Homology Flows, Cohomology Cuts*

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 maximum flows in surface-embedded graphs in near-linear time. 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. Except for the special case of planar graphs, for which an $O(n \log n)$-time algorithm has been known for 20 years, the best previous time bounds for maximum flows in surface-embedded graphs follow from algorithms for general sparse graphs. Our key insight is to optimize the relative homology class of the flow, rather than directly optimizing the flow itself. A dual formulation of our algorithm computes the minimum-cost cycle or circulation in a given (real or integer) homology class.

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

General Terms: Algorithms, Performance

Keywords: computational topology, combinatorial optimization

Errors, like straws, upon the surface flow;
He who would search for pearls must dive below.
— John Dryden, *All for Love*, Prologue (1677)

1. INTRODUCTION

Planar graphs are natural targets for study. In addition to modeling real-world scenarios ranging from road networks to VLSI layouts, they often admit much faster algorithms compared to more general graphs. Most algorithms for planar graphs have been generalized to larger families of graphs, such as graphs of higher genus, graphs with forbidden minors, or graphs with small separators. Examples include single-source and multiple-source shortest paths [14, 29, 43, 53, 54, 56, 72]; minimum spanning trees [67, 57]; graph and subgraph isomorphism [24, 25, 35, 45, 60]; and approximation algorithms for the traveling salesman problem, Steiner trees, and other NP-hard problems [9, 10, 11, 21, 25].

A stark exception to this general pattern is the classical maximum flow problem and its dual, the minimum cut problem. Flow and cuts were originally developed as tools for studying railway and other transportation networks [39], which are naturally modeled as planar graphs; Ford and Fulkerson's seminal paper [30] includes an algorithm for planar networks where the source and target vertices lie on the same face. A long series of results has led to planar maximum-flow algorithms that run in $O(n \log n)$ time, first for undirected graphs [31, 41, 68] and more recently for directed graphs [8, 12, 13]. Despite more than half a century of attention on flows in planar graphs, surprisingly little is known about flows in these more general graph families. Even for graphs embedded on the torus, the fastest algorithms to compute maximum flows are no faster than for arbitrary sparse graphs.

This paper describes the first algorithms to find maximum flows in surface-embedded graphs in near-linear time when the genus is fixed. The input to our problem is an undirected graph $G = (V, E)$ embedded on an orientable surface of genus g, along with two vertices s and t and a capacity function $c : E \to \mathbb{R}^+$. For any fixed genus g and polynomially-bounded capacities, both our algorithms run in $O(n \operatorname{polylog} n)$ time. In a companion paper [16], we describe the first algorithm to compute minimum cuts in surface-embedded graphs in $O(n \log n)$ time for any fixed genus.

Before describing our results in more detail, we review several previous related results.

Flows in sparse graphs. Euler's formula implies that an 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 [71] and Goldberg and Tarjan [34], run in time $O(n^2 \log n)$. The *minimum-cost* maximum flow can be computed in $O(n^2 \log^2 n)$ time using an algorithm of Orlin [65]. (For graphs with small separators, the running time of Orlin's algorithm can be improved to $O(n^2 \log n)$ by replacing Dijkstra's algorithm with a linear-time

*Research partially supported by NSF grant DMS-0528086.

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

shortest-path algorithm [43, 72].) The fastest algorithm known for integer capacities, due to Goldberg and Rao [33], runs in $O(\min\{n^{2/3}, m^{1/2}\} m \log(n^2/m) \log U) = O(n^{3/2} \log n \log U)$ time, where U is an upper bound on the edge capacities. The recent algorithm of Diatch and Spielman [22] computes the minimum-cost maximum flow in time $O(n^{3/2} \operatorname{polylog} n \log U)$.

For further background on maximum flow algorithms and related results, we refer the reader to monographs by Ahuja *et al.* [4] and Schrijver [69].

Flows in planar graphs. Maximum flows in planar graphs have received considerable attention for more than 50 years. Weihe [77] and Borradaile and Klein [8, 12, 13] describe the history of planar flow algorithms in detail; we describe only a few important highlights.

Itai and Shiloach exploited the connection between maximum flows in an undirected planar graph and shortest paths in its dual graph to obtain an $O(n \log n)$-time algorithm when the source and sink vertices lie on a common face [49]; see also Hassin [40].

Reif [68] developed a divide-and-conquer algorithm to compute a minimum cut, and thus the maximum flow *value*, in a planar undirected network in $O(n \log^2 n)$ time. Reif's algorithm 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 [41]. Frederickson subsequently improved Reif's algorithm to $O(n \log n)$ time [31]. Frederickson's improvement can also be obtained more directly using more recent planar shortest-path algorithms [14, 43, 53, 72].

Maximum flows in *directed* planar graphs were first investigated by Johnson and Venkatesan [50], who described an algorithm based on recursive separator decompositions with running time $O(n^{3/2} \log n)$. Venkatesan [75] observed that a feasible flow with a given *value*, if such a flow exists, can be computed in $O(n^{3/2})$ time by computing a single-source shortest path tree in a dual graph with both positive and negative edge weights, using an algorithm of Lipton, Rose, and Tarjan [56]; see also [61]. Binary search over the possible flow values gives a max-flow algorithm that runs in $O(n^{3/2} \log C)$ time, where C is the sum of the capacities. This running time is improved by recent planar shortest path algorithms [43, 29]; the algorithm of Klein, Mozes, and Weimann [54] implies a running time of $O(n \log^2 n \log C)$. Returning to the classical augmenting path technique, Weihe [77, 76] described a planar maximum-flow algorithm that runs in $O(n \log n)$ time, provided the input graph satisfies a certain connectivity condition. Finally, Borradaile and Klein [8, 12, 13] described an $O(n \log n)$-time algorithm to find maximum flows in arbitrary directed planar graphs.

Generalizations of planar graphs. Surprisingly little is known about the complexity of flow algorithms for generalizations of planar graphs. Miller and Naor [61] generalized Johnson and Venkatesan's algorithm to planar (single commodity) flow networks with multiple sources and sinks. A recent algorithm of Hochstein and Wiehe [44] 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 [34].

To our knowledge, the only prior result that applies to graphs of positive genus, but not to arbitrary sparse graphs, is an algorithm of Imai and Iwano [48] that computes minimum-cost flows in graphs with small balanced separators, using a combination of nested dissection [56, 66], interior-point methods [74], and fast matrix multiplication. Their algorithm can be adapted to compute maximum flows 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 [33, 22].

New results. Our key insight generalizes the relationship between flows and dual shortest paths in planar graphs first observed by Venkatesan [75]. We prove that given any flow f, one can find a feasible flow in the same *homology class* in near-linear time by computing a single-source shortest path tree in the dual of the residual network G_f. This observation allows us to optimize the homology class of the flow, rather than directly optimizing the flow itself; instead of optimizing a vector of $O(n)$ flow values, our algorithm optimizes a vector of $2g+1$ homology coefficients. We perform this optimization implicitly using two different techniques. The central-cut ellipsoid method [38, 37] yields an algorithm that runs in $O(g^7 n \log^2 n \log^2 C)$ time for integer capacities that sum to C. Alternately, multidimensional parametric search [1] yeilds an algorithm with running time $g^{O(g)} n \log^{2g+4} n$. Both running times are $O(n \operatorname{polylog} n)$ for any fixed genus g.

A dual formulation of our algorithm finds the minimum-cost circulation in the same homology class as a given circulation, in roughly the same time as computing a maximum flow. If the capacities are integers, the resulting circulation is the minimum-cost *integer* circulation in the desired homology class. The minimum-cost circulation is always the weighted sum of at most $2g$ directed cycles.

In a companion paper [16], we describe an algorithm to compute minimum cuts in $g^{O(g)} n \log n$ time, using very different techniques than in this paper [15, 55]. Essentially the same algorithm computes the shortest cycle in every $\mathbb{Z}_2$-homology class, in the same running time. Unlike the corresponding problem for circulations considered in this paper, we prove that computing the shortest cycle in a *single* $\mathbb{Z}_2$-homology class is NP-hard.

2. DRAMATIS PERSONAE

We begin by recalling several useful definitions from topological graph theory and algebraic topology. For more comprehensive background, we refer the interested reader to Gross and Tucker [36] or Mohar and Thommasen [62] for topological graph theory; and Hatcher [42], Massey [58], or Spanier [63] for algebraic topology.

2.1 Surfaces

A ***surface*** (more formally, a *2-manifold*) is a Hausdorff topological space in which every point has an open neighborhood homeomorphic to $\mathbb{R}^2$. A ***cycle*** in a surface Σ is (the image of) a continuous map $\gamma : S^1 \to \Sigma$; a cycle is ***simple*** if this map is injective. The ***genus*** of a surface Σ is the maximum number of simple, disjoint, non-separating cycles $\gamma_1, \gamma_2, \ldots, \gamma_g$ in Σ; that is, $\gamma_i \cap \gamma_j = \emptyset$ for all i and j, and the space $\Sigma \setminus (\gamma_1 \cup \cdots \cup \gamma_g)$ 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, namely the sphere with g handles attached. We will also assume that $g = o(n)$, since our new algorithms improve existing results only when g is small.

2.2 Graphs and Embeddings

Let $G = (V, E)$ be an undirected graph. We define an associated directed graph $\vec{G} = (V, \vec{E})$ by replacing each undirected edge in E with an antisymmetric pair of directed edges. The graphs G and $\vec{G}$ are represented by the same adjacency matrix. Following Borradaile and Klein [8, 12, 13], we refer to the directed edges in $\vec{E}$ as ***darts***. Each dart connects two (possibly equal) vertices, called its ***tail*** and its ***head***; we say that the dart ***leaves*** its tail and ***enters*** its head. Each dart $\vec{e}$ has a unique ***reversal***, denoted $rev(\vec{e})$ and defined by swapping its endpoints: $head(rev(\vec{e})) = tail(\vec{e})$ and $tail(rev(\vec{e})) = head(\vec{e})$. We will often write $u{\to}v$ to denote a dart with tail u and head v; thus, $rev(u{\to}v) = v{\to}u$.

Informally, an ***embedding*** of a graph G on a surface Σ is a drawing of the graph on Σ, such that vertices are mapped to distinct points and edges are mapped to non-crossing curves. 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* [62]) if every face is an open topological disk. Any cellular embedding can be represented combinatorially by a ***rotation system***, which is a permutation π of the darts of G, where $\pi(\vec{e})$ is the dart that appears immediately after $\vec{e}$ in the counterclockwise ordering of darts leaving $tail(\vec{e})$.

Suppose G is a simple n-vertex graph cellularly embedded on an orientable surface of genus g. Euler's formula $|V| - |E| + |F| = 2 - 2g$ implies that G has at most $3n - 6 + 6g$ edges and at most $2n - 4 + 4g$ faces, with equality if every face of the embedding is a triangle. In this paper, we consider only graphs and surfaces with $1 \le g \ll n$, so the overall complexity of any embedding is $O(n)$.

Every dart in an embedded graph G separates two (possibly equal) faces, called the ***left shore*** and ***right shore***. We say that a dart $\vec{e}$ ***winds counterclockwise*** around $left(\vec{e})$ and ***winds clockwise*** around $right(\vec{e})$. Reversing the dart swaps these two faces: $left(rev(\vec{e})) = right(\vec{e})$ and $right(rev(\vec{e})) = left(\vec{e})$. We will sometimes write $f{\uparrow}g$ to denote a dart whose left shore is f and whose right shore is g; thus, $rev(f{\uparrow}g) = g{\uparrow}f$. See Figure 1.

2.3 Chains, Circulations, and Flows

Let $G = (V, E)$ be an undirected graph embedded on a surface Σ, and let F denote the set of faces of the embedding. A ***k-chain*** is a function that assigns a real weight to all cells of dimension k. Thus, a ***0-chain*** is a function $\omega : V \to \mathbb{R}$; a ***1-chain*** is a function $\phi : E \to \mathbb{R}$; and a ***2-chain*** is a function $\alpha : F \to \mathbb{R}$.

It is useful to think of each 1-chain as assigning both an orientation and a *non-negative* value to each edge in G. We implicitly extend any 1-chain to a function on the darts of G; for each edge uv, we arbitrarily choose one of its darts $u{\to}v$ and define $\phi(u{\to}v) = \phi(uv)$ and $\phi(v{\to}u) = -\phi(uv)$.

The ***boundary*** of a 1-chain ϕ is the 0-chain $\partial\phi : V \to \mathbb{R}$ defined as follows:

$$\partial\phi(v) := \sum_{u\,:\,u{\to}v \in \vec{E}} \phi(u{\to}v)$$

A ***circulation*** is a 1-chain ϕ such that $\partial\phi(v) = 0$ for every vertex $v \in V$. The equation $\partial\phi(v) = 0$ is often called the *flow conservation constraint* at v; intuitively, the total flow into v equals the total flow out of v. For any two vertices s and t, an ***(s, t)-flow*** is a 1-chain ϕ such that $\partial\phi(v) = 0$ for every vertex $v \in V \setminus \{s, t\}$. The ***value*** of a flow ϕ is $\partial\phi(t) = -\partial\phi(s)$; a circulation is simply a flow with value 0.

The (first) ***chain space C(G)*** is the vector space of all 1-chains in G, which is isomorphic to $\mathbb{R}^{|E|}$; this is sometimes also called the *edge space*. The ***cycle space Z(G)*** is the vector space of all circulations in G, which is isomorphic to $\mathbb{R}^{|E|-|V|+1}$. The ***flow space Z(G; st)*** is the vector space of all (s,t)-flows in G, which is isomorphic to $\mathbb{R}^{|E|-|V|+2}$. The cycle space is (redundantly) generated by the indicator functions of all simple directed cycles in $\vec{G}$, and the flow group is (redundantly) generated by the indicator functions of all directed walks from s to t in $\vec{G}$.

2.4 Boundary Circulations and Homology

The boundary of a 2-chain $\alpha : F \to \mathbb{R}$ is the 1-chain $\partial\alpha : E \to \mathbb{R}$ defined by setting $\partial\alpha(\vec{e}) := \alpha(right(\vec{e})) - \alpha(left(\vec{e}))$. One can easily verify that the boundary of any 2-chain is a circulation. A ***boundary circulation*** is the boundary of some 2-chain. In planar graphs, every circulation is a boundary circulation, but this is not true for higher-genus embeddings. The ***boundary space B(G)*** is the vector space of all boundary circulations; this is a linear subspace of $Z(G)$, isomorphic to $\mathbb{R}^{|F|-1}$.

We say that two flows or circulations ϕ and ψ are ***homologous***, or in the same ***homology class***, if their difference $\phi - \psi$ is a boundary circulation. We write $\phi \simeq \psi$ to denote that ϕ and ψ are homologous.

The ***homology space H(G)*** is the vector space of all homology classes of circulations in G, which is isomorphic to $Z(G)/B(G) \cong \mathbb{R}^{|E|-|V|-|F|+2} = \mathbb{R}^{2g}$ by Euler's formula. Similarly, the ***(s, t)-flow homology space***, which we denote by $H(G; st)$, is the vector space of all homology classes of (s,t)-flows in G, which is isomorphic to $Z(G; st)/B(G) \cong \mathbb{R}^{|E|-|V|-|F|+1} = \mathbb{R}^{2g+1}$.

2.5 Capacities and Residual Networks

Now fix a positive ***capacity*** function $c : E \to \mathbb{R}^+$. A flow or circulation ϕ is ***feasible*** (with respect to c) if and only if $|\phi(e)| \le c(e)$ for every edge $e \in E$. The ***residual capacity*** function $c_\phi : \vec{E} \to \mathbb{R}$ is defined by setting $c_\phi(u{\to}v) = c(uv) - \phi(u{\to}v)$. The ***residual network G_ϕ*** is just the graph $\vec{G}$ with darts weighted by the residual capacity function c_ϕ. Clearly, ϕ is feasible if and only if every dart in G_ϕ has non-negative residual capacity. Moreover, ϕ is a *maximum* flow if and only if there is no directed path in G_ϕ from s to t in which every dart has positive residual capacity. We emphasize that c and c_ϕ are *not* 1-chains.

2.6 Dual Graphs, Cocycles, and Cohomology

The ***dual graph G^**** of an embedded graph G is the (multi-)graph whose vertices are the faces of G, where two faces are joined by a (dual) edge if and only if they 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, so that faces of G^* correspond exactly to vertices of G. For any vertex v of G, we let v^* denote the corresponding face of G^*. We orient the darts of G^* by defining $(u{\to}v)^* := u^*{\uparrow}v^*$ and $(f{\uparrow}g)^* := f^*{\to}g^*$. Duality is an involution—the dual of G^* is isomorphic to the original graph G. However, G and G^* use opposite orientations of the underlying surface Σ to distinguish left from right.

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 the subgraph H is a cycle, we call H^* a ***cocycle***. The bijection between the edges of G and the edges of G^* extends to a bijection between 1-chains in G and in G^*. A ***cocirculation*** is a 1-chain whose dual is a circulation in G^*; a

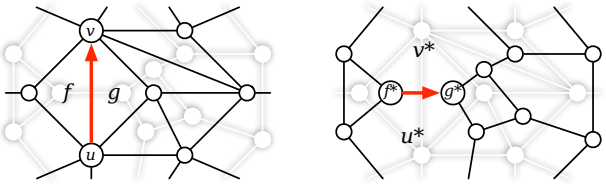

Figure 1. Graph duality. One dart $u{\to}v = f{\uparrow}g$ and its dual $f^*{\to}g^* = u^*{\uparrow}v^*$ are emphasized.

coboundary is a 1-chain whose dual is a boundary circulation in G^*; two cocirculations are ***cohomologous*** if their difference is a coboundary, or equivalently, if their dual circulations are homologous.

3. HOMOLOGY FLOWS

Throughout this section, we fix an undirected graph G, a cellular embedding of G on an orientable surface Σ of genus g, a capacity function $c\colon E(G) \to \mathbb{R}^+$, and two vertices s and t.

3.1 Homologous Feasible Flows

Let $\phi\colon E \to \mathbb{R}$ be an arbitrary (i.e. not necessarily feasible) flow in G. The ***dual residual network*** G^*_ϕ is the directed dual graph $\vec{G}^*$, where every dual dart $\vec{e}^*$ has a *cost* $c_\phi(\vec{e}^*)$ equal to the residual capacity of its corresponding primal dart: $c_\phi(\vec{e}^*) = c_\phi(\vec{e})$.

Lemma 3.1. *There is a feasible (s,t)-flow in G homologous to a given (s,t)-flow ϕ if and only if the dual residual network G^*_ϕ contains no negative-cost cycles.*

Proof: Let λ^* be an arbitrary directed cycle in G^*_ϕ, and let λ denote the corresponding directed cocycle in $\vec{G}$. The total cost of λ^* is the difference between the total capacity of λ and the total flow through λ:

$$c_\phi(\lambda^*) \;=\; c(\lambda) - \phi(\lambda) \;=\; \sum_{\vec{e}\in\lambda} c(e) - \sum_{\vec{e}\in\lambda} \phi(\vec{e}).$$

For any 2-chain $\alpha\colon F \to \mathbb{R}$, we have

$$\begin{aligned}\sum_{\vec{e}\in\lambda} \partial\alpha(\vec{e}) \;&=\; \sum_{f\uparrow g\in\lambda} \big(\alpha(g) - \alpha(f)\big) \\ &=\; \sum_{f^*\to g^*\in\lambda^*} \big(\alpha(g) - \alpha(f)\big) \;=\; 0.\end{aligned}$$

(The last equality follows from the fact that λ^* is a cycle.) Thus, for any flow ψ homologous to ϕ, we have $\psi(\lambda) = \phi(\lambda)$, which immediately implies that $c_\psi(\lambda^*) = c_\phi(\lambda^*)$.

If the cycle λ^* has negative cost, then for any flow ψ homologous to ϕ, we have $c_\psi(\lambda^*) = c_\phi(\lambda^*) < 0$. It follows immediately that $c_\psi(\vec{e}) < 0$ for at least one dart $\vec{e}$ in λ; in other words, ψ is infeasible.

On the other hand, suppose G^*_ϕ has no negative cycles. Fix an arbitrary source vertex x^* in G^*_ϕ. For any face f of G, let $\alpha(f)$ denote the shortest-path distance from x^* to f^* in G^*_ϕ; these distances are well-defined precisely because G^*_ϕ has no negative cycles. Finally, consider the flow $\psi := \phi + \partial\alpha$, which is clearly homologous to ϕ. Because α is defined by shortest-path distances, we have $c_\phi(f{\uparrow}g) = c_\phi(f^*{\to}g^*) \geq \alpha(g) - \alpha(f)$, and therefore

$$\begin{aligned}\psi(f{\uparrow}g) &= \phi(f{\uparrow}g) + \alpha(g) - \alpha(f) \\ &\leq \phi(f{\uparrow}g) + c_\phi(f{\uparrow}g) \\ &= c(f{\uparrow}g)\end{aligned}$$

for every dart $f{\uparrow}g$. In other words, ψ is feasible. □

Theorem 3.2. *Given an (s,t)-flow ϕ in G, we can find a feasible (s,t)-flow in G that is homologous with ϕ, or determine that no homologous feasible flow exists, using $O(gn\log^2 n)$ arithmetic operations.*

Proof: The recent algorithm of Klein, Mozes, and Weimann [54] computes either a single-source shortest path tree or a negative cycle in an embedded directed graph in $O(gn\log^2 n)$ time. Klein *et al.* describe their algorithm only in the context of planar graphs. However, their algorithm generalizes directly to higher-genus graphs via the observation that any n-vertex graph of genus g can be separated into planar subgraphs, each with at most $2n/3$ vertices, by removing $O(\sqrt{gn})$ edges [23, 32, 47, 46]. Moreover, such a separator can be computed in $O(n)$ time [5, 26]. □

3.2 Flow Homology Basis

Every (s,t)-flow can be expressed as a weighted sum of walks[1] from s to t. Consequently, every homology class of (s,t)-flows is a weighted sum of homology classes of (s,t)-walks. It follows immediately that the flow homology space $Z(G,st) \cong \mathbb{R}^{2g+1}$ can be generated by the homology classes of $2g+1$ (s,t)-walks. We call such a collection of walks a ***flow homology basis***.

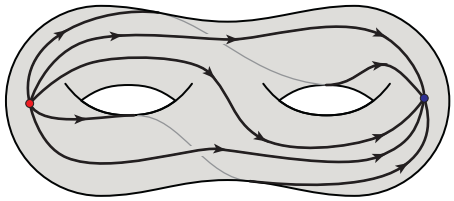

Figure 2. A flow homology basis for a surface of genus 2.

Lemma 3.3. *We can compute a flow homology basis for G in $O(gn)$ time.*

Proof: We begin by computing a tree-cotree decomposition [26]. Let T be an arbitrary spanning tree of G; let C^* be an arbitrary spanning tree of $(G \setminus T)^*$; and finally, let $X = G \setminus (T \cup C)$. Euler's formula implies that X contains exactly $2g$ edges; call them $e_1, e_2, \ldots, e_{2g}$. Orient these edges arbitrarily.

We define $2g+1$ walks $w_0, w_1, \ldots, w_{2g}$ as follows. Let w_0 denote the unique path from s to t in T. For each index i between 1 and $2g$, let w_i denote the walk obtained by concatenating the unique path in T from s to $\mathit{tail}(\vec{e}_i)$, followed by the dart $\vec{e}_i$, followed by the unique path in T from $\mathit{head}(\vec{e}_i)$ to t. (Note that in each walk w_i, each edge in G is traversed at most twice, and

[1] These walks are not necessarily simple paths; vertices and edges may be repeated. Indeed, an edge of G can carry flow even though it does not lie on *any* simple path from s to t. In principle, we can discard such 'useless' edges, because there is always at least one maximum flow that does not use them. However, no algorithm is known to find all useless edges in subquadratic time, even in planar graphs [7, 12, 13].

the repeated edges (if any) lie on a simple path.) We claim that the walks $w_0, w_1, \ldots, w_{2g}$ lie in independent homology classes, and hence comprise a basis for the flow homology space $H(G, st)$.

Suppose to the contrary that $w_j \cong \sum_{i<j} a_i w_i$ for some index $j \neq 0$ and some real coefficients a_i. Then the difference $\varphi = w_j - \sum_{i<j} a_i w_i$ is a boundary circulation, which has nonzero value only on edges of $T \cup X$. Let $\alpha : F \to \mathbb{R}$ be a 2-chain such that $\varphi = \partial\alpha$. For any cotree edge $e \in C$, we have $\varphi(e) = 0$. Thus, if a cotree edge separates two faces f and f', we must have $\alpha(f) = \alpha(f')$. Because C^* is a spanning tree for G^*, it follows that α assigns the same value to *every* face of G, which implies that φ is identically zero. On the other hand, walk w_j contains an edge e_j that does not lie on any other basis walk, so $\varphi(e_j) = w_j(e_j) = 1$. We have a contradiction.

The tree T and cotree U can each be constructed in $O(n)$ time using (for example) depth-first search, after which each walk w_i can be easily constructed in $O(n)$ time. □

Fix a flow homology basis $w_0, w_1, \ldots, w_{2g}$ for G. A ***basic flow*** is any flow ϕ of the form $\sum_{i=0}^{2g} \phi_i \cdot w_i$ for some coefficients $\phi_0, \phi_1, \ldots, \phi_{2g}$. Equivalently, a flow ϕ is basic if and only if $\phi(e) = 0$ for every cotree edge $e \in C$. Every flow in G is homologous to exactly one basic flow.

Corollary 3.4. *Given the coefficients $\phi_0, \phi_1, \ldots, \phi_{2g}$ of a basic flow ϕ, we can compute a feasible (s,t)-flow homologous with ϕ, or determine that no homologous feasible flow exists, using $O(gn \log^2 n)$ arithmetic operations.*

3.3 Optimization

The preceding results imply that to compute a maximum (s,t)-flow in G, it suffices to find a basic flow ϕ of maximum value such that the dual residual network G^*_ϕ contains no negative cycles. We can formulate this optimization problem as a linear program as follows. For any directed cocycle λ, let

$$c(\lambda) := \sum_{\vec{e} \in \lambda} c(e) \qquad \text{and} \qquad w_i(\lambda) = \sum_{\vec{e} \in \lambda} w_i(\vec{e}).$$

The optimal basic flow is the solution to the following linear programming problem.

$$\begin{aligned} \max \quad & \sum_{i=0}^{2g} \phi_i \\ \text{s.t.} \quad & \sum_{i=0}^{2g} \phi_i \cdot w_i(\lambda) \le c(\lambda) \quad \text{for each cocycle } \lambda \text{ in } G \end{aligned} \tag{LP}$$

In fact, most of the constraints in this linear program are redundant; it suffices to consider only cocycles λ whose dual cycles λ^* are simple and have minimum cost (that is, primal capacity) in their homology class. However, there could be as many as $n^{\Theta(g)}$ non-redundant constraints; consequently, we must solve (LP) implicitly.

3.3.1 Ellipsoid Method

We now transform our decision procedure into an optimization algorithm using the *central-cut ellipsoid method*. A full technical description of the method is given by Grötschel *et al.* [38, 37]. We briefly sketch the method here, deferring most of the technical details to the appendix.

Let Φ denote the polytope of feasible basic flow coefficients, and let ϕ_{OPT} denote the optimum basic flow. The ellipsoid algorithm maintains an ellipsoid $\mathcal{E}$ guaranteed to contain ϕ_{OPT}. First, we perturb the objective function slightly to guarantee that ϕ_{OPT} is unique; let $\tilde{c}$ denote the perturbed objective vector. Initially, $\mathcal{E}$ is large enough to contain the entire polytope Φ. At each iteration, we call the decision procedure from Corollary 3.4 to determine whether the centroid x of $\mathcal{E}$ is feasible. If x is infeasible, the decision procedure returns a negative cycle, which corresponds to a violated constraint in the linear program. If x is feasible, we add the artificial constraint $\langle \tilde{c}, \phi \rangle \ge \langle \tilde{c}, x \rangle$. In either case, we obtain a halfspace h that contains ϕ_{OPT}. We then compute a close approximation of the smallest ellipsoid containing $\mathcal{E} \cap h$, and let this be the new ellipsoid $\mathcal{E}$. Each iteration reduces the volume of $\mathcal{E}$ by a factor of $e^{1/O(g)}$. Let $\Phi_\varepsilon = \Phi \cap h_\varepsilon$, where h_ε is the halfspace $\langle \tilde{c}, \phi \rangle \ge \langle \tilde{c}, \phi_{OPT} \rangle - \varepsilon$, where ε is chosen so that Φ_ε lies inside a ball of radius $1/3$. The iterations stop when the volume of $\mathcal{E}$ is smaller than the volume of Φ_ε. Even though the constraint matrix defining Φ is *not* totally unimodular, we prove in the full paper that the vertices of Φ have integer coordinates. Thus, we obtain the optimal flow value ϕ_{OPT} by rounding the last feasible point found (which is guaranteed to lie inside Φ_ε) to the integer grid.

In the full paper, we prove that Φ is contained in a sphere of radius $O(Cg)$ centered at the origin, where C is the sum of the edge capacities; we can take this ball to be our initial ellipsoid. We also prove that for an appropriate perturbed objective function $\tilde{c}$, the volume of Φ_ε is $C^{-O(g^2)}$. These two facts imply that the ellipsoid algorithm halts after $O(g^3 \log C)$ iterations. In each iteration, the decision procedure executes $O(gn \log^2 n)$ additions, subtractions, and comparisons. Grötschel *et al.* prove that to maintain sufficient precision over K iterations, it suffices to round all numbers to $O(K)$ bits [37, 38]. Thus, each arithmetic operation in our decision procedure requires $O(g^3 \log C)$ time.

Theorem 3.5. *Given a graph $G = (V, E)$ embedded on a surface of genus g, a positive integer capacity function $c : E \to \mathbb{Z}^+$, and two vertices $s, t \in V$, a maximum (s,t)-flow in G can be computed in time $O(g^7 n \log^2 n \log^2 C)$, where C is the sum of the edge capacities.*

3.3.2 Multidimensional Parametric Search

We now describe a more combinatorial algorithm whose running time has better dependence on the complexity of the graph, but exponential dependence on the genus of the underlying surface. Our algorithm uses the *multidimensional search* paradigm independently developed by Cohen and Megiddo [17, 18, 19], Norton, Plotkin, and Tardos [64], and Aneja and Kabadi [6], and extended further by several other authors [1, 2, 3, 20, 51, 52, 73]. Specifically, we use the version of the technique described by Agarwala and Fernández-Baca [1].

The method requires two black-box algorithms for the decision problem, one serial and the other parallel. The method also requires that the parallel algorithm can be modeled as a *linear decision tree* with respect to the input parameters. That is, every branch in the parallel algorithm is based on the sign of an affine combination of the input parameters. Let T_s denote the running time of the serial decision algorithm, T_p the running time of the parallel decision algorithm, P the number of processors used by the parallel decision algorithm, and d the number of parameters to be resolved. The running time of the resulting optimization algorithm is $d^{O(d)} \cdot (T_p + \log P)^d \cdot T_s$. (Agarwala and Fernández-Baca do not report the dependence of their running time on d; the bound stated here follows form the analysis by Agarwal, Sharir, and Toledo [3].)

In our application, we have $d = 2g+1$; the algorithm of Klein *et al.* [54] gives us $T_s = O(gn\log^2 n)$; and a parallel version of the Floyd-Warshall algorithm [59] gives us $T_p = O(\log n \log\log n)$ and $P = O(n^3)$. Thus, the overall running time of our algorithm is $g^{O(g)}\, n\, (\log n)^{2g+3}\, (\log\log n)^{2g+1}$.

Theorem 3.6. *Given a graph $G = (V,E)$ embedded on a surface of genus g, a positive capacity function $c : E \to \mathbb{R}^+$, and two vertices $s, t \in V$, a maximum (s,t)-flow in G can be computed in $g^{O(g)} n \log^{2g+4} n$ time.*

4. COHOMOLOGY CUTS

A minor modification of our algorithm allows us to solve two interesting special cases of the minimum-cost (or equivalently, maximum-value) circulation problem in roughly the same time as a maximum (s,t)-flow.

4.1 Maximum-Value Circulations

Suppose we are given a graph G (with no source or sink), a positive capacity function $c : E \to \mathbb{R}^+$, and a ***value*** function $\theta : \vec{E} \to \mathbb{R}$. The value of a circulation ϕ is the inner product $\langle \phi, \theta \rangle = \sum_{\vec{e} \in \vec{E}} \phi(\vec{e}) \cdot \theta(\vec{e})$. Like the capacity function c, the value function θ is not (in general) a 1-chain; the values of a dart and its reversal need not have any relationship. In particular, some darts may have negative value (that is, positive cost).

The maximum-flow algorithm described in the previous section can be easily modified to compute maximum-value circulations, provided all circulations in the same homology class have the same value. We call the value function $\theta : \vec{E} \to \mathbb{R}$ is ***homology invariant*** if $\langle \phi, \theta \rangle = \langle \psi, \theta \rangle$ for any two homologous circulations $\phi \simeq \psi$, or equivalently, if $\langle \partial\alpha, \theta \rangle = 0$ for any 2-chain α.

Theorem 4.1. *Given a graph $G = (V,E)$ embedded on a surface of genus g, a capacity function $c : E \to \mathbb{R}^+$, and a homology-invariant value function $\theta : \vec{E} \to \mathbb{R}$, we can compute a maximum-value circulation in $g^{O(g)} n \log^{2g+3} n$ time, or in $O(g^7 n \log^2 n \log^2 C)$ time if capacities are integers that sum to C.*

Proof (sketch): The homology space $H(G) \cong \mathbb{R}^{2g}$ can be generated by (the homology classes of) $2g$ directed cycles $\gamma_1, \gamma_2, \ldots, \gamma_{2g}$ in independent homology classes. Using an algorithm similar to Lemma 3.3, we can construct such a set of cycles in $O(gn)$ time [27, 28].

Corollary 3.4 implies that it suffices to find the homology class of the maximum-value circulation. Specifically, we need to find a feasible homology vector $(\phi_1, \ldots, \phi_{2g})$ such that the cost function

$$\left\langle \sum_{i=1}^{2g} \phi_i \cdot \gamma_i,\ \theta \right\rangle = \sum_{i=1}^{2g} \phi_i \cdot \langle \gamma_i, \theta \rangle$$

is maximized. Corollary 3.4 gives us strong membership and separation oracles for this linear optimization problem, so we can apply either the central-cut ellipsoid method or multidimensional parametric search, exactly as we did for the standard maximum-flow problem. Except for the change of objective function and the slightly smaller dimension, the optimization algorithm is identical. □

The following lemma exactly characterizes homology-invariant value functions. Recall that a 1-chain $\theta : E \to \mathbb{R}$ is a ***cocirculation*** if its dual 1-chain $\theta^* : E^* \to \mathbb{R}$, defined by setting $\theta^*(\vec{e}^*) = \theta(\vec{e})$, is a circulation in G^*.

Lemma 4.2. *A value function $\theta : \vec{E} \to \mathbb{R}$ is homology invariant if and only if θ is a cocirculation.*

Proof: If the function $\theta : E \to \mathbb{R}$ is not a cocirculation, then for some face f, we have

$$\langle \partial f, \theta \rangle = \sum_{\vec{e}:\, left(\vec{e}) = f} \theta(\vec{e}) \neq 0$$

Because θ gives non-zero value to the boundary circulation ∂f, it cannot be homology invariant.

On the other hand, suppose θ is an arbitrary cocirculation and $\alpha : F \to \mathbb{R}$ is an arbitrary 2-chain. We can easily verify that $\langle \partial\alpha, \theta \rangle = 0$ as follows:

$$\begin{aligned}
\langle \partial\alpha, \theta \rangle &= \sum_{\vec{e} \in \vec{E}} \partial\alpha(\vec{e}) \cdot \theta(\vec{e}) \\
&= \sum_{\vec{e} \in \vec{E}} \big(\alpha(left(\vec{e})) - \alpha(right(\vec{e}))\big) \cdot \theta(\vec{e}) \\
&= \sum_{f \in F} \left(\sum_{\vec{e}:\, left(\vec{e}) = f} \alpha(f) \cdot \theta(\vec{e}) \ - \sum_{\vec{e}:\, right(\vec{e}) = f} \alpha(f) \cdot \theta(\vec{e}) \right) \\
&= \sum_{f \in F} \left(\alpha(f) \cdot \left(\sum_{\vec{e}:\, left(\vec{e}) = f} \theta(\vec{e}) \ - \sum_{\vec{e}:\, right(\vec{e}) = f} \theta(\vec{e}) \right) \right) \\
&= \sum_{f \in F} \left(\alpha(f) \cdot \sum_{\vec{e}:\, left(\vec{e}) = f} \Big(\theta(\vec{e}) - \theta(rev(\vec{e})) \Big) \right) \\
&= 2 \cdot \sum_{f \in F} \left(\alpha(f) \cdot \sum_{\vec{e}:\, left(\vec{e}) = f} \theta(\vec{e}) \right) \\
&= 2 \cdot \sum_{f \in F} \alpha(f) \cdot 0 \ = \ 0.
\end{aligned}$$

It follows that θ is homology-invariant. □

4.2 Min-Cost Homologous Circulation

The problem considered in the previous section has the following natural and useful dual interpretation. Consider the following linear programming formulation of the maximum-value circulation problem.

$$\begin{aligned}
\max \quad & \sum_{u \to v} \phi(u \to v) \cdot \theta(u \to v) \\
\text{s.t.} \quad & \sum_{u : uv \in E} \big(\phi(u \to v) - \phi(v \to u)\big) = 0 && \text{for all } v \in V \\
& \phi(u \to v) \le c(u \to v) && \text{for all } u \to v \in \vec{E} \\
& \phi(u \to v) \ge 0 && \text{for all } u \to v \in \vec{E}
\end{aligned}$$

The dual of this linear program has a variable $\alpha(v)$ for each vertex v and a variable $x(u \to v)$ for each dart $u \to v$.

$$\begin{aligned}
\min \quad & \sum_{u \to v} x(u \to v) \cdot c(u \to v) \\
\text{s.t.} \quad & \alpha(u) - \alpha(v) + x(u \to v) \ge \theta(u \to v) && \text{for all } u \to v \in \vec{E} \\
& x(u \to v) \ge 0 && \text{for all } u \to v \in \vec{E}
\end{aligned}$$

This dual linear program is more naturally cast in terms of the dual graph G^*, as follows:

$$\begin{aligned}
\min \quad & \sum_{f \uparrow g} x(f \uparrow g) \cdot c(f \uparrow g) \\
\text{s.t.} \quad & \alpha(f) - \alpha(g) + x(f \uparrow g) \ge \theta(f \uparrow g) && \text{for all } f \uparrow g \in \vec{E}^* \\
& x(f \uparrow g) \ge 0 && \text{for all } f \uparrow g \in \vec{E}^*
\end{aligned}$$

Let $\alpha_{OPT}(f)$ and $x_{OPT}(f\uparrow g)$ denote the variables in the optimum solution to this dual-dual linear program. We view the vector α_{OPT} of face variables as a 2-chain. We define a 1-chain $\vartheta : E^* \to \mathbb{R}$ by setting $\vartheta_{OPT}(f\uparrow g) := x_{OPT}(f\uparrow g) - x_{OPT}(g\uparrow f)$ for every dart $f\uparrow g$. Because every primal capacity $c(u\to v)$ is positive, each dart variable $x_{OPT}(f\uparrow g)$ is individually as small as possible without violating any constraint; that is,

$$x_{OPT}(f\uparrow g) = \max\{0,\ \theta(f\uparrow g) - \alpha(f) + \alpha(g)\}.$$

It follows immediately that $\vartheta_{OPT} = \theta - \partial\alpha$; thus, ϑ_{OPT} is a circulation in G^*, homologous with the circulation θ. Equivalently, ϑ_{OPT} is a *cocirculation* in G, in the same *cohomology* class as θ. Moreover, the optimal objective value can be rewritten as follows:

$$\sum_{f\uparrow g} x_{OPT}(f\uparrow g)\cdot c(f\uparrow g) = \sum_{e^*\in E^*} c(e^*)\cdot|\vartheta_{OPT}(e^*)|$$

We conclude that ϑ_{OPT} is the minimum-capacity cocirculation in the same cohomology class as θ.

Theorem 4.3 (Homological Maxflow/Mincut). *Let $G = (V, E)$ be an undirected graph embedded on a surface of genus g, let $c : E \to \mathbb{R}^+$ be a capacity function, and let $\theta : E \to \mathbb{R}$ be a cocirculation in G. The maximum value $\langle\phi, \theta\rangle$ of any feasible circulation ϕ in G is equal to the minimum capacity of any cocirculation cohomologous with θ.*

A simple modification of our maximum-value circulation algorithm computes the minimum-cost circulation in a given homology class.

Theorem 4.4. *Suppose we are given an undirected graph $G = (V, E)$ embedded on a surface of genus g, a cost function $c : E \to \mathbb{R}^+$, and a circulation $\theta : E \to \mathbb{R}$. We can compute a minimum-cost circulation homologous with θ in $g^{O(g)}n\log^{2g+4} n$ time, or in time $O(g^7 n\log^2 n\log^2 C)$ if all capacities are integers that sum to C.*

Proof (sketch): Within the stated time bounds, we can compute a maximum-value feasible circulation ϕ^*_{OPT} in the dual graph G^*, using c as a capacity function and θ as a homology-invariant value function. Our algorithm optimizes the homology class of the circulation, using a linear-programming formulation similar to (LP):

$$\begin{aligned} \max\quad & \sum_{i=1}^{2g} \phi_i^* \cdot \langle\gamma_i, \theta\rangle \\ \text{s.t.}\quad & \sum_{i=1}^{2g} \phi_i^* \cdot \lambda_i^*(\gamma) \ \le\ c(\gamma) \quad \text{for every cycle } \gamma \text{ in } \vec{G} \end{aligned} \tag{LP2}$$

Here, $\lambda_1^*, \lambda_2^*, \ldots, \lambda_{2g}^*$ are cycles in G^* that generate the homology space of G^*. As described in the proof of Theorem 4.1, we can construct these cycles in $O(gn)$ time.

In any feasible basis for the homology linear program (LP2), exactly $2g$ of the linear constraints are satisfied with equality. These constraints are defined by $2g$ cycles $\gamma_1, \gamma_2, \ldots, \gamma_{2g}$ in $\vec{G}$ whose total residual cost is zero. Each cycle γ_i is the minimum-cost cycle in its homology class. Moreover, these $2g$ cycles lie in independent homology classes, and therefore comprise a basis of the homology space of G.

Let ϑ_{OPT} denote the minimum-cost circulation homologous with θ. This circulation has non-zero value only on edges of G whose dual edges are saturated by ϕ^*_{OPT}. Let X denote the subgraph of G whose dual edges are saturated by G; every circulation in X is a linear combination of the $2g$ basis cycles $\gamma_1, \gamma_2, \ldots, \gamma_{2g}$. In particular, for some real coefficients $a_1, a_2, \ldots, a_{2g}$, we have $\vartheta_{OPT} = \sum_{i=1}^{2g} a_i\gamma_i$.

We compute these coefficients as follows. For each index j, we have a linear equation

$$\langle\theta, \lambda_j^*\rangle = \langle\vartheta_{OPT}, \lambda_j^*\rangle = \left\langle \sum_{i=1}^{2g} a_i\gamma_i,\ \lambda_j^* \right\rangle = \sum_{i=1}^{2g} a_i\langle\gamma_i, \lambda_j^*\rangle.$$

Each inner product $\langle\gamma_i, \lambda_j^*\rangle$ can be trivially computed in $O(n)$ time. Thus, we can compute $a_1, a_2, \ldots, a_{2g}$ in $O(g^2 n)$ time by setting up and solving a system of $2g$ linear equations. □

Finally, consider the special case where the input circulation θ is a single directed cycle. The minimum-cost circulation ϑ_{OPT} homologous to θ is not necessarily a cycle. However, the constraint matrix for the linear program defining ϑ_{OPT} is totally unimodular, which implies that ϑ_{OPT} is an *integer* circulation.

5. CONCLUSIONS AND OPEN PROBLEMS

We have described the first algorithms to compute maximum flows in surface-embedded undirected graphs in near-linear time. Our algorithms can be adapted to directed graphs and to non-orientable surfaces with no change in the running time, although the algorithms are slightly more complicated; details will appear in the full version of the paper. In a companion paper [16], we describe an algorithm to compute minimum cuts in surface-embedded graphs in $O(n\log n)$ time for any fixed genus.

Many improvements and open questions remain. We conjecture that 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 [8, 12, 13]. Even the special case of unit capacities is open.

It would be interesting to generalize our results to compute minimum-cost flows or circulations with no topological restrictions on the cost function. We are unaware of any near-linear-time algorithms to compute minimum-cost flows even in planar graphs.

Acknowledgments. We would like to thank Cora Borradaile, Chandra Chekuri, Sariel Har-Peled, Aparna Sundar, and Kim Whittlesey for helpful discussions.

References

[1] R. Agarwala and D. Fernández-Baca. Weighted multidimensional search and its application to convex optimization. *SIAM J. Comput.* 25:83–99, 1996.

[2] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.* 30:412–458, 1998.

[3] P. K. Agarwal, M. Sharir, and S. Toledo. An efficient multidimensional searching technique and its applications. Tech. Rep. CS-1993-20, Dept. Comp. Sci., Duke Univ., August 1993. ⟨ftp://ftp.cs.duke.edu/pub/dist/techreport/1993/1993-20.ps.gz⟩.

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

[5] L. Aleksandrov and H. Djidjev. Linear algorithms for partitioning embedded graphs of bounded genus. *SIAM J. Discrete Math* 9(1):129–150, 1996.

[6] Y. P. Aneja and S. N. Kabadi. Polynomial algorithms for Lagrangean relaxations in combinatorial problems. Faculty of Business Working Paper Series W91-03, University of Windsor, 1991. Cited in [52].

[7] T. C. Biedl, B. Brejová, and T. Vinař. Simplifying flow networks. *Proc. 25th Symp. Math. Found. Comput. Sci.*, 192–201, 2000. Lecture Notes Comput. Sci. 1893, Springer-Verlag.

[8] 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⟩.

[9] 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/⟩.

[10] 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.

[11] 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.

[12] 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.

[13] 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⟩.

[14] 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.

[15] 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.

[16] E. W. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. *Proc. 25th Ann. ACM Symp. Comput. Geom.*, 2009. ⟨http://www.cs.uiuc.edu/~jeffe/pubs/surfcut.html⟩.

[17] E. Cohen. *Combinatorial Algorithms for Optimization Problems*. Ph.D. thesis, Dept. Comput. Sci., Stanford Univ., June 1991. Tech. Report STAN-CS-91-1366.

[18] E. Cohen and N. Megiddo. Maximizing concave functions in fixed dimension. *Complexity in Numerical Optimization*, 74–87, 1993. World Scientific.

[19] E. Cohen and N. Megiddo. Strongly polynomial-time and NC algorithms for detecting cycles in periodic graphs. *J. Assoc. Comput. Mach.* 40(4):791–830, 1993.

[20] E. Cohen and N. Megiddo. Algorithms and complexity analysis for some flow problems. *Algorithmica* 11(3):320–340, 1994.

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

[22] S. I. Diatch and D. A. Spielman. Faster lossy generalized flow via interior point algorithms. *Proc. 40th ACM Symp. Theory Comput.*, 451–460, 2008.

[23] H. N. Djidjev and S. M. Venkatesan. Planarization of graphs embedded on surfaces. *Proc. 21st Workshop Graph-Theoretic Concepts Comput. Sci.*, 62–72, 1995. Lecture Notes Comput. Sci. 1017, Springer-Verlag.

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

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

[26] D. Eppstein. Dynamic generators of topologically embedded graphs. *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms*, 599–608, 2003.

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

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

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

[32] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms* 5(3):391–407, 1984.

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

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

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

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

[37] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1(2):169–197, 1981.

[38] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, 2nd edition. Algorithms and Combinatorics 2. Springer-Verlag, 1993.

[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 [70].

[40] R. Hassin. Maximum flow in (s, t) planar networks. *Inform. Proc. Lett.* 13:107, 1981.

[41] 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.

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

[43] 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.

[44] 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.

[45] 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.

[46] J. P. Hutchinson. On genus-reducing and planarizing algorithms for embedded graphs. *Graphs and Algorithms, Proc. AMS-IMS-SIAM Joint Summer Res. Conf.*, 19–26, 1989. Contemporary Mathematics 89, American Mathematical Society.

[47] J. P. Hutchinson and G. L. Miller. Deleting vertices to make graphs of positive genus planar. *Discrete Algorithms and Complexity Theory, Proceedings of the Japan-US Joint Seminar, Kyoto, Japan*, 81–98, 1987. Academic Press.

[48] 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.

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

[50] D. B. Johnson and S. M. Venkatesan. Partition of planar flow networks (preliminary version). *Proc. 24th IEEE Symp. Found. Comput. Sci.*, 259–264, 1983. IEEE Computer Society.

[51] S. N. Kabadi and Y. P. Aneja. ϵ-approximation minimization of convex functions in fixed dimension. *Oper. Res. Lett.* 18:171–176, 1996.

[52] S. N. Kabadi and Y. P. Aneja. Equivalence of ϵ-approximate separation and optimization in fixed dimensions. *Algorithmica* 29:582–594, 2001.

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

[54] 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.

[55] 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.

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

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

[58] W. S. Massey. *A basic course in algebraic topology*. Springer-Verlag, 1991.

[59] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach.* 30(4):852–865, 1983.

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

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

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

[63] J. R. Munkres. *Topology*, 2nd edition. Prentice-Hall, 2000.

[64] C. H. Norton, S. A. Plotkin, and É. Tardos. Using separation algorithms in fixed dimension. *J. Algorithms* 13(1):79–98, 1992.

[65] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Oper. Res.* 41(2):338–350, 1993.

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

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

[68] 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.

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

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

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

[72] 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.

[73] S. Toledo. Maximizing non-linear concave functions in fixed dimension. *Complexity in Numerical Optimization*, 429–447, 1993. World Scientific.

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

[75] S. M. Venkatesan. *Algorithms for network flows*. Ph.D. thesis, The Pennsylvania State University, 1983. Cited in [50].

[76] K. Weihe. Edge-disjoint (s,t)-paths in undirected planar graphs in linear time. *J. Algorithms* 23(1):121–138, 1997.

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