Algorithms for the Edge-Width of an Embedded Graph[*]

Sergio Cabello[†] Éric Colin de Verdière[‡] Francis Lazarus[§]

July 29, 2010

Abstract

Let G be an *unweighted* graph of complexity n cellularly embedded in a surface (orientable or not) of genus g. We describe improved algorithms to compute (the length of) a shortest non-contractible and a shortest non-separating cycle of G.

If k is an integer, we can compute such a non-trivial cycle with length at most k in $O(gnk)$ time, or correctly report that no such cycle exists. In particular, on a fixed surface, we can test in linear time whether the edge-width or face-width of a graph is bounded from above by a constant. This also implies an output-sensitive algorithm to compute a shortest non-trivial cycle that runs in $O(gnk)$ time, where k is the length of the cycle.

We also give an approximation algorithm for the shortest non-trivial cycle. If a parameter $0 < \varepsilon < 1$ is given, we compute in $O(gn/\varepsilon)$ time a non-trivial cycle whose length is at most $1 + \varepsilon$ times the length of the shortest non-trivial cycle.

Keywords: Topological graph theory, computational topology, edge-width, face-width, surface, embedded graph.

1 Introduction

Let Σ be a surface (orientable or not) of genus g with b boundaries. Let G be an (unweighted, undirected) graph embedded on Σ of *complexity* n (this is the total number of vertices and edges of G), where all the faces of G are disks. In this paper, we are interested in the *edge-width* and *face-width* of G, which roughly measure how G "locally looks planar". Specifically, the *edge-width* of G is the minimum number of edges in a non-contractible cycle of G [2, 23]. The *face-width* of G (also known as *representativity*) is the minimum number of points in the image of G that are contained in a non-contractible curve on Σ [24]. There are also the corresponding concepts of non-separating edge-width and non-separating face-width, where the requirement of being non-contractible is strengthened into being non-separating.

This paper gives improved algorithms to compute these parameters. Before describing our new results in detail, we present some motivations and related works.

[*]A preliminary version appeared in Proc. ACM Symp. on Computational Geometry, 2010, pp. 147–155. 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. E-mail: `sergio.cabello@fmf.uni-lj.si`.

[‡]Laboratoire d'informatique, École normale supérieure, CNRS, Paris, France. Email: `Eric.Colin.de.Verdiere@ens.fr`.

[§]GIPSA-Lab, CNRS, Grenoble, France. E-mail: `Francis.Lazarus@gipsa-lab.grenoble-inp.fr`.

Topological Graph Theory. Edge-width and face-width were introduced in the field of topological graph theory (see Mohar and Thomassen [22, Chapter 5] for a survey). Graph embeddings with large face-width share many properties with planar graphs; we list a selection of them. Every graph embedded in a surface with sufficiently large face-width or edge-width, that depends on the surface, is 5-choosable and thus 5-colorable [27, 9]. Every graph with large face-width can be made planar by cutting along cycles that are far apart from each other [28]. A given graph is a minor of every graph of face-width k embedded on the same surface, when k is sufficiently large [23]. Finally, for any fixed surface there exists a constant bounding the number of embeddings with face-width at least 3 that any 3-connected graph may have [21, 16], thus extending Whitney's result for planar graphs to arbitrary surfaces.

Related Algorithmic Results. The computational aspects of the edge-width and face-width have also been studied. The face-width of G is half the edge-width of the *vertex-face incidence graph* of G, and similarly for the non-separating edge-width; therefore, the problem boils down to computing a shortest *non-trivial* cycle on an embedded graph, where non-trivial means either non-contractible or non-separating. Such cycles are the fundamental tool to perform surgery on embedded graphs. The first algorithm, by Thomassen [26], finds a shortest non-trivial cycle in cubic time. In essence, for each vertex of G, the algorithm computes a breadth-first search (BFS) tree T rooted at that vertex, and, for every non-tree edge e, tests whether the cycle formed by T and e is non-trivial; finally, the shortest such cycle is returned. In particular, the shortest non-trivial cycle has no repeated vertex.

Other papers on this topic study the computation of shortest non-trivial cycles in the more general situation of (non-negatively) *weighted* graphs. Erickson and Har-Peled [11] extend Thomassen's algorithm to this setting and decrease its complexity to $O(n^2 \log n)$, by interleaving Dijkstra's algorithm with tests for triviality. This is the best current result, and an algorithm with subquadratic running time would give rise to a subquadratic-time algorithm to find the girth of sparse graphs [5]. In the same paper, Erickson and Har-Peled [11, Corollary 5.8] give an $O(gn \log n)$-time algorithm that computes a non-trivial cycle whose length is at most twice the length of the shortest non-trivial cycle.

Efficient algorithms for the case where the genus is small have also been investigated [6, 19]. The best known algorithm by Cabello and Chambers [3] computes the shortest non-contractible (resp. non-separating) cycle on an orientable surface in $O((g + b)g^2 n \log n)$ (resp. $O(g^3 n \log n)$). As a consequence, the (possibly non-separating) edge-width and face-width of a graph in a fixed orientable surface can be computed in $O(n \log n)$ time. In another paper [4], we also consider the more general scenario of finding shortest non-trivial cycles in *directed* graphs.

Kawarabayashi and Mohar (private communication) point out that their results in [16] imply that the *face-width* k of a graph can be computed in $2^{O(gk)}n$ time, assuming that $k \geq 3$. This approach relies on graph minors, and in particular, the relations between the face-width and tree-width of embedded graphs. Their algorithm has an exponential dependency on the genus and the face-width, it uses an unknown (but computable) list of minimal graphs, and it does not extend to the problem of computing the edge-width.

The Unweighted Case. It is natural to ask whether the algorithms for the weighted case can be made faster when restricted to unweighted graphs. They all use shortest path trees, computed in $O(n \log n)$ time in the weighted case using Dijkstra's algorithm; this step can be speeded up to linear time in the unweighted case, for which BFS trees are shortest path trees. However, the current results are also relying on the minimum cut problem in planar graphs [19] or on dynamic trees [3], and require an extra logarithmic factor that is independent from the shortest path tree computation. The $O(n^2 \log n)$ time algorithm by Erickson and Har-Peled [11, Lemma 5.2] immediately gives an $O(n^2)$ time algorithm for the non-contractible case, but in the non-separating case, the complexity is still $O(n^2 \log n)$, because the extra logarithmic factor appears in their recurrence, independently of the shortest path tree computation.

Our Results. Recall that, in this paper, G is an unweighted graph of complexity n cellularly embedded on a surface Σ (orientable or not) of genus g with b boundaries. Our first result is an efficient algorithm to decide whether the edge-width is bounded from above by a given integer:

Theorem 1. *Given an integer k, we can decide in $O((g+b)nk)$ time (resp. $O(gnk)$ time) if the edge-width (resp. non-separating edge-width) of G is at most k. If it is the case, we can also obtain a shortest non-contractible (resp. non-separating) cycle in G.*

In particular, on a fixed surface, we can decide in linear time whether the edge-width is bounded from above by a constant.

By running the algorithm of Theorem 1 for exponentially increasing values of k, we obtain an efficient output-sensitive algorithm:

Corollary 2. *Let k be the edge-width (resp. non-separating edge-width) of G. We can compute a shortest non-contractible (resp. non-separating) cycle in G in $O(gnk)$ (resp. $O((g+b)nk)$) time.*

We also give an efficient algorithm to compute an approximation of the edge-width; this is an enhancement over the 2-approximation algorithm by Erickson and Har-Peled [11, Corollary 5.8] mentioned above:

Theorem 3. *Let ε be a parameter such that $0 < \varepsilon < 1$. We can compute a non-contractible (resp. non-separating) cycle in G whose length is at most $1+\varepsilon$ times the length of the shortest such cycle, in $O((g+b)n/\varepsilon)$ (resp. $O(gn/\varepsilon)$) time. In particular, we approximate the (possibly non-separating) edge-width of G up to a factor of ε.*

Incidentally, we give alternate algorithms to find shortest non-trivial loops (possibly in weighted graphs). Our algorithms are (arguably) simpler to implement than those by Erickson and Har-Peled [11, Lemma 5.2]; some ideas of the proof are inspired from the paper by Erickson and Whittlesey to compute shortest homotopy generators [12]. Compared to Erickson and Har-Peled [11], our algorithms are faster by a logarithmic factor for the non-separating case in unweighted graphs, and have the same asymptotic running-time otherwise:

Theorem 4. *We can compute a shortest non-contractible or non-separating loop through a given basepoint in G in $O(n)$ time.*

Again, we note that all the results above about the (possibly non-separating) edge-width parameter have immediate counterparts for the (possibly non-separating) face-width, because the *vertex-face incidence graph* Γ of G (also called *radial graph*) is also embedded on Σ, has the same asymptotic complexity as G, and the face-width of G equals half of the edge-width of Γ [22, Proposition 5.5.4]. We also emphasize that all of our algorithms are quite simple and do not require heavy data structures like self-adjusting top trees (needed by Cabello and Chambers [3]).

Our output-sensitive complexity can be combined with combinatorial bounds on the edge-width and face-width that are known. Hutchinson [15] showed that a triangulation with m vertices in an orientable surface without boundary has edge-width $O(\sqrt{m/g} \log g)$ if $g \leq m$ and $O(\log g)$ if $g > m$. This result can be extended to non-orientable surfaces, and the same bound applies to the face-width of arbitrary embedded graphs [6, Lemma 13 and Theorem 14]. Therefore, our algorithm implies that the edge-width of a triangulation and the face-width of an arbitrary graph in a surface (orientable or not) without boundary can be computed in $O(n^{3/2}g^{1/2}\log g)$ time, provided that $g \leq n$. This time bound is subquadratic unless $g \log^2 g = \Omega(n)$.

Applications. In topological graph theory there are several results of the following form: for any fixed surface Σ there exists a constant $c = c(\Sigma, \Pi)$ such that any graph embedded in Σ with face-width (or edge-width) at least c has property Π. See for example [27, 9, 28, 23, 21], and [22, Chapter 5]. Our results provide a linear-time algorithm to test if the hypotheses of those results are fulfilled.

Getting bounds on the edge-width or the face-width has been explicitly used as subroutine in algorithms that work in linear time on a surface of bounded genus, for computing crossing numbers of graphs [17], for graph isomorphism of graphs that admit polyhedral embeddings [16], and for finding certain induced cycles in embedded graphs [18]. In these papers, the authors use the 2-approximation of the edge-width mentioned above. Using this 2-approximation instead of the real edge-width affects exponentially the running time; however, this is hidden in the O-notation because the authors consider a fixed surface. Using our Theorem 1 instead removes this overhead.

Also, there is a closed formula telling the orientable genus of a graph that can be embedded in the projective plane. Indeed, Fiedler et al. [13] have shown that a graph G that can be embedded in the projective plane with face-width $k \neq 2$ has orientable genus $\lfloor k/2 \rfloor$. Our results imply an algorithm to compute the orientable genus $g(G)$ of such graphs in $O(g(G)n)$ time.

The techniques we use here to prove Theorem 4 are also useful in another paper [4], where we obtain efficient algorithms that find shortest non-trivial cycles in a more general setting than previously studied, namely, for *directed* weighted graphs on surfaces.

Overview of the Techniques. Our algorithm to obtain Theorem 1 consists of two steps: (1) We first compute a set of vertices K such that any non-trivial cycle has to use some vertex of K. (2) For every vertex s in K, we compute the shortest non-trivial cycle passing through s in the graph induced by the vertices at distance at most $k/2$ from s. The key idea in our approach is to find an efficient way to carry Step (2) simultaneously for several basepoints that are far apart on the surface. This idea, in turn, requires to choose K in Step (1) adequately. This strategy also requires to be able to test in constant time whether a cycle with a special structure is trivial; we introduce a technique for this purpose, which implies Theorem 4, of independent interest.

For the proof of Theorem 3, we start by computing a 2-approximation of the edge-width. Then we proceed as in the previous paragraph, but since we are only interested in an approximately shortest non-trivial cycle, we can discard some vertices of K to speed up the algorithm.

The rest of the paper is organized as follows. After some preliminaries (Section 2), we prove Theorem 4 in Section 3. In Section 4, we introduce two other tools to compute shortest non-trivial loops with given basepoints in "parallel" and to achieve Step (1) above; we also recall the 2-approximation algorithm. Finally, in Section 5, we prove the main results: Theorem 1, Corollary 2, and Theorem 3.

2 Background and Terminology

2.1 Graph Theory

All the considered graphs may have loop edges and multiple edges. We sometimes denote by xy an edge with endpoints x and y: even if this notation is ambiguous in presence of multiple edges, it will always be clear from the context which edge is considered. A *walk* in a graph 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 $i = 1, \ldots, m-1$; such a walk is *closed* if the target of e_m is the source of e_1. A *path* is a walk without repeated vertices; a *cycle* is a closed walk without repeated vertices. A *loop* with basepoint s is a closed walk with a distinguished occurrence of vertex s. We denote by $V(G)$ and $E(G)$ the set of vertices and edges of a graph G, respectively. For a subset $X \subseteq V(G)$, we use $G[X]$ to denote the subgraph of G induced by X. For an edge e of G, we denote by $G - e$ the graph G without that edge.

Suppose G is connected; consider a spanning tree T of G. For any vertex s and any edge uv of G, we denote by $\tau(T, s, uv)$ the loop consisting of the path in T from s to u, the edge uv, and the path in T from v to s. We also denote by $\tau(T, uv)$ the closed walk consisting of the edge uv and the path in T between u and v. Note that $\tau(T, uv)$ is a cycle, if uv is not in T.

2.2 Surfaces

We review some basic topology of surfaces. See any of the books by Hatcher [14], Massey [20], or Stillwell [25] 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. For simplicity, we define the *reduced genus* $\bar{g}$ to be $2g$, if Σ is orientable, and g, otherwise. $\overline{\Sigma}$ denotes the surface without boundary obtained by attaching a disk to each boundary component of Σ.

2.3 Graph Embeddings

An *embedding* of a graph G in a surface Σ is a drawing of G on Σ without crossings. More precisely, the vertices of G are mapped to distinct points of the interior 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 in Σ.

In this paper, G is *cellularly embedded* on Σ if the faces of G on $\overline{\Sigma}$ are (homeomorphic to) open 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. For a cellularly embedded graph G with V vertices, E edges, and F faces, *Euler's formula* states that $V - E + F = 2 - \bar{g}$.

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 [10], or rotation systems [22]. For orientable surfaces, one can also use the DCEL that is customary in Computational Geometry (see, e.g., de Berg et al. [8]). 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.

If G is a graph embedded on Σ without isolated vertices, we will denote by $\Sigma \backslash\!\backslash G$ the surface with boundary obtained after cutting Σ along G. Note that $\Sigma \backslash\!\backslash G$ is a surface with boundary. In contrast, $\Sigma \setminus G$ is a set operation where we remove from Σ the points in (the image of the embedding of) G. In particular, if G is cellularly embedded, $\overline{\Sigma} \backslash\!\backslash G$ is a union of closed disks, whereas $\overline{\Sigma} \setminus G$ is a union of open disks.

We say that G *separates* Σ if $\Sigma \backslash\!\backslash G$ (equivalently $\Sigma \setminus G$) has at least two connected components.

2.4 Homotopy and Homology

Let G be a graph embedded on Σ. Homotopy and homology are two equivalence relations on the set of loops in G with a given basepoint. Here, we stick to a concise description of these notions and refer to one of the aforementioned books for a more formal and detailed treatment.

Let us fix a common basepoint for all loops considered in this section. Two loops in G are *homotopic* if one can be deformed continuously to the other on the surface, keeping the basepoint fixed during the deformation. It turns out that the equivalence classes, called *homotopy classes*, form a group, where the multiplication in the group corresponds to the concatenation of the loops. The zero element is the set of loops that are homotopic to the constant loop; such loops are called *contractible*, or *homotopically trivial.* An important characterization is that a simple loop is contractible if and only if it bounds a disk on the surface.

Homology is a coarser equivalence relation than homotopy. The homology group is the abelianization of the homotopy group. A loop in G is *null-homologous*, or *homologically trivial*, if it belongs to the zero homology class. We also have a nice characterization for simple loops: a simple loop is null-homologous on $\overline{\Sigma}$ if and only if it separates $\overline{\Sigma}$ (or, equivalently, Σ). As in previous papers, when we write "separating on Σ", we really mean "null-homologous on $\overline{\Sigma}$": these two notions coincide for simple loops, and the latter is also defined for non-simple loops, which turns out to be useful. Therefore, contractible implies separating, even for non-simple loops.

If G is cellularly embedded on Σ, then G contains non-contractible cycles except when Σ is the sphere or the disk. Also, G contains non-separating cycles except when $\overline{\Sigma}$ is the sphere.

When considering the problem of finding a shortest non-contractible loops or cycles, we assume every face of G contains at most one boundary of Σ. This is possible since several boundary components of Σ in one face of G can be replaced with one single boundary component without changing the contractibility character of the loops in G. When considering the computation of shortest non-separating loops or cycles, we assume our input surface Σ has no boundary. This is valid since a cycle is separating in Σ if and only if it is separating in $\overline{\Sigma}$.

Henceforth, we use the term *non-trivial* as a shorthand for non-contractible or non-separating.

2.5 Duality

Let G be a graph cellularly embedded in Σ. Its *dual graph*, denoted by G^*, has for vertices the set of faces of Σ and for edges the set of edges (dual to) $E(G)$: 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. The dual graph G^* has a natural embedding in Σ: each vertex f^* of G^*, corresponding to face f of G, is assigned to a point p_f of the interior of the face f; for each edge uv of G, incident to faces f and f', the dual edge $(uv)^*$ is assigned a curve that connects the points p_f and $p_{f'}$ and crosses G precisely at the edge uv. For our discussion, it is convenient to make the following assumptions: for every face f of G containing a boundary component (which is unique in that face by our assumption of Section 2.4) the point p_f belongs to that boundary component, and we add to G^* a loop edge, whose image is the boundary component. (Such loop edges correspond to no edge of the primal graph G.) For a set of edges $A \subseteq E(G)$, we use the notation $A^* = \{e^* \mid e \in A\}$.

2.6 Deformation retract

A subspace A of a topological space X is a *deformation retract* of X if there is a continuous map $F : X \times [0,1] \to X$ such that for every x in X and a in A, we have $F(x,0) = x$, $F(x,1) \in A$, and $F(a,1) = a$.

It is known that if A is a deformation retract of X, then X and A have the same *homotopy type*. The (quite intuitive) consequence that will be useful to us is that, under this condition, A and X have the same number of connected components, and A has a non-contractible loop if and only if X has a non-contractible loop.

3 Shortest Non-Trivial Loops

In this section, we prove Theorem 4; the tools developed here will be useful for the proof of Theorems 1 and 3 as well.

As already noted, for the non-contractible case, Theorem 4 follows directly from Erickson and Har-Peled [11, Lemma 5.2] by replacing Dijkstra's algorithm with a breadth-first search. For the non-separating case, this result is new and improves upon previous papers [11, 6, 3] in the case of unweighted graphs. The ideas are closely related to Erickson and Whittlesey's algorithm to compute a shortest system of loops [12] (specifically, a shortest non-separating loop is the first loop computed by their greedy algorithm, although they do not compute it in linear time). The idea of computing shortest non-trivial loops using this method appears in one of the authors' course notes [7].

Let T be an arbitrary spanning tree of G; let C^* be the subgraph of G^* with the same vertex set as G^* and edge set $E(G^*) \setminus E(T)^*$. (In particular, C^* contains every loop edge of G^* in a boundary component of Σ.)

Lemma 5. *C^* is a cut graph of Σ; that is, $\Sigma \setminus C^*$ is (homeomorphic to) an open disk.*

Proof. $\Sigma \setminus C^*$ can be obtained by taking all faces of G^* (which are open disks) and gluing them in a tree-like fashion according to the tree T, i.e., along the edges of $E(T)^*$. Since attaching disks in this way gives a disk, we obtain that $\Sigma \setminus C^*$ is a disk. □

The following lemma was also noted and used by Erickson and Whittlesey [12, Section 3.4].

Lemma 6. *Let $e \in E(G) \setminus E(T)$. Then $C^* - e^*$ is a deformation retract of $\Sigma \setminus \tau(T, e)$.*

Proof. The cycle $\tau(T, e)$ cuts C^* at exactly one point. Therefore this cycle corresponds to a path intersecting the boundary of the closed disk $\Sigma \backslash\backslash C^*$ exactly at its endpoints. (See Figure 1.) Both halves of the disk retract to the corresponding portion of the boundary of the disk. Therefore, $\Sigma \setminus \tau(T, e)$ retracts to $C^* \setminus (e^* \cap \tau(T, e))$. This in turn retracts to $C^* - e^*$. □

Corollary 7. *Let $e \in E(G) \setminus E(T)$. The cycle $\tau(T, e)$ is separating on Σ if and only if $C^* - e^*$ is not connected. The cycle $\tau(T, e)$ is contractible if and only if $C^* - e^*$ has a connected component that is a tree (possibly reduced to a single vertex).*

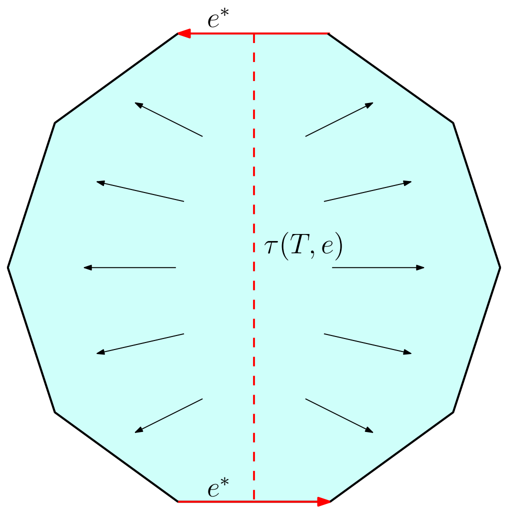

Figure 1: The retraction in the proof of Lemma 6. The boundary of the disk is the boundary of $\Sigma \backslash\backslash C^*$.

Proof. The cycle $\tau(T, e)$ is separating if and only if $\Sigma \setminus \tau(T, e)$ is not connected; by Lemma 6, this holds if and only if $C^* - e^*$ is not connected.

$\tau(T, e)$ is contractible if and only if one component of $\Sigma \setminus \tau(T, e)$ is a disk, i.e., has no non-contractible loop. By Lemma 6, this holds if and only if one component of $C^* - e^*$ has no non-contractible loop, i.e., is a tree. □

A cycle $\tau(T, e)$ is of one of the following three topological types: contractible, non-contractible but separating, and non-separating. We can partition the edges e^* of C^* into three sets, depending on the corresponding type of $\tau(T, e)$. Figure 2 illustrates this classification.

For later use, it will be convenient to use $E_{\text{non-con}}(T)$ (resp. $E_{\text{non-sep}}(T)$) for the set of edges e in $E(G) \setminus E(T)$ such that $\tau(T, e)$ is non-contractible (resp. non-separating). As before, we use $E_{\text{non-triv}}(T)$ as an ambiguous term to refer to $E_{\text{non-con}}(T)$ or $E_{\text{non-sep}}(T)$ as needed.

Lemma 8. *The sets $E_{\text{non-con}}(T)$ and $E_{\text{non-sep}}(T)$ can be computed in $O(n)$ time.*

Proof. We construct the cut graph C^* in linear time. By Corollary 7, $E_{\text{non-con}}(T)$ can be obtained by the following procedure: starting with C^*, we repeatedly remove edges with an incident vertex of degree one; the edge set of the resulting graph is then exactly $E_{\text{non-con}}(T)^*$. To obtain $E_{\text{non-sep}}(T)$, note that, by Corollary 7, $E_{\text{non-sep}}(T)^*$ 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 (see Aho et al. [1, Section 5.3] for the similar problem of determining biconnected components). □

Let s be an arbitrary vertex of G. We have the following structural result on shortest non-trivial loops based at s.

Lemma 9. *Assume T is a BFS tree from root s. Every shortest loop among the loops $\tau(T, s, e)$, where $e \in E_{\text{non-triv}}$, is a shortest non-trivial loop through s.*

Proof. A proof appears in Thomassen [26] in a more general setting. We provide an ad hoc short proof. Let ℓ be a non-trivial loop with basepoint s. We show that some non-trivial loop $\tau(T, s, e)$ is no longer than ℓ. Let $e_1, e_2, \ldots, e_k$ be the edges of ℓ, in the same order as they appear along ℓ.

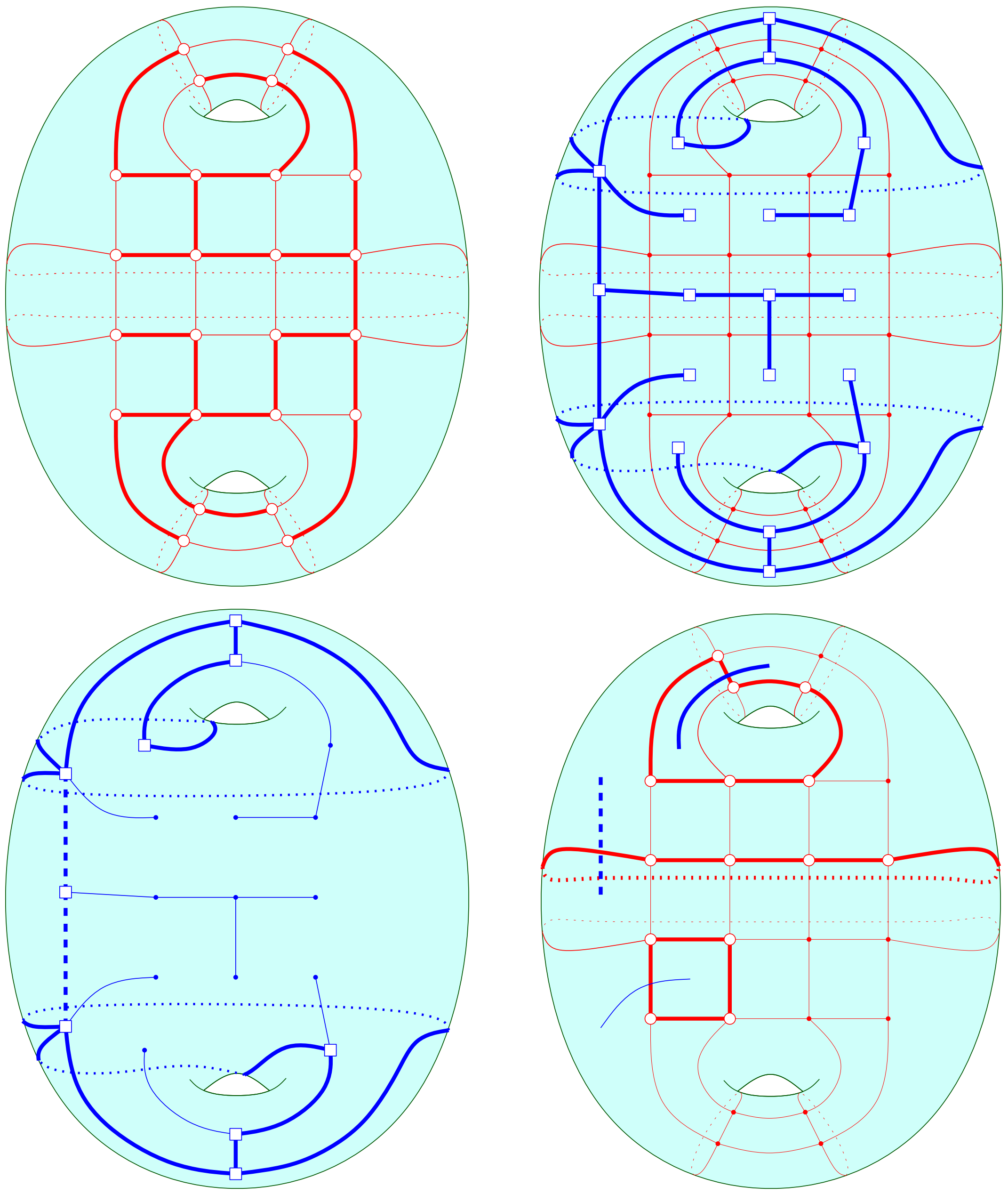

Figure 2: Top left: A graph G embedded in a double torus with a spanning tree T marked with bolder edges. Top right: The cut graph C^* is marked with bold edges; thinner edges are from the primal graph. Bottom left: classification of the edges of C^*. The bold solid edges are $E_{\text{non-sep}}(T)^*$; the bold dashed edges are $(E_{\text{non-con}}(T) \setminus E_{\text{non-sep}}(T))^*$; the thin edges are $E(C^*) \setminus E_{\text{non-con}}(T)^*$. The dotted parts of edges do not provide any information about their type. Bottom right: examples of the loop $\tau(T, e)$ in bold for different types of $e^* \in C^*$.

Since ℓ is homotopic in Σ (and therefore also homologous in $\overline{\Sigma}$) to the concatenation of loops $\tau(T,s,e_1)\cdot\tau(T,s,e_2)\cdots\tau(T,s,e_k)$, at least one of the loops $\tau(T,s,e_i)$ is non-trivial. However, $\tau(T,s,e_i)$ is a shortest loop through s that contains e_i because T is a BFS tree, whence $\tau(T,s,e_i)$ is no longer than ℓ. Furthermore, e_i cannot belong to T, for otherwise the loop $\tau(T,s,e_i)$ would be contractible (and separating). $\square$

We conclude the proof of Theorem 4.

Proof of Theorem 4. We construct a BFS tree T of G from s in linear time. We attach to each vertex u of G a label $d(u)$ equal to its distance from s. We then compute $E_{\text{non-triv}}(T)$, again in linear time, using Lemma 8. Among the edges e of $E_{\text{non-triv}}(T)$, we select an edge e_0 minimizing the length of $\tau(T,s,e)$, or equivalently, minimizing the sum $d(u)+d(v)$ where u and v are the endpoints of e. Finally, we report the loop $\tau(T,s,e_0)$. The procedure takes linear time; its correctness follows from Lemma 9 and from the fact that $\tau(T,e)$ is non-trivial if and only if $\tau(T,s,e)$ is non-trivial. $\square$

Our algorithm trivially extends to the weighted case, at the expense of a logarithmic factor, by replacing the BFS with a shortest path tree computation.

Also, the same ideas yield algorithms to compute shortest loops with an odd number of edges and shortest one-sided loops (which reverse the orientation of the surface, when it is non-orientable). Lemma 9 extends immediately to these two problems. To compute a shortest loop with an odd number of edges, it suffices to store, on each vertex of G, the parity of the number of edges to the root; then $\tau(T,s,e)$ has an odd number of edges if and only if the parities of the vertices of G incident with e are the same. To compute a shortest one-sided loop, it suffices to choose local orientations of the surface at each vertex that are consistent across each edge of T; then $\tau(T,s,e)$ is one-sided if and only if the orientations of the two vertices of G incident with e do not match across edge e. The results of the next section extend to the problem of finding shortest one-sided cycles, but do *not* extend to the problem of finding a shortest cycle with an odd number of edges.

4 Tools for Shortest Non-Trivial Cycles

4.1 Shortest Non-Trivial Loops with Remote Sources in Parallel

The following result will be our tool to work with several sources simultaneously.

Lemma 10. *Let $V_1,\ldots,V_t$ be subsets of $V(G)$ that are pairwise disjoint, let $s_1,\ldots,s_t$ be vertices with $s_i\in V_i$ for each i, and let $\mathbb{L}_i$ be the set of non-trivial loops through s_i contained in $G[V_i]$. In $O(n)$ time we can find a shortest loop in $\bigcup_i \mathbb{L}_i$, or correctly report that $\bigcup_i \mathbb{L}_i$ is empty.*

Proof. We first describe the algorithm interlaced with an analysis of its running time, and then discuss its correctness.

For each i, we construct a BFS tree T_i with root s_i of the component of $G[V_i]$ that contains s_i. To each vertex u of G, we attach two labels, $c(u)$ and $d(u)$. The label $c(u)$ has value i if u is in the same connected component of $G[V_i]$ as s_i, and has value 0 otherwise. The label $d(u)$ is the distance between u and $s_{c(u)}$ if $c(u)\geq 1$, and undefined if $c(u)=0$. Since the sets $V_1,\ldots,V_t$ are disjoint, the labels $c(u)$ and $d(u)$ are uniquely defined. These labels can be computed in $O(n)$ time using the BFS trees $T_1,\ldots,T_t$.

We then extend the forest $T_1,\ldots,T_t$ to a spanning tree T of G. This can also be done in linear time. Next, we compute $E_{\text{non-triv}}(T)$ using Lemma 8. Let $\tilde{E}$ be the subset of the edges

$uv \in E_{\text{non-triv}}(T)$ such that $c(u) = c(v)$ and this number is non-zero. If $\tilde{E}$ is empty, we report that $\bigcup_i \mathbb{L}_i$ is empty. Otherwise, we compute

$$xy = \arg\min\{d(u) + d(v) \mid uv \in \tilde{E}\}$$

and return the loop $\tau(T, s_{c(x)}, xy)$. The construction of $E_{\text{non-triv}}(T)$ and $\tilde{E}$ takes linear time, and we spend additional constant time per edge in $\tilde{E}$ to find the edge xy. This concludes the description of the algorithm.

We now show the correctness of the algorithm. Let E_i be the set of edges uv in $E(G)$ with endpoints in T_i, i.e., such that $c(u) = c(v) = i$. Also, let $E_{\text{non-triv}}(T_i)$ be the subset of the edges e in E_i such that $\tau(T_i, s_i, e)$ is non-trivial.

By the same arguments as in the proof of Lemma 9, if $\mathbb{L}_i$ is not empty, then it contains a shortest loop of the form $\tau(T_i, s_i, e)$ for some edge e in E_i. As a consequence, a shortest non-trivial loop in $\mathbb{L}_i$ is given by $\tau(T_i, s_i, e)$ where

$$e = \arg\min\{d(u) + d(v) \mid uv \in E_{\text{non-triv}}(T_i)\}.$$

For any edge e in E_i, it holds that $\tau(T_i, s_i, e) = \tau(T, s_i, e)$ and $\tau(T_i, e) = \tau(T, e)$. It follows that

$$E_{\text{non-triv}}(T_i) = E_{\text{non-triv}}(T) \cap E_i,$$

whence $\tilde{E} = \bigcup_i E_{\text{non-triv}}(T_i)$. The correctness of the algorithm directly follows. □

4.2 Surface Decomposition

Lemma 11. *Let s be a vertex of G. In $O(n)$ time, we can compute a set K of vertices of G that intersects every non-trivial closed walk in G and is the union of the vertex sets of m shortest paths from s, where $m = 2\bar{g} + b$ for the non-contractible case and $2\bar{g}$ for the non-separating case.*

Proof. Assume first that Σ has no boundary (this is the only relevant situation in the non-separating case). We use the tree-cotree decomposition of Eppstein [10]. Consider a BFS tree T from the vertex s. Let $(T')^*$ be an arbitrary spanning tree of $G^* - E(T)^*$. Thus T and T' are edge-disjoint in G; we let X be the remaining edges of G. It follows from Euler's formula that X has $\bar{g}$ edges.

Consider the set of loops $\mathbb{L} = \{\tau(T, s, e) \mid e \in X\}$: their union $\bigcup \mathbb{L}$ is a cut graph. Indeed, $\Sigma \setminus\!\!\setminus (T \cup X)$ is a set of faces connected according to the dual tree T', hence a disk. But $\bigcup \mathbb{L}$ is obtained from $T \cup X$ by iteratively removing a degree one vertex with its incident edge, and this operation preserves the fact that the complement is a disk. Let K be the set of vertices in $\mathbb{L}$. Since $\bigcup \mathbb{L}$ is a cut graph, each non-trivial closed walk must intersect K. Moreover, since T is a BFS tree, each of the $\bar{g}$ loops $\tau(T, s, e)$ in $\mathbb{L}$ consists of two shortest paths from s.

K can be computed in linear time, as we describe next. Computing a BFS tree T from s, the spanning tree in the dual graph, and computing the set of edges X takes linear time. We mark in T the vertex s and the endpoints of the edges in X. By definition of the loops in $\mathbb{L}$, the set K is the set of vertices of the minimal subtree of T that includes the marked vertices. This subtree is obtained in linear time by recursively removing unmarked degree one vertices.

It remains to consider the case where we want to compute a shortest non-contractible cycle on a surface Σ with boundary. In this case we attach a handle to every boundary component of Σ, obtaining a surface $\hat{\Sigma}$ without boundary. To make G cellular on $\hat{\Sigma}$, we just have to add two loop edges per boundary component of Σ; see Figure 3. Let $\hat{G}$ be the new graph. Then we apply the previous construction to this new graph. We obtain a set $\hat{K}$ that intersects every cycle of $\hat{G}$ that is non-trivial on $\hat{\Sigma}$; furthermore, $\hat{K}$ is in the union of $2\bar{g} + b$ shortest paths from the source s. (The

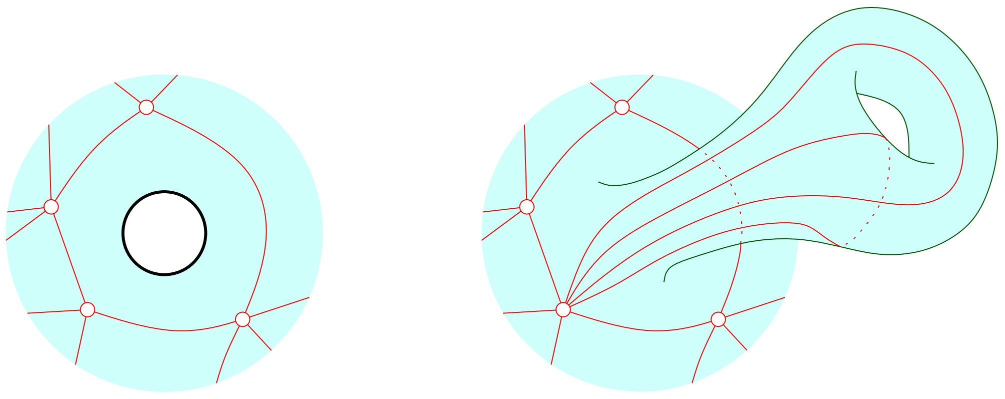

Figure 3: Detail of the graph $\hat{G}$ cellularly embedded in $\hat{\Sigma}$, as constructed in the proof of Lemma 11. We attach a handle to each boundary component and add two loop edges to obtain a cellular embedding.

two loop edges e_1 and e_2 defining a handle contribute to a single shortest path to K, namely, the shortest path from s to the common endpoint of e_1 and e_2.) Hence $\hat{K}$ intersects also every cycle in G that is non-trivial in $\hat{\Sigma}$. Furthermore, the shortest paths from s in G and $\hat{G}$ are the same, because we only added loop edges, so $\hat{K}$ lies on $2\bar{g}+b$ shortest paths from s in G. To conclude, note that a cycle is contractible in $\hat{\Sigma}$ if and only if it is contractible in Σ. So we can take $K = \hat{K}$. ☐

4.3 2-Approximation for the Edge-Width

Another tool we will need is the algorithm by Erickson and Har-Peled [11, Corollary 5.8] to compute a 2-approximation of the edge-width. They describe it in the case of weighted graphs; in our setting, we can shave off a logarithmic factor using Theorem 4. We sketch the proof for convenience, and also because most of the tools have already been presented.

Proposition 12. *We can compute a non-trivial cycle whose length is at most twice the length of a shortest non-trivial cycle, in $O((g+b)n)$ time for the non-contractible case and $O(gn)$ time for the non-separating case.*

Proof. Let π be a shortest path in G. Erickson and Har-Peled [11, Lemma 5.6] give an algorithm that computes in $O(n \log n)$ time a non-trivial loop intersecting π whose length is at most twice the length of the shortest non-trivial loop intersecting π. Their algorithm is the following: (1) contract π to a single point; (2) compute the shortest non-trivial loop ℓ through that point; (3) expand back π, and return the loop resulting from ℓ after the expansion. Since π is a shortest path, the expansion can at most double the length of the loop, which is therefore a 2-approximation of the shortest non-trivial loop intersecting π. Since our graph is unweighted, we can use Theorem 4 in step (2), and therefore the algorithm runs in $O(n)$ time.

By Lemma 11, we can compute a set of $O(g+b)$ or $O(g)$ shortest paths intersecting every non-trivial loop. We can apply the previous algorithm to each of these shortest paths to obtain a non-trivial loop ℓ' whose length is at most twice the length of a shortest non-trivial cycle. By construction ℓ' is either a cycle or a loop composed of a starting path concatenated with a cycle and the reverse of the starting path. In the latter case, we simply remove the starting and ending paths to get a non-trivial cycle. ☐

5 Proofs of the Main Results

We are now ready to give the proofs of our main results.

5.1 Output-Sensitive Algorithm

Proof of Theorem 1. We start by computing the set K of vertices as in Lemma 11. It then suffices to compute a shortest non-trivial loop based at some vertex in K, or to determine that every such loop has length larger than k.

Let S_i be the set of vertices in K at distance exactly i from s ($0 \leq i \leq n$). Each S_i has cardinality at most $2\bar{g}+b$ (non-contractible case) or $2\bar{g}$ (non-separating case). For simplicity, in the remaining part of the proof, we only consider the non-contractible case; the non-separating case is the same, except that we can replace $2\bar{g}+b$ by $2\bar{g}$.

For each j, $0 \leq j \leq k$, we put the elements of

$$S_j, S_{(k+1)+j}, S_{2(k+1)+j}, \ldots$$

into at most $2\bar{g}+b$ batches, each containing at most one element from each of these S_i. In total, we have a partition of K into $O((g+b)k)$ batches such that any two vertices in the same batch are at distance at least $k+1$ from each other (because an element in S_i and an element in S_j are at distance at least $|i-j|$ from each other by the triangle inequality).

Now, consider a single batch $\{s_1, \ldots, s_t\}$. For each i, let V_i be the set of vertices at distance at most $k/2$ from s_i; the V_i's are pairwise disjoint. We can thus apply Lemma 10: if there exists a non-trivial loop based at some s_i that has length at most k, we obtain the shortest such loop.

We apply this operation for every batch; thus we computed, for each vertex of K, the shortest loop based at that vertex, unless that loop has length larger than k. If the shortest of the resulting loops has length at most k, this is the output of the algorithm. Otherwise (or if no loop has been found), we report that no non-trivial loop with length at most k exists.

The set K and the $O((g+b)k)$ batches can be computed in $O(n)$ time. Then the $O(n)$ time algorithm of Lemma 10 is applied once for each of the $O((g+b)k)$ batches; thus the total running time is $O((g+b)nk)$. □

The proof of our output-sensitive algorithm is now easy:

Proof of Corollary 2. The edge-width can be computed applying Theorem 1 with $k = 2^0, 2^1, \ldots, 2^i, \ldots$ until a non-contractible cycle is found. The total cost is

$$O((g+b)n(2^{\lceil \log k \rceil + 1})) = O((g+b)nk),$$

where k is the edge-width. (Alternately, we may use Proposition 12 to compute a 2-approximation of the edge-width, and then apply Theorem 1 only once.) The same arguments, with $g+b$ replaced by g, yield the result for the non-separating edge-width. □

5.2 Approximation Algorithm

Finally, we prove Theorem 3.

Proof of Theorem 3. Again, we only consider the non-contractible case; the non-separating case is handled in the same way. Let k be the edge-width of G. Using Proposition 12, we compute in $O((g+b)n)$ time an integer k' such that $k \leq k' \leq 2k$.

Let K be the set of vertices of G obtained by applying Lemma 11; in particular, K intersects every non-trivial cycle. For $i \geq 0$, let S_i be the set of vertices of K at distance i from the fixed vertex s. We put $S'_i = S_{i \cdot \lceil k'\varepsilon/4 \rceil}$ and consider the subset $K' = \cup_i S'_i$ of K.

We claim that some non-trivial loop of length at most $(1+\varepsilon)k$ must intersect K'. Indeed, let ℓ be a shortest non-trivial cycle; ℓ contains a vertex v of K. From the construction of K, the vertices of the shortest path π from v to s are in K. This shortest path contains a vertex w in K' at distance at most $k'\varepsilon/4 \leq k\varepsilon/2$ from v. Using the subpath of π between w and v as an approach path to ℓ, we obtain a non-trivial loop of length at most $(1+\varepsilon)k$ through w. This proves the claim.

Recall that, as a consequence of Lemma 11, every set S'_i contains at most $2\bar{g} + b$ elements. Let $t = \left\lceil \frac{(1+\varepsilon)(k'+1)}{\lceil k'\varepsilon/4 \rceil} \right\rceil = O(1/\varepsilon)$. For each j, $0 \leq j < t$, we put the elements of

$$S'_j, S'_{t+j}, S'_{2t+j}, \ldots$$

into at most $2\bar{g} + b$ batches, each containing at most one element from each of these S'_i. In total, we have a partition of K' into $O((g+b)/\varepsilon)$ batches such that any two vertices in the same batch are at distance at least $t\lceil k'\varepsilon/4 \rceil \geq (1+\varepsilon)(k'+1)$ from each other (because an element in S_i and an element in S_j are at distance at least $|i-j|$ from each other by the triangle inequality). Within each batch, we apply Lemma 10, where for each source s_i in the batch we take V_i as the set of vertices at distance at most $(1+\varepsilon)k'/2$ from s_i. In $O(n)$ time, we find a shortest non-trivial loop of length at most $(1+\varepsilon)k'$ through a given vertex in the batch, or determine that no such loop exists.

We do this for each batch. This takes $O((g+b)n/\varepsilon)$ time in total. Finally, we return the shortest loop between the shortest non-trivial loops found over all batches. We know by the above claim that the computation will find in some batch a non-trivial loop of length at most $(1+\varepsilon)k$. As at the end of the proof of Proposition 12, it remains to extract a cycle from this loop, if needed. ☐

Acknowledgments

We thank Jeff Erickson for pointing out reference [15] and its implications, and Luca Castelli Aleardi for an inspiring discussion.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer programs.* Addison-Wesley, 1974.

[2] M. O. Albertson and J. P. Hutchinson. The independence ratio and genus of a graph. *Transactions of the American Mathematical Society*, 226:161–173, 1977.

[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. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG)*, pages 156–165, 2010.

[5] S. Cabello, M. DeVos, J. Erickson, and B. Mohar. Finding one tight cycle. *ACM Transactions on Algorithms*, 2010. To appear. Preliminary version in SODA'08.

[6] 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.

[7] É. Colin de Verdière. Algorithms for graphs on surfaces. Course notes, available at `http://www.di.ens.fr/~colin/cours/algo-graphs-surfaces.pdf`, 2008.

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

[9] M. DeVos, K.-i. Kawarabayashi, and B. Mohar. Locally planar graphs are 5-choosable. *J. Comb. Theory Ser. B*, 98(6):1215–1232, 2008.

[10] 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.

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

[12] 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.

[13] J. R. Fiedler, J. P. Huneke, R. B. Richter, and N. Robertson. Computing the orientable genus of projective graphs. *J. Graph Theory*, 20(3):297–308, 1995.

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

[15] J. P. Hutchinson. On short noncontractible cycles in embedded graphs. *SIAM Journal on Discrete Mathematics*, 1(2):185–192, 1988.

[16] K.-i. Kawarabayashi and B. 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.

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

[18] Y. Kobayashi and K.-i. 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.

[19] 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.

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

[21] B. Mohar and N. Robertson. Flexibility of polyhedral embeddings of graphs in surfaces. *J. Comb. Theory Ser. B*, 83(1):38–57, 2001.

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

[23] N. Robertson and P. D. Seymour. Graph minors. VII. Disjoint paths on a surface. *Journal of Combinatorial Theory, Series B*, 45:212–254, 1988.

[24] N. Robertson and R. P. Vitray. Representativity of surface embeddings. In B. Korte, L. Lovász, and Prömel, editors, *Paths, flows, and VLSI-layout*, pages 293–328. Springer-Verlag, Berlin, 1990.

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

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

[27] C. Thomassen. Five-coloring maps on surfaces. *J. Comb. Theory Ser. B*, 59(1):89–105, 1993.

[28] X. Yu. Disjoint paths, planarizing cycles, and spanning walks. *Transactions of the American Mathematical Society*, 349:1333–1358, 1997.