Counting and Sampling Minimum Cuts in Genus g Graphs

Erin W. Chambers[*] Kyle Fox[†] Amir Nayyeri[‡]

December 1, 2012

Abstract

Let G be a directed graph with n vertices embedded on an orientable surface of genus g with two designated vertices s and t. We show that counting the number of minimum (s,t)-cuts in G is fixed parameter tractable in g. Specially, we give a $2^{O(g)}n^2 + \min\{n^2 \log n, g^{O(g)}n^{3/2}\}$ time algorithm for this problem. Our algorithm requires counting sets of cycles in a particular integer homology class. That we can count these cycles is an interesting result in itself as there are few prior results that are fixed parameter tractable and deal directly with integer homology. We also describe an algorithm which, after running our algorithm to count the number of cuts once, can sample repeatedly for a minimum cut in $O(g^2 n)$ time per sample.

[*]Department of Computer Science and Mathematics, Saint Louis University; echambe5@slu.edu. Research supported in part by the National Science Foundation under Grant No. CCF 1054779.

[†]Department of Computer Science, University of Illinois, Urbana-Champaign; kylefox2@illinois.edu. Research supported in part by the Department of Energy Office of Science Graduate Fellowship Program (DOE SCGF), made possible in part by the American Recovery and Reinvestment Act of 2009, administered by ORISE-ORAU under contract no. DE-AC05-06OR23100.

[‡]Department of Computer Science, University of Illinois, Urbana-Champaign; nayyeri2@illinois.edu. Research supported in part by the National Science Foundation under Grant No. CCF 09-15519.

1 Introduction

Given a weighted directed graph $G = (V, E)$ on an orientable surface Σ of genus g with two vertices $s, t \in V$, we consider the problem of counting the minimum (s, t)-cuts of G. This problem is #P-complete, and can be reduced to the problem of counting maximal anitchains in a poset [50]. Ball and Provan [50] first considered the problem of counting minimum cuts and gave an algorithm to compute the number of minimum (s, t)-cuts in an (s, t)-planar graph. Later, Bezáková and Friedlander [2] generalized the algorithm for arbitrary locations of s and t in a planar graph.

Counting the number of minimum cuts is of interest due to connections with many other areas. For example, the number of minimum cuts is closely related to the probabilistic connectedness of a stochastic graph, where each edge may fail with a certain probability [1], and so it is fundamentally important in several network reliability problems [1, 15, 35, 47].

In addition, cuts have strong connections to problems from computer vision. In image segmentation, the image is represented as a (generally planar) graph on the pixels with edges connecting neighboring pixels weighted according to how similar the pixels are; a minimum cut between two locations corresponds to a good segmentation of the original image [7]. Being able to count minimum cuts is closely related to sampling such cuts [34], implying our ability to count minimum cuts allows us to sample from the collection of high quality segmentations of an image.

For graphs on surfaces, good segmentation algorithms are key for problems such as texture mapping, metamorphosis, simplification, and compression; see [52] for a recent survey of techniques and applications. Many of these algorithms wish to minimize stretch, so that patches of the surface are separated via small separators and local distances within each patch stay close to the original distance. Minimum cuts have strong potential here; by looking at flow along the mesh, separating along these cuts will again result in a good segmentation. Even an algorithm with large dependence on the genus is useful, because many meshing algorithms attempt to keep the genus small as a way of reducing noise in the mesh.

1.1 Flows and cuts in restrictive graph families

Along with counting minimum cuts, *computing* minimum cuts (and maximum flows) is a fundamental problem in combinatorial optimization. Because of their utility, much effort has been spent on finding faster algorithms in restricted but useful settings.

Planar graphs are a natural family of graphs to consider, both because they arise naturally in many settings and because the extra structure in planar graphs can be exploited to get faster algorithms. Examples of problems aided by planarity include minimum spanning trees [42, 49]; single-source shortest paths [13, 30, 38, 41, 45, 53]; multiple-source shortest paths [8, 37]; replacement paths [22, 56]; graph and subgraph isomorphism [19, 20, 26, 31, 43]; and approximation of several NP-hard problems [3–5, 17, 20].

Of course, much work is also dedicated to computing cuts and flows in planar graphs, both undirected [10, 24, 28, 32, 33, 40, 51] and directed [6, 54]. The current best algorithms run in $O(n \log \log n)$ time in undirected graphs [33] and $O(n \log n)$ time in directed graphs [6], where both considerably improve running times for arbitrary graphs.

Because of this success in the planar setting, there has been a considerable amount of recent work on computing flows and cuts in generalizations of planar graphs, which include surface embedded graphs [13, 14, 21, 23], minor-free families [12], and graphs with bounded treewidth [27]. Unfortunately, some properties of planar graphs that the planar flow and cut algorithms take for granted such as every cycle being separating do not apply in these more general settings. The lack of structure not only slows progress toward creating new algorithms, but it introduces surprising dependencies in running time. For example, the current best algorithms for computing a minimum cut in an undirected surface graph run in $2^{O(g)} n \log n$ time [23] and $g^{O(g)} n \log \log n$ time [33].

1.2 Our contributions

In this paper, we describe a near-quadratic time algorithm to compute the number of minimum (s,t)-cuts for a weighted graph embedded on a surface of constant genus. Specifically, our algorithm runs in $2^{O(g)} n^2 + n^2 \log n$ time assuming a *cellular* embedding is given onto a surface of genus g. If no embedding is given, then we can compute one in $2^{O(g)} n$ time [36, 55]. The $O(n^2 \log n)$ time dependency is for computing a maximum flow using Goldberg and Tarjan's algorithm [25] for arbitrary sparse graphs. Alternatively, we can use Chambers *et al.*'s algorithm [13] to compute a maximum flow in a surface embedded graph to obtain a $2^{O(g)} n^2 + g^{O(g)} n^{3/2}$ time algorithm for counting minimum cuts. Since counting the number of cuts is generally #P-complete [50], finding a fixed parameter tractable algorithm to compute the number of cuts for a surface embedded graph represents a significant and perhaps optimal improvement in known results for a large family of graphs. Our algorithm requires only a few simple assumptions on the input graph; every edge has positive capacity, and there exists a directed path from s to every vertex in G and a directed path from every vertex in G to t.

Our approach uses a connection between cuts and (co-)homology in a non-trivial way, as well as generalizing the tools from [2] to more general surfaces. As in [2], our algorithm first reduces the problem of counting minimum cuts in G to the problem of counting *forward* (t,s)*-cuts* in a directed acyclic graph. Our algorithm then used a connection between forward (t,s)-cuts and circulations of a certain homology class in the dual graph. Both reductions are briefly described in Section 3; details are deferred to Appendices. The characterization of cuts using circulations in the dual is not original to this work [14, 21, 23, 48], but there are some key changes in our characterization and how we use it for counting cuts. In [14, 21, 23], minimum cuts in *undirected* graphs are characterized using homology with coefficients in $\mathbb{Z}_2$. However, G has directed edges, so we must use coefficients in $\mathbb{Z}$. Integer coefficients are used in [48] to compute edge expansion in genus g graphs. However, the algorithm used to compute edge expansion has running time $n^{O(g^2)}$ and is therefore not fixed parameter tractable. To the best of our knowledge, there is no fixed parameter tractable algorithm that deals with integer homology directly.

In Section 4 we describe an algorithm to compute the number of circulations in a certain homology class if the primal directed acyclic graph is triangulated. Finally, in Section 5 we generalize our algorithm to work for non-triangulated graphs as well. Unlike many other problems where triangulating the input is trivial, we must actually change the surface itself to form a triangulation without affecting the number of forward (t,s)-cuts. Fortunately, some surprising properties of non-crossing cycles allows us to limit the complexity of our modified surface.

Our algorithm can also be used to randomly *sample* a minimum (s,t)-cut. The sampling algorithm follows almost as an immediate consequence of our main result and the sampling technique given in [2]. After running the counting cut algorithm, we only need $O(g^2 n)$ time per sampling, so several samples can be computed quite fast. We defer the presentation of the sampling algorithm to Appedix F.

2 Preliminaries

We give a brief overview of necessary definitions of tools we use. For full coverage, we direct the reader to existing books and surveys on topology [29, 46], computational topology [18, 57], and graphs on surfaces [16, 44].

2.1 Graphs

Let $G = (V, E)$ be a directed graph. For each edge $e = (u \to v) \in E$ we define its ***tail*** and ***head*** to be the vertices u and v, respectively. If $u = v$ then e is a ***loop***. The ***indegree*** of a vertex v is the total number of edges in E whose head are v. Similarly, the ***outdegree*** of v is the total number of edges in E whose tail are v. A directed (u, v)-walk in G is a sequence of vertices $W = (u = w_1, w_2, \cdots, w_k = v)$ such that $(w_i \to w_{i+1}) \in E$ for all $1 \leq i < k$; we denote a (u, v)-***walk*** by $u \rightsquigarrow w$; we also use $u \rightsquigarrow v$ to designate there is a a directed walk from u to v. A (u, w)-walk is ***closed*** if $u = w$, it is a ***path*** if it has no repeated vertices, and it is a ***cycle*** if it is a path with $u = w$ its only repeated vertex. Let W_1 be a (u, v)-walk and W_2 be a (v, w)-walk; their ***concatenation*** $W_1 \cdot W_2$ is defined as the concatenation of their corresponding vertex sequences.

A directed graph G is a ***directed acyclic graph (DAG)*** if and only if it does not contain any directed cycle. A vertex $v \in V$ of a DAG is a source if its indegree is zero and a sink if its outdegree is zero.

A ***cut*** in G is defined as a subset of vertices $S \subseteq V$; we refer to S and $T = V \backslash S$ as different ***sides*** of the cut. Given two vertices a and b with $a \in S$ and $b \in T$, we say S is a ***forward*** $\boldsymbol{(a, b)}$***-cut*** if there is no edge $(u \to v) \in E$ such that $u \in T$ and $v \in S$.

An edge $e = (u \to v)$ ***crosses*** a cut S if and only if u and v are on different sides of S; particularly e crosses S in the ***forward*** direction if $u \in S$ and in the ***backward*** direction if $v \in S$. For a cut S we use the notation $\boldsymbol{\Gamma^+(S)}$ to denote the set of all edges that cross S in the *forward* direction. We define $\boldsymbol{\Gamma^-(S)}$ as the set of edges that cross S in the backward direction. A walk W crosses a cut S k times if and only if there are k edges of W that cross S.

A ***spanning tree*** τ of a connected graph $G = (V, E)$ is a maximal subgraph of G that contains no cycles. The tree τ is a ***forward spanning tree*** with ***root*** $r \in V$ if and only if τ contains a directed path from r to any vertex $u \in V$; we will also say that is a directed tree with root r. Similarly, τ is a ***backward spanning tree*** with root r if it contains a path from any vertex $u \in V$ to r.

2.2 Surfaces and embeddings

A ***surface*** (or a 2-manifold) with boundary is a (Hausdorff) space in which every point has a neighborhood homeomorphic to the Euclidian plane or the closed half plane. The union of all points that are homeomorphic to the closed half plane compose the ***boundary*** of the surface. Every boundary component is homeomorphic to a disk. A ***cycle*** in a surface Σ is a continuous function $\gamma : S^1 \to \Sigma$; the cycle γ is *simple* if and only if γ is injective. A ***path*** in a surface Σ is a continuous function $p : [0, 1] \to \Sigma$; the path p is *simple* if and only if p is injective. A simple cycle γ is ***separating*** if and only if $\Sigma \backslash \gamma$ is not connected. Further, γ is ***contractible*** if $\Sigma \backslash \gamma$ is composed of two components and at least one of them is homeomorphic to an open disc. The ***genus*** of a surface Σ, denoted by g, is the maximum number of disjoint simple cycles $\gamma_1, \gamma_2, \ldots, \gamma_g$ in Σ such that $\Sigma \backslash (\gamma_1 \cup \gamma_2 \cup \cdots \cup \gamma_g)$ is connected. A surface Σ is ***non-orientable*** if and only if it contains a subspace homeomorphic to the Möbius band and is ***orientable*** otherwise. It is known that any surface can be specified up to homeomorphism with its genus and orientability. For this paper, we consider only compact, connected, orientable surfaces.

An ***embedding*** of a graph $G = (V, E)$ on a surface Σ is a drawing of G on Σ, such that vertices are mapped to distinct points in Σ and edges are mapped to *internally* disjoint simple paths in Σ. A ***face*** of an embedding is a maximal connected set in Σ that does not intersect the image of G. An embedding is ***cellular*** if all of its faces are homeomorphic to a topological open disc. For a face f, we let ∂f define the clockwise cycle bounding the face. Any cellular embedding in an orientable surface can be combinatorially described using a ***rotation system***, which records the cyclic order of the incident edges of each vertex. For such an embedding, Euler's formula implies $|V| - |E| + |F| = 2 - 2g - b$, where F is the set of faces of the embedding and b is the number of boundary components. For this paper, we assume $g = O(n)$. Euler's formula then implies $m = O(n)$ if G is simple.

2.3 Topology

Let $G = (V, E)$ be a graph embedded on a surface Σ and let F be the set of faces of the embedding. We optionally refer to the vertices of G as ***cells of dimension* $\mathbf{0}$**, the edges as ***cells of dimension* $\mathbf{1}$**, and the faces as ***cells of dimension* $\mathbf{2}$**. A ***k-chain*** $(1 \leq k \leq 2)$ is a function that assigns a set of values to cells of dimension k. A k-chain is ***trivial*** if it assigns 0 to all cells, it is ***non-negative*** if it assigns non-negative values to all cells, and it is a **$(\mathbf{0}, \mathbf{1})$*-chain*** if it assigns values from $\{0, 1\}$ to all cells.

Let $\phi : E \to \mathbb{Z}$ be a 1-chain. Then, the boundary of ϕ is a 0-chain $\partial\phi : V \to \mathbb{Z}$ defined as $\partial\phi(v) = \sum_{((v \to w)) \in E} \phi(v \to w)$ for each $v \in V$. It helps to think of ϕ as a function that assigns non-negative values to each undirected edge together with a direction, so that $\phi(u \to v) = -\phi(v \to u)$.

A ***circulation*** is a 1-chain ϕ such that $\partial\phi(v) = 0$ for all $v \in V$. The ***cycle space*** of a graph G, denoted by $Z(G)$, is the vector space of 1-chains that are circulations in G. One can easily verify that $Z(G)$ is isomorphic to $\mathbb{Z}^{|E|-|V|+1}$. A ***trivial circulation***, a ***non-negative circulation*** and a **$(\mathbf{0}, \mathbf{1})$*-circulation*** are special cases of such 1-chains. We say a set of *directed* cycles $\mathcal{C}$ ***trivially generates*** a circulation ϕ if ϕ is the 1-chain that assigns a value to each edge equal to the number of times the edge appears in $\mathcal{C}$. We may abuse terminology by equating $\mathcal{C}$ with the circulation trivially generated by $\mathcal{C}$.

2.4 Homology

The boundary of a 2-chain $\alpha : F \to \mathbb{Z}$ is a 1-chain $\partial\alpha : E \to \mathbb{Z}$ such that $\partial\alpha(e) = right(e) - left(e)$, where $left(e)$ and $right(e)$ are the faces to the left and right of the edge e with a certain direction. It is straightforward to verify that the boundary of any 2-chain, which is called a ***boundary circulation***, is a circulation. The ***boundary space*** of G, denoted by $B(G)$ is the space of all boundary circulations. It follows from the definition that $B(G)$ is a linear subspace of $Z(G)$, and it is isomorphic to $\mathbb{Z}^{|F|-1}$ if $b = 0$ or $\mathbb{Z}^{|F|}$ if $b \geq 1$, where b is the number of boundary components in Σ.

Two circulations ϕ and ψ are ***homologous***, or they are in the same ***homology class***, if and only if their difference $\phi - \psi$ is a boundary circulation. Thus the ***homology space*** is the vector space of homology classes, which is isomorphic to $Z(G) \backslash B(G) \cong \mathbb{Z}^{2g+\max\{0,b-1\}}$ by Euler's formula. We sometimes refer to the homology class of a set of directed cycles, where we more precisely mean the homology class of the circulation trivially generated by that set.

2.5 Dual graphs

Let $G = (V, E)$ be a graph embedded on a surface Σ, and let F be the set of faces in the embedding. The ***dual graph* G^*** is defined as an embedded graph on Σ that has a vertex f^* for each face $f \in F$. There is an edge $(f^* \to g^*)$ in G^* if and only of there is a directed edge $e \in E$ that has f on its left side and g on its right side; we denote such a situation by $f \uparrow g$. For a subgraph H of G, we abuse notation by letting H^* denote the subgraph of G^* with the same edges as H.

The graph G is a ***triangulation*** if every face of G is a triangle; this is equivalent to G^* being ***3-regular***, where every vertex of G^* has degree 3.

Let $G = (V, E)$ be a graph embedded on a surface Σ of genus g. A tool we use is a ***tree-cotree*** decomposition, where E is decomposed into three disjoint sets (τ, L, C) such that τ is a spanning tree of G, L^* is a spanning tree of G^*, and C is composed of $2g$ edges. It is well known that for any $e \in C$, $\tau \cup e$ contains exactly one undirected cycle, which is non-separating.

2.6 Flows

For $s, t \in V$, an (s,t)-***flow*** is a 1-chain $\phi : E \to \mathbb{R}$ such that $\partial\phi(v) = 0$ for all $v \in V \setminus \{s, t\}$. For a capacity function $c : E \to \mathbb{R}$, the flow ϕ is ***feasible*** if $\phi(e) \leq c(e)$ for all $e \in E$. Each flow can be decomposed to a set of weighted paths and cycles. A flow is ***acyclic*** if it can be decomposed into a set of simple (s,t)-paths.

For a flow ϕ, the ***residual capacity*** function $c_\phi : E \to \mathbb{R}$ is defined as $c_\phi(e) = c(e) - \phi(e)$. The ***residual graph*** $\boldsymbol{G_\phi}$ is the graph with edges weighted according to the residual capacity function.

3 Minimum cuts, forward cuts, and cocirculations

Following Bezáková and Friedlander, we begin by reducing the problem of counting minimum cuts to the problem of counting forward cuts. Let G be a directed acyclic (multi) graph. Let a be a vertex of G with indegree 0 and let b be a vertex of G with outdegree 0. Finally, let S be a subset of vertices such that $a \in S$ and $b \notin S$. Recall that X is a forward (a,b)-cut of G if there is no edge $u \to v$ such that $v \in X$ and $u \notin X$.

The following theorem is a slight generalization of Bezáková and Friedlander [2, Theorem 4]; the proof appears in Appendix A.

Theorem 3.1. *Let $G = (V, E, c)$ be a (directed) flow network with edge capacities $c : E \to \mathbb{R}^+$ embedded on a surface of genus g. Let $s \in V$ be the source and $t \in V$ be the sink. There exists a directed acyclic (DAG) graph $\tilde{G} = (\tilde{V}, \tilde{E})$, possibly with self-loops, embedded on a surface of genus at most g and vertices $\tilde{s}, \tilde{t} \in \tilde{V}$ such that the number of minimum (s,t)-cuts in G is equal to the number of forward $(\tilde{t}, \tilde{s})$-cuts in $\tilde{G}$.*

The following corollary is immediate from Theorem 3.1 as well as the original theorem in [2].

Corollary 3.2 (Bezáková and Friedlander [2, Corollary 5]). *Suppose there exits a path from s to every vertex of G and a path from every vertex of G to t. Then $\tilde{t}$ is the only vertex of indegree 0 and $\tilde{s}$ is the only vertex of outdegree 0 in $\tilde{G}$.*

Based on the above theorem and corollary, we focus on the problem of counting forward (t,s)-cuts in a directed acyclic graph G possibly with self loops embedded on a surface Σ of genus g where t is the only source in G and s is the only sink. Let $\Sigma' = \Sigma \setminus (s^* \cup t^*)$. Simply knowing that G is a DAG immediately gives us the following lemma which generalizes Claim 1 of [2]. This lemma helps us characterize the edges leaving forward (t,s)-cuts as particular circulations in the dual and makes it possible to count these circulations.

Lemma 3.3. *There exist no non-trivial non-negative boundary circulations of G^* in the surface Σ'.*

Proof: For the sake of contradiction, let ϕ be a non-trivial non-negative boundary circulation of G^* in the surface Σ'. We note that no edge dual to a loop in G can have non-zero value in any boundary circulation, since its dual is bordered by the same face on both sides.

Let $(u{\to}w)^*$ be a directed edge with $\phi((u{\to}w)^*) > 0$. By Corollary 3.2, u is reachable from t and w can reach s. Therefore, there exists a simple *directed* path $p = v_0{\to}v_1{\to}v_2{\to}\ldots{\to}v_k$ from t to s through $u{\to}w$, where $v_0 = t$, $v_k = s$; also for some $0 \le i \le k-1$, $v_i = u$ and $v_{i+1} = w$.

Boundary circulation ϕ is equal to $\partial\alpha$ for some 2-chain α of G^* in the surface Σ where $\alpha(v_0^*) = \alpha(v_k^*) = 0$ (because $v_0^* = t^*$ and $v_k^* = s^*$ are boundaries). For each edge $v_i{\to}v_{i+1}$ of p, we have $\alpha(v_{i+1}^*) - \alpha(v_i^*) \le 0$, because there exists no edge e in G with $\phi(e^*) < 0$. Further, we have $\alpha(w^*) - \alpha(u^*) < 0$. Therefore, $\alpha(v_k^*) < 0$, a contradiction. □

Theorem 3.1 reduces the problem of counting minimum cuts to the problem of counting forward cuts in a DAG that possible contains self loops. The following results reduce counting forward cuts to the problem of counting circulations in a certain homology class. These results borrow ideas from minimum cut algorithms in surface embedded graphs [14, 21, 23], but require substantially more technical detail to work with integer homology.

Lemma 3.4. *Let T be a forward (t,s)-cut in G, and let ϕ_T be the 1-chain in G^* where $\phi_T(e) = 1$ if e^* crosses T and $\phi_T(e) = 0$ otherwise. Then, ϕ_T is a $(0,1)$-circulation of G^* homologous to ∂t^* in the surface Σ'.*

Proof: We define a 2-chain α of G^* in the surface Σ'. For each vertex $v \in V \setminus \{t,s\}$, let $\alpha(v^*) = 1$ if $v \in T \setminus \{t\}$, and let $\alpha(v^*) = 0$ otherwise. Consider the circulation $\partial\alpha$ and any directed edge $e = u{\to}v$ of G. If e is a loop and $v = u$, then $\alpha(v^*) - \alpha(u^*) = 0$. If $u = t$ and $v \in T$, then $\partial\alpha(e^*) = -1$. If $u = t$ and $v \notin T$, then $\partial\alpha(e^*) = 0$. If $u \in T \setminus \{t\}$ and $v \notin T$, then $\partial\alpha(e^*) = 1$. In all other situations, $\partial\alpha(e^*) = 0$. We see $\partial\alpha = \phi_T - \partial t^*$. □

Lemma 3.5. *Let ϕ be a non-negative circulation in G^* that is homologous to ∂t^* in the surface Σ'. Then there exists a forward (t,s)-cut T of G, such that for each edge $e \in E$, e crosses T if and only if $\phi(e^*) = 1$. Further, ϕ is a $(0,1)$-circulation in G^*.*

Proof: We begin by showing the existence of some (not necessary forward) (t,s)-cut T where for every edge e that crosses T in forward direction we have $\phi(e^*) \ge 1$; the dual of the collection of edges with value 1 in ϕ separates t from s.

Let ϕ_t be the circulation trivially generated by ∂t^*. By assumption, there exists a 2-chain α of G^* in the surface Σ such that $\partial\alpha = \phi - \phi_t$ and $\alpha(t^*) = \alpha(s^*) = 0$. Let $p = (v_0{\to}v_1{\to}\ldots v_k)$ be any simple path in G such that $v_0 = t$ and $v_k = s$. Suppose there exists no edge e of p with $\phi(e^*) \ge 1$. We have $\alpha(v_1^*) - \alpha(v_0^*) \ge 1$ and $\alpha(v_{i+1}^*) - \alpha(v_i^*) \ge 0$ for all $0 < i \le k$. We immediately have a contradiction on $\alpha(v_0^*) = \alpha(v_k^*) = 0$. Thus, any simple (t,s)-path contains an edge $e \in E$ such that $\phi(e^*) \ge 1$; so the collection of such edges separate t from s.

Now, let $T \subsetneq V$ be the set of vertices reachable from t. Recall $\Gamma^+(T)$ and $\Gamma^-(T)$ are the sets of edges that cross T in forward and backward direction, respectively. Let ϕ_T be the 1-chain of G^* with $\phi_T(e^*) = 1$ for every directed edge $e \in \Gamma^+(T)$, $\phi_T(e^*) = -1$ for every directed edge $e \in \Gamma^-(T)$, and $\phi_T(e^*) = 0$ for every other edge in G. Let G' be the graph G with every edge of $\Gamma^-(T)$ reversed. We see T is a forward (t,s)-cut in G'. By Lemma 3.4, ϕ_T is a $(0,1)$-circulation homologous to ∂t^* in G'^* on the surface Σ'. By assumption, ϕ is homologous to ∂t^* on Σ', too. Therefore, ϕ_T is also homologous to ϕ on Σ'.

Now, let $\phi' = \phi - \phi_T$. The 1-chain ϕ' is a non-negative boundary circulation. Thus, Lemma 3.3 implies that ϕ' should be trivial. It follows that $\phi = \phi_T$ and so T is a forward (t,s)-cut. Further, ϕ is a $(0,1)$-circulation. □

Lemmas 3.4 and 3.5 imply a bijection between forward (t,s)-cuts in G and $(0,1)$-circulations in G^* of a particular homology class. We immediately get the following theorem which drives the remaining algorithm design and analysis.

Theorem 3.6. *Let $G = (V, E)$ be a directed graph embedded on a surface Σ with vertices $s, t \in V$ such that there exist no directed cycles in G other than single edge loops and where every vertex of G is reachable from t and every vertex can reach s, and let $\Sigma' = \Sigma \setminus (t^* \cup s^*)$. The number of forward (t, s)-cuts in G is equal to the number of $(0, 1)$-circulations of G^* homologous to ∂t^* in the surface Σ'.*

4 Counting circulations in triangulations

In this section, we give our algorithm for counting forward (t, s)-cuts of G assuming G is embedded in Σ as a triangulation. We relax our assumption that G is a triangulation in Section 5. We first note that G^* is 3-regular. Therefore, any set of edge-disjoint directed cycles in G^* must also be vertex disjoint. We immediately see every $(0, 1)$-circulation ϕ of G^* is trivially generated by a unique set of edge-disjoint cycles in G^* found by tracing along edges e of G where $\phi(e^*) = 1$. Theorem 3.6 then implies we can count the forward (t, s)-cuts of G by counting such collections of cycles in a particular homology class. In fact, the second part of Lemma 3.5 implies we can safely count *all* sets of cycles that trivially generate a circulation in the correct homology class without explicitly checking if they are edge disjoint. The rest of this section is devoted to counting these collections of cycles.

Our algorithm begins with the following construction. It creates a tree-cotree decomposition (τ, C, L) where τ is an arbitrary directed spanning tree of G rooted at t. Euler's formula implies that L contains exactly $2g$ directed edges which we label $u_1 \to v_1, u_2 \to v_2, \ldots, u_{2g} \to v_{2g}$. Let $\tau[v]$ denote the directed path from t to v in τ. For each $i \in \{1, \ldots, 2g\}$, let p_i^+ denote the directed path $\tau[u_i] \cdot (u_i \to v_i)$, and let $p_i^- = \tau[v_i]$. Let $p_0^+ = \tau[s]$ and let p_0^- denote the trivial walk from s to itself. Let p_i denote the directed path $p_i^+ \cdot rev(p_i^-)$. Finally, let $P = \{p_0, p_1, \ldots, p_{2g}\}$. The intersection of P and Σ' forms a *system of arcs* in Σ'; cutting Σ' along P creates a topological disk [11].

Lemma 4.1. *Let G be a directed, acyclic graph, T be a forward (t, s)-cut, and p be a simple directed path in G. At most one edge of p crosses T.*

Proof: Since T is a forward (t, s)-cut, by definition there is no edge going from $V \setminus T$ to T. Therefore, once any directed walk enters $V \setminus T$, it cannot cross to T again, giving at most one edge on the walk crossing the cut (which must go from T to $V \setminus T$). □

Corollary 4.2. *At most one edge of each path p_i^+, p_i^- crosses T.*

Let $\mathcal{C}$ be an arbitrary set of directed simple cycles in G^* and let ϕ be the circulation trivially generated by $\mathcal{C}$. We show how to determine the homology class of ϕ in Σ' by (essentially) counting the number of times the cycles in $\mathcal{C}$ cross P. For any directed cycle γ in G^* and index $0 \le i \le 2g$, let $x_i^+(\gamma)$ be the net number of times γ crosses p_i^+, and let $x_i^-(\gamma)$ be the net number of times γ crosses p_i^-. Let $x_i(\gamma) = x_i^+(\gamma) - x_i^-(\gamma)$. Similar to [14, 23], we define the ***subdivided crossing vector*** $x^{|}(\gamma)$ to be $(x_0^+(\gamma), x_0^-(\gamma), \ldots, x_{2g}^+(\gamma), x_{2g}^-(\gamma))$. We define the ***arc crossing vector*** $x(\gamma)$ to be $(x_0(\gamma), \ldots, x_{2g}(\gamma))$. The subdivided/arc crossing vector of $x(\mathcal{C})$ is the sum of the respective crossing vectors of $\mathcal{C}$'s individual elements. Observe that $x(\partial t^*) = (1, 0, 0, \ldots)$. The following lemma and its proof are based on [23, Lemma 3.2]. We defer the proof to Appendix B.

Lemma 4.3. *A set of cycles $\mathcal{C}$ in G^* trivially generates a boundary circulation ϕ in Σ' if and only if $x(\mathcal{C}) = 0$.*

Corollary 4.4. *Two sets of cycles $\mathcal{C}$ and $\mathcal{C}'$ are homologous if and only if $x(\mathcal{C}) = x(\mathcal{C}')$.*

Let γ be a directed cycle in G^*. We define the ***crossing sequence*** of γ to be its cyclic order of crossings of $\{p_0^-, p_0^+, \ldots, p_{2g}^-, p_{2g}^+\}$. We compute the total number of forward (t,s)-cuts in G (or the total number of sets of cycles in G^* that trivially generate a circulation homologous to ∂t^*), by enumerating sets of *abstract* cycles in the dual, where an abstract cycle is specified by a crossing sequence. For any set of abstract cycles $\mathcal{C}_\mathcal{A}$, we compute the total number of corresponding circulations in G^*.

We use a modification of a method based on previous works [11,14] to enumerate abstract sets of cycles. Our algorithm cuts Σ' along P and replaces each copy of p_i^+, p_i^- with a single edge to obtain an abstract polygonal schema which we denote as S. We emphasize that our algorithm replaces each *path* p_i^+, p_i^- by a single edge and *not* each arc p_i of Σ' as in previous works. Each path p_i^+, p_i^- corresponds to two edges of S.

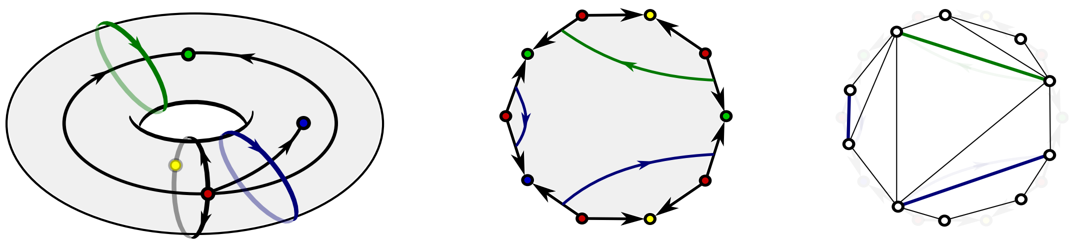

Figure 1. Building the polygonal schema (left to right): The set of paths P is given in black, and the dual of a forward cut is given in green and blue; polygonal schema S; the triangulated dual of S.

Now consider a set of cycles $\mathcal{C}$ that trivially generates a circulation homologous to ∂t^* in Σ', and let $\mathcal{C}_\mathcal{A}$ be its corresponding abstract set of cycles. In polygonal schema S, each of the cycles of $\mathcal{C}_\mathcal{A}$ is cut into arcs which cross the schema; by Lemma 3.5, the arcs will be non-crossing in the interior of S, and by Corollary 4.2 each edge of the schema contains at most one endpoint of any arc; in particular, no two arcs have endpoints on the same pair of boundary.

Our algorithm dualizes the polygonal schema by taking the original abstract schema and replacing each edge with a vertex and each vertex with an edge. It then connects two vertices in the dual if there is an arc between their corresponding edges in the original schema. Now, each arc from $\mathcal{C}_\mathcal{A}$ represents an edge between vertices in the $8g+2$-gon. (Note that we can ignore copies of p_0^- because p_0^- is a trivial walk and no directed paths cross it.) Since none of the arcs can cross, this abstraction gives a subdivision of the dualized schema with no parallel edges. We triangulate this subdivision by adding edges of weight zero. We say that two triangulations of the abstract polygonal schema are *equivalent* if and only if they are identical ignoring zero weight edges. See Figure 1 for illustration.

Every equivalence class of weighted triangulations of the dualized schema corresponds to a collection of non-crossing abstract cycles in G^*. Our algorithm enumerates such equivalence classes. Note that the weight of each edge is either zero or one, which implies that our algorithm needs to consider only $2^{O(g)}$ different triangulations.

Our algorithm first checks whether a triangulation corresponds to a set of cycles that trivially generate the homology class of ∂t^* using Corollary 4.2 and Corollary 4.4. We note that each edge of the triangulation has a clear direction specified entirely by the vertices it goes through, because each vertex corresponds to a directed path in G. Then, it computes an abstract collection of cycles corresponding to the triangulation by brute force and then computes the *crossing sequence* for each cycle in the collection and P. All that remains is to compute the total number of cycle collections in G^* with a given set of crossing sequences.

4.1 Counting cycles with a crossing sequence

We briefly sketch our strategy to count collections of cycles in G^* with a given set of crossing sequences. Let X be an arbitrary crossing sequence corresponding to an abstract directed cycle γ. In a manner similar to [9, 39], our algorithm builds a finite portion of the *universal covering space* of size $O(|X|n)$. This subset of the covering space contains a *lift* of G^* we call G_X such that G_X is a directed acyclic graph and every cycle with crossing sequence X lifts to a directed path in G_X. We can easily count these directed paths in $O(|X|n^2)$ time using a strategy similar to the one found in [2]. See Appendix C for details.

In order to compute the total number of cycle collections in G^* with a given set of crossing sequences, our algorithm simply needs to multiply the numbers of cycles for each individual crossing sequence. It then adds the number of cycles corresponding to each equivalence class of weighted triangulations of the dualized polygonal schema. We get the following lemma.

Lemma 4.5. *Let G be a triangulated DAG with possible self-loops embedded on a surface Σ of genus g with t and s the only source and sink, respectively. There is a $2^{O(g)}n^2$ time algorithm to compute the total number of forward (t, s)-cuts.*

5 Handling non-triangulations

In this section, we remove our assumption that G is embedded in Σ as a triangulation. We sketch an algorithm to build a triangulated graph G_Δ embedded on a surface Σ_Δ of genus $O(g)$ with the same total number of forward (t, s)-cuts. We can then use the algorithm of section 4 to count the number of forward cuts in G_Δ and so in G.

We use the following lemma to add edges to G in order to create a triangulation; see Appendix D for the proof.

Lemma 5.1. *Let $G = (V, E)$ be a DAG with possible self-loops and t and s the only source and sink vertices, respectively. Assume $u, v \in V$, $(u\to v) \notin E$ and $u \rightsquigarrow v$. Then, $G \cup (u\to v)$ is a DAG with possible self-loops that has the same number of forward (t, s)-cuts as G.*

A face f of G is ***irreducible*** if and only if for any two vertices $u, v \in V$ that are not adjacent on f (1) $u \neq v$, (2) $u \not\rightsquigarrow v$, and (3) $v \not\rightsquigarrow u$. In particular, the boundary of an irreducible face is composed of an even number of edges with alternating clockwise and counterclockwise directions. See Figure 3 for examples of irreducible faces.

An embedded DAG with possible self loops is ***maximally triangulated*** if and only if any non-triangle face of it is irreducible. Applying Lemma 5.1 and adding self-loops let us create a maximally triangulated graph G_δ, but this process does not necessarily result in a triangulation. We cannot add edges on an irreducible face without possibly changing the number of forward (t, s)-cuts. Fortunately, we have the following lemma based on a surprising application of a bound on the number of non-crossing non-homotopic loops on a surface. A proof appears in Appendix E.

Lemma 5.2. *Let G be a DAG with possible self-loops embedded on a surface Σ such that t and s are the only source and sink. Then, there is an $O(n^2)$ time algorithm to compute a maximally triangulated DAG, G_δ, embedded on Σ that has the same number of forward (t, s)-cuts. Further, G_δ has $O(g)$ irreducible faces.*

To get rid of irreducible faces we further triangulate G_δ intuitively by connecting the vertices that appear on the boundary of irreducible faces to s; Lemma 5.1 particularly implies that we can always add edges to s without changing the total number of minimum cuts. However, adding edges with endpoints in different faces results in changing the underlying surface Σ; intuitively, we need to glue more handles to the surface to avoid edge crossings. The following theorem shows that all irreducible faces can be triangulated by adding only $O(g)$ handles to Σ.

Lemma 5.3. *Let G_δ be a maximally triangulated DAG with possible self-loops embedded on a surface Σ of genus g, and t and s be the only source and sink, respectively. Then, there exists a triangulated supergraph G_Δ of G_δ embedded on a surface Σ_Δ of genus $O(g)$ such that the number of forward (t, s)-cuts in G_δ and G_Δ are equal. Further, G_Δ can be computed in $O(n)$ time.*

Proof: Lemma 5.2 implies that G_δ has $O(g)$ irreducible faces. Lemma 5.1 implies that s is not on the boundary of any irreducible faces.

Let f be an irreducible face and f' be a triangle incident to s. Let the vertices on the boundary of f and f' be $(v_0, v_2, \ldots, v_{k-1})$ and (s, s', s'') respectively, in clockwise order. We add a handle to connect f and f', and use it to add edges from all v_i's to s. Combinatorially, for all $0 \leq i < k$, we add the $e_i = v_i \to s$ edges such that (1) for all $0 \leq i < k$, e_i is between $(v_{i\ominus 1}, v_i)$ and $(v_{i\oplus 1}, v_i)$ in the list of v_i, where $\ominus$ and $\oplus$ are subtraction and addition modulo k, (2) for all $1 \leq i < k-1$, e_i is between $(v_{i\ominus 1}, s)$ and $(v_{i\oplus 1}, s)$ in the list of s, (3) (v_0, s) is between (v_1, s) and (s'', s) in the list of s, and (4) (v_{k-1}, s) is between (v_{k-2}, s) and (s', s) in the list of s; see Figure 2.

It is easy to check that for any $0 \leq i < k-1$, the triangle (v_i, s, v_{i+1}) is a face of the new graph. The only face that is not a triangle is $(v_0, s, s'', s', s, v_{k-1})$, which can be triangulated by adding the following edges: $(s'' \to s)$, $(s \to s)$ and $(v_{k-1} \to s)$; see Figure 2. Since all new edges are towards s, Lemma 5.1 implies that adding them does not change the number of forward (t, s)-cuts.

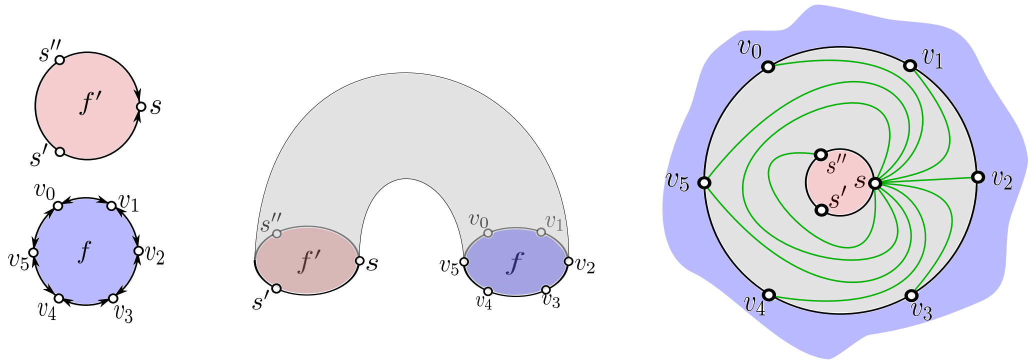

Figure 2. Triangulating an irreducible face (left to right): f', a triangle incident to s, and f an irreducible face of degree 6; the handle to connect f to f'; the flat view of the handle as an annulus and its triangulation with green edges (all the green edges are directed towards s.

Each of $O(g)$ irreducible faces of degree d can be triangulated by adding one handle in $O(d)$ time. It follows that we can iteratively triangulate G_δ to obtain G_Δ by adding $O(g)$ handles in $O(n)$ time. □

Finally, the main theorem of this section follows from Lemma 5.2 and Lemma 5.3.

Theorem 5.4. *Let $G = (V, E, c)$ be a (directed) flow network with edge capacities $c : E \to \mathbb{R}^+$ embedded on a surface Σ of genus g. Let $s \in V$ be the source and $t \in V$ be the sink where there exists a path from s to every vertex in V and a path from every vertex in V to t. There exists a $2^{O(g)}n^2 + \min\left\{n^2 \log n, g^{O(g)}n^{3/2}\right\}$ time algorithm to calculate the number of minimum (s, t)-cuts in G.*

References

[1] Michael O. Ball and Scott J. Provan. Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13:253–278, 1983.

[2] Ivona Bezáková and Adam J. Friedlander. Counting and sampling minimum (s,t)-cuts in weighted planar graphs in polynomial time. *Theoret. Comput. Sci.*, 417:2–11, 2012.

[3] Glencora Borradaile, Erik D. Demaine, and Siamak Tazari. Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. In *Proc. 26th Int. Symp. Theoretical Aspects Comput. Sci.*, volume 3 of *Leibniz Int. Proc. Informatics*, pages 171–182. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2009.

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

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

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

[7] Y. Boykov and O. Veksler. Graph cuts in vision and graphics: Theories and applications. In Nikos Paragios, Yunmei Chen, and Olivier Faugeras, editors, *Handbook of Mathematical Models in Computer Vision*, pages 79–96. Springer US, 2006.

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

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

[10] Parinya Chalermsook, Jittat Fakcharoenphol, and Danupon Nanongkai. A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. In *Proc. 15th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 828–829, 2004.

[11] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. *Comput. Geom. Theory Appl.*, 41(1–2):94–110, 2008.

[12] Erin W. Chambers and David Eppstein. Flows in one-crossing-minor-free graphs. In *Proc. 21st International Symposium on Algorithms and Computation (ISAAC 2010)*, Lecture Notes in Computer Science, pages 241–252. Springer-Verlag, 2010.

[13] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Homology flows, cohomology cuts. In *Proc. 42nd Ann. ACM Symp. Theory Comput.*, pages 273–282, 2009. Full version to appear in *SIAM J. Comput.*

[14] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proc. 25th Ann. Symp. Comput. Geom.*, pages 377–385, 2009.

[15] Charles J. Colbourn. Combinatorial aspects of network reliability. *Annals Operations Research*, 33:1–15, 1991.

[16] Éric Colin de Verdière. Topological algorithms for graphs on surfaces. Habilitation thesis, May 2012.

[17] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Bojan Mohar. Approximation algorithms via contraction decomposition. In *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 278–287, 2007.

[18] Herbert Edelsbrunner and John Harer. *Computational Topology, An Introduction.* American Mathematical Society, 2010.

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

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

[21] Jeff Erickson, Kyle Fox, and Amir Nayyeri. Global minimum cuts in surface embedded graphs. In *Proc. 23rd Ann. ACM-SIAM Symp. Discrete Algorithms*, 2012.

[22] Jeff Erickson and Amir Nayyeri. Computing replacement paths in surface graphs. In *Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 1347–1354, 2011.

[23] Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 1166–1176, 2011.

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

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

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

[27] Torben Hagerup, Jyrki Katajainen, Naomi Nishimura, and Prabhakar Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57(3):366–375, 1998.

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

[29] Allen Hatcher. *Algebraic Topology.* Cambridge Univ. Press, 2002.

[30] Monika R. Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997.

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

[32] Alon Itai and Yossi Shiloach. Maximum flow in planar networks. *SIAM J. Comput.*, 8:135–150, 1979.

[33] Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proc. 43rd Ann. ACM Symp. Theory Comput.*, pages 313–322, 2011.

[34] Mark Jerrum. Random generation of combinatiorial structures from a uniform distribution. In Wilfried Brauer, editor, *Automata, Languages and Programming*, volume 194 of *Lecture Notes in Computer Science*, pages 290–299. Springer Berlin / Heidelberg, 1985. 10.1007/BFb0015754.

[35] David R. Karger. A randomized fully polynomial time approximation scheme for the all terminal network reliability problem. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95, pages 11–17, New York, NY, USA, 1995. ACM.

[36] Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *Proc. 49th IEEE Symp. Found. Comput. Sci.*, pages 771–780, 2008.

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

[38] Philip Klein, Shay Mozes, and Oren 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.

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

[40] Jakub Łącki and Piotr Sankowski. Min-cuts and shortest cycles in planar graphs in $O(n \log \log n)$ time. In *Proc. 19th Ann. Europ. Symp. Algorithms*, number 6942 in Lecture Notes Comput. Sci., pages 155–166. Springer, 2011.

[41] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16:346–358, 1979.

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

[43] Gary L. Miller. Isomorphism testing for graphs of bounded genus. In *Proc. 12th Ann. ACM Symp. Theory Comput.*, pages 225–235, 1980.

[44] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces.* Johns Hopkins Univ. Press, 2001.

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

[46] James R. Munkres. *Topology.* Prentice-Hall, 2nd edition, 2000.

[47] Hiroshi Nagamoch, Zheng Sun, and Toshihide Ibaraki. Counting the number of minimum cuts in undirected multigraphs. *IEEE Transactions on Reliability*, 40:610–614, 1991.

[48] Viresh Patel. Determining edge expansion and other connectivity measures of graphs of bounded genus. In *Proc. 18th Ann. Europ. Symp. Algorithms*, ESA'10, pages 561–572, Berlin, Heidelberg, 2010. Springer-Verlag.

[49] Dana Pe'er. On minimum spanning trees. Master's thesis, Hebrew University, 1998.

[50] Scott J. Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing*, 12:777–788, 1983.

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

[52] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[53] Siamak Tazari and Matthias 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.

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

[55] A. White. Orientable embeddings of cayley graphs. *Duke math J.*, pages 353–371, 1974.

[56] Christian Wulff-Nilsen. Solving the replacement paths problem for planar directed graphs in $O(n \log n)$ time. In *Proc. 21st Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 756–765, 2010.

[57] Afra Zomorodian. *Topology for Computing.* Cambridge Univ. Press, 2005.

Appendix

A Forward cuts and embeddings

We now prove the following theorem:

Theorem A.1 (Bezáková and Friedlander [2, Theorem 4]). *Let $G = (V, E, c)$ be a (directed) flow network with edge capacities $c : E \to \mathbb{R}^+$. Let $s \in V$ be the source and $t \in V$ be the sink. There exists a directed acyclic (DAG) graph $\tilde{G} = (\tilde{V}, \tilde{E})$ and vertices $\tilde{s}, \tilde{t} \in \tilde{V}$ such that the number of minimum (s,t)-cuts in G is equal to the number of forward $(\tilde{t}, \tilde{s})$-cuts in $\tilde{G}$.*

The proof of Theorem A.1 relies upon constructing a maximum flow f for the graph G, and then computing the residual graph G_f and iteratively removing cycles to result in an acyclic residual graph. The graph $\tilde{G}$ is then formed by taking the acyclic residual graph and contracting the strongly connected components. The ***edge contraction*** operation is defined as removing an edge $u{\to}v$ of the graph G and identifying its endpoints, u and v. Corollary 3.2 follows immediately.

Corollary 3.2. *[Bezáková and Friedlander [2, Corollary 5]] Suppose there exits a path from s to every vertex of G and a path from every vertex of G to t. Then $\tilde{t}$ is the only vertex of indegree 0 and $\tilde{s}$ is the only vertex of outdegree 0 in $\tilde{G}$.*

In [2], Bezáková and Friedlander use the fact that contraction of an edge in a planar graph yields another planar graph with an inherited embedding, as well as the existing algorithms for computing flows in planar graphs quickly, to get their algorithm to count the number of cuts.

In our setting, we must similarly exploit the embedding of the graph on a surface. Let G be embedded on a surface Σ of genus g. We note $O(n^2 \log n)$ is the best known time bound for computing maximum flows in arbitrary sparse graphs [25], which includes surface embedded graphs, so we use this algorithm to compute flows on our graph. We could also use a $g^{O(g)} n^{3/2}$ time algorithm [13] if we are not concerned with the larger dependency on g. The proof of Theorem A.1 implies $\tilde{G}$ can be computed in $O(n^2 \log n)$, because forming the residual and removing cycles from $\tilde{G}$ takes $O(n^2)$ time in a sparse graph. The proof for Theorem A.1 will not quite get the correct structure for the DAG in our setting, however, since we will still require the graph to be cellularly embedded on the surface Σ, and contracting strongly connected components may destroy the topology of the underlying surface which is essential to the algorithm.

We therefore modify the original construction as follows. First, find strongly connected components as in the original construction. All of the edges will be contracted iteratively unless the edge is a loop; these loops will not be contracted. The contraction of a single edge can be done in linear time while maintaining the same embedding on the underlying surface, so the overall contraction algorithm is still $O(n^2)$. Note that the underlying topology is maintained, since a loop will remain in the graph for each handle of the surface. Theorem 3.1 follows immediately.

Theorem 3.1. *Let $G = (V, E, c)$ be a (directed) flow network with edge capacities $c : E \to \mathbb{R}^+$ embedded on a surface of genus g. Let $s \in V$ be the source and $t \in V$ be the sink. There exists a directed acyclic (DAG) graph $\tilde{G} = (\tilde{V}, \tilde{E})$ possibly with self-loops embedded on a surface of genus at most g and vertices $\tilde{s}, \tilde{t} \in \tilde{V}$ such that the number of minimum (s,t)-cuts in G is equal to the number of forward $(\tilde{t}, \tilde{s})$-cuts in $\tilde{G}$.*

While this graph is not a DAG, our proofs are not affected by the existence of self loops. As an alternative to the above procedure, we can compute a new embedding of $\tilde{G}$ without loops onto a new surface of genus at most g [36], but we must consider loops anyway due to technicalities introduced by the procedure in Section 5. Note the minimum embedding of a connected graph is known to be cellular [55].

Lemmas 3.4 and 3.5 imply a bijection between forward (t,s)-cuts in G and $(0,1)$-circulations in G^* of a particular homology class. We immediately get the following theorem which drives the remaining algorithm design and analysis.

Theorem 3.6. *Let $G = (V, E)$ be a directed graph embedded on a surface Σ with vertices $s, t \in V$ such that there exist no directed cycles in G other than single edge loops and where every vertex of G is reachable from t and every vertex can reach s, and let $\Sigma' = \Sigma \setminus (t^* \cup s^*)$. The number of forward (t,s)-cuts in G is equal to the number of $(0,1)$-circulations of G^* homologous to ∂t^* in the surface Σ'.*

B Proof of Lemma 4.3

Lemma 4.3. *A set of cycles $\mathcal{C}$ in G^* trivially generates a boundary circulation ϕ in Σ' if and only if $x(\mathcal{C}) = 0$.*

Proof: Suppose $\mathcal{C}$ trivially generates boundary circulation ϕ. By definition, $\phi = \partial\alpha$ for some

2-chain α of G^* in the surface Σ'. The boundary of any face of G^* has crossing vector 0. We see

$$x(\mathcal{C}) = x(\sum_{v \in V} \alpha(v^*) \cdot \partial v^*) = \sum_{v \in V} \alpha(v^*) x(\partial v^*) = 0.$$

Now, suppose $x(\mathcal{C}) = 0$. We create a graph G^+ by modifying G^* as follows. For every loop $p_i \in P$ and for every adjacent pair of edge e_1, e_2 in p_i, we subdivide e_1 and e_2 by replacing e_1 (respectively e_2) with a vertex v_{e_1} (v_{e_2}) adjacent to both endpoints of e_1 (e_2). We then add an edge $(v_{e_1} v_{e_2})$ to G^+, subdividing a face incident to both e_1^* and e_2^*. Essentially, we are augmenting G^* with paths that follow the images of loops in P. We then prove the lemma by considering the cycles $\mathcal{C}$ in G^+. Subdividing edges and faces does not change homology.

Let a and b be two intersection (crossing) points between a cycle $\gamma \in \mathcal{C}$ and some arc p_i where γ crosses p_i from left to right through a and γ crosses p_i from right to left through b. If such crossing points do not exist each cycle of $\mathcal{C}$ lies in the disk $\Sigma' \setminus P$, and the lemma follows. Let $p_i[a, b]$ be the path added to G^+ between a and b along with image of p_i. Alter $\mathcal{C}$ by replacing γ's crossings through a and b with copies of $p_i[a, b]$ and $rev(p_i[a, b])$. The transformation does not change the homology class of $\mathcal{C}$ and it reduces the number of crossings of P. By induction, we see ϕ is homologous to a circulation trivially generated from a set of cycles $\mathcal{C}'$ that do not cross P. Each cycle of $\mathcal{C}'$ lies in the disk $\Sigma' \setminus P$, meaning they trivially generate a boundary circulation. □

C Counting cycles with a given crossing sequence

Let X be an arbitrary crossing sequence corresponding to an abstract directed cycle γ. We represent the elements of X using the integers $0, \ldots, 2g+1$ along with their negations (where the existence of element -0 is optional). Element i represents γ crossing p_i^+ and element $-i$ represents γ crossing p_i^-. Abusing notation, we assign each edge e of G^* a crossing sequence $X(e)$ using the following linear time recursive procedure. For the sake of definition, we add a vertex s' to G within an arbitrary face incident to s along with a directed edge $s{\to}s'$. We set $X((s{\to}s')^*) = 0$. Recall that we use the spanning tree τ and the set L of distinct extra edges in the tree-cotree decomposition to construct P. For every edge $u_i{\to}v_i$ in L, we set $X((u_i{\to}v_i)^*) = i$. For every edge $u{\to}v$ in τ in bottom up order from the leaves, we set $X((u{\to}v)^*)$ by concatenating the crossing sequences for edges of G incident to v in clockwise order starting with the edge immediately clockwise to $u{\to}v$. During the concatenation, we negate the singleton crossing sequence of any edge $w{\to}v$ that lies in L. Finally, we set $X(e) = \varepsilon$ for every other edge in G^*. For any cycle c, we have $X(c)$ equal to the concatenation of crossing sequences for c's individual edges.

We construct the following graph G_X along with a mapping from vertices and edges of G_X to G^*. The vertices are of G_X are pairs (f^*, X') where f^* is a vertex of G^* and X' is a prefix of X (including the empty sequence ε). Graph G_X contains edges $(f^*, X'){\to}(h^*, X'')$ where X' is a *proper* prefix of X (not including X itself), $f^*{\to}h^*$ is an edge of G^*, and $X'' = X' \cdot X(f^*{\to}h^*)$. Vertices and edges of G_X map to vertices and edges of G^* by simply dropping the second component of their pairs. We can also define G_X along with an embedding on a disk Σ_X using a standard construction [9,39]. Cut along P's image in Σ' to create a disk D we call the fundamental domain. Create one copy of D denoted $D_{X'}$ for every prefix X' of X. For every pair of prefixes X' and X'' where $X'' = X' \cdot i$, paste together $D_{X'}$ and $D_{X''}$ along p_i^+. If $X'' = X' \cdot -i$, then paste along p_i^-. Finally, remove all outgoing edges from any vertex in D_X. Here Σ_X is a subset of the *universal cover* of Σ'. The next lemma follows from our construction.

Lemma C.1. *Let f be a face of G incident to path p_i^+ (p_i^-). If X contains i ($-i$), then there* *exists a bijection between loops in G^* with crossing sequence X and basepoint f^* and paths in G_X from (f^*, ε) to (f^*, X).*

Lemmas 4.3 and 3.3 imply the following lemma.

Lemma C.2. *Graph G_X is a directed acyclic graph.*

A simple dynamic programming algorithm computes the number of paths from a vertex u in a DAG to a vertex v in linear time (see, for example, [2, Observation 6]). For each face f of G incident to a path p_i^+ (p_i^-) where X contains i ($-i$), our algorithm computes the number of paths in G_X from (f^*, ε) to (f^*, X). It then sums the results.

Lemma C.3. *Let X be a non-empty crossing sequence of an abstract cycle. Then, there is a $O(|X|n^2)$ time algorithm to compute the number of directed cycles with crossing sequence X.*

D Proof of Lemma 5.1

Lemma 5.1. *Let $G = (V, E)$ be a DAG with possible self-loops and t and s the only source and sink vertices, respectively. Assume $u, v \in V$, $(u \rightarrow v) \notin E$ and $u \rightsquigarrow v$. Then, $G \cup (u \rightarrow v)$ is a DAG with possible self-loops that has the same number of forward (t, s)-cuts as G.*

Proof: If $u = v$ the lemma is trivial because a self-loop never shows up in a forward cut, so we may assume $u \neq v$.

Assume $G \cup (u \rightarrow v)$ has a directed cycle γ of length at least two. Since G does not contain any directed cycles, we know $(u \rightarrow v) \in \gamma$. It immediately follows that $[\gamma \setminus (u \rightarrow v)] \cup (u \rightsquigarrow v)$ is a closed walk of length at least two in G, which contradicts the lemma assumption.

Now consider any forward cut T in G. Since $u \rightsquigarrow v$ it cannot be the case that $u \in S$ and $v \in T$; it follows that T is a forward cut in $G \cup (u \rightarrow v)$ as well.

On the other hand, a forward cut in $G \cup (u \rightarrow v)$ is indeed a forward cut in its subgraph G, and the proof is complete. $\square$

E Proof of Lemma 5.2

We begin with some definitions. Let v be a vertex on the boundary of an irreducible face f. Then, v is ***incoming*** on f if and only if both incident edges to v on f are incoming. Similarly, v is ***outgoing*** on f if and only if both incident edges to v on f are outgoing. Observe that any vertex v on the boundary of any irreducible face f is either incoming or outgoing on f; see Figure 3 for sample irreducible faces.

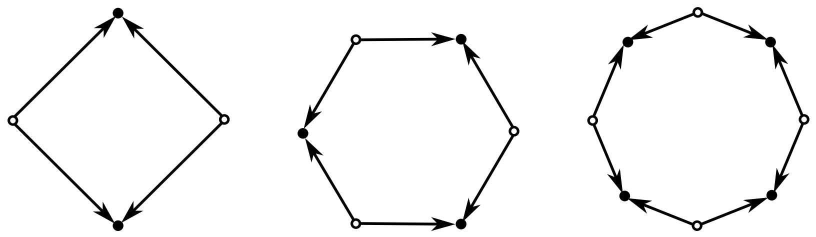

Figure 3. Irreducible faces of degree four, six and eight; circles: outgoing vertices; bullets: incoming vertices.

Lemma E.1. *Let G be a maximally triangulated DAG with possible self-loops embedded on a surface Σ and f be an irreducible face. Then, all vertices on the boundary of f are distinct.*

Proof: Any two non-adjacent vertices on the boundary of f are distinct by the definition of an irreducible face.

Assume that $(u \rightarrow v)$ appears on the boundary of f and that $u = v$. Let w be the other neighbor of v on the boundary of f, and so $(w \rightarrow v) \in E$. Because u and v are identical, it follows that $(w \rightarrow u) \in E$, in particular $w \rightsquigarrow u$. Since the degree of an irreducible face is at least 4, w and u cannot be adjacent on the boundary of v, which implies that f is not irreducible. $\square$

Let $\mathcal{S}$ be a spanning tree of G that contains a single directed path $\mathcal{S}[v]$ from any vertex $v \in V$ to s.

Lemma E.2. *Let G be a maximally triangulated DAG with possible self-loops embedded on a surface Σ, f be an irreducible face and u and v be outgoing and incoming vertices on f, respectively. Then, there is no directed path from any vertex of $\mathcal{S}[v]$ to u; in particular, no directed (t, u)-path intersects $\mathcal{S}[v]$.*

Proof: Let $\gamma = \mathcal{S}[v]$ and assume, for the purpose of contradiction, that there exist a directed path τ from a vertex $x \in \gamma$ to u. It follows that there exists a directed path, $\gamma[v, x] \cdot \tau[x, u]$, from v to u. Since G does not contain a directed cycle whose length is larger than one, we have $(u \rightarrow v) \notin E$, which implies that f is reducible; see Figure 4.

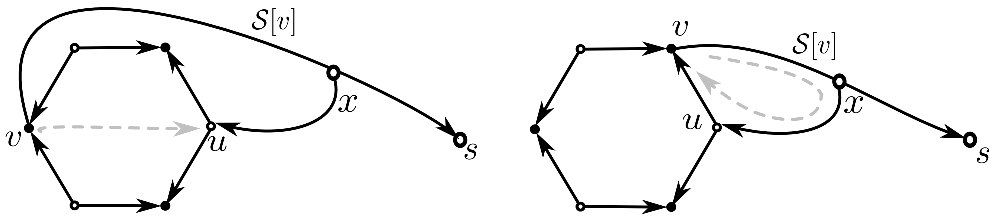

Figure 4. The setting for Lemma E.2.

$\square$

Let u be an outgoing vertex on an irreducible face f and v_1 and v_2 be u's neighbors on f. We define $\boldsymbol{C(\mathcal{S}, f, u)}$ to be the undirected cycle that is composed of $(u \rightarrow v_1) \cdot \mathcal{S}[v_1]$ and $(u \rightarrow v_2) \cdot \mathcal{S}[v_2]$; see Figure 5, left.

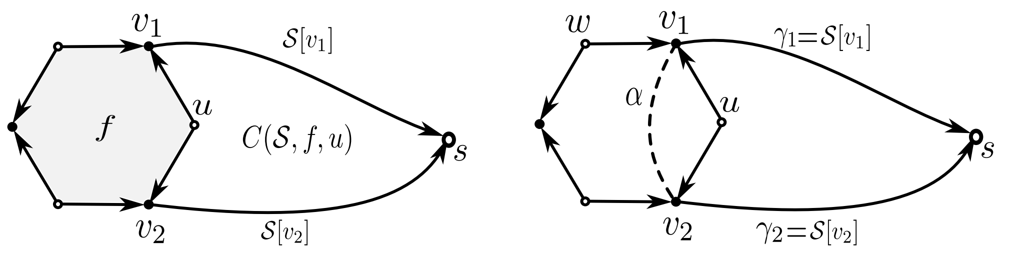

Figure 5. Left: definition of $C(\mathcal{S}, f, u)$. Right: proof of Lemma E.3

Lemma E.3. *Let G be a maximally triangulated DAG with possible self-loops embedded on a surface Σ, f be an irreducible face and u be an outgoing vertex on f. Then, $C(\mathcal{S}, f, u)$ is a non-separating cycle on Σ.*

Proof: Assume, for the purpose of contradiction, that $C(\mathcal{S}, f, u)$ is separating. Let v_1 and v_2 be the neighbors of u on f and α be a (v_1, v_2)-path (in the surface and not in the graph), which is strictly inside f except for its endpoints that are on the boundary of f. Also, let $\gamma_1 = \mathcal{S}[v_1]$ and $\gamma_2 = \mathcal{S}[v_2]$. (See Figure 5, right.)

Since $C(\mathcal{S}, f, u)$ and $\alpha \cup (u, v_1) \cup (u, v_2)$ are separating, $\gamma = \alpha \cup \gamma_1 \cup \gamma_2$ is separating as well. Assume that γ separates Σ into two surfaces Σ_u and Σ_w. Let w be the other neighbor of v_1 on f and observe that u and w are on two different sides of γ. Without loss of generality assume that $u \in \Sigma_u$ and $w \in \Sigma_w$.

Since t is the source of the DAG and its indegree is zero it cannot be on γ, so it is either in Σ_u or in Σ_w. In the former case any (t, w)-path has to intersect γ and in the latter case any (t, u)-path has to intersect γ. In either case there is a directed path from $\gamma_1 = \mathcal{S}[v_1]$ or $\gamma_2 = \mathcal{S}[v_2]$ to u or w, which contradicts Lemma E.2.

□

Lemma E.4. *Let G be a maximally triangulated DAG with possible self-loops embedded on a surface Σ, f and f' be irreducible faces and u and u' be outgoing vertices on f and f', respectively. Then, $C(\mathcal{S}, f, u) \cup C(\mathcal{S}, f', u')$ does not separate Σ; in particular, $C(\mathcal{S}, f, u)$ and $C(\mathcal{S}, f', u')$ are not homotopic.*

Proof: Let v_1 and v_2 be the neighbors of u on f and α be a directed (v_1, v_2)-path (in the surface and not in the graph), which is strictly inside f except for its endpoints that are on the boundary of f. Similarly, let v'_1 and v'_2 be the neighbors of u' on f' and α' be a directed (v'_1, v'_2)-path, which is strictly inside f' except for its endpoints that are on the boundary of f'. Let $w, w' \in V$ be the other neighbors of v_1 and v'_1 on f and f', respectively. Further, let $\gamma_1 = \mathcal{S}[v_1]$, $\gamma_2 = \mathcal{S}[v_2]$, $\gamma'_1 = \mathcal{S}[v'_1]$, $\gamma'_2 = \mathcal{S}[v'_2]$, $\gamma = \alpha \cup \gamma_1 \cup \gamma_2$ and $\gamma' = \alpha' \cup \gamma'_1 \cup \gamma'_2$.

Assume, for the purpose of contradiction, that $C(\mathcal{S}, f, u) \cup C(\mathcal{S}, f', u')$ is separating. Since $\alpha \cup (u, v_1) \cup (u, v_2)$ and $\alpha' \cup (u', v'_1) \cup (u', v'_2)$ are contractible cycles and $C(\mathcal{S}, f, u) \cup C(\mathcal{S}, f', u')$ is separating, $\gamma \cup \gamma'$ is also separating. Assume that $\gamma \cup \gamma'$ separates Σ to two surfaces Σ_u and Σ_w. Observe that u and w are on two different sides of $\gamma \cup \gamma'$. Without loss of generality assume that $u \in \Sigma_u$ and $w \in \Sigma_w$.

There are four possible cases to consider depending on whether $t \in \Sigma_u$ and whether $u' \in \Sigma_u$. Here we assume $t \in \Sigma_u$ and $u' \in \Sigma_u$; the argument for the other cases are the same (in fact easier).

Observe that $u' \in \Sigma_u$ implies that $w' \in \Sigma_w$. Since $t \in \Sigma_u$, any (t, w')-directed path τ' intersects $\gamma \cup \gamma'$. Lemma E.2 implies that τ' cannot intersect γ'. So, it intersects γ_1 or γ_2. Without loss of generality assume that τ' intersects γ_1, and x' is any vertex in the intersection. Then, $\delta = \gamma_1[v_1, x'] \cdot \tau'[x', w'] \cdot (w' \rightarrow v'_1)$ is a directed (v_1, v'_1) path.

Similarly, any directed (t, w)-path, τ, intersects γ'_1 or γ'_2.

Case 1: If there is a vertex $x \in \tau \cap \gamma'_1$, then there is a (v'_1, v_1)-directed path $\delta' = \gamma'_1[v'_1, x] \cdot \tau[x, w] \cdot (w \rightarrow v_1)$, and $\delta \cdot \delta'$ is a cycle of length at least two contradicting the assumption that G is a DAG.

Case 2: If there is a vertex $x \in \tau \cap \gamma'_2$, then there is a (v'_2, v_1)-directed path $\delta' = \gamma'_2[v'_2, x] \cdot \tau[x, w](w \rightarrow v_1)$, and $\delta' \cdot \delta$ is a (v'_2, w')-directed path contradicting either the assumption that G has not cycle of length larger than one (if $w' \rightarrow v'_2$) or the assumption that f' is irreducible (otherwise).

□

We may finally complete our proof of Lemma E.5.

Lemma E.5. *Let G be a maximally triangulated DAG embedded on a surface Σ of genus g. Then, the total degree of irreducible faces of G is at most $12g$.*

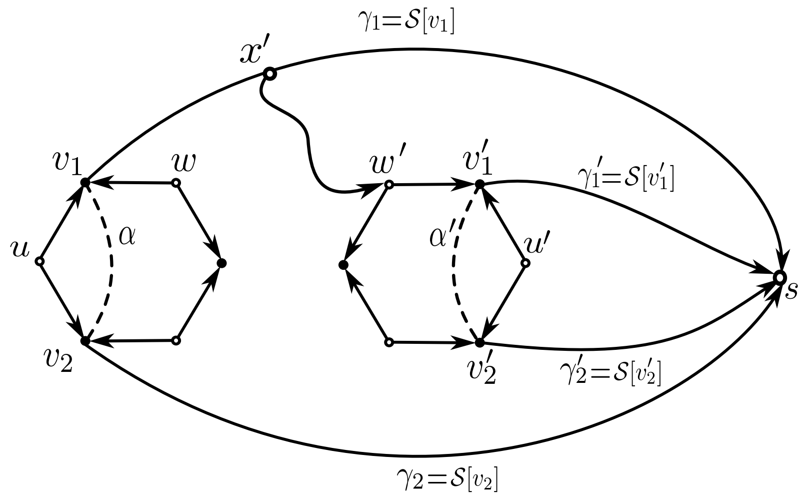
$\gamma_1 = \mathcal{S}[v_1]$
x'
v_1
w
w'
v_1'
$\gamma_1' = \mathcal{S}[v_1']$
u
α
α'
u'
s
v_2
v_2'
$\gamma_2' = \mathcal{S}[v_2']$
$\gamma_2 = \mathcal{S}[v_2]$

Figure 6. The setting for Lemma E.4.

Proof: Consider a backward spanning tree $\mathcal{S}$ rooted at s. Let $u_1, u_2, \ldots, u_k$ be the list of outgoing vertices (with possible multiplicity) on all irreducible faces of G; for $1 \le i \le k$ assume u_i is on the irreducible face f_i. By construction, the cycles (closed walks) $C(T, f_i, u_i)$ $(1 \le i \le k)$ are mutually non-crossing. Lemma E.3 implies that for any $1 \le i \le k$, $C(\mathcal{S}, f_i, u_i)$ is non-separating, and Lemma E.4 implies that for any pair $1 \le i, j \le k$, $C(\mathcal{S}, f_i, u_i)$ and $C(\mathcal{S}, f_j, u_j)$ are non-homotopic. It follows that $k \le 6g$; see [11, Lemma 2.1].

On the other hand, k equals half of the total degree of the irreducible faces and the statement follows. $\square$

The following lemma states that we can compute a maximally triangulated DAG.

Lemma 5.2. *Let G be a DAG with possible self-loops embedded on a surface Σ such that t and s are the only source and sink. Then, there is an $O(n^2)$ time algorithm to compute a maximally triangulated DAG, G_δ, embedded on Σ that has the same number of forward (t,s)-cuts. Further, G_δ has $O(g)$ irreducible faces.*

Proof: Lemma 5.1 implies that for any pair of vertices $u, v \in V$ such that $u \rightsquigarrow v$ we can add $u \to v$ without changing the total number of forward (t,s)-cuts. In particular, we can also add self-loops without changing the total number of forward (t,s)-cuts.

First, in $O(n^2)$ time, for all pairs of vertices $u, v \in V$ we figure out whether $u \rightsquigarrow v$ by running BFS's from all vertices of G. Then, for each vertex $u \in V$, we add all $u \to v$ edges such that u and v are on a same face and $u \rightsquigarrow v$. This takes $O(n)$ time to detect in the BFS tree, and the graph embedded can be updated in the same amount of time. It is straight forward to check that the resulting graph, G_δ, is maximally triangulated. Then, Lemma E.5 implies that G_δ contains $O(g)$ irreducible faces. $\square$

F Sampling minimum cuts

In this Appendix, we given an algorithm to sample a minimum (s,t)-cut from a graph uniformly at random. Let $G = (V, E)$ be a directed acyclic graph plus a set of loops embedded on a surface Σ of genus g with a unique source t and unique sink s. Let G^* be the dual graph of G and let $\Sigma' = \Sigma \setminus (t^* \cup s^*)$. By Theorem A.1, it suffices to give an algorithm to sample forward (t,s)-cuts in G.

Our sampling algorithm combines the ideas from earlier in this paper with the algorithm given in [2]. We assume G is embedded as a triangulation without loss of generality (see Section 5) and run the counting algorithm given in Section 4. We assume familiarity with the counting algorithm as given.

Our counting algorithm enumerates weighted triangulations of a dualized polygonal schema with a particular *arc crossing signature* relative to a system of $2g+1$ arcs. For each such triangulation, it counts the directed cycles in G^* that correspond to the crossing sequences represented in the triangulation. See Section 4 for details. For each such triangulation Δ_i, let c_i be the number of collections of directed cycles corresponding to Δ_i. Our sampling algorithm samples a single weighted triangulation where each triangulation Δ_i is picked with probability $c_i / \sum_k c_k$.

Let Δ be the triangulation sampled. Our *counting* algorithm computes an abstract collection of cycles corresponding to Δ. For each cycle in the abstract collection, it then counts the number of real cycles with the same *crossing sequence* relative to a system of $4g+2$ paths. Our sampling algorithm just needs to pick a cycle uniformly at random for *each* of these crossing sequences. Let X be one such crossing sequence.

Given X, our counting algorithm creates a directed acyclic graph G_X. It then counts the directed paths in G_X between several pairs of endpoints. For each pair of endpoints $((f_i^*, \varepsilon), (f_i^*, X))$, let d_i be the number of directed paths between (f_i^*, ε) and (f_i^*, X) in G_X. Our sampling algorithm picks a pair of endpoints where each pair $((f_i^*, \varepsilon), (f_i^*, X))$ is picked with probability $d_i / \sum_k d_k$.

Finally, we describe how to sample a directed path between a pair of endpoints (f^*, ε) and (f^*, X). Let $x_0 = (f^*, X)$. For every $k = 1, 2, \ldots$, our sampling algorithm selects x_k from the set of immediate predecessors to x_{k-1} with probability proportional to the number of paths between (f^*, ε) and the predecessor. The reverse of $x_0, x_1, \ldots$ gives us a randomly sampled path in G_X or equivalently a randomly sampled cycle in G^*.

All of the information required for the sampling algorithm is computed by the counting algorithm. As a byproduct of our counting algorithm, we obtain an enumeration of all $2^{O(g)}$ possible weighted triangulations. We can pick a random triangulation Δ from this list in $O(g)$ time. Then, we need $O(g^2 n)$ time to sample a cut that corresponds to Δ. In fact, the above procedure runs in only $O(g^2 n)$ additional time over the time it takes to count in the first place. We can actually sample several cuts in $O(g^2 n)$ time each while relying on the preprocessing done by only a single instantiation of the counting algorithm.