figure 2. 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 $T[s,x] \cdot xy$, $R[y,s]^{-1}$, and $T[s,y]$ to deduce that $\text{loop}_T(xy)$ 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 $\overline{\mathbb{L}}$. □

The previous two lemmas show 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 one is 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 element.

3.3 Algorithm

THEOREM 1. *Let G be an undirected graph with asymmetric weights, V vertices, and E edges, and let s be one of its vertices. Let $\mathbb{L}$ be a family of loops in G that satisfies the 3-path condition. Assume that there is an oracle that decides in time $\tau(k)$ if a loop with k edges is in $\mathbb{L}$. Then we can find a shortest loop based at s not in $\mathbb{L}$, or correctly report that no such loop exists, in time $O(E\tau(E) + V \log V)$.*

PROOF. The algorithm is as follows. Firstly, we construct a shortest path tree T and a reversed shortest path tree R for vertex s. Secondly, for each oriented edge xy, we construct explicitly the closed walk $\text{loop}_{T,R}(xy)$, check its membership in $\mathbb{L}$, and select the shortest one among the ones not in $\mathbb{L}$ (unless all of them are in $\mathbb{L}$, in which case we report that no such loop exists).

Correctness is clear by Lemma 1. As for the running time, note that T and R can be constructed in time $O(E+V \log V)$ time, using Dijkstra's algorithm [13] speeded up with Fibonacci heaps [20]. Once the trees T and R are available, we can preprocess the graph such that any distance query from or to vertex s takes $O(1)$ time. For each oriented edge xy, we need time $O(1+\tau(E)) = O(\tau(E))$ to compute the length of $\text{loop}_{T,R}(xy)$ and to check membership in $\mathbb{L}$. The result follows. □

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.

LEMMA 3. *A 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. □

THEOREM 2. *Let G be an undirected graph with asymmetric weights, V vertices, and E edges. Let $\mathbb{W}$ be a family of closed walks in G that satisfies the 3-path condition. Assume that there is an oracle that decides in time $\tau(k)$ if a closed walk with k edges is in $\mathbb{W}$. Then in time $O(VE\tau(E) + V^2 \log V)$ we can find a shortest closed walk not in $\mathbb{W}$ (actually, this is a cycle by Lemma 3), or correctly report that no such walk exists.*

PROOF. We run the algorithm of Theorem 1, taking for s each vertex in turn, and return the shortest closed walk found. This is valid since the family of loops in $\mathbb{W}$ with a given basepoint satisfies the 3-path condition for loops. □

5. COMPUTING SHORTEST NON-TRIVIAL LOOPS AND CYCLES ON SURFACES

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

Recall the definition of $\mathbb{N}$ given in Equation (1). We want to find the oriented edges that define a loop in $\mathbb{N}$. This is equivalent to the problem of deciding if $\text{loop}_T(xy)$ is in $\mathbb{L}$. For this purpose, we can regard T as an undirected graph, and decide if $\text{loop}_T(xy)$ is in $\mathbb{L}$. This can be done efficiently using a characterization from our companion paper [4].

PROPOSITION 1. *In $O(E+V\log V)$ time, we can compute the shortest path tree T and determine the set of edges xy of $E(G)\setminus E(T)$ such that $\text{loop}_T(xy)$ is contractible (resp. separating).*

PROOF. The techniques are very similar to our companion paper [4, Section 3]; we briefly indicate the general strategy for completeness.

In this proof we will need the the *dual graph* G^*: it has for vertices the (dual) set of faces of G, for edges the (dual) set of edges of G, and two faces are adjacent if they share an edge of G. The edge dual to e is denoted by e^*, and it connects the two faces adjacent to e in the embedding. Furthermore, for every face f of G containing a boundary component, we add to G^* a loop edge. Such loop edges correspond to no edge of the primal graph G.)

We compute the shortest path tree T in $O(n\log n)$ time using Dijkstra's algorithm. Let C^* be the subgraph of G^* with the same vertex set as G^* and edge set $E(G^*)\setminus\{e^* \mid e\in E(T)\}$. The graph C^* can be constructed in linear time. Note that $\text{loop}_T(e)$ is non-contractible or non-separating if and only if $\text{cycle}_T(e)$ is non-contractible or non-separating, respectively.

In our companion paper [4], Corollary 2, we show the following characterization: The cycle $\text{cycle}_T(e)$ is non-separating if and only if C^*-e^* is connected. The cycle $\text{cycle}_T(e)$ is contractible if and only if C^*-e^* has a connected component that is a tree (possibly reduced to a single vertex)

Therefore, the set of dual edges e^* such that $\text{loop}_T(e)$ is non-contractible can be obtained in linear time from the set of edges of C^* by repeatedly removing edges with an incident vertex of degree one. To obtain the set of edges e such that $\text{loop}_T(e)$ is non-separating, note that its dual set is precisely the set of non-bridge edges in C^*. The computation of the bridge edges of a graph in linear time using depth-first search is part of the folklore. □

THEOREM 3. *Let G be an undirected graph with asymmetric weights, V vertices, and E edges, that is cellularly embedded on Σ. Let s be one of its vertices. In time $O(E+V\log V)$ we can find a shortest non-contractible, respectively non-separating, loop based at s.*

PROOF. By Lemma 2, it suffices to compute the shortest element in $\mathbb{N}$; that is, the shortest $\text{loop}_{T,R}(xy)$ such that $xy\in E(G)\setminus E(T)$ and $\text{loop}_T(xy)\in\overline{\mathbb{L}}$. Proposition 1 allows to list the edges xy such that $\text{loop}_T(xy)\in\overline{\mathbb{L}}$ in $O(E+V\log V)$ time. For each xy, the length of $\text{loop}_{T,R}(xy)$ can be computed in constant time. Hence, we can find the shortest element of $\mathbb{N}$ in $O(E+V\log V)$ time. □

5.2 Computing Shortest Non-Trivial Cycles on Surfaces

THEOREM 4. *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.*

PROOF. The algorithm consists essentially in applying V times Theorem 3, choosing each vertex in turn to be the basepoint. However, this gives a running-time of $O(VE+V^2\log V)$. To decrease this to $O(E+Vg+V^2\log V)$, we use Proposition 2 below (which allows only for a slight improvement here, but will be used crucially in Section 6). This proposition says that, after $O(E)$ pre- and post-processing, we may assume $E=O(V+g)$. So the complexity of the algorithm is now $O(E+V(V+g)+V^2\log V)$, as claimed.

The output of the algorithm is a cycle by Lemma 3. □

PROPOSITION 2. *Assume we want to compute a shortest non-contractible or non-separating cycle in G. With $O(E)$ time pre-processing and post-processing, it is sufficient to compute the shortest such cycle on another graph G' with V vertices and $E'=\Theta(V+g)$ edges, cellularly embedded on a surface Σ', such that Σ and Σ' have the same orientability character, the same genus, and the number of boundaries of Σ' is at most the number of boundaries of 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 one face has degree one, we may remove the incident loop edge, which is contractible. Whenever one face has degree two, we may replace the two homotopic edges e_1 and e_2, with endpoints x and y, by an edge e with the same endpoints (removing the face in-between). The weight of e, oriented from x to y, is the minimum of the two weights of the edges of e_1 and e_2 from x to y, and similarly for the weights from y to x.

In linear time, we removed all the faces with degree one or two. The shortest cycle in G' within a given homotopy class corresponds to a shortest cycle in G within that homotopy class. So the shortest non-separating cycle in G' corresponds to the shortest non-separating cycle in G; recovering the cycle in G from the cycle in G' takes $O(E)$ time.

Clearly, G' has the same number V of vertices as G; let E' and F' be its number of edges and faces. The bound $V+g=O(E')$ comes from Euler's formula. The reverse bound follows from $F'\leq 2E'/3$ (by double-counting the incidences between the edges and the faces) and $E'=V+F'+O(g)$ (by Euler's formula).

For the non-contractible case, the same algorithm works, but we cannot assume anymore that the surface has no boundary, so this proof only gives us $E'=O(V+g+b)$. We next show that a slightly more complicated argument still gives us $E'=O(V+g)$, as claimed. Recall that our representation of the embedded graph G on the surface $\overline{\Sigma}$ is obtained by attaching disks to each boundary component of Σ, and recording, in each face of G, the number of boundaries inside it. (Actually, the exact number is irrelevant as far as contractibility is concerned; only the presence or absence of boundaries matter.)

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'. So 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) a 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.

Assume one face f (of the embedded graph G on $\overline{\Sigma}$) has degree one. If it contains no boundary component of Σ, we can remove it as before. Otherwise, the loop edge β incident to it is non-contractible; we remove β and the corresponding face f, and mark the other face incident to β as containing boundaries. Also, we let α be the shortest cycle among the current α and β.

Assume one face has degree two, and contains no boundary component of Σ. Then as before, we may remove that face, and replace both incident edges by a single edge e with appropriate weights: the weight of e, oriented from x to y, is the minimum of the two weights of the edges of e_1 and e_2, oriented from x to y, and similarly for the opposite orientation.

Assume that four edges e_1, e_2, e_3, and e_4 have the same endpoints and form three faces of degree two, whose boundaries are $e_i e_{i+1}$ $(i = 1, 2, 3)$. Then, assuming the previous step has been applied as much as possible, each face contains (at least) one boundary component of Σ. 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 the current α, $e_2 \cdot e_3^{-1}$, and $e_2^{-1} \cdot e_3$. 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 is non-contractible, 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 $E' = \Theta(V + g)$. Again, the only non-trivial inequality is $E' = O(V + g)$. 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. □

6. IMPROVED ALGORITHM FOR SMALL 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: we obtain algorithms with running time $O(\sqrt{g}\, n^{3/2} \log n)$ to compute a shortest non-contractible or non-separating cycle on Σ. The same technique yields near-linear-time algorithms for graphs with bounded treewidth, because they admit small separators.

6.1 Small Genus

THEOREM 5. *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 $O(E + Vg \log g + \sqrt{g}\, V^{3/2} \log V)$ time.*

The proof uses *topology-preserving divide-and-conquer*: we compute a separator of the embedded graph, and recurse on two instances in which a different part of the graph is simplified, but the topology of the surface is preserved. Figure 3 illustrates this strategy.

LEMMA 4. *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 cellularly embedded on Σ and 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; otherwise the embedding G would not be cellular. Let e be one such edge; we add the edges of T and the edge e to the set F.

The union of the edges in F and their incident vertices 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.

The contraction of all the edges of F can be done in $O(E)$ time. Note that, after the contraction of F, we can update the pointers from each vertex-edge-face incidence to its incident vertex in $O(E)$ total time. □

PROOF OF THEOREM 5. By Proposition 2, 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 4. Thus, it only remains to consider the case $V = \Omega(g)$, which implies $E = \Theta(V)$. The rest of the proof discusses this case. We describe the algorithm for non-separating cycles; the same argument applies to non-contractible cycles.

In $O(V)$ time, we find a graph separator X for G of size $O(\sqrt{gV})$ [14, Theorem 5]; 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. Now, by Theorem 3, we can compute the shortest non-separating loop α_0 passing through any of the vertices in X, in $O(\sqrt{gV}\,(V + g + V \log V)) = O(\sqrt{gV}\, V \log V)$ time.[5] 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 4), 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 2 takes $O(V)$ time). Similarly, we 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.

[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 component in Σ is not contained in a disk, but in a sphere with (possibly multiple) punctures, and thus may contain non-trivial cycles.

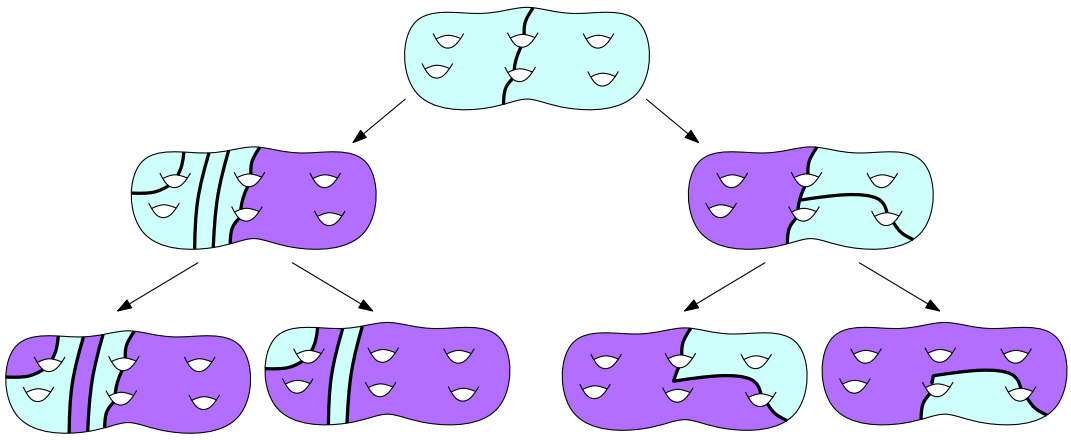

Figure 3. Topologically preserving divide and conquer. The light part of the surface represents the graph. The darker part represents faces without vertices. The bold curves represent the separator of the graph.

We stop the recursion when the number of vertices is $\Theta(g)$ (hence the number of edges is $\Theta(g)$). In this base case, we compute the shortest non-separating cycle using Theorem 4. This costs $O(g^2 \log g)$ time for each subproblem. Note that there are $O(V/g)$ base subproblems because every vertex of G is found in at most one base subproblem, so the total time spent in these base cases is $O(Vg \log g)$.

Let $T(V)$ be the total time spent in the recursive calls, without counting the base subproblems, for an input with V vertices. We have

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

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 $T(V) = O(\sqrt{g}\, V^{3/2} \log V)$. Including the $O(E)$ pre- and post-processing time, and also the time for the base subproblems, this gives a total running time of $O(E + Vg \log g + \sqrt{g}\, V^{3/2} \log V)$. □

6.2 Constant Treewidth

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

THEOREM 6. *Let G be an undirected graph with asymmetric weights and V vertices that is cellularly embedded on Σ. Assume G has treewidth at most k_0, where k_0 is a constant. We can compute the shortest non-contractible or non-separating cycle in G in $O(Vg \log g + V \log^2 V)$.*

PROOF. The proof is the same as that of Theorem 5. The only difference is that, for graphs of bounded treewidth, it holds $E = O(V)$ and we can compute separators of constant size in linear time [1]. The recurrence is now

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

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 $T(V) = O(V \log^2 V)$. Adding the $O(E) = O(V)$ time pre- and post-processing and the $O(Vg \log g)$ term for the base cases 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.

THEOREM 7. *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.

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^* [11, 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 either using the algorithm by Janiga and Koubek [23], speeded up with the linear time shortest path algorithm by Henzinger et al. [22], or by computing a maximum flow [2, 15] 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. □

References

[1] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.

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

[3] S. Cabello and E. W. Chambers. Multiple source shortest paths in a genus g graph. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 89–97, 2007.

[4] S. Cabello, É. Colin de Verdière, and F. Lazarus. Output-sensitive algorithm for the edge-width of an embedded graph. In *These Proceedings*, 2010.

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

[6] P. Chalermsook, J. Fakcharoenphol, and D. 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.

[7] E. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th Annual ACM Symposium on Computational Geometry (SOCG)*, pages 377–385, 2009.

[8] E. W. Chambers, É. Colin de Verdière, J. Erickson, F. Lazarus, and K. Whittlesey. Splitting (complicated) surfaces is hard. *Computational Geometry: Theory and Applications*, 41(1–2):94–110, 2008.

[9] E. W. Chambers, J. Erickson, and A. Nayyeri. Homology flows, cohomology cuts. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing (STOC)*, pages 273–282, 2009.

[10] É. Colin de Verdière and J. Erickson. Tightening non-simple paths and cycles on surfaces. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 192–201, jan 2006.

[11] É. Colin de Verdière and A. Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms*, To appear. Preliminary version in STACS'08.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

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

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

[15] J. Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010. To appear.

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

[17] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1038–1046, 2005.

[18] J. Erickson and P. Worah. Computing the shortest essential cycle. *Discrete & Computational Geometry*, 2010. To appear.

[19] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006.

[20] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved netweok optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

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

[22] M. R. Henzinger, Ph. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1, part 1):3–23, 1997.

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

[24] P. N. Klein, S. Mozes, and O. Weimann. Shortest paths in directed planar graphs with negative lengths: a linear-space $O(n \log^2 n)$-time algorithm. *ACM Transactions on Algorithms*, To appear. Preliminary version in SODA'09.

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

[26] W. S. Massey. *Algebraic Topology: An Introduction*, volume 56 of *Graduate Texts in Mathematics*. Springer-Verlag, 1977.

[27] B. Mohar and C. Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. John Hopkins University Press, 2001.

[28] R. Motwani and P. Raghavan. *Randomized algorithms*. Press Syndicate of the University of Cambridge, 1995.

[29] J. 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.

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

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

APPENDIX

A. WITHOUT UNIQUE SHORTEST PATHS

Throughout the paper we have assumed for simplicity that there is a unique shortest path between any pair of vertices. In this section, we explain how to remove this assumption while keeping the same running times. In practice, this assumption was only used in the proofs of Lemmas 1 and 2, for which we now provide alternative proofs without assuming uniqueness of shortest paths. Without loss of generality we can assume that all the edges have positive weights. Indeed, we can add an infinitesimal positive weight to edges of zero weight[6]. This does not change the order of the lengths of paths in the graph.

[6]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.

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 length 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 $\lambda(\alpha) = (\ell(\alpha), -\lambda_T(\alpha))$. For the rest of this section, the length of two loops through s will be compared using the lexicographic order under $\lambda(\cdot)$. Thus, α is λ-*shorter* than another loop β through s if $\lambda(\alpha) \preceq \lambda(\beta)$. 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. We can now give a proof of Lemma 1.

Proof of Lemma 1. 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. By definition of the λ-length, we have $\text{loop}_{T,R}(xy) \preceq \alpha$, where $\text{loop}_{T,R}(xy) = \alpha[s,y] \cdot R[y,s]$. To prove the lemma, it is thus enough to show that $\text{loop}_{T,R}(xy)$ is in $\overline{\mathbb{L}}$, hence 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 $\ell(T[s,x]) < \ell(T[s,y])$. In both cases $\lambda(\beta) \prec \lambda(\alpha)$, 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.

We can now conclude exactly as in the first proof of this lemma. By invariance under reversal $\beta^{-1} = R[y,s]^{-1} \cdot T[s,y]^{-1}$ is also in $\mathbb{L}$, and we can 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 $\text{loop}_{T,R}(xy)$ 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 $\mathbb{L}$, a contradiction. □

We replace Lemma 2 by the following version, which uses λ-length.

Lemma 5. *If $\overline{\mathbb{L}}$ is non-empty, then a λ-shortest element of $\mathbb{N}$ is a shortest element of $\overline{\mathbb{L}}$.*

Proof. Assume $\overline{\mathbb{L}}$ is non-empty; by the refined proof of Lemma 1, $\overline{\mathbb{L}}$ has a λ-shortest element of the form $\alpha = \text{loop}_{T,R}(xy)$ for some edge $xy \in E(G) \setminus E(T)$. We first prove that α belongs to $\mathbb{N}$, or equivalently, that $\text{loop}_T(xy) \in \overline{\mathbb{L}}$. The loop $T[s,y] \cdot R[y,s]$ is in $\mathbb{L}$. Indeed, as in the above refined proof we have $\lambda(T[s,y] \cdot R[y,s]) \prec \lambda(\alpha)$ while α is λ-minimal in $\overline{\mathbb{L}}$. 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 conclude that α is also in $\mathbb{L}$, a contradiction.

Let β be a λ-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 $\overline{\mathbb{L}}$. By the preceding paragraph $\lambda(\beta) \preceq \lambda(\alpha)$. By the same reasoning as above $\lambda(T[s,y'] \cdot R[y',s]) \prec \lambda(\beta)$. Hence, $T[s,y'] \cdot R[y',s]$ is strictly λ-shorter than α, so 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 $\mathbb{L}$, again a contradiction. □

The rest of the discussion in Section 3 goes through unchanged.