Optimizing Cycles and Bases:
Notes for an AMS Short Course on Computational Topology[*]

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

1 Overview

Many applications of computational topology require efficient descriptions of topological features. This lecture will survey recent algorithms and hardness results for computing several different descriptions of optimal cost. We consider the following specific problems for a given topological space X:

- Find the set of loops of minimum total length whose homotopy classes generate $\pi_1(X, x)$.
- Find the set of cycles of minimum total length whose homology classes generate $H_1(X)$.
- Find the set of p-cycles of minimum total weight whose homology classes generate $H_p(X)$.
- Find the shortest cycle in X that is freely homotopic to a given cycle.
- Find the shortest cycle in X that is homologous to a given cycle.
- Find the minimum-weight p-cycle in X that is homologous to a given p-cycles.

We consider these problems only over *combinatorial* spaces: simplicial, cubical, or CW complexes whose cells have non-negative *weights*. The output for each problem consists of one or more subcomplexes. In particular, for homotopy and first homology problems, we consider only paths, cycles, and subgraphs in the 1-skeleton; the weight (or 'length') of such a subgraph is the sum of the weights of its edges. Similarly, for questions about kth homology, the output is a subcomplex of the k-skeleton, or more generally a k-chain. Thus, the internal geometry of the faces is completely irrelevant (or more accurately, never defined).

For several of the problems, we further assume that the input space is a 2-manifold. Classical results of Markov [44, 45, 46] imply that most computational questions about homotopy are undecidable in non-manifold 2-complexes, or in manifolds of dimension 4 and higher. As we mention below, several questions about homology that can be decided in polynomial time for surfaces become NP-hard in non-manifold complexes or manifolds of higher dimension. We consider only orientable manifolds in these notes.

For optimization problems involving homology, the choice of coefficient ring is of critical importance. For computing optimal homology bases, there are efficient greedy algorithms for any coefficient *field*, but no results are known for $\mathbb{Z}$ or other rings without division. Localizing a particular $\mathbb{Z}_2$-homology class is NP-hard, even for combinatorial surfaces; however, there are polynomial-time algorithms to localize $\mathbb{R}$-homology classes, or to localize $\mathbb{Z}$-homology classes in manifolds.

[*]This work was partially supported by NSF grant CCF 09-15519.

2 Combinatorial Surfaces

Before discussing any algorithms, let us establish some standard nomenclature and notation for combinatorial surfaces, mostly inherited from topological graph theory [30, 41, 48].

An ***embedding*** of an undirected graph G on a 2-manifold Σ maps vertices to distinct points and edges to interior-disjoint curves. The *faces* of the embedding are maximal connected subsets of Σ that are disjoint from the image of the graph. An embedding is ***cellular*** if each of its faces is homeomorphic to the plane. Any cellular embedding can be represented combinatorially by a ***rotation system***, which is an encoding of the counterclockwise order of edges incident to each vertex. ***Euler's formula*** implies that any cellularly embedded graph with n vertices, n_1 edges, and n_2 faces lies on a surface with Euler characteristic $\chi = n - n_1 + n_2 = 2 - 2g$, which implies that $n_1 = O(n+g)$ and $n_2 = O(n+g)$. To simplify our notation, we consider only cellular embeddings of genus $g = O(n)$, so that the overall complexity of the embedding is $O(n)$.

Any undirected graph G embedded on a surface Σ has an associated ***dual graph G^****, defined as follows. The dual graph G^* has a vertex f^* for each face f of G. Two vertices in G^* are joined by an edge e^* if and only if the corresponding faces of G are separated by an edge e. The dual graph G^* has a natural cellular embedding in the same surface Σ; each face v^* of this embedding corresponds to a vertex v of the primal graph G. Duality is an involution: $(G^*)^* = G$. For any subgraph H of G, we abuse notation by letting H^* denote the corresponding subgraph of the dual graph G^*.

Finally, a ***combinatorial surface*** is an abstract surface Σ together with a cellularly embedded graph G with positively weighted edges. The only paths or cycles we consider in combinatorial surfaces are walks in its graph; the length of any such walk is the sum of its edge weights, counted with appropriate multiplicity.

3 Optimal Homotopy Bases

We now consider the problem of computing a set of loops of minimum total length whose homotopy classes generate the fundamental group $\pi_1(\Sigma, x)$ of some space Σ.

We first describe an algorithm of Erickson and Whittlesey [26] for the special case of combinatorial surfaces. Their algorithm is a modification of a simple linear-time algorithm to compute an *arbitrary* homotopy basis, due to Eppstein [23]. Their algorithm actually constructs a *system of loops*, that is, a set of $2g$ loops $\ell_1, \dots, \ell_{2g}$ with a common basepoint, such that $\Sigma \setminus (\ell_1 \cup \cdots \cup \ell_{2g})$ is a topological disk. Any system of loops with basepoint x is also a basis for the fundamental group $\pi_1(\Sigma, x)$. Not every homotopy basis is a system of loops, however; homotopy bases may contain (self-)intersections that cannot be removed by homotopy.

A *spanning tree* (or *maximal tree*) of G is any connected acyclic subgraph of G that contains every vertex of G. We call a subgraph C a *spanning cotree* of an embedded graph G if the dual subgraph C^* is a spanning tree of the dual graph G^*. A ***tree-cotree decomposition*** (T, L, C) of G is a partition of its edges into three disjoint subsets: a spanning tree T, a spanning cotree C, and the leftover edges $L = G \setminus (T \cup C)$. The spanning tree T and spanning cotree C can be chosen arbitrarily, as long as they do not share an edge. If G is embedded on an orientable surface of genus g, Euler's formula implies that in any tree-cotree decomposition (T, L, C) of G, the set L contains exactly $2g$ edges. Indeed, tree-cotree decompositions of planar graphs (with $L = \varnothing$) were introduced by von Staudt [51] in his proof of Euler's formula.

Given any embedded graph G with n vertices, it is easy to construct a tree-cotree decomposition (T, L, C) in $O(n)$ time as follows. Construct a spanning tree T of G using depth-first search; then construct a spanning tree C^* of the dual subgraph $(G \setminus T)^*$, again by depth-first search; and finally, let $L = G \setminus (T \cup C)$.

Finally, given a tree-cotree decomposition, we can construct basis for $\pi_1(\Sigma, x)$ as follows. For each edge $uv \in L$, let $\ell(uv)$ be the loop formed by concatenating the unique path in T from x to u, the edge uv, and the unique path in T from v back to x. Eppstein [23] proves that the set $\{\ell(e) \mid e \in L\}$ is a system of loops and therefore a homotopy basis. Each loop $\ell(e)$ can be computed from T and e in $O(1)$ time per edge. Thus, the overall time to complete these $2g$ loops is $O(n + k_{out})$, where k_{out} is the complexity of the output. Each loop $\ell(e)$ traverses each edge of the combinatorial surface at most twice, so $k_{out} = O(ng)$; this bound is tight in the worst case.

To construct the *shortest* homotopy basis with a given basepoint x, Erickson and Whittlesey [26] modify Eppstein's algorithm by choosing a particular *greedy* tree-cotree decomposition (T, L, C). In this decomposition, T is a single-source shortest path tree rooted at the basepoint x, and C^* is the maximum spanning tree of G^*, where the weight of each dual edge e^* is the length of the corresponding primal loop $\ell(e)$. Standard textbook algorithms to compute shortest-path trees and minimum spanning trees let us construct this tree-cotree decomposition in $O(n \log n)$ time. If $g = O(n^{1-\varepsilon})$ for any constant $\varepsilon > 0$, both the shortest-path tree and the dual maximum spanning tree can be computed in $O(n)$ time [33, 43].

Erickson and Whittlesey [26] prove that the greedy tree-cotree decomposition defines the shortest system of loops using a complex exchange argument. A much simpler proof was more recently given by Colin de Verdière [15], who also described a similar greedy algorithm to compute the minimum-length *cut graph* with a prescribed set of vertices. A cut graph is a subgraph X of the graph G such that $\Sigma \setminus X$ is an open disk; computing the shortest cut graph with no prescribed vertex set is NP-hard [24].

If no basepoint is specified in advance, we can compute the shortest homotopy basis in $O(n^2 \log n + k_{out})$ time by running the previous algorithm at every vertex.

4 Optimal Homology Bases

Any set of loops that define a basis for the fundamental group also immediately define a basis for the first homology group $H_1(\Sigma)$, over any coefficient ring. Thus, Eppstein's linear-time tree-cotree algorithm can also be used to construct a first homology basis for any combinatorial surface in $O(n + k_{out}) = O(ng)$ time. (We can reduce the output size slightly by replacing each loop $\ell(e)$ with the unique cycle in the graph $T \cup \{e\}$, but the output size is still $\Theta(gn)$ in the worst case.) However, the *minimum-length* first homology basis is not generally consistent with any tree-cotree decomposition.

Erickson and Whittlesey [26] describe an efficient algorithm to compute the shortest homology basis with respect to any coefficient *field* R, generalizing an earlier algorithm of Horton [37] to compute the shortest cycle basis of an edge-weighted graph. Their algorithm is simplified by the following observation of Dey *et al.* [19]: Every generator in the optimal homology basis is also a generator in the optimal *homotopy* basis for some basepoint. Thus, in $O(n^2 \log n + gn^2)$ time, we can compute a set of $O(gn)$ candidate cycles that must include the optimal homology basis. For each candidate cycle, we can also compute its length and a vector of length $2g$ encoding its homology class, in total time $O(gn^2)$.

We are now faced with the following *matroid optimization* problem: Given a collection of $O(gn)$ vectors, each with a positive weight, find a subset of minimum total weight that generates the *vector space* R^{2g}. This problem can be solved by a greedy algorithm, similar to Kruskal's classical algorithm to compute minimum spanning trees [40]. Starting with an empty basis, consider the vectors one at a time in order of increasing weight; whenever we encounter a vector that is linearly independent of the vectors already in the basis, add it to the basis. This portion of the algorithm can be implemented to run in $O(gn \log n + g^3 n)$ time. The total time to compute an optimal homology basis is $O(n^2 \log n + gn^2 + g^3 n)$; for surfaces of genus $O(n^{1-\varepsilon})$, the $O(n^2 \log n)$ term can be removed from the running time.

Dey *et al.* [19] and Chen and Freedman [14] extend this greedy strategy to compute optimal bases for the first homology group (again, over any coefficient field) of an arbitrary simplicial complex in polynomial time using persistent homology [20, 21, 53]. Chen and Friedman generalize the algorithm further to compute minimum-cost bases for higher-dimensional homology groups, where the cost of a single generator is the *radius* of the smallest ball that contains it. It is more natural to define the cost of a generator to be the number (or total weight) of the simplices it contains, but finding the minimum-cost basis for $H_p(\Sigma)$ with this measure, for any $p > 1$, is NP-hard [12].

The restriction to coefficient *fields* is unfortunately necessary for all of these algorithms. For coefficient rings without division, such as the integers $\mathbb{Z}$, homology groups are not vector spaces, and thus a maximal independent set of homology classes is not necessarily a basis. Gortler and Thurston (personal communication) have shown that the greedy algorithm can return a set of $2g$ cycles in independent $\mathbb{Z}$-homology classes but do *not* generate the first homology group, even for a combinatorial surface of genus 2. No algorithm or hardness result is currently known for computing minimal $\mathbb{Z}$-homology bases.

5 Shortest Homotopic Paths and Cycles

The ***shortest homotopic path*** problem asks, given a path π in some space Σ, to compute the shortest path in Σ that is homotopic to π. The definition of the universal cover Σ implies that the shortest path homotopic to π is the projection of the shortest path in $\tilde{\Sigma}$ between the endpoints of some lift $\tilde{\pi}$ of π. This characterization does not give us an actual algorithm, because the universal cover has unbounded complexity. Algorithms to solve the homotopic shortest path problem first construct a small finite portion of the $\tilde{\Sigma}$ and then compute a shortest path between the endpoints of $\tilde{\pi}$ in this relevant region.

The first efficient algorithm for this problem was described by Hershberger and Snoeyink [34], who considered the problem in simple Euclidean polygons with holes. Their algorithm proceeds in five stages. (1) In a preprocessing phase, compute a triangulation of the polygon P, and assign each diagonal in the triangulation a unique label. (2) Compute the sequence of diagonals crossed by a point moving along the input path π. (3) Reduce this crossing sequence by repeatedly removing adjacent pairs of the same diagonal label. (4) Construct the *sleeve* of triangles containing every path whose crossing sequence is the reduced crossing sequence of π. The sleeve is a topological disk, but not necessarily a simple Euclidean polygon; rather, it is a subset of the universal cover of the triangulated polygon P. (5) Compute the shortest path in the sleeve between the endpoints of π using the *funnel* algorithm of Chazelle [11] and Lee and Preparata [42]. The overall running time of Hershberger and Snoeyink's algorithm is $O(n \log n + nk_{in})$, where k_{in} is the number of segments in the input path. A similar algorithm constructs the shortest cycle freely homotopic to a given cycle in the same time bound. Extensions of this algorithm were developed by Bespamyatnikh [3, 4], Cabello *et al.* [7], and Efrat *et al.* [22].

Colin de Verdière and Erickson follow a similar strategy for computing shortest homotopic paths in combinatorial surfaces [17]. Instead of a triangulation, their preprocessing algorithm constructs a *tight octagonal decomposition* of the given surface Σ in $O(gn \log n)$ time; this is a collection of $O(g)$ cycles, each as short as possible in its homotopy class, that decompose the surface into octagons meeting four at a vertex. The universal cover of the octagonal decomposition is combinatorially isomorphic to a tiling of the hyperbolic plane by right-angled octagons. In particular, each cycle in the decomposition lifts to an infinite geodesic in $\tilde{\Sigma}$, which crosses any other shortest path in $\tilde{\Sigma}$ at most once. Given a path π in Σ, the algorithm exploits this regular structure to construct a finite portion of the infinite octagonal tiling containing the shortest path between the endpoints of $\tilde{\pi}$, then fills each relevant octagon with the corresponding planar portion of the input graph, and finally computes the shortest path in the resulting finite portion of the universal cover $\tilde{\Sigma}$. Assuming $k \geq 2$, the query phase of the algorithm runs in $O(gnk_{in})$ time, where k_{in} is the number of edges in the input path. A slightly more complicated algorithm constructs the shortest cycle freely homotopic to a given cycle in $O(gnk_{in}\log(nk_{in}))$ time.

6 Optimizing in $\mathbb{Z}_2$-Homology Classes

The complexity of finding optimal representatives in a given *homology* class depends crucially on the choice of coefficient ring. Somewhat surprisingly, optimization over finite fields turns out to be significantly harder than optimization over either the reals or the integers.

We first consider homology over $\mathbb{Z}_2$. With this coefficient field, a 1-chain in a combinatorial surface is a *subgraph* of its graph; a 1-cycle is a subgraph in which every vertex has even degree; a *1-boundary* is the boundary of the union of a subset of faces; and two even subgraphs are homologous if and only if their symmetric difference is a boundary subgraph. Euler's formula immediately implies that the first homology group is the vector space $\mathbb{Z}_2^{2g}$.

Unfortunately, any reasonable variant of localizing a $\mathbb{Z}_2$-homology class is NP-hard, even in combinatorial surfaces. An argument of Chambers *et al.* [8] can be modified to show that finding either the shortest *cycle* or the shortest *closed walk* $\mathbb{Z}_2$-homologous to a given cycle is NP-hard, by reduction from the Hamiltonian cycle problem in planar grid graphs [38]. Later Chambers, Erickson, and Nayyeri [10] proved that finding the minimum-length *even subgraph* $\mathbb{Z}_2$-homologous to a given cycle is NP-hard, by reduction from MinCut in graphs with negative edges [47]. In more general simplicial complexes, Chen and Friedman [13] prove that approximating the shortest *even subgraph* in a given homology class to within any constant factor is NP-hard, even when the rank of the first homology group is 1, by reduction from the nearest codeword problem [2].

However, for surfaces of constant genus, we can find minimal representatives in any $\mathbb{Z}_2$-homology class in $O(n\log n)$ time. Specifically, Erickson and Nayyeri [25] describe an algorithm to compute either the shortest *closed walk* or the shortest *even subgraph* in a given $\mathbb{Z}_2$-homology class in $2^{O(g)}n\log n$ time, simplifying and improving an earlier algorithm by Chambers *et al.* [8] that runs in $g^{O(g)}n\log n$ time.

Erickson and Nayyeri's algorithm first constructs the ***$\mathbb{Z}_2$-homology cover*** $\overline{\Sigma}$ of the combinatorial surface Σ; this is the unique connected covering space of Σ whose group of deck transformations is $H_1(\Sigma;\mathbb{Z}_2)\cong\mathbb{Z}_2^{2g}$. More concretely, this covering space can be constructed as follows. Let $\ell_1,\dots,\ell_{2g}$ be any systems of loops for Σ, such as the one constructed from a tree-cotree decomposition by Eppstein's algorithm. The surface $D:=\Sigma\setminus(\ell_1\cup\cdots\cup\ell_{2g})$ is a topological disk; each loop ℓ_i appears on the boundary of D as two boundary segments ℓ_i^+ and ℓ_i^-. For each homology class $h\in(\mathbb{Z}_2)^{2g}$, we create a disjoint copy (D,h) of D; for each index i, let (ℓ_i^+,h) and (ℓ_i^-,h) denote the copies of ℓ_i^+ and ℓ_i^- in the disk (D,h). For each index i, let b_i denote the $2g$-bit vector whose ith bit is equal 1 and whose other $2g-1$ bits are all equal to 0. The $\mathbb{Z}_2$-homology cover $\overline{\Sigma}$ is constructed by gluing the copies of D together by identifying boundary paths (ℓ_i^+,h) and $(\ell_i^-,h\oplus b_i)$, for every index i and every homology class h. The resulting combinatorial surface has $\overline{n}=2^{2g}n$ vertices, each labeled by a pair (v,h) for some vertex v in Σ and some homology class h, and genus $\overline{g}=2^{2g}(g-1)+1$.

Let ω be a closed walk in Σ, and let $[\omega]$ denote its $\mathbb{Z}_2$-homology class. the walk ω is the projection of a path in $\overline{\Sigma}$ from $(v,0)$ to $(v,[\omega])$, where v is any vertex in ω. Thus, the *shortest* closed walk in homology class h is the projection of the shortest path in $\overline{\Sigma}$ from some vertex $(v,0)$ to the corresponding vertex (v,h). This shortest path can be found in $O(n\cdot\overline{n})=2^{O(g)}n^2$ time by computing a shortest-path tree at every vertex $(v,0)$. Erickson and Nayerri [25] reduce the running time to $O(\overline{gn}\log\overline{n})=2^{O(g)}n\log n$ using more complex data structures [6].

The minimum-weight even subgraph in any $\mathbb{Z}_2$-homology class has at most g connected components, each of which is (the image of) the shortest closed walk in its own $\mathbb{Z}_2$-homology class. To find the optimal even subgraph, Erickson and Nayerri first compute the shortest closed walk in *every* $\mathbb{Z}_2$-homology class, in $2^{O(g)} n \log n$ time, and then assemble the components using a simple dynamic programming algorithm in $2^{O(g)}$ additional time. Specifically, let $C(h,k)$ denote the minimum total weight of any set of at most k closed walks whose homology classes sum to h. This function obeys the recurrence $C(h,k) = \min_{h'} \left(C(h',k-1) + C(h \oplus h', 1)\right)$, where h' ranges over all homology classes. The base cases are $C(0,k) = 0$ and $C(h,1)$, which has already been computed for each h.

7 Optimizing in $\mathbb{R}$- and $\mathbb{Z}$-Homology Classes

We now switch to finding minimum-cost representatives in real and integer homology classes. For homology over the reals, we can describe polynomial-time algorithms for arbitrary weighted simplicial complexes via linear programming. Under appropriate conditions, the resulting linear programs actually have integral solutions, and thus can be used to find optimal representatives in *integer* homology classes.

Fix a simplicial complex X and a non-negative integer $p \geq 0$. Let m and n respectively denote the number of p-simplices and $(p+1)$-simplices in X. In real simplicial homology, a *p-chain* is a formal linear combination of oriented p-simplices in X, which we identify with a real vector $\boldsymbol{c} = (c_1, c_2, \dots, c_n) in \mathbb{R}^m$. Two p-chains are homologous if their difference is the boundary of some $(p+1)$-chain. Note that we are *not* restricting our homology classes to chains with empty boundaries.

Suppose each *unoriented* p-simplex in X has a non-negative real weight; we represent these weights by another vector $\boldsymbol{w} = (w_1, \dots, w_n) \in \mathbb{R}^m$. The *weight* of a p-chain $\boldsymbol{c}$ is defined as the sum $w(\boldsymbol{c}) := \sum_i |c_i| w_i$. Given a real p-chain $\boldsymbol{c}$ as input, our goal is to another p-chain $\boldsymbol{x}$ of minimum weight $w(\boldsymbol{x})$ in the real-homology class of $\boldsymbol{c}$. Dey, Hirani, and Krishnamoorthy [18] observe that this problem has the following linear programming formulation:

$$\begin{array}{ll} \text{minimize} & \sum_i w_i \cdot (x_i^+ + x_i^-) \\ \text{subject to} & \boldsymbol{x}^+ - \boldsymbol{x}^- = \boldsymbol{c} + [\partial_{p+1}](\boldsymbol{y}^+ - \boldsymbol{y}^-) \\ & \boldsymbol{x}^+, \boldsymbol{x}^-, \boldsymbol{y}^+, \boldsymbol{y}^- \geq 0 \end{array} \tag{LP1}$$

Here, the vector $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^- \in \mathbb{R}^m$ is the target p-chain, the vector $\boldsymbol{y} = \boldsymbol{y}^+ - \boldsymbol{y}^- \in \mathbb{R}^n$ is a $(p+1)$-chain, and $[\partial_{p+1}]$ is the $m \times n$ matrix representing the boundary operator $\partial_{p+1} \colon \mathbb{R}^n \to \mathbb{R}^m$.

Finding the minimum-cost representative in a given *integer* homology class is the solution to an *integer* program, obtained by adding the constraints $\boldsymbol{x}^+, \boldsymbol{x}^- \in \mathbb{Z}^m$ and $\boldsymbol{y}^+, \boldsymbol{y}^- \in \mathbb{Z}^n$ to the linear program (LP1). Unlike linear programming, which can be solved by strongly polynomial-time algorithms integer programming is in general NP-hard.

A matrix is *totally unimodular* if every square minor has determinant 0, 1, or -1. Cramer's rule implies that for any totally unimodular matrix A and any integer vector $\boldsymbol{b}$, every vertex of the polyhedron $\{\boldsymbol{x} \mid A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\}$ is integral [36]. Thus, any linear program defined by a totally unimodular constraint matrix and an integral constraint vector has an integral solution. Dey, Hirani, and Krishnamoorthy [18] prove that the boundary matrix $[\partial_{p+1}]$ is totally unimodular if and only if certain relative pth homology groups of subcomplexes of X are torsion-free. In particular, any orientable compact manifold satisfies this condition, as does any finite simplicial d-complex embedded in $\mathbb{R}^d$. Thus, for complexes with totally unimodular boundary matrices, the linear program (LP1) automatically has integral solutions, and thus can be used to optimize within any integer homology class in polynomial time.

Moreover, for any chain $\boldsymbol{c} \in \{-1, 0, 1\}^n$, we can obtain the minimum-cost homologous chain $\boldsymbol{x} \in \{-1, 0, 1\}^n$ of minimum total weight in the $\mathbb{Z}$-homology class of $\boldsymbol{c}$ by adding constraints $\boldsymbol{x}^+, \boldsymbol{x}^- \leq 1$ to (LP1) and solving the resulting *linear* program.

Dey, Hirani, and Krishnamoorthy's result generalizes an earlier result of Sullivan [52], who described a polynomial-time algorithm to find the minimum-weight p-chain in a given $\mathbb{Z}$-homology class, in a $(p+1)$-*manifold* cell complex. Sullivan observed that the dual of linear program (LP1) describes the following *minimum-cost circulation* problem. Let G be the 1-skeleton of the dual cell complex; conveniently, this graph has n vertices and m edges. We interpret G as a symmetric directed graph $\vec{G} = (V, \vec{E})$ with the same adjacency matrix. Assign each directed edge $\vec{e}$ of $\vec{G}$ a *capacity* $c(\vec{e})$ equal to the weight of the corresponding *unoriented* primal p-cell, and a cost $\$(\vec{e})$ equal to the coefficient of the corresponding *oriented* p-cell in the input chain $\boldsymbol{c}$. We emphasize here that the capacity function is symmetric, but the cost function is antisymmetric. Sullivan applies a primal-dual algorithm [] to solve both the primal and dual liner programs in time polynomial in n, m, and $\|\boldsymbol{c}\|_1$.

For complexes that do not satisfy the total unimodularity condition, it is not known whether minimum-cost z-homologous chains can be computed in polynomial time, or if the problem is NP-hard.

8 Maximum flows in surface graphs

The minimum-cost circulation problem is a generalization of the classical *maximum flow* problem, which is described as follows. The input is a directed graph $G = (V, E)$, with two specified vertices s and t, and a non-negative *capacity* function $c\colon E \to \mathbb{R}^+$. An (s,t)*-flow* in G is a function $\phi\colon E \to \mathbb{R}$ such that $\sum_u \phi(u{\to}v) = \sum_w \phi(v{\to}w)$ for every vertex v *except* s and t; this equation is usually called the *conservation constraint*. A flow ϕ is *feasible* if $0 \le \phi(e) \le c(e)$ for every edge e. The *value* of ϕ is $|f| := \sum_w \phi(s{\to}w) - \sum_u \phi(u{\to}s)$, the total net flow out of s. The maximum flow problem asks to compute a feasible flow in G with maximum value. For graphs with n vertices and $O(n)$ edges, the fastest maximum-flow algorithms run in $O(n^2 \log n)$ time [29, 27, 35, 50] or in $O(n^{3/2} \log n \log U)$ time, where U is an upper bound on the edge capacities [28].

We close this lecture by discussing a recent algorithm by Chambers, Erickson, and Nayerri [9] to compute maximum flows in surface-embedded graphs. For surfaces of constant genus, this algorithm is faster than the best maximum-flow algorithms for sparse graphs by roughly a factor of $O(\sqrt{n})$. The algorithm can be generalized to solve the LP-dual problem of localizing a real or integer homology class in the same asymptotic running time.

Let G be a graph that is cellularly embedding on a 2-manifold Σ. In more topological language, an (s,t)-flow ϕ in G is any real 1-chain whose boundary is the 0-chain $|\phi|(t-s)$. A *circulation* is a flow with value 0, or in other words, a real 1-cycle. Two flows in G are homologous if their difference is a formal sum of contractible cycles, also called a *boundary circulation*. The set of all homology classes of flows is the relative homology group $H_1(\Sigma, \{s,t\}) \cong \mathbb{R}^{2g+1}$. A homology class of flows is *feasible* if it contains a feasible flow.

Flow homology can also be characterized in terms of the dual graph G^* as follows. A *cocycle* λ in G is any subgraph whose dual λ^* is a directed cycle in G^*. The capacity $c(\lambda)$ of a cocycle λ is just the sum of the capacities of its edges. Similarly, for any flow ϕ, we let $\phi(\lambda) = \sum_{e\in\lambda} \phi(e)$. It is easy to prove that two flows ϕ and ψ are homologous if and only if $\phi(\lambda) = \psi(\lambda)$ for every cocycle λ; in particular, any two homologous flows have the same value.

Chambers, Erickson, and Nayerri [9] prove that the homology class of a flow ϕ is feasible if and only if $\phi(\lambda) \le c(\lambda)$ for every cocycle λ. This condition can be tested by running a single-source shortest path algorithm in the dual graph G^* with appropriate (possibly negative) edge weights. Chambers *et al.* describe a suitable shortest-path algorithm that runs in $O(g^2 n \log^2 n)$ time, generalizes an earlier algorithm for planar graphs by Klein, Mozes, and Weimann [39, 49]. This algorithm returns either a feasible flow homologous to ϕ or a cocycle that is over-saturated by ϕ.

For any cocycle λ, the inequality $\phi(\lambda) \le c(\lambda)$ imposes a *linear* constraint on the homology class of ϕ. Thus, the set of all feasible homology classes is a convex polyhedron in $\mathbb{R}^{2g+1}$, and finding the feasible homology class of maximum value is a linear programming problem. Unfortunately, this linear program appears to have $n^{O(g)}$ non-redundant constraints, so we cannot solve it directly. However, the linear program can be solved using implicit methods that apply the dual shortest-path algorithm as a membership/separation oracle. Specifically, if the edge capacities are integers less than C, the central-cut ellipsoid method [31, 32] solves the linear program in $O(g^8 n \log^2 n \log^2 C)$ time. Alternatively, for arbitrary capacities, multidimensional parametric search [1] gives us a combinatorial algorithm that runs in $g^{O(g)} n^{3/2}$ time. Both time bounds are faster by roughly a factor of $\sqrt{n}$ than the fastest algorithms for general sparse graphs when the genus is constant.

References

[1] P. K. Agarwal, M. Sharir, and S. Toledo. An efficient multi-dimensional 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⟩.

[2] S. Arora, L. Babai, J. Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.* 54(2):317–331, 1997.

[3] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms* 49(2):284–303, 2003.

[4] S. Bespamyatnikh. Encoding homotopy of paths in the plane. *Proc. LATIN 2004: Theoretical Infomatics*, 329–338, 2004. Lecture Notes Comput. Sci. 2976, Springer-Verlag.

[5] 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.

[6] S. Cabello, E. W. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs. Full version of [5], in preparation.

[7] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. *Discrete Comput. Geom.* 31:61–81, 2004.

[8] 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.

[9] E. W. Chambers, J. Erickson, and A. Nayyeri. Homology flows, cohomology cuts. *Proc. 42nd Ann. ACM Symp. Theory Comput.*, 273–282, 2009. Full version available at http://www.cs.uiuc.edu/~jeffe/pubs/surflow.html.

[10] E. W. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. *Proc. 25th Ann. ACM Symp. Comput. Geom.*, 377–385, 2009.

[11] B. Chazelle. A theorem on polygon cutting with applications. *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, 339–349, 1982.

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

[13] C. Chen and D. Freedman. Hardness results for homology localization. *Proc. 21st ACM-SIAM Symp. Discrete Algorithms*, 1594–1604, 2010.

[14] C. Chen and D. Freedman. Measuring and computing natural generators for homology groups. *Comput. Geom. Theory Appl.* 43(2):169–181, 2010.

[15] É. Colin de Verdière. Shortest cut graph of a surface with prescribed vertex set. *Proc. 18th Ann. Europ. Symp. Algorithms*, 100–111, 2010. Lecture Notes Comput. Sci. 6347.

[16] 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.

[17] É. Colin de Verdière and J. Erickson. Tightening non-simple paths and cycles on surfaces. *SIAM J. Comput.* to appear, 2011. Full version of [16].

[18] T. K. Dey, A. N. Hirani, and B. Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *Proc. 42nd Ann. ACM Symp. Theory Comput.*, 221–230, 2010.

[19] T. K. Dey, J. Sun, and Y. Wang. Approximating loops in a shortest homology basis from point data. *Proc. 26th Ann. ACM Symp. Comput. Geom.*, 166–175, 2010.

[20] H. Edelsbrunner and J. Harer. Persistent homology—a survey. *Essays on Discrete and Computational Geometry: Twenty Years Later*, 257–282, 2008. Contemporary Mathematics 453, American Mathematical Society.

[21] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.* 28:511–533, 2002.

[22] A. Efrat, S. G. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. *Comput. Geom. Theory Appl.* 35(3):162–172, 2006.

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

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

[25] J. Erickson and A. Nayyeri. Shortest homologous cycles and minimum cuts via homology covers. *Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, p. to appear, 2011. ⟨http://www.cs.uiuc.edu/~jeffe/homcover.html⟩.

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

[27] A. V. Goldberg, M. D. Grigoriadis, and R. E. Tarjan. Use of dynamic trees in a network simplex algorithm for the maximum flow problem. *Math. Program.* 50:277–290, 1991.

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

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

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

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

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

[33] 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.

[34] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.* 4:63–98, 1994.

[35] D. S. Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum flow problem. *Oper. Res.* 58(4):992–1009, 2008.

[36] A. J. Hoffman and J. B. Kruskal. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems*, 223–246, 1956. Annals of Mathematical Studies 38, Princeton University Press.

[37] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.* 16:358–366, 1987.

[38] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.* 11:676–686, 1982.

[39] 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. *ACM Trans. Algorithms* 6(2):article 30, 2010.

[40] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.* 7(1):48–50, 1956.

[41] S. K. Lando and A. K. Zvonkin. *Graphs on Surfaces and Their Applications*. Low-Dimensional Topology II. Springer-Verlag, 2004.

[42] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks* 14:393–410, 1984.

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

[44] A. A. Markov. Impossibility of algorithms for recognizing some properties of associative systems. *Dokl. Akad. Nauk SSSR* 77:953–956, 1951. In Russian.

[45] A. A. Markov. The insolubility of the problem of homeomorphy. *Dokl. Akad. Nauk SSSR* 121:218–220, 1958.

[46] A. A. Markov. Unsolvability of certain problems in topology. *Dokl. Akad. Nauk SSSR* 123:978–980, 1958.

[47] 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.

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

[49] S. Mozes and C. Wolff-Nilsen. Shortest paths in planar graphs with real lengths in $O(n\log^2 n/\log\log n)$ time. *Proc. 18th Ann. Europ. Symp. Algorithms*, 206–217, 2010. Lecture Notes Comput. Sci. 6347, Springer-Verlag.

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

[51] K. G. C. von Staudt. *Geometrie der Lage [Geometry of the Plane]*. Verlag von Bauer and Rapse (Julius Merz), Nürnberg, 1847.

[52] J. M. Sullivan. *A Crystalline Approximation Theorem for Hypersurfaces*. Ph.D. thesis, Princeton Univ., October 1990. ⟨http://torus.math.uiuc.edu/jms/Papers/thesis/thesis.pdf⟩.

[53] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.* 33(2):249–274, 2005.