Minimum Cuts and Shortest Homologous Cycles*

Erin W. Chambers
Department of Computer Science and Mathematics
Saint Louis University
echambe5@slu.edu

Jeff Erickson
Department of Computer Science
University of Illinois, Urbana-Champaign
jeffe@cs.uiuc.edu

Amir Nayyeri
Department of Computer Science
University of Illinois, Urbana-Champaign
nayyeri2@illinois.edu

ABSTRACT

We describe the first algorithms to compute minimum cuts in surface-embedded graphs in near-linear time. Given an undirected graph embedded on an orientable surface of genus g, with two specified vertices s and t, our algorithm computes a minimum (s,t)-cut in $g^{O(g)}n\log n$ time. Except for the special case of planar graphs, for which $O(n\log n)$-time algorithms have been known for more than 20 years, the best previous time bounds for finding minimum cuts in embedded graphs follow from algorithms for general sparse graphs. A slight generalization of our minimum-cut algorithm computes a minimum-cost subgraph in every $\mathbb{Z}_2$-homology class. We also prove that finding a minimum-cost subgraph homologous to a single input cycle is NP-hard.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.2.2 [Discrete Mathematics]: Graph theory—*Graph algorithms*

General Terms: Algorithms, Performance

Keywords: computational topology, topological graph theory

Bin ölç, bir kes.
[Measure a thousand times, cut once.]
— Turkish proverb

1. INTRODUCTION

Planar graphs have been a natural focus of study for algorithms research for decades, both because they accurately model many real-world networks, and because they often admit simpler and/or more efficient algorithms for many problems than general graphs. Most planar-graph algorithms either apply immediately or have been quickly generalized to larger families of graphs, such as graphs of higher genus, graphs with forbidden minors, or graphs with small separators. Examples include minimum spanning trees [61, 54]; single-source and multiple-source shortest paths [9, 30, 42, 50, 51, 53, 69]; graph and subgraph isomorphism [36, 44, 56, 25, 26]; and approximation algorithms for the traveling salesman problem, Steiner trees, and other NP-hard problems [3, 4, 5, 22, 26].

The classical minimum cut problem and its dual, the maximum flow problem, are stark exceptions to this general pattern. Flows and cuts were introduced in the 1950s as tools for studying transportation networks, which are naturally modeled as planar graphs [39]. Ford and Fulkerson's seminal paper [31] includes an algorithm to compute maximum flows in planar networks where the source and target lie on the same face. A long series of results eventually led to planar minimum-cut algorithms that run in $O(n\log n)$ time, first for undirected graphs [62, 40, 32] and later for directed graphs [48, 42]. Strangely, however, almost nothing is known about computing flows and cuts in generalizations of planar graphs. Even for graphs embedded on the torus, the fastest known minimum-cut algorithms have no better performance than for general sparse graphs.

This paper describes the first algorithm to compute minimum cuts in surface-embedded graphs of fixed genus in near-linear time. Before describing our results in detail, we first review several related results; technical terms are defined in Section 2.

Planar minimum cuts. Recall that for any two vertices s and t in a graph G, an ***(s,t)-cut*** is a subset of the edges of G that intersects every path from s to t. A *minimum* (s,t)-cut is an (s,t)-cut of minimum size, or minimum total weight if the edges of G are weighted.

Itai and Shiloach [47] observed that the minimum (s,t)-cut in a planar graph G is dual to the minimum-cost cycle that separates faces s^* and t^* in the dual graph G^*. They also observed that this separating cycle intersects any shortest path from a vertex of s^* to a vertex of t^* exactly once. Thus, one can compute the minimum cut by cutting the dual graph G^* along a shortest path π from s^* to t^*; duplicating every vertex and edge of π; and then computing, for each vertex u of π, the shortest path between the two copies of u in the resulting planar graph. Applying Dijkstra's shortest-path algorithm at each vertex of π immediately yields a running time of $O(n^2\log n)$.

Reif [62] improved the running time of this algorithm to $O(n\log^2 n)$ using a divide-and-conquer strategy. Reif's algorithm

*Research partially supported by NSF grant DMS-0528086.

SCG'09, June 8–10, 2009, Aarhus, Denmark.

was extended by Hassin and Johnson to compute the actual maximum flow in $O(n \log n)$ additional time, using a carefully structured dual shortest-path computation [40]. Frederickson [32] subsequently improved the running time of Reif's algorithm to $O(n \log n)$ using a balanced separator decomposition to speed up the shortest-path computations. Janiga and Koubek [48] adapted Reif's $O(n \log^2 n)$-time algorithm to directed planar graphs. Henzinger *et al.* [42] generalized Frederickson's technique to obtain an $O(n)$-time planar shortest-path algorithm; using this algorithm in place of Dijkstra's algorithm improves the running times of both Reif's and Janiga and Koubek's algorithms to $O(n \log n)$. The same improvement can also be obtained using more recent multiple-source shortest path algorithms by Klein [50] and Cabello and Chambers [9].

Minimum cuts in directed planar graphs can also be obtained in $O(n \log n)$ time using the planar maximum-flow algorithms of Weihe [72] (if the graph satisfies certain connectivity restrictions) and Borradaile and Klein [2, 6, 7].

Generalizations of planar graphs. Surprisingly little is known about the complexity of computing maximum flows or minimum cuts in generalizations of planar graphs. In particular, we know of no algorithm to compute minimum cuts in non-planar graphs that does not first compute a maximum flow.

By combining a technique of Miller and Naor [57] with the planar directed flow algorithm of Borradaile and Klein [2, 6, 7], one can compute maximum (single-commodity) flows in a planar graph with k sources and sinks in $O(k^2 n \log n)$ time. A recent algorithm of Hochstein and Weihe [43] computes a maximum flow in a planar graph with k additional edges in $O(k^3 n \log n)$ time, using a clever simulation of Goldberg and Tarjan's push-relabel algorithm [35].

To our knowledge, the only prior max-flow algorithm that applies to graphs of positive genus, but not to arbitrary sparse graphs, is an algorithm of Imai and Iwano [45] that computes minimum-cost flows in graphs with small balanced separators, using a combination of nested dissection [53, 60], interior-point methods [71], and fast matrix multiplication. Their algorithm can be adapted to compute maximum flows (and therefore minimum cuts) in any graph of constant genus in time $O(n^{1.595} \log C)$, where C is the sum of the capacities. However, this is slower than more recent and more general algorithms [34].

Euler's formula implies that a simple n-vertex graph embedded on a surface of genus $O(n)$ has at most $O(n)$ edges. The fastest known combinatorial maximum-flow algorithms for sparse graphs, due to Sleator and Tarjan [67] and Goldberg and Tarjan [35], run in time $O(n^2 \log n)$. The fastest algorithm known for integer capacities, due to Goldberg and Rao [34], runs in time $O(n^{3/2} \log n \log U)$, where U is an upper bound on the edge capacities. These are also the fastest algorithms previously known for computing maximum flows or minimum cuts in graphs of any positive genus.

For further background on maximum flows, minimum cuts, and related problems, we refer the reader to monographs by Ahuja *et al.* [1] and Schrijver [65].

Short interesting cycles. Many different problems, such as finding approximate traveling salesman tours [22] and Steiner trees [3] in surface embedded graphs, embedding high-genus graphs into the plane with low distortion [46] or with few crossings [49], and feature detection and simplification in meshes [38, 24], require algorithms to find short but topologically nontrivial cycles.

Thomassen described the first efficient algorithm to find the shortest non-contractible or non-separating cycle [70, 58]. After many intermediate improvements [8, 11, 27, 52], Cabello and Chambers [9] described the fastest algorithm currently known for this problem, which runs in $O(g^3 n \log n)$ time. Splitting cycles are non-contractible and separating; finding the shortest such cycle is NP-hard, although there is an $O(n \log n)$-time algorithm for graphs of any fixed genus [12, 13]. Colin de Verdière and Erickson [18] prove that the shortest path or cycle in a given homotopy class can be computed in polynomial time, improving earlier results of Colin de Verdière and Lazarus [17, 20, 21]. Erickson and Whittlesey describe a greedy algorithm to find a minimum-length set of cycles that generate the first homology group of a surface [28].

Several practical heuristics have been developed for finding short cycles that work well in practice, although they have no theoretical guarantees. For example, Guskov and Wood [38] describe an algorithm to find and remove intersecting pairs of short non-contractible cycles ('topological noise') from surface meshes. Zomorodian and Carlsson [74] define the *localized* homology of an arbitrary topological case with respect to a subspace covering; they also describe algorithms to compute localized homology generators of simplicial complexes using persistent homology; however, they do not describe how to compute covers that would lead to cycles of minimum size. Chen and Friedman [15, 16] describe a polynomial-time algorithm to compute a cycle of minimum *radius* in a given homology class, in an arbitrary edge-weighted simplicial complex; however, the length of this cycle could be arbitrarily longer than optimal.

Dey *et al.* [23, 24] describe algorithms to find short *handle* and *tunnel* cycles in surface meshes embedded in $\mathbb{R}^3$. Any embedded surface subdivides $\mathbb{R}^3$ into an inner handlebody I and an outer handlebody O; handle cycles are null-homologous in I, while tunnel cycles are null-homologous in O. The algorithm of Dey *et al.* finds g short handle cycles and g short tunnel cycles that collectively generate the first homology group of the input surface. Their algorithm uses several heuristics to reduce the length of the output cycles, in part because no algorithm is known to find the shortest cycle in a given homology class.

New Results. The input to our algorithm is an undirected edge-weighted graph G embedded on an orientable surface of genus g. Given vertices s and t, our algorithm computes a minimum-weight (s,t)-cut in $g^{O(g)} n \log n$ time. For any fixed positive genus, this improves the best previous time bound by a factor of $\min\{n, \sqrt{n} \log C\}$.

Our minimum-cut algorithm is a special case of a more general algorithm to compute a minimum-cost *subgraph* in a given $\mathbb{Z}_2$-homology class. Even when the homology class is specified by a simple cycle, the output representative may be the union of several cycles; see Figure 2. For surfaces with genus g and b boundary components, our algorithm runs in $(g+b)^{O(g+b)} n \log n$ time. We also show that this more general problem is strongly NP-hard, even if the input homology class is specified by a simple cycle; thus, the exponential dependence on the topology of the surface is unavoidable unless P=NP. We are not aware of any previous algorithmic results for our more general problem.

Of course, the original minimum-cut problem is not NP-hard! We conjecture that minimum cuts can be computed in time $O(g^c n \log n)$ for some small constant c.

In a companion paper [14], we describe algorithms to compute a maximum flow in surface embedded graphs, using very different techniques from this paper. Specifically, given an undirected

graph embedded on an orientable surface of genus g, with two specified vertices s and t, we can compute a maximum (s,t)-flow in $O(g^7 n \log^2 n \log^2 C)$ time for integer capacities that sum to C, or in $(g \log n)^{O(g)} n$ time for real capacities. Our key insight is that it suffices to optimize the relative homology class of the flow, rather than directly optimizing the flow itself.

2. DRAMATIS PERSONAE

We begin by recalling several useful definitions related to graphs embedded on surfaces. For more comprehensive treatments, we refer the interested reader to Gross and Tucker [37] and Mohar and Thomassen [58] for topological graph theory, and to Hatcher [41] and Stillwell [68] for topology.

2.1 Surfaces and Curves

A ***surface*** (more formally, a *2-manifold*) Σ is a compact topological space in which every point has an open neighborhood homeomorphic to either the plane $\mathbb{R}^2$ or a closed halfplane $\{(x,y) \in \mathbb{R}^2 \mid x \geq 0\}$. The points with halfplane neighborhoods comprise the ***boundary*** of the surface; every connected component of the boundary is homeomorphic to a circle. A surface is *non-orientable* if it contains a subset homeomorphic to the Möbius band, and *orientable* otherwise.

A ***path*** in a surface Σ is a continuous function $p\colon [0,1] \to \Sigma$. A ***loop*** is a path whose endpoints $p(0)$ and $p(1)$ coincide; we refer to this common endpoint as the ***basepoint*** of the loop. An ***arc*** is a path whose endpoints lie on the boundary of Σ. A ***cycle*** is a continuous function $\gamma\colon S^1 \to \Sigma$. We refer to paths, loops, arcs, and cycles as ***curves***. We will usually not distinguish between a path/cycle and its image in Σ. A curve is ***simple*** if the function that defines it is injective, except for the basepoint in the case of loops. The ***reversal*** $\overline{p}$ of a path p is defined by setting $\overline{p}(t) = p(1-t)$. The ***concatenation*** $p \cdot q$ of two paths p and q with $p(1) = q(0)$ is the path created by setting $(p \cdot q)(t) = p(2t)$ for all $t \leq 1/2$ and $(p \cdot q)(t) = q(2t-1)$ for all $t \geq 1/2$.

The ***genus*** of a surface Σ is the maximum number of simple, disjoint, non-separating cycles $\gamma_1, \gamma_2, \ldots, \gamma_g$; that is, $\gamma_i \cap \gamma_j = \emptyset$ for all i and j, and the surface $\Sigma \setminus (\gamma_1 \cup \cdots \cup \gamma_g)$ is connected. This paper will consider only compact, connected, orientable surfaces. Up to homeomorphism, there is exactly one such surface with any non-negative genus g and any non-negative number of boundaries b.

2.2 Graph Embeddings

An ***embedding*** of an undirected graph G on a surface Σ consists of a mapping of vertices of G to distinct points in Σ and a collection of mappings from edges of G to simple paths in Σ that intersect only at common endpoints. A ***face*** of an embedding is a maximal connected subset of Σ that does not intersect the image of any edge or vertex. An embedding is ***cellular*** (or *2-cell* [58]) if every face is an open topological disk; in particular, in any cellular embedding, the boundary of Σ is covered by edges of G.

Suppose G is a simple n-vertex graph cellularly embedded on an orientable surface of genus g with b boundaries. Euler's formula $|V| - |E| + |F| = 2 - 2g - b$ implies that G has at most $3n - 6 + 6g + 3b$ edges and at most $2n - 4 + 4g + 2b$ faces, with equality if every face of the embedding and every boundary cycle is a triangle. Thus, the overall complexity of an embedding of an n-vertex graph is $O(n + g + b) = O(n)$.

Cellular graph embeddings are equivalent to the ***combinatorial surfaces*** introduced by Colin de Verdière [17] and used by several other authors to formulate optimization problems for surface-embedded graphs [9, 10, 11, 13, 18, 19, 20, 21, 27, 28, 29, 52]. A combinatorial surface $S = (\Sigma, G)$ consists of an abstract surface Σ together with a cellularly embedded graph G with positively weighted edges. Paths and cycles are required to be walks in the graph; the length of an curve is the sum of its edge weights, counted with appropriate multiplicity.

Two paths in a combinatorial surface ***cross*** if no continuous infinitesimal perturbation makes them disjoint; if such a perturbation exists, then the paths are ***non-crossing***. We say that a cycle γ is ***non-self-crossing*** if no two sub-paths of γ cross, ***weakly simple*** if γ is non-self-crossing and traverses each edge at most once, and ***(strictly) simple*** if γ visits each vertex at most once.

Cutting a combinatorial surface along a cycle or an arc modifies both the surface and the embedded graph. For any combinatorial surface $S = (\Sigma, G)$ and any cycle or arc γ in G, we define a new combinatorial surface $S \setminus \gamma$ by taking the topological closure of $\Sigma \setminus \gamma$ as the new underlying surface; the new embedded graph contains two copies of each vertex and edge of γ, each bordering a new boundary.

2.3 Homotopy and Homology

Homotopy is an equivalence relation between curves that captures the notion of continuous deformation. Two paths p and p' are ***homotopic*** if there is a continuous map $h\colon [0,1] \times [0,1] \to \Sigma$ such that $h(0,t) = p(t)$ and $h(1,t) = p'(t)$ for all t, and $h(\cdot,0)$ and $h(\cdot,1)$ are constant maps. Two cycles γ and γ' are ***(freely) homotopic*** if there is a continuous map $h\colon [0,1] \times S^1 \to \Sigma$ such that $h(0,t) = \gamma(t)$ and $h(1,t) = \gamma'(t)$ for all t. A loop or cycle is ***contractible*** if it is homotopic to a constant map; an arc is contractible if it is homotopic to a subpath of a boundary cycle. A simple cycle γ is ***separating*** if $\Sigma \setminus \gamma$ is disconnected.

Homology is a coarser equivalence relation than homotopy, with nicer algebraic properties. Like several earlier papers [15, 16, 23, 24], we will consider only one-dimensional cellular homology with coefficients in the finite field $\mathbb{Z}_2$; this restriction allows us to radically simplify our definitions. For a more general treatment of homology, we refer the reader to our companion paper [14] or to standard references on topology [41, 68, 73].

Fix a cellular embedding of an undirected graph G on a surface with genus g and b boundaries. An ***even subgraph*** is a subgraph of G in which every node has even degree, or equivalently, the union of edge-disjoint cycles. A ***boundary subgraph*** is the boundary of the union of a subset of faces of G; for example, every separating cycle is a boundary subgraph. Two even subgraphs are ***homologous***, or in the same ***homology class***, if their symmetric difference is a boundary subgraph. Boundary subgraphs are also called ***null-homologous***. Any two homotopic cycles are homologous, but homologous cycles are not necessarily homotopic; see Figure 1. Moreover, the homology class of a cycle can contain even subgraphs that are not cycles; see Figure 2.

The subgraphs of G define a vector space isomorphic to $\mathbb{Z}_2^{|E|}$, whose addition operation is symmetric difference, denoted $\oplus$. Even subgraphs, boundary subgraphs, and homology classes all define subspaces of this vector space as follows.

The ***cycle space*** $Z(G)$ is the vector space of all even subgraphs, which is (redundantly) generated by all cycles in G. Every even subgraph satisfies $|V|$ linear constraints, one at each vertex; however, exactly one of these constraints is redundant. Thus, the cycle space is isomorphic to $\mathbb{Z}_2^{|E|-|V|+1}$.

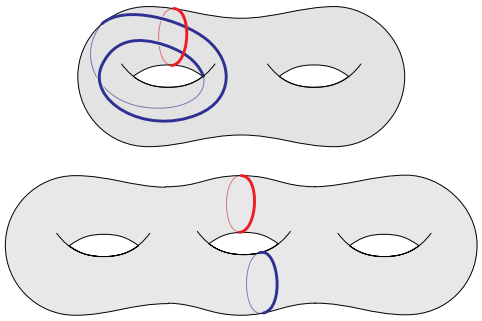

Figure 1. Homologous pairs of cycles that are not homotopic. (Lighter portions of the curves are on the back side of the surface.)

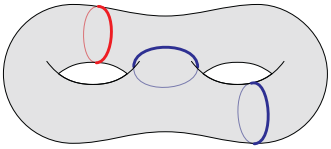

Figure 2. Each cycle is homologous to the union of the other two.

The ***boundary space*** $B(G)$ is the vector space of all boundary subgraphs, which is (redundantly) generated by the boundaries of faces of the embedding of G. Any boundary subgraph is specified by a subset of the faces; a subset and its complement define the same boundary subgraph if and only if the surface has no boundary. Thus $B(G)$ is isomorphic to $\mathbb{Z}_2^{|F|-1}$ if $b = 0$, and $\mathbb{Z}_2^{|F|}$ if $b \geq 1$. Because every boundary subgraph is even, $B(G)$ is a linear subspace of $Z(G)$.

Finally, the ***homology space*** $H(G)$ is the vector space of all homology classes of even subgraphs, which is isomorphic to $Z(G)/B(G)$. Euler's formula implies that $H(G)$ has dimension $(|E| - |V| + 1) - (|F| - 1) = 2g$ if $b = 0$, or $(|E| - |V| + 1) - |F| = 2g + b - 1$ if $b > 0$. In particular, any two graphs embedded on the same surface define isomorphic homology spaces, and the number of different homology classes is exactly $2^{2g+\max\{0,b-1\}}$. The dimension $\beta = 2g + \max\{0, b-1\}$ of $H(G)$ is called the *first Betti number* of the surface.

2.4 Dual Graphs, Cocycles, and Cuts

For any graph G on a surface without boundary,[1] we can define a canonical ***dual graph*** G^*. The vertices of G^* correspond to the faces of G, and two vertices in G^* are are joined by a (dual) edge if and only if the corresponding faces of G are separated by an edge of G. Thus, every edge e in G has a corresponding dual edge in G^*, denoted e^*. For any face f of G, we let f^* denote the corresponding vertex of G^*. The dual graph G^* has a cellular embedding on Σ, whose faces correspond exactly to the vertices of G. For any vertex v of G, we let v^* denote the corresponding face of G^*. Duality is an involution—the dual of G^* is isomorphic to the original graph G.

When the graph G is fixed, we abuse notation by writing H^* to denote the subgraph of G^* containing the edges dual to the edges of a subgraph H of G. If a subgraph H is a cycle, we call its dual H^* a ***cocycle***; if a subgraph H is a boundary subgraph, its dual H^* is a ***cut***. In planar graphs, every cocycle is a cut, and every minimal cut is a cocycle; however, these equivalences do not extend to higher-genus graphs.

[1] Graph duality can be generalized to surfaces with boundary [18, 29], but this paper will not require such a generalization.

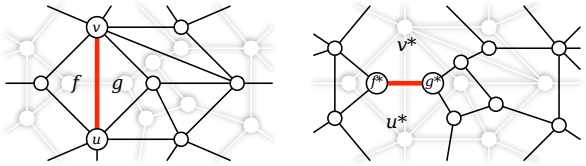

Figure 3. Graph duality. One edge uv and its dual $(uv)^* = f^*g^*$ are emphasized.

3. MINIMUM HOMOLOGOUS SUBGRAPHS

Let G be an n-vertex undirected graph, embedded on an orientable surface with genus g and b boundary components, and let H be an even subgraph of G. In this section, we describe an algorithm to compute the minimum-cost even subgraph homologous with H in $(g+b)^{O(g+b)} n \log n$ time. In fact, our algorithm can be modified easily to compute a minimum-cost representative in *every* homology class in the same asymptotic running time; there are exactly $2^{2g+\max\{b-1,0\}}$ such classes.

Our algorithm closely resembles the algorithm of Chambers *et al.* [13] for computing a shortest splitting cycle; in fact, our algorithm is somewhat simpler. The first stage of our algorithm cuts the underlying combinatorial surface into a topological disk by a network of shortest paths. Next, we enumerate all possible ways for an even subgraph to intersect each shortest path in the decomposition network $O(g+b)$ times. We quickly discard any crossing pattern that does not correspond to an even subgraph in the desired homology class. Each crossing pattern is realized by several *homotopy* classes of sets of non-crossing cycles, which we easily enumerate. Within each homotopy class, we find a minimum-length set of non-crossing cycles with each crossing pattern using an algorithm of Kutz [52]. The union of those cycles is an even subgraph in the desired homology class; we return the lightest such subgraph as our output.

To simplify the presentation of our results, we assume that there is a unique shortest path $\sigma(u, v)$ between any pair of vertices u and v in the input graph G. If necessary, this assumption can be enforced (at least with high probability) using standard perturbation techniques [59].

3.1 Minimum Cuts

Before we describe our algorithm, we first show that the minimum-weight homologous subgraph problem includes (the combinatorial dual of) the classical minimum-cut problem as a special case.

Lemma 3.1. *Let $G = (V, E)$ be an edge-weighted graph embedded on a surface Σ without boundary, and let s and t be vertices of G. Finally, let X be the minimum-weight (s,t)-cut in G. Then X^* is the minimum-weight even subgraph of G^* homologous with the boundary of s^* in the surface $\Sigma \setminus (s^* \cup t^*)$.*

Proof: Let ∂s^* denote the boundary of s^*, and let Σ' denote the surface $\Sigma \setminus (s^* \cup t^*)$.

Let X be an arbitrary (s,t)-cut in G. This cut partitions the vertices of G into two disjoint subsets, S and T, respectively containing vertices s and t. Thus, the dual subgraph X^* partitions the faces of G^* into two disjoint subsets, S^* and T^*, respectively containing faces s^* and t^*. In particular, X^* is the boundary of the union of the faces in S^*, which implies that X^* is null-homologous in Σ. The subgraph $X^* \oplus \partial s^*$ is the boundary of

the union of $S^* \setminus \{s^*\}$, which is a subset of the faces of Σ'. Thus, $X^* \oplus \partial s^*$ is null-homologous in Σ'. We conclude that X^* and ∂s^* are homologous in Σ'.

Conversely, let X^* be an arbitrary even subgraph of G^* homologous to ∂s^* in Σ'. The subgraph $X^* \oplus \partial s^*$ is null-homologous in Σ'. This immediately implies that X^* is null-homologous in Σ; moreover, faces s^* and t^* are on opposite sides of X^*. any path from s to t in the original graph G must traverse at least one edge of X. We conclude that X is an (s,t)-cut. □

3.2 Crossing Bound

Our main technical lemma establishes an upper bound on the number of crossings between an arbitrary shortest path and the minimum-weight even subgraph in any homology class. Crossing-number arguments were first used by Cabello and Mohar [11] to develop the first subquadratic algorithms for shortest non-contractible and non-separating cycles; their arguments are the foundation of all later improvements of their algorithm [8, 52, 9]. Our proof is quite similar to the argument of Chambers *et al.* [13] that the shortest *splitting* cycle crosses any shortest path $O(g+b)$ times. However, our new proof is simpler, because the structure we seek is a true subgraph, which need not be connected, rather than a single (weakly) simple closed walk.

Stating our crossing bound is rather subtle, because we cannot consistently define when a shortest path crosses an even subgraph. Instead, we partition the even subgraph into a well-behaved collection of cycles, and then consider the total number of crossings between a shortest path and the cycles in that collection. Specifically, we define a ***cycle decomposition*** of an even subgraph H to be a set of edge-disjoint, non-crossing, weakly simple cycles whose union is H.

Lemma 3.2. *Every even subgraph of an embedded graph has a cycle decomposition.*

Proof: Let H be an even subgraph of G. We can decompose H into cycles by specifying, at each vertex v, which pairs of incident edges of H are consecutive. Any pairing that does not create a crossing at v is sufficient. For example, if $e_1, e_2, \ldots, e_{2d}$ are the edges of H incident to v, indexed in clockwise order around v, we could pair edges e_{2i-1} and e_{2i} for each i. □

We emphasize that each cycle in a cycle decomposition may visit vertices multiple times; indeed, some even subgraphs cannot be decomposed into strictly simple cycles.

Lemma 3.3. *Let G be an edge-weighted graph embedded on a surface with genus g and b boundary components. Let H be a subgraph of G of minimum weight in its $\mathbb{Z}_2$-homology class, and let $\gamma_1, \gamma_2, \ldots, \gamma_r$ be a cycle decomposition of H. The total number of crossings between any shortest path in G and the cycles $\gamma_1, \gamma_2, \ldots, \gamma_r$ is at most $6g + 2b - 3$.*

Proof: Let $\sigma(y,z)$ denote the shortest path between any two vertices y and z, and let $\sigma = \sigma(u,v)$ for some vertices u and v. Uniqueness of shortest paths implies that if y and z are vertices of σ, either the shortest path $\sigma(y,z)$ or its reversal $\sigma(z,y)$ is a subpath of σ. Without loss of generality, we can assume that σ crosses each cycle γ_i at least once. For each i, let x_i denote the number of times σ and γ_i cross, and let $x = x_1 + x_2 + \cdots + x_r$. We need to prove that $x \le 6g + 2b - 3$.

Consider the graph G/σ obtained from G by contracting σ to a single vertex uv. This graph inherits a cellular embedding on Σ from the cellular embedding of G. Each cycle γ_i is contracted to the union of x_i simple non-crossing loops in G/σ with basepoint uv. Altogether, we obtain x loops, which we denote $\ell_1, \ell_2, \ldots, \ell_x$.

Suppose some loop ℓ_i is contractible. This loop is the contraction of a path π_i in G whose endpoints u_i and v_i lie in σ. The cycle $\delta = \pi_i \cdot \sigma(v_i, u_i)$ is also contractible. Thus, the even subgraph $H \oplus \delta$ is homologous with H. Moreover, the uniqueness of shortest paths implies that the weight of $H \oplus \delta = H \cup \sigma(v_i, u_i) \setminus \pi_i$ is smaller than the weight of H. But this contradicts our assumption that H has minimum weight in its homology class.

Now suppose some pair of loops ℓ_i and ℓ_j are homotopic; by definition, the cycle $\ell_i \cdot \overline{\ell_j}$ is contractible. These two loops are contractions of paths π_i and π_j in G with endpoints in σ. Let u_i and v_i denote the endpoints of π_i, and let u_j and v_j denote the endpoints of π_j. The cycle $\pi_i \cdot \sigma(v_i, v_j) \cdot \overline{\pi_j} \cdot \sigma(u_j, u_i)$ in G is also contractible. Let δ denote the set of edges of G that appear in this cycle exactly once. If the sub-paths $\sigma(v_i, v_j)$ and $\sigma(u_j, u_i)$ are edge-disjoint, then δ is a contractible cycle; otherwise, δ is the union of two non-crossing homotopic cycles. In either case, δ is a boundary subgraph, so the symmetric difference $H \oplus \delta$ is homologous with H. Moreover, $H \oplus \delta$ has smaller weight than H, and we obtain another contradiction.

We conclude that the loops $\ell_1, \ell_2, \ldots, \ell_x$ lie in distinct nontrivial homotopy classes. Thus, these loops define an embedding of a single-vertex graph with x edges onto Σ, where no face of the embedding is a disk bounded by less than three edges. Euler's formula now implies that $x \le 6g + 2b - 3$ [13, Lemma 2.1]. □

We emphasize that different cycle decompositions of the same even subgraph may lead to different numbers of crossings. Our crossing bound applies to *any* cycle decomposition.

3.3 Systems of Paths and Crossing Vectors

We are now ready to describe our algorithm. The input consists of an edge-weighted graph G, embedded on a surface Σ with genus g and b boundary components, along with an even subgraph H of G. Our algorithm computes the minimum-weight even subgraph of G that is homologous with H.

Our algorithm begins by computing a set P of paths, each of which is the concatenation of two shortest paths (possibly meeting in the interior of an edge), such that the surface $\Sigma \setminus P$ is a topological disk. If the Σ has no boundary, the paths in P are non-contractible loops with an arbitrary common basepoint; Euler's formula implies that the number of loops is $2g$. Specifically, we construct a *greedy system of loops* in $O(n \log n + gn)$ time, using an algorithm of Erickson and Whittlesey [28]. If the surface has boundary, the paths in P are non-contractible arcs; Euler's formula implies that the number of arcs is $2g + b - 1$. Specifically, we again construct a *greedy system of arcs* in $O(n \log n + (b+g)n)$ time, using a modification of Erickson and Whittlesey's algorithm described by Chambers *et al.* [13].

Let $p_1, p_2, \ldots, p_\beta$ denote the paths in P, where $\beta = 2g + \max\{0, b-1\}$. It is no coincidence that the number of paths in P is equal to the dimension of the homology group $H(G)$. Indeed, we can identify the homology class of any even subgraph by considering the number of times it crosses each path in P, as follows.

For any cycle γ and any index i, let $x_i(\gamma)$ denote the number of times γ crosses the path p_i. The *crossing vector* $x(\gamma)$ is the vector $(x_1(\gamma), \ldots, x_\beta(\gamma))$. The crossing vector of a set of cycles is the sum of the crossing vectors of its elements. Crossing vectors are not well-defined for arbitrary even subgraphs; different cycle

decompositions can yield different crossing numbers. However, the *parity* of the crossing numbers is independent of the cycle decomposition. The *crossing parity vector* of any even subgraph H is the bit vector $\bar{x}(H) = (\bar{x}_1, \ldots, \bar{x}_\beta)$, where $\bar{x}_i = 1$ if the path p_i crosses (any cycle decomposition of) H an odd number of times, and $x_i = 0$ otherwise.

Lemma 3.4. *Two even subgraphs are homologous if and only if their crossing parity vectors are equal.*

Proof: Every boundary subgraph is the symmetric difference of facial cycles. Any non-contractible loop or arc crosses facial cycle an even number of times; thus, the crossing parity vector of any facial cycle is the zero vector. Every pair of even subgraphs H and H' satisfies the identity $x(H \oplus H') = x(H) \oplus x(H')$. Thus, the crossing parity vector of any boundary subgraph is the zero vector. □

Lemma 3.5. *We can compute the crossing parity vector of any even subgraph in $O(gn)$ time.*

Proof: We can compute a cycle decomposition $\gamma_1, \ldots, \gamma_r$ of H in $O(n)$ time, by following the proof of Lemma 3.3. We can compute the number of crossings between any cycle γ_i and any path p_j in time proportional to the number of edges in γ_i. Thus, we can compute each bit $\bar{x}_j(H)$ in $O(n)$ time. □

We call an integer vector $(x_1, \ldots, x_\beta)$ a *valid crossing vector* if $x_i \le 12g + 4b - 6$ for all i and $(x_1 \bmod 2, \ldots, x_\beta \bmod 2) = \bar{x}(H)$. Our algorithm enumerates all $(g+b)^{O(g+b)}$ valid crossing vectors by brute force in $(g+b)^{O(g+b)}$ time. Then for each valid crossing vector, our algorithm computes the minimum-weight collection of cycles with that crossing vector, as we describe next.

3.4 Triangulations and Crossing Sequences

We can cut our combinatorial surface $\Sigma \setminus P$ into a 2β-gon, or *abstract polygonal schema*, by cutting along each path in P and replacing each copy of each path in the cut surface with a single edge. Thus, each path in P corresponds to two edges of this polygon. The vertices correspond to copies of the common basepoint of P if Σ has no boundary, or to boundary paths between endpoints of P otherwise.

We dualize the abstract polygonal schema by replacing each edge with a vertex, and connecting vertices which correspond to adjacent edges in the primal schema. Any collection of non-crossing, non-self-crossing cycles corresponds to a *weighted triangulation* [13], where we draw an edge between two vertices of the dual abstract polygonal schema if and only if some cycle consecutively crosses the corresponding pair of paths in the greedy system of loops. Each edge is weighted by the number of times such a crossing occurs in our collection. Conversely, a weighted triangulation corresponds to a collection of non-crossing, non-self-crossing cycles as long as corresponding vertices are incident to edges of equal total weight. Lemma 3.3 implies that we only need to consider weights between 0 and $O(g+b)$. Thus, there are $(g+b)^{O(g+b)}$ different weighted triangulations for each valid crossing vector.

For each valid weighted triangulation, we can compute a corresponding collection of abstract cycles in $O((g+b)^2)$ time by brute force. In the same time, we can also compute the *sequence* of crossings of each abstract cycle with the paths in P. An algorithm of Kutz [52] computes the shortest cycle in G with a given crossing sequence of length x in $O(xn \log n)$ time, by gluing together x copies of the planar surface $\Sigma \setminus P$ into an annulus and calling Frederickson's planar minimum-cut algorithm [32]. Thus, for each weighted triangulation, we obtain the shortest corresponding set of cycles in $O((g+b)^2 n \log n)$ time.

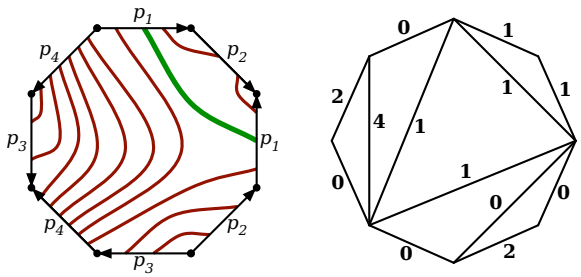

Figure 4. Two disjoint simple cycles on a surface of genus 2, and the corresponding weighted triangulation.

Theorem 3.6. *Let G be a graph with positively weighted edges embedded on a surface with genus g and b boundary components, and let H be an even subgraph of G. We can compute the minimum-cost even subgraph homologous with H in $(g+b)^{O(g+b)} n \log n$ time.*

Corollary 3.7. *Let G be an edge-weighted graph embedded on a surface with genus g and b boundary components, and let s and t be vertices of G. We can compute the minimum-weight (s,t)-cut in G in $g^{O(g)} n \log n$ time.*

4. NP-HARDNESS

In this section, we show that finding the minimum-weight even subgraph in a given homology class is NP-hard, even when the underlying surface has no boundary.

Chen and Freedman [16, 15] proved a similar hardness result (by reduction from a special case of MAX2SAT) for general simplicial complexes; however, the complexes output by their reduction are never manifolds. Chambers *et al.* [13] prove that finding the shortest *splitting* cycle is NP-hard; a cycle is splitting if it is non-self-crossing, non-contractible, and null-homologous. A simple modification of their reduction (from Hamiltonian cycle in planar grid graphs) implies that finding the shortest *strictly simple cycle* in a given homology class is NP-hard. Our proof closely follows a reduction of McCormick *et al.* [55] from MIN2SAT to a special case of MAXCUT.

Theorem 4.1. *Computing the minimum-cost even subgraph in a given homology class on a surface without boundary is equivalent to computing a minimum-capacity cut in an embedded edge-weighted graph G whose negative-cost edges are dual to an even subgraph in G^*.*

Proof: Fix a graph G embedded on a surface Σ without boundary, together with a cost function $c : E \to \mathbb{R}$. For any even subgraph H of G, let $c(H) = \sum_{e \in H} c(e)$, and let MINHOM$(H, c)$ denote the even subgraph of minimum cost in the homology class of H.

Consider the *residual cost* function $c_H : E \to \mathbb{R}$ defined by setting $c_H(e) = c(e)$ for each edge $e \notin H$, and $c_H(e) = -c(e)$ for each edge $e \in H$. For any subgraph H' of G, we have $c(H') = c_H(H \oplus H') + c(H)$, which immediately implies that MINHOM$(H, c) = H \oplus$ MINHOM$(\emptyset, c_H)$.

Every null-homologous even subgraph of G is dual to a cut in the dual graph G^*. Thus, we have reduced our problem to computing the minimum cut in G^* with respect to the cost function c_H. Since the empty set is a valid cut with zero cost, the cost of the minimum cut is never positive. In particular, H is the minimum-cost even subgraph in its homology class if and only if the cut in G^* with minimum residual cost is empty.

In fact, our reduction is reversible. Suppose we want to find the minimum cut in an embedded graph $G = (V, E)$ with respect to the cost function $c : E \to \mathbb{R}$, where every face of G is incident to an even number of edges with negative cost. Let $H = \{e \in E \mid c(e) < 0\}$ be the subgraph of negative-cost edges, and let X denote the (possibly empty) set of edges in the minimum cut of G. Consider the *absolute cost* function $|c| : E^* \to \mathbb{R}$ defined as $|c|(e^*) = |c(e)|$. Then $(H \oplus X)^*$ is the even subgraph of G^* of minimum absolute cost that is homologous to H^*. □

We now prove that this special case of the minimum cut problem is NP-hard, by reduction from MinCut in graphs with negative edges. This problem includes MaxCut as a special case (when every edge has negative cost), but many other special cases are also NP-hard [55]. The output of our reduction is a simple triangulation; the reduction can be simplified if graphs with loops and parallel edges are allowed.

Suppose we are given an *arbitrary* graph $G = (V, E)$ with n vertices and an *arbitrary* cost function $c : E \to \mathbb{R}$. We begin by computing a cellular embedding of G on some surface. If we don't care whether the surface is orientable, we can simply impose a cyclic order on the edges incident to each vertex. The maximum-genus *orientable* cellular embedding can be computed in polynomial time [33]. Alternately, we can add zero-length edges to make the graph complete and then use classical results of Ringel, Youngs, and others [64, 63] to compute a minimum-genus orientable embedding of K_n in polynomial time. Once we have an embedding, we add vertices and zero-cost edges to obtain a triangulation.

Let C be the sum of the absolute values of the edge costs: $C := \sum_e |c(e)|$. We locally modify both the surface and the embedding to transform each negative-weight edge into a cocycle, as follows. We transform the edges one at a time; after each iteration, the embedding is a simple triangulation. (Our reduction can be simplified if a simple graph is not required.) For each edge uv with $c(uv) < 0$, remove uv to create a quadrilateral face. Triangulate this face as shown in Figure 5; we call the new faces uu_1u_2 and vv_1v_2 *endpoint triangles*. Assign cost C to the edges of the endpoint triangles and cost zero to the other new edges. Glue a new handle to the endpoint triangles, and triangulate the handle with a cycle of six edges, each with cost $c(uv)/6$. These six edges form a cocycle of cost $c(uv)$, which we call an *edge cocycle*, in the new embedding. Each iteration adds 5 vertices and 21 edges to the graph and increases the genus of the underlying surface by 1.

Let G' denote the transformed graph and $c' : E(G') \to \mathbb{R}$ its associated cost function. The minimum cut in G' cannot contain any edge of an endpoint triangle. Thus, for each edge cocycle, either all six edges cross the cut, or none of them cross the cut. It follows that the minimum cut in G' corresponds to a cut with equal cost in the original graph G. Conversely, any cut in G can be transformed into a cut in G' of equal cost. Thus, computing the minimum cut in G' is equivalent to computing the minimum cut in G.

Theorem 4.2. *Given an even subgraph H of an edge-weighted graph G embedded on a surface without boundary, computing the minimum-weight even subgraph homologous to H is strongly NP-hard.*

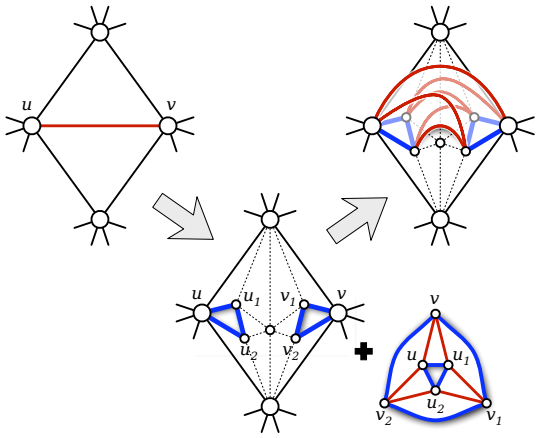

Figure 5. Adding a handle to transform a negative edge into a negative cocycle. Thick (blue) edges have cost C; dashed edges have cost zero.

Our reduction can be modified further to impose other desirable properties on the output instances, for example, that the graph is unweighted, every vertex has degree 3, or the input subgraph H is a simple cycle. We leave the details as easy exercises for the reader.

Finally, we emphasize that the NP-hardness of this problem relies crucially on the fact that we are using homology with coefficients taken from the finite field $\mathbb{Z}_2$. The corresponding problem for homology with real or integer coefficients is a minimum-cost circulation problem, and thus can be solved in polynomial time. In our companion paper [14], we show that this circulation problem can be solved in near-linear time for graphs of constant genus, using very different techniques.

5. CONCLUSIONS AND OPEN PROBLEMS

Many potential improvements and open questions remain. For example, unlike the flow algorithms described in our companion paper [14], our minimum-cut algorithm does not seem to extend easily to embedded *directed* graphs. Computing minimum cuts in directed graphs embedded on a surface in near-linear time remains an open question.

We conjecture that both maximum flows and minimum cuts in embedded graphs can be computed in $O(g^k n \log n)$ time for some small constant k, perhaps using a generalization of the network simplex algorithm of Borradaile and Klein [2, 6, 7]. Even the special case of unit capacities is open.

It seems likely that our minimum-cut algorithm will extend to graphs embedded on non-orientable surfaces; the main challenge is extending Lemma 3.3. (In contrast, our techniques for computing maximum flows will not extend to non-orientable surfaces without substantial modification, in part because our definition of duality for directed graphs requires a consistent surface orientation.) No near-linear-time algorithms are known in this setting, even for graphs embedded on the projective plane.

Acknowledgments. The authors would like to thank Chandra Chekuri and Aparna Sundar for helpful discussions, and the anonymous reviewers for their comments and suggestions.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] G. Borradaile. *Exploiting Planarity for Network Flow and Connectivity Problems*. Ph.D. thesis, Brown University, May 2008. ⟨http://www.cs.brown.edu/research/pubs/theses/phd/2008/glencora.pdf⟩.

[3] G. Borradaile, E. D. Demaine, and S. Tazari. Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. *Proc. 26th Int. Symp. Theoretical Aspects Comput. Sci.*, 171–182, 2009. Dagstuhl Seminar Proceedings. ⟨http://drops.dagstuhl.de/opus/volltexte/2009/1835/⟩.

[4] G. Borradaile, C. Kenyon-Mathieu, and P. N. Klein. A polynomial-time approximation scheme for Steiner tree in planar graphs. *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms*, 1285–1294, 2007.

[5] G. Borradaile, C. Kenyon-Mathieu, and P. N. Klein. Steiner tree in planar graphs: An $O(n \log n)$ approximation scheme with singly-exponential dependence on epsilon. *Proc. 10th Ann. Workshop on Algorithms and Data Structures*, 275–286, 2007.

[6] G. Borradaile and P. Klein. An $O(n \log n)$-time algorithm for maximum st-flow in a directed planar graph. *Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithms*, 524–533, 2006.

[7] G. Borradaile and P. Klein. An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *J. ACM*, to appear, 2009. ⟨http://www.math.uwaterloo.ca/~glencora/downloads/maxflow-full.pdf⟩.

[8] S. Cabello. Many distances in planar graphs. *Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithms*, 1213–1220, 2006.

[9] S. Cabello and E. W. Chambers. Multiple source shortest paths in a genus g graph. *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms*, 89–97, 2007.

[10] S. Cabello, M. DeVos, J. Erickson, and B. Mohar. Finding one tight cycle. *Proc. 19th Ann. ACM-SIAM Symp. Discrete Algorithms*, 527–531, 2008.

[11] S. Cabello and B. Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discrete Comput. Geom.* 37:213–235, 2007.

[12] E. W. Chambers, É. Colin de Verdière, J. Erickson, F. Lazarus, and K. Whittlesey. Splitting (complicated) surfaces is hard. *Proc. 22nd Ann. ACM Symp. Comput. Geom.*, 421–429, 2006.

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

[14] E. W. Chambers, J. Erickson, and A. Nayyeri. Homology flows, cohomology cuts. *Proc. 41st Ann. ACM Symp. Theory Comput.*, 2009.

[15] C. Chen and D. Freedman. Quantifying homology classes II: Localization and stability. Preprint, 2007. ⟨http://arxiv.org/abs/0709.2512v2⟩.

[16] C. Chen and D. Freedman. Quantifying homology classes. *Proc. 25th Ann. Symp. Theoretical Aspects Comp. Sci.*, 169–180, 2008. Dagstuhl Seminar Proceedings. ⟨http://drops.dagstuhl.de/opus/volltexte/2008/1343/⟩.

[17] É. Colin de Verdière. *Raccourcissement de courbes et décomposition de surfaces [Shortening of Curves and Decomposition of Surfaces]*. Ph.D. thesis, University of Paris 7, Dec. 2003. ⟨http://www.di.ens.fr/~colin/textes/these.html⟩.

[18] É. Colin de Verdière and J. Erickson. Tightening non-simple paths and cycles on surfaces. *Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithms*, 192–201, 2006.

[19] É. Colin de Verdière and F. Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *Proc. 11th Sympos. Graph Drawing*, 478–490, 2003. Lecture Notes Comput. Sci. 2912.

[20] É. Colin de Verdière and F. Lazarus. Optimal system of loops on an orientable surface. *Discrete Comput. Geom.* 33(3):507–534, 2005.

[21] É. Colin de Verdière and F. Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *J. ACM* 54(4), 2007.

[22] E. D. Demaine, M. Hajiaghayi, and B. Mohar. Approximation algorithms via contraction decomposition. *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms ACM-SIAM symposium on Discrete algorithms*, 278–287, 2007.

[23] T. K. Dey, K. Li, and J. Sun. On computing handle and tunnel loops. *IEEE Proc. Int. Conf. Cyberworlds*, 357–366, 2007.

[24] T. K. Dey, K. Li, J. Sun, and D. Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3D models. *ACM Trans. Graphics* 27(3):1–9, 2008. Proc. SIGGRAPH 2008.

[25] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms and Applications* 3(3):1–27, 1999.

[26] D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica* 27:275–291, 2000.

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

[28] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. *Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms*, 1038–1046, 2005.

[29] J. Erickson and P. Worah. Computing the shortest essential cycle. Preprint, November 2008. ⟨http://www.cs.uiuc.edu/~jeffe/pubs/essential.html⟩.

[30] 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.

[31] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian J. Math.* 8(399–404), 1956.

[32] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs with applications. *SIAM J. Comput.* 16(6):1004–1004, 1987.

[33] M. L. Furst, J. L. Gross, and L. A. McGeoch. Finding a maximum-genus graph imbedding. *J. Assoc. Comput. Mach.* 35(3):523–534, 1988.

[34] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM* 45(5):783–797, 1998.

[35] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. Assoc. Comput. Mach.* 35(4):921–940, 1988.

[36] M. Grohe. Isomorphism testing for embeddable graphs through definability. *Proc. 32nd ACM Symp. Theory Comput.*, 63–72, 2000.

[37] J. L. Gross and T. W. Tucker. *Topological graph theory*. Dover Publications, 2001.

[38] I. Guskov and Z. Wood. Topological noise removal. *Proc. Graphics Interface*, 19–26, 2001.

[39] T. E. Harris and F. S. Ross. Fundamentals of a method for evaluating rail net capacities. Tech. rep., The RAND Corporation, Santa Monica, California, October 24 1955. Cited in [66].

[40] R. Hassin and D. B. Johnson. An $O(n \log^2 n)$ algorithm for maximum flow in undirected planar networks. *SIAM J. Comput.* 14(3):612–624, 1985.

[41] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001. ⟨http://www.math.cornell.edu/∼hatcher/⟩.

[42] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.* 55(1):3–23, 1997.

[43] J. M. Hochstein and K. Weihe. Maximum s-t-flow with k crossings in $O(k^3 n \log n)$ time. *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms*, 843–847, 2007.

[44] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). *Proc. 6th ACM Symp. Theory Comput.*, 172–184, 1974.

[45] H. Imai and K. Iwano. Efficient sequential and parallel algorithms for planar minimum cost flow. *Proc. SIGAL Int. Symp. Algorithms*, 21–30, 1990. Lecture Notes Comput. Sci. 450, Springer-Verlag.

[46] P. Indyk and A. Sidiropoulos. Probabilistic embeddings of bounded genus graphs into planar graphs. *Proc. 23rd Ann. ACM Symp. Comput. Geom.*, 204–209, 2007.

[47] A. Itai and Y. Shiloach. Maximum flow in planar networks. *SIAM J. Comput.* 8:135–150, 1979.

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

[49] K. Kawarabayashi and B. Reed. Computing crossing number in linear time. *Proc. 39th Ann ACM Symp. Theory Comput.*, 382–390, 2007.

[50] P. Klein. Multiple-source shortest paths in planar graphs. *Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms*, 146–155, 2005.

[51] P. 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. *Proc. 20th Ann. ACM-SIAM Symp. Discrete Algorithms*, 236–245, 2009.

[52] M. Kutz. Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time. *Proc. 22nd Ann. ACM Symp. Comput. Geom.*, 430–438, 2006.

[53] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.* 16:346–358, 1979.

[54] M. Mareš. Two linear time algorithms for MST on minor closed graph classes. *Archivum Mathematicum* 40(3):315–320, 2004.

[55] S. T. McCormick, M. R. Rao, and G. Rinaldi. Easy and difficult objective functions for max cut. *Math. Program., Ser. B* 94:459–466, 2003.

[56] G. L. Miller. Isomorphism testing for graphs of bounded genus. *Proc. 12th ACM Symp. Theory Comput.*, 225–235, 1980.

[57] G. L. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. *SIAM J. Comput.* 24(5):1002–10017, 1995.

[58] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[59] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica* 7:105–113, 1987.

[60] V. Y. Pan and J. H. Reif. Fast and efficient parallel solution of sparse linear systems. *SIAM J. Comput.* 22(6):1227–1250, 1993.

[61] D. Pe'er. On minimum spanning trees. Master's thesis, Hebrew University, 1998. ⟨http://www.math.ias.edu/∼avi/STUDENTS/dpthesis.pdf⟩.

[62] J. Reif. Minimum s-t cut of a planar undirected network in $O(n \log^2 n)$ time. *SIAM J. Comput.* 12:71–81, 1983.

[63] G. Ringel. *Map Color Theorem*. Springer-Verlag, 1974.

[64] G. Ringel and J. W. T. Youngs. Solution of the Heawood map-coloring problem. *Proc. Nat. Acad. Sci. USA* 60:438–445, 1968.

[65] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics 24. Springer-Verlag, 2003.

[66] A. Schrijver. On the history of combinatorial optimization (till 1960). *Handbook of Discrete Optimization*, 1–68, 2005. Elsevier.

[67] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.* 26(3):362–391, 1983.

[68] J. Stillwell. *Classical Topology and Combinatorial Group Theory*, 2nd edition. Graduate Texts in Mathematics 72. Springer-Verlag, 1993.

[69] S. Tazari and M. Müller-Hannemann. Shortest paths in linear time on minor-closed graph classes, with an application to Steiner tree approximation. *Discrete Appl. Math.* 157:673–684, 2009.

[70] C. Thomassen. Embeddings of graphs with no short non-contractible cycles. *J. Comb. Theory Ser. B* 48(2):155–177, 1990.

[71] P. M. Vaidya. Speeding-up linear programming using fast matrix multiplication. *Proc. 30th IEEE Symp. Found. Comput. Sci.*, 332–337, 1989.

[72] K. Weihe. Maximum (s,t)-flows in planar networks in $O(|V| \log |V|)$-time. *J. Comput. Syst. Sci.* 55(3):454–476, 1997.

[73] A. Zomorodian. *Topology and Computing*. Cambridge University Press, 2005.

[74] A. Zomorodian and G. Carlsson. Localized homology. *Proc. IEEE Int Conf. Shape Modeling Appl. (SMI)*, 189–198, 2007.