Discrete Comput Geom (2014) 52:450–475
DOI 10.1007/s00454-014-9623-4

Counting and Sampling Minimum Cuts in Genus g Graphs

Erin W. Chambers · Kyle Fox · Amir Nayyeri

Received: 17 September 2013 / Revised: 16 May 2014 / Accepted: 8 July 2014 /
Published online: 3 September 2014

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 computing the number of minimum (s,t)-cuts in G is fixed-parameter tractable in g. Specifically, we give a $2^{O(g)}n^2$ 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 no 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 minimum cuts once, can sample an (s,t)-minimum cut uniformly at random in $O(n \log n)$ time per sample.

Keywords Topological graph theory · Computational topology · Minimum cuts

Mathematics Subject Classification 68W05

1 Introduction

Given a weighted directed graph $G = (V, E)$ with n vertices and m edges on an orientable surface Σ of genus g with two vertices $s, t \in V$, we consider the prob-

E. W. Chambers
Department of Mathematics and Computer Science, Saint Louis University, Saint Louis, MO, USA
e-mail: echambe5@slu.edu

K. Fox
Department of Computer Science, Duke University, Durham, NC, USA
e-mail: kylefox@cs.duke.edu

A. Nayyeri
School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA
e-mail: nayyeria@eecs.oregonstate.edu

lem of counting the minimum (s,t)-cuts of G. This problem in general graphs is #P-complete [54], and can be reduced to the problem of counting maximal antichains in a poset [1]. Ball and Provan [1] first considered the problem of counting minimum cuts and gave an algorithm to compute the number of minimum cardinality (s,t)-cuts in an (s,t)-planar graph (where the source and sink are on the same face). Later, Bezáková and Friedlander [2] generalized the algorithm for arbitrary locations of s and t in a planar graph and arbitrary edge weights.

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 [1,16,38,50].

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 [8]. Being able to count minimum cuts is closely related to sampling such cuts [37], implying that 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 [56] 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; 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 [45,53]; single-source shortest paths [15,33,41,44,48,57]; multiple-source shortest paths [9,40]; replacement paths [23,60]; graph and subgraph isomorphism [20,21,29,34,46]; and approximation of several NP-hard problems [5–7,18,21].

Of course, much work is also dedicated to computing cuts and flows in planar graphs, both undirected [11,28,31,35,36,43,55] and directed [4,58]. The current best algorithms run in $O(n \log\log n)$ time in undirected graphs [36] and $O(n \log n)$ time

in directed graphs [4], considerably improving on running times for arbitrary sparse 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 [14,15,24,25], minor-free families [12], and graphs with bounded treewidth [30]. 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 [24] and $g^{O(g)}n \log\log n$ time [36].

1.2 Our Contributions

In this paper, we describe a 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 $2^{O(g)}n^2$ 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 [39,59]. Since counting the number of cuts is generally #P-complete [54], 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. We make the second assumption so that vertices with no effect on the connectivity of s and t do not influence the number of minimum (s,t)-cuts. See [2, Sect. 4].

Our approach uses a connection between cuts and (co-)homology in a non-trivial way, as well as generalizing 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 uses a bijection between forward (t,s)-cuts and circulations of a certain homology class in the dual graph. These reductions are described in Sects. 3 and 4, respectively. The characterization of cuts using circulations in the dual is not original to this work [14, 24,25,52], but there are some key changes in our characterization and how we use it for counting cuts. In [14,24,25], minimum cuts in *undirected* graphs are characterized using homology with coefficients in $\mathbb{Z}_2$. However, in our case, G has directed edges, so we must use coefficients in $\mathbb{Z}$. Integer coefficients are used in [52] 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 Sect. 5 we describe an algorithm to compute the number of dual circulations in a certain homology class if the primal directed acyclic graph is triangulated. Finally, in Sect. 6 we generalize our algorithm to work for non-triangulated primal graphs as well. Unlike many other problems where triangulating an input graph without

changing the output 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 allow us to limit the complexity of our modified surface.

Our algorithm can also be used to *sample* a minimum (s, t)-cut uniformly at random. 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 $O(n \log n)$ time per sampling, so several samples can be computed quickly. We describe the sampling algorithm in Sect. 7.

2 Preliminaries

We give a brief overview of necessary definitions for tools we use. For full coverage, we direct the reader to existing books and surveys on topology [32,49], computational topology [19,61], and graphs on surfaces [17,47].

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. Our graph may contain parallel edges with identical endpoints; in this case the edge referred to by $(u \to v)$ should be clear from context. If $u = v$ then e is a *loop*. The *indegree* of a vertex v is the total number of edges in E whose heads are v. Similarly, the *outdegree* of v is the total number of edges in E whose tails are v, and the simple *degree* of a vertex v is the sum of its indegree and outdegree. A directed (u, v)-walk in G is a sequence of vertices $W = (u = w_1, w_2, \ldots, w_k = v)$ such that $(w_i \to w_{i+1}) \in E$ for all $1 \leq i < k$; we sometimes denote a (u, v)-*walk* by $u \rightsquigarrow v$; we also use $u \rightsquigarrow v$ to designate that there is a a directed walk from u to v. A (u, v)-walk is *closed* if $u = v$, it is a (directed) *path* if it has no repeated vertices, and it is a (directed) *cycle* if $u = v$ is its only repeated vertex. An *undirected walk*, *path*, or *cycle* has the same definition as above, except we allow either $(w_i \to w_{i+1}) \in E$ or $(w_{i+1} \to w_i) \in E$ for any consecutive vertices in the vertex sequence. Note that we still respect the ordering of vertices in an undirected walk even though we are ignoring the orientation of its edges within G. 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 (excepting the first instance of v in W_2). Let $rev(W_1)$ denote the reversal of walk W_1.

A directed graph G is a *directed acyclic graph (DAG)* 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 $T = V \setminus S$ as different *sides* of the cut. Given two vertices a and b with $a \in S$ and $b \in T$, we say S is an $(a, b) - cut$. Further S is a *forward(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 exactly one of u and v lie in S. In particular, 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 $\Gamma^+(S)$ to denote the set of all edges that cross S in the *forward* direction. We define $\Gamma^-(S)$ as the set of edges that cross S in the backward direction. A walk W crosses a cut S k times if there are k edges of W that cross S. Given an edge capacity function $c : E \to \mathbb{R}^+$, the value of a cut S is $\sum_{e\in\Gamma^+(S)} c(e)$ (note that this sum is taken over edges in $\Gamma^+(S)$ only). A *minimum (s,t)-cut* of G with respect to capacity function c is an (s, t)-cut of least value.

A *spanning tree* τ of a connected graph $G = (V, E)$ is a maximal subgraph of G that contains no undirected 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 τ is a directed tree with root r. Similarly, τ is a *backward spanning tree* with root r if it contains a directed path from every vertex $u \in V$ to r.

2.2 Surfaces and Embeddings

A *surface* (or a 2-manifold) with boundary is a compact Hausdorff space in which every point has a neighborhood homeomorphic to the Euclidean plane or the closed half plane. The union of all points in open neighborhoods homeomorphic to the closed half plane compose the *boundary* of the surface. Every boundary component is homeomorphic to a circle. A *cycle* in a surface Σ is a continuous function $\gamma : S^1 \to \Sigma$, where S^1 is the unit circle; the cycle γ is *simple* if γ is injective. A *loop* is a cycle along with a designated basepoint on the cycle. A *path* in a surface Σ is a continuous function $p : [0, 1] \to \Sigma$; the path p is *simple* if p is injective. Two paths or cycles in a surface *cross* if no continuous infinitesimal perturbation makes them disjoint; if such a perturbation exists, then the paths are *non-crossing*. Two paths p and q in Σ are *homotopic* if one can be continuously deformed into the other without changing their endpoints. More formally, a *homotopy* between p and q is a continuous map $h\colon [0, 1] \times [0, 1] \to \Sigma$ such that $h(0, \cdot) = p$, $h(1, \cdot) = q$, $h(\cdot, 0) = p(0) = q(0)$, and $h(\cdot, 1) = p(1) = q(1)$.

A simple cycle γ is *separating* 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. A (not necessarily simple) *loop* ℓ is *contractible* if it is homotopic to its basepoint. When ℓ is simple, these two definitions of contractible are equivalent. 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 between the edges' vertices' corresponding points. A *face* of an embedding is a maximal connected subset of 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 (clockwise) 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 all embeddings are cellular. We assume $g = O(n)$, because otherwise the trivial algorithm of enumerating all s,t-cuts and counting the ones that are minimum would meet our desired time bound. Finally, we assume the input graph G is simple. Self-loops do not appear in minimum s,t-cuts and parallel edges can be replaced by single edges without increasing the genus of the embedding. By Euler's formula, our assumptions guarantee $m = O(n)$.

2.3 Topology

Let $G = (V, E)$ be a directed 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* 0, the edges as *cells of dimension* 1, and the faces as *cells of dimension* 2. A $k-chain$ ($0 \leq k \leq 2$) is a function that assigns a set of real 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 $\{0, 1\}$-*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 $\partial\phi(v) = \sum_{(v\to w)\in E} \phi(v \to w) - \sum_{(w\to v)\in E} \phi(w \to v)$ for each $v \in V$. 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 integral 1-chains that are circulations in G. The cycle space $Z(G)$ is isomorphic to $\mathbb{Z}^{|E|-|V|+1}$ [32]. A *trivial circulation*, a *non-negative circulation* and a $\{0, 1\}$-*circulation* are special cases of these 1-chains in the cycle space.

We say a set of undirected 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 $\mathcal{C}$ in the forward direction minus the number of times it appears in the backwards direction. Note that if $\mathcal{C}$ contains only *directed* cycles, then it generates a non-negative circulation. We will abuse terminology slightly 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 $\partial\alpha(e) = \partial\alpha(\text{right}(e)) - \partial\alpha(\text{left}(e))$, where $\text{left}(e)$ and $\text{right}(e)$ are the faces to the left and right of the directed edge e. If e's right (or left) side is a boundary component then we take $\partial\alpha(\text{right}(e)) = 0$ (or $\partial\alpha(\text{left}(e)) = 0$). 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 integral 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 [32].

Two integral 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 $Z(G)/B(G) \cong \mathbb{Z}^{2g+\max\{0,b-1\}}$ 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 and Triangulations

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 if 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.

Let $G = (V, E)$ be a graph embedded on a surface Σ of genus g. A tool we use is a *tree-cotree* decomposition, introduced by Eppstein [22], where E is decomposed into three disjoint sets (τ, L, C) such that τ is a spanning tree of G, C^* is a spanning tree of G^*, and L is composed of $2g$ edges. For any $e \in L$, $\tau \cup e$ contains exactly one undirected cycle, which is non-separating.

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

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 $v \in V \backslash \{s, t\}$. For a capacity function $c : E \to \mathbb{R}^+$, the flow ϕ is *feasible* if $0 \le \phi(e) \le c(e)$ for all $e \in E$. Each flow can be decomposed to a set of weighted paths and cycles; here, each edge in a particular cycle or path has the same amount of flow present. A flow is *acyclic* if it can be decomposed into a set of simple (s, t)-paths. The value of an (s, t)-flow ϕ is $\partial\phi(s) = \sum_{s \to v} \phi(s \to v) - \sum_{u \to s} \phi(u \to s)$. A *maximum(s,t)-flow* with respect to a capacity function c is a feasible flow of highest value. A well known theorem of Ford and Fulkerson [26] states that for any capacity function c, the value of a maximum feasible flow is equal to the value of a minimum (s, t)-cut.

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* $\boldsymbol{G_\phi}$ contains two edges $(u \to v)$ and $(v \to u)$ for every edge $e = (u \to v)$ in G (the residual graph G_ϕ may have multiple edges from v to u for this definition). The edges are weighted so $c_\phi(u \to v) = c(e) - \phi(e)$ and $c_\phi(v \to u) = \phi(e)$.

3 Minimum Cuts and Forward Cuts

Following Bezáková and Friedlander [2], we begin by reducing the problem of counting the minimum (s, t)-cuts in G to the problem of counting forward $(\tilde{t}, \tilde{s})$-cuts in a

related graph $\tilde{G}$. Let G be a directed acyclic (multi-)graph. Let a and b be vertices of G. Finally, let S be a subset of vertices such that $a \in S$ and $b \notin S$. Recall that S is a forward (a, b)-cut of G if there is no edge $u \to v$ such that $v \in S$ and $u \notin S$. We begin by considering the following theorem:

Theorem 3.1 (Bezáková–Friedlander [2, Theorem 4]) *Let $G = (V, E, c)$ be a (directed) graph 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 graph (DAG) $\tilde{G} = (\tilde{V}, \tilde{E})$ 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 $\tilde{G}$. Moreover, $|\tilde{V}| \leq |V|$, $|\tilde{E}| \leq |E|$, and it is possible to construct $\tilde{G}$ in time $O(|V|^3 + |E|^2)$. Also, if G is planar, then $\tilde{G}$ is planar as well, and $\tilde{G}$ can be constructed in time $O(|V| \log |V|)$.*

The proof of this theorem relies upon constructing a maximum flow f for the graph G and removing directed cycles containing flow from f. The graph $\tilde{G}$ is then formed by taking the residual graph G_f, removing 0 weight edges, 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. The final part of the theorem comes from the existence of fast algorithms for computing maximum flows in planar graphs [3] as well as the fact that contraction of an edge in a planar graph yields a planar graph with an inherited embedding. Corollary 3.2 follows immediately.

Corollary 3.2 (Bezáková–Friedlander [2, Corollary 5]) *Suppose there exists 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 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. The current best running time for computing a maximum flow in an arbitrary graph where $m = O(n)$ is $O(n^2/\log n)$ due to a recent algorithm of Orlin [51]. The proof of Theorem 3.1 implies $\tilde{G}$ can be computed in $O(n^2)$ time, because computing a maximum flow, forming the residual graph, and removing cycles from $\tilde{G}$ takes $O(n^2)$ time total when $m = O(n)$. The proof for Theorem 3.1 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, our algorithm finds strongly connected components in the residual graph as in the original construction. All of the edges are contracted iteratively unless the edge is a loop; these loops are not contracted so that we can retain a cellular embedding of the graph. 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. For the simplicity of exposition, we assume the new graph contains no faces of degree 2 or 1. Our algorithm enforces this assumption by iteratively removing faces of degree 2 or 1 by deleting one of their edges. Again, this process can be easily completed in $O(n^2)$ time. Theorem 3.3 follows immediately.

Theorem 3.3 *Let $G = (V, E, c)$ be a (directed) graph 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 graph (DAG) $\tilde{G} = (\tilde{V}, \tilde{E})$, 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 $\tilde{G}$. Moreover, $\tilde{G}$ has $O(n)$ vertices and edges, $\tilde{G}$ has no faces of degree 2 or 1, and $\tilde{G}$ can be computed in $O(n^2)$ time.*

While this graph is not a strictly DAG, our algorithm is resilient to the existence of self loops. As an alternative to the above procedure, our algorithm can perform the original contraction procedure of Bezáková and Friedlander [2] and then compute a new embedding of $\tilde{G}$ without loops onto a new surface of genus at most g [39], but we must consider loops anyway due to technicalities introduced by the procedure in Sect. 6. Note the minimum embedding of a connected graph is known to be cellular [59].

4 Forward Cuts and Cocirculations

Based on Theorem 3.3 and Corollary 3.2, we focus on the problem of counting forward (t, s)-cuts in a directed acyclic graph; therefore, from here on G is a directed acyclic graph, 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]. While the original claim deals with single cycles in planar graphs, our lemma describes more general circulations. 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 4.1 *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 $\phi((u \to w)^*) > 0$. Vertex u is reachable from t since t is the only source in G. Likewise, w can reach s. Therefore, there exists a simple *directed* path $p = (v_0 \to v_1 \to v_2 \to \cdots \to v_k)$ from t to s through $u \to w$, where $v_0 = t$, $v_k = s$; also for some $0 \le i < k$, $v_i = u$ 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 in Σ'). 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, either $\alpha(v_0^*) > 0$ or $\alpha(v_k^*) < 0$, a contradiction. □

Theorem 3.3 reduces the problem of counting minimum cuts to the problem of counting forward cuts in a DAG that possibly 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 [14,24,25], but require substantially more technical detail to work with integer homology.

Lemma 4.2 *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 so $v = u$, 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 4.3 *Let $\phi : \mathrm{E}^* \to \mathbb{Z}$ 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 necessarily forward) (t, s)-cut T where for every edge e that crosses T in the forward direction we have $\phi(e^*) \geq 1$; i.e., the dual of the collection of edges with value at least 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 $p = (v_0 \to v_1 \to \cdots v_k)$ be any directed path in G such that $v_0 = t$ and $v_k = s$. Suppose there exists no edge e of p with $\phi(e^*) \geq 1$. We have $\alpha(v_1^*) - \alpha(v_0^*) \geq 1$ and $\alpha(v_{i+1}^*) - \alpha(v_i^*) \geq 0$ for all $0 < i < k$. 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 $\phi(e^*) \geq 1$. The collection of such edges separates t from s.

Now, let $T \subsetneq V$ be the set of vertices reachable from t using only edges e with $\phi(e^*) = 0$. Recall $\Gamma^+(T)$ and $\Gamma^-(T)$ are the sets of edges that cross T in the forward and backward directions, 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'. Further, G' is acyclic, because every edge in G' not also present in G lies on that forward (t, s)-cut. By Lemma 4.2, 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. Since homology is a transitive property and both are homologous to ∂t^*, we know that 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 on G^*. Thus, Lemma 4.1 implies that ϕ' should be trivial. It follows that $\phi = \phi_T$. Circulation ϕ_T is non-negative, so T is a forward (t, s)-cut. Further, ϕ is a $\{0, 1\}$-circulation. □

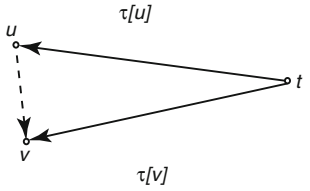
Lemmas 4.2 and 4.3 imply a bijection between forward (t, s)-cuts in G and $\{0, 1\}$-circulations in G^* of a particular homology class. We immediately get the following theorem which drives the remaining algorithm design and analysis.

Theorem 4.4 *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'.*

5 Counting Cuts 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 Sect. 6. We first recall 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 directed cycles in G^* found by tracing along edges e of G where $\phi(e^*) = 1$. Theorem 4.4 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 4.3 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 or simple. We focus on counting these collections of cycles.

5.1 Crossing Sequences and Vectors

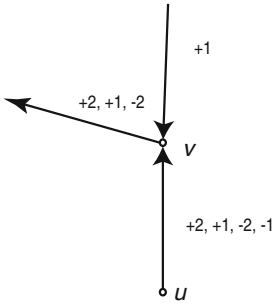


Our algorithm begins with the following construction. It creates a tree-cotree decomposition (τ, L, C) where τ is an arbitrary directed spanning tree of G rooted at t. Euler's formula implies that L contains exactly $2g$ directed edges which we label $u_1 \to v_1, \ldots, u_{2g} \to v_{2g}$. Let $\tau[v]$ 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 closed walk $p_i^+ \cdot rev(p_i^-)$. 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 [13].

Lemma 5.1 *Let G be a directed acyclic graph with single edge loops, 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. □

Corollary 5.2 *At most one edge of each path p_i^+, p_i^- crosses T.*



Let γ be a directed cycle in G^*. We define the *crossing sequence* of γ to be its cyclic order of crossings of $\{p_0^-, p_0^+, \ldots, p_{2g}^-, p_{2g}^+\}$; this is well defined if we perturb the paths infinitesimally to be disjoint and non-crossing. More formally, let X be an arbitrary crossing sequence corresponding to a 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^-. Suppose we perturb the arcs of P so they are disjoint except at t. Abusing notation, we assign each edge e^* of G^* a computed crossing sequence $X(e^*)$ for the perturbed arcs 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 $s \to s'$. 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 each edge $u_i \to v_i$ in L, set $X((u_i \to v_i)^*) = i$. For each edge e^* outside τ or L set $X(e^*) = \varepsilon$. Finally, for each edge $u \to v$ in τ in bottom up order from the leaves, set $X((u \to v)^*)$ using the following iterative subroutine. The crossing sequence $X((u \to v)^*)$ begins as the empty sequence ε. For each edge e' incident to v in clockwise order starting with the edge immediately clockwise to $u \to v$, append to $X((u \to v)^*)$ the crossing sequence $X(e'^*)$ if e' has tail v. If e' does not have tail v, then $e' = (u_i \to v_i)$ for some index i. Append to $X((u \to v)^*)$ the element $-i$. As each edge appears at most twice in each arc of P, the above procedure to construct these crossing sequences runs in $O(gn)$ time.

Lemma 5.3 *For any directed dual cycle c, we have $X(c)$ equal to the concatenation of crossing sequences for c's individual edges in order.*

Proof It suffices to prove that for any edge e, the crossing sequence $X(e^*)$ accurately lists the arcs of P crossed by e^* in order. The statement is trivially true for any e outside τ or L. Each edge $u_i \to v_i$ in L appears in exactly one member of P so the statement is true for those edges as well. Finally, for any edge $e = u \to v$ of τ, assume the computation is accurate for all descendants of e in the rooted tree τ. If $u \to v$ or $v \to u$ appears in any arc P it must be immediately followed by or proceeded by

an edge incident to v. Let e' be the first edge in the clockwise rotation system of v following e. Edge e' is one of $s \to s'$, a member of L, a descendent of e in τ, or an edge outside of L and τ so we may assume $X(e'^*)$ is accurate (and possibly empty). If $X(e'^*)$ is non-empty and e' has v as its tail then the arcs passing through e' must be the leftmost arcs passing through $u \to v$ to avoid a crossing and e'^* passes through them in the same order and direction as e^*. If $X(e'^*)$ is non-empty and e' has v as its head, then the arcs passing through e' must still be the leftmost arcs passing through $u \to v$. However, e'^* passes through the arcs in the opposite order and direction as e^*. In all cases, we see the concatenation of $X(e'^*)$ or its reversal is correct. The remainder of $X(e^*)$'s computation is correct by induction. □

Now, let $\mathcal{C}$ be an arbitrary set of *undirected* 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 computing the net number of times the cycles in $\mathcal{C}$ cross members of P. For any undirected cycle $\gamma = (f_1, f_2, \ldots, f_k)$ in G^* and index $0 \leq i \leq 2g$, let $x_i^+(\gamma)$ be the number of edges $(f_j \uparrow f_{j+1})$ in p_i^+ minus the number of edges $(f_{j+1} \uparrow f_j)$ in p_i^+. Let $x_i^-(\gamma)$ be defined similarly for p_i^-. Equivalently, if γ is directed, then $x_i^+(\gamma)$ is the number of times i appears in the crossing sequence for γ and $x_i^-(\gamma)$ is the number of times $-i$ appears in the crossing sequence. Let $x_i(\gamma) = x_i^+(\gamma) - x_i^-(\gamma)$. Similar to [14,24], 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 [24, Lemma 3.2].

Lemma 5.4 *A set of undirected 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 arc crossing vector 0. We see

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

where $\sum_{v \in V} \alpha(v^*) \cdot \partial v^*$ denotes the collection of cycles where for each vertex v, ∂v^* appears $\alpha(v^*)$ times.

Now, suppose $x(\mathcal{C}) = 0$. We create a graph G^+ by modifying G^* as follows. For every arc $p_i \in P$ and for every edge e in p_i, we subdivide $e^* = (u \to v)$ by replacing e^* with a vertex v_{e^*} and edges $(u \to v_{e^*})$ and $(v_{e^*} \to w)$. Then, for every adjacent pair of edges e_1, e_2 in p_i, we add an edge $(v_{e_1^*} v_{e_2^*})$ to G^+, subdividing a face originally incident to both e_1^* and e_2^*. Essentially, we augment G^* with paths that follow the images of loops in P. See Fig. 1. We 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 cycles $\gamma_1, \gamma_2 \in \mathcal{C}$ and the image of some arc p_i where γ_1 crosses p_i from left to right through a and γ_2 crosses

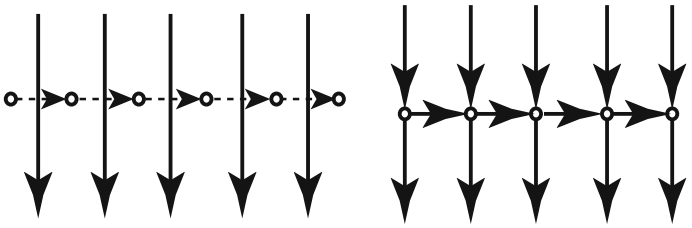


Fig. 1 *Left* An arc p (*dashed*) and its dual edges (*solid*). *Right* A new path in G^+ follows the image of p

p_i from right to left through b. Note that γ_1 and γ_2 may be the same cycle. 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 the image of p_i. Alter $\mathcal{C}$ by replacing γ_1 and γ_2's crossings through a and b with copies of $p_i[a, b]$ and $rev(p_i[a, b])$. The transformation may change the number of cycles in $\mathcal{C}$. However, 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. □

Corollary 5.5 *Two sets of undirected cycles $\mathcal{C}$ and $\mathcal{C}'$ are homologous if and only if $x(\mathcal{C}) = x(\mathcal{C}')$.*

5.2 Enumerating Crossing Sequences

Recall that for a directed cycle γ in G^*, the crossing sequence is the 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 method based on previous works [13,14,27] 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; see Fig. 2. 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^+ or p_i^- corresponds to two edges 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 4.3, the arcs will be non-crossing in the interior of S, and by Corollary 5.2 each edge of the schema contains at most one endpoint of any arc; in particular, no two arcs have endpoints on the same edge of the 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

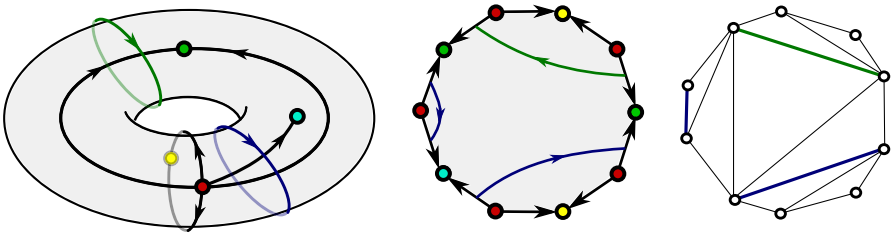


Fig. 2 Building the polygonal schema (*left* to *right*): The set of paths P is given in *black*, and the dual of a forward cut is given in *green* and *blue*; polygonal schema S; the triangulated dual of S

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 weighted triangulations of the abstract polygonal schema are *equivalent* if and only if they are identical ignoring zero weight edges. See Fig. 2 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. 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 5.5. 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. Our algorithm then 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.

5.3 Counting Cycles with a Given Crossing Sequence

Let X be an arbitrary crossing sequence corresponding to an abstract directed cycle γ. We construct the following graph G_X along with a mapping from vertices and edges of G_X to G^*. The vertices 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 [10,42]. 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', which we are constructing as is common in previous work [42]. The next lemma follows from our construction.

Lemma 5.6 *Let f be a face of G on the right side of an edge in path p_i^+ (p_i^-). If X ends with i ($-i$), then there exists a bijection between cycles in G^* with crossing sequence X with first vertex f^* and paths in G_X from (f^*, ε) to (f^*, X).*

Lemmas 5.4 and 4.1 imply the following lemma.

Lemma 5.7 *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 on the right side of an edge in 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 5.8 *Let X be a non-empty crossing sequence of an abstract cycle. Then, there is an $O(|X|n^2)$ time algorithm to compute the number of directed cycles with crossing sequence X.*

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 number 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 5.9 *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.*

6 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 Sect. 5 to count the forward cuts in G_Δ and so in G. The following lemma has a key role in constructing G_Δ.

Lemma 6.1 *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.*

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

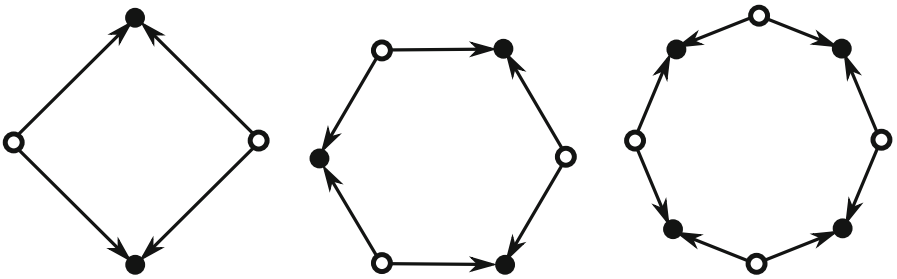


Fig. 3 Irreducible faces of degree four, six and eight; *circles* outgoing vertices; *bullets* incoming vertices

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

Now consider any forward (t, s)-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 $G \cup (u \to v)$ as well.

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

A face f of G with degree at least 4 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 Fig. 3.

Lemma 6.2 *Let 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 $(u \to v)$ 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 $(w \to v) \in E$. Because u and v are identical, it follows that $(w \to u) \in E$, in particular $w \rightsquigarrow u$. Since the boundary of an irreducible face is a closed walk of length at least 4, w and u cannot be adjacent on the boundary of f, which implies that f is not irreducible. □

An embedded DAG with possible self loops is *maximally triangulated* if and only if any non-triangle face of it is irreducible. Applying Lemma 6.1 lets us create a maximally triangulated graph G_δ:

Lemma 6.3 *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 as G.*

Proof Lemma 6.1 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.

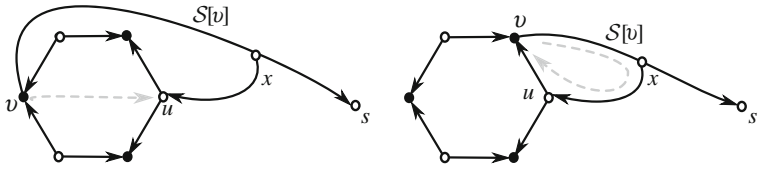


Fig. 4 The setting for Lemma 6.4

First, in $O(n^2)$ time, for all pairs of vertices $u, v \in V$, our algorithm figures out whether $u \rightsquigarrow v$ by running breadth first searches from all vertices of G. It then iteratively adds edges $(u \to v)$ such that $u \rightsquigarrow v$ and adding edge $(u \to v)$ subdivides a face of degree $d \geq 4$ into two faces of degree strictly less than d.

It is straight forward to check that the resulting graph G_δ is maximally triangulated. □

Unfortunately, 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 can prove that the total number of irreducible faces is $O(g)$ in any maximally triangulated DAG.

Let v be a vertex on the boundary of an irreducible face f of the maximally triangulated graph G_δ. 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 Fig. 3.

Let $\mathcal{S}$ be a backwards spanning tree of G with root s. Let $\mathcal{S}[v]$ denote the directed path from vertex $v \in V$ to s.

Lemma 6.4 *Let f be an irreducible face and u and v be an outgoing vertex and an incoming vertex 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 exists 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 1, we have $(u \to v) \notin E$, which in particular implies that u and v are not adjacent. Thus, f must be reducible; see Fig. 4. □

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 $L(\mathcal{S}, f, u)$ to be the undirected loop $rev(\mathcal{S}[v_1]) \cdot (v_1 \to u) \cdot (u \to v_2) \cdot \mathcal{S}[v_2]$. We define $C(\mathcal{S}, f, u)$ to be the undirected cycle composed of directed edges from G_δ appearing in $L(\mathcal{S}, f, u)$ exactly once. See Fig. 5, left.

Lemma 6.5 *Let 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

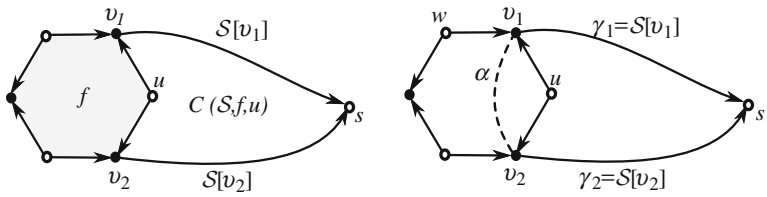


Fig. 5 *Left* The definition of $C(\mathcal{S}, f, u)$. *Right* The proof of Lemma 6.5

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 Fig. 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. Cycle 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 6.4. □

Lemma 6.6 *Let f and f' be distinct irreducible faces and u and u' be two 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. Also, let $y, y' \in V$ be the other neighbors of v_2 and v_2' 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 $\gamma' = \alpha' \cup \gamma_1' \cup \gamma_2'$. See Fig. 6 for illustration.

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.

We prove that if $\gamma \cup \gamma'$ is separating then the following three contradictory statements hold: (1) ($v_1 \neq v_1'$ and $v_1 \rightsquigarrow v_1'$) or ($v_2 \neq v_1'$ and $v_2 \rightsquigarrow v_1'$), (2) ($v_1 \neq v_1'$ and $v_1' \rightsquigarrow v_1$) or ($v_1 \neq v_2'$ and $v_2' \rightsquigarrow v_1$), and (3) ($v_1 \neq v_2'$ and $v_1 \rightsquigarrow v_2'$) or ($v_2 \neq v_2'$ and $v_2 \rightsquigarrow v_2'$).

The following case analysis shows that the three statements are in contradiction. There are eight cases to consider. We group the cases to make the analysis concise. Each group is labeled by three characters from $\{0, 1, *\}$ showing which condition of the three statements above is true. Specifically, a 0 implies the first condition is true, a 1 implies the second condition is true, and a $*$ implies one or both conditions are true. For example in $(01*)$ indicates that the first condition of statement (1), the second condition of statement (2) and one of the conditions of statement (3) are true.

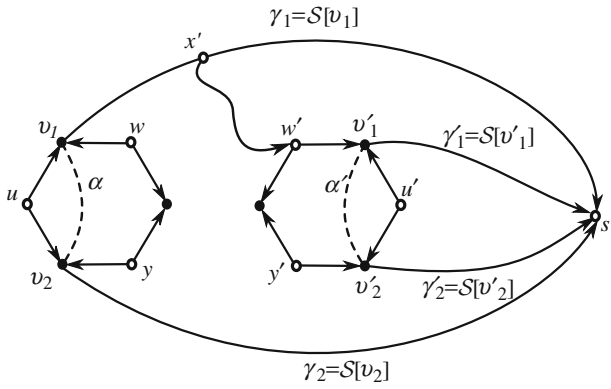


Fig. 6 The setting for Lemma 6.6

Case (00*) If $v_1 \neq v'_1$, $v_1 \rightsquigarrow v'_1$ and $v'_1 \rightsquigarrow v_1$ then $(v_1 \rightsquigarrow v'_1) \cdot (v'_1 \rightsquigarrow v_1)$ is a non-trivial directed cycle.

Case (01*) If $v_1 \rightsquigarrow v'_1$ and $v'_2 \rightsquigarrow v_1$ then $(v'_2 \rightsquigarrow v_1) \cdot (v_1 \rightsquigarrow v'_1)$ is a directed (v'_2, v'_1) walk.

Case (10*) If $v_2 \rightsquigarrow v'_1$ and $v'_1 \rightsquigarrow v_1$ then $(v_2 \rightsquigarrow v'_1) \cdot (v'_1 \rightsquigarrow v_1)$ is a directed (v_2, v_1) walk.

Case (*10) If $v_1 \neq v'_2$, $v'_2 \rightsquigarrow v_1$ and $v_1 \rightsquigarrow v'_2$ then $(v'_2 \rightsquigarrow v_1) \cdot (v_1 \rightsquigarrow v'_2)$ is a non-trivial directed cycle.

Case (*11) If $v'_2 \rightsquigarrow v_1$ and $v_2 \rightsquigarrow v'_2$ then $(v_2 \rightsquigarrow v'_2) \cdot (v'_2 \rightsquigarrow v_1)$ is a directed (v_2, v_1)-walk.

Cases $(00*)$ and $(*10)$ are in contradiction with G being a DAG, case $(01*)$ is in contradiction with f' being irreducible, and cases $(10*)$ and $(*11)$ are in contradiction with f being irreducible.

It remains to prove (1), (2) and (3). The proof of statement (1), ($v_1 \neq v'_1$ and $v_1 \rightsquigarrow v'_1$) or ($v_2 \neq v'_1$ and $v_2 \rightsquigarrow v'_1$), follows. The argument for the other two cases is exactly the same.

Observe that $w', u' \notin \gamma'$, otherwise G contains a directed cycle or a (v'_2, v'_1)-directed path. On the other hand, if $w' \in \gamma$ or $u' \in \gamma$ then there is a directed walk δ from v_1 or v_2 to v'_1 through w' or u'. Since $w' \neq v'_1$ and $u' \neq v'_1$, δ is not trivial, and so it has distinct endpoints (otherwise δ is a non-trivial directed closed walk.) Thus, at least one of the conditions of statement (1) is true.

Hence, we assume $w', u' \notin \gamma \cup \gamma'$, and they are on different sides of the supposedly separating set $\gamma \cup \gamma'$. Suppose $\gamma \cup \gamma'$ separates Σ into two surfaces Σ_u and Σ_w, where $u' \in \Sigma_u$ and $w' \in \Sigma_w$.

We consider the two cases $t \in \Sigma_u$ and $t \in \Sigma_w$.

First, suppose $t \in \Sigma_u$, and let τ be a (t, w')-path. Because $t \in \Sigma_u$ and $w' \in \Sigma_w$ the directed path τ intersects $\gamma \cup \gamma'$. Since, by Lemma 6.4, τ does not intersect γ', it intersects γ, and in particular γ_1 or γ_2. If τ intersects γ_1 at x' (see Fig. 6) then $\delta_1 = \gamma_1[v_1, x'] \cdot \tau[x', w'] \cdot (w' \to v'_1)$ is a (v_1, v'_1)-directed path, and so $v_1 \rightsquigarrow v'_1$. Since $w' \neq v'_1$, δ_1 is not trivial, and so, $v_1 \neq v'_1$ (otherwise δ_1 is a non-trivial directed

closed walk.) If τ intersects γ_2 at x' then $\delta_2 = \gamma_2[v_2, x'] \cdot \tau[x', w'] \cdot (w' \to v_1')$ is a (v_2, v_1')-directed path, and so $v_2 \rightsquigarrow v_1'$. Since $w' \neq v_1'$, δ_2 is not trivial, and so, $v_2 \neq v_1'$ (otherwise δ_2 is a non-trivial directed closed walk.)

Second, suppose $t \in \Sigma_w$, and let τ be a (t, u')-path. Because $t \in \Sigma_w$ and $u' \in \Sigma_u$ the directed path τ intersects $\gamma \cup \gamma'$. Since, by Lemma 6.4, τ does not intersect γ', it intersects γ, and in particular γ_1 or γ_2. If τ intersects γ_1 at x' then $\delta_1 = \gamma_1[v_1, x'] \cdot \tau[x', u'] \cdot (u' \to v_1')$ is a (v_1, v_1')-directed path, and so $v_1 \rightsquigarrow v_1'$. Since $u' \neq v_1'$, δ_1 is not trivial, and so, $v_1 \neq v_1'$ (otherwise δ_1 is a non-trivial directed closed walk.) If τ intersects γ_2 at x' then $\delta_2 = \gamma_2[v_2, x'] \cdot \tau[x', u'] \cdot (u' \to v_1')$ is a (v_2, v_1')-directed path, and so $v_2 \rightsquigarrow v_1'$. Since $u' \neq v_1'$, δ_2 is not trivial, and so, $v_1 \neq v_1'$ (otherwise δ_2 is a non-trivial directed closed walk.) □

Using Lemmas 6.5 and 6.6 we obtain an upper bound on the number of irreducible faces.

Lemma 6.7 *The total degree of irreducible faces of G is at most $12g$.*

Proof Consider a backward spanning tree $\mathcal{S}$ rooted at s. Let $u_1, \ldots, u_k$ be the list of outgoing vertices (with possible multiplicity) on all irreducible faces of G; for $1 \leq i \leq k$ assume u_i is on the irreducible face f_i. By construction, the closed walks $L(\mathcal{S}, f_i, u_i)$ $(1 \leq i \leq k)$ are mutually non-crossing except at s. Lemma 6.5 implies that for any $1 \leq i \leq k$, $L(\mathcal{S}, f_i, u_i)$ is non-contractible; otherwise $C(\mathcal{S}, f_i, u_i)$ would be separating. Finally, Lemma 6.6 implies that for any pair $1 \leq i < j \leq k$, $L(\mathcal{S}, f_i, u_i)$ and $L(\mathcal{S}, f_j, u_j)$ are non-homotopic; otherwise $C(\mathcal{S}, f_i, u_i)$ and $C(\mathcal{S}, f_j, u_j)$ would be homotopic. It follows that $k \leq 6g$; see [13, Lemma 2.1]. As k equals half of the total degree of the irreducible faces, the lemma follows. □

To get rid of irreducible faces we further triangulate G_δ by connecting the vertices that appear on the boundary of irreducible faces to s; Lemma 6.1 implies that we can always add edges to s without changing the total number of forward 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 lemma shows that all irreducible faces can be triangulated by adding only $O(g)$ handles to Σ.

Lemma 6.8 *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 6.3 implies G_δ has $O(g)$ irreducible faces. Lemma 6.1 implies s is not on the boundary of any irreducible face. Additionally, we recall that we collapsed faces of degree 1 and 2 during the creation of $\tilde{G}$.

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 $(v_0, v_1, \ldots, v_{k-1})$ 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 edge $e_i = v_i \to s$ such that (1)

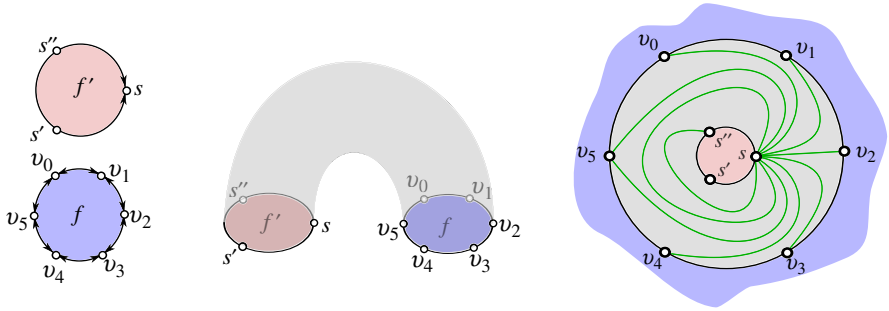


Fig. 7 Triangulating an irreducible face (*left* to *right*): f', a triangle incident to s, and f an irreducible face of degree 6; the handle to connect f to f'; the flat view of the handle as an annulus and its triangulation with *green* edges (all the *green* edges are directed towards s)

for all $0 \leq i < k$, e_i is between $(v_{i\oplus 1}, v_i)$ and $(v_{i\ominus 1}, v_i)$ in the clockwise rotation system edge list of v_i, where $\oplus$ and $\ominus$ are addition and subtraction 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 (s'', s) and (v_1, 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 Fig. 7.

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 Fig. 7. Since all new edges are towards s, Lemma 6.1 implies that adding them does not change the number of forward (t, s)-cuts.

Each irreducible face 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. □

Theorem 3.3 reduces the problem of counting minimum cuts in a surface embedded graph of genus g to the problem of counting forward cuts in a graph embedded on the same surface. Lemmas 6.3 and 6.8 reduce the latter problem to counting forward cuts in a triangulation of a surface of genus $O(g)$. Finally, Lemma 5.9 provides an algorithm to count forward cuts in embedded triangulations. Thus, we derive our main theorem.

Theorem 6.9 *Let $G = (V, E, c)$ be a (directed) graph with edge capacities $c : E \to \mathbb{R}^+$ embedded on an orientable 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 $2^{O(g)}n^2$ time algorithm to calculate the number of minimum (s, t)-cuts in G.*

7 Sampling Minimum Cuts

In this section, we give 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 3.3, 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 Sect. 6) and run the counting algorithm given in Sect. 5. 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 Sect. 5 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 counts the number of real cycles with the same *crossing sequence* relative to a system of $4g + 2$ paths. Our sampling algorithm picks 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. When sampling for a given crossing sequence, a random base vertex for the loop is chosen (without explicitly building the subset of the universal cover); this takes $O(\log n)$ time since we are sampling among n vertices. As we walk backwards along a random directed path, it takes at most $O(\log n)$ time to pick x_k from the set of predecessors, where lookups are done in the table for the dynamic programming. Since there are at most n vertices on the directed path, the total time is at most $O(n \log n)$. (Note that if one builds the universal cover for the weighted triangulation explicitly rather than storing the information in a dynamic programming table, it results in an extra factor of g, giving $O(gn \log n)$ instead.)

Finally, we get the following result:

Theorem 7.1 *Let $G = (V, E, c)$ be a (directed) graph with edge capacities $c : E \to \mathbb{R}^+$ embedded on an orientable 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 an algorithm to sample minimum (s, t)-cuts uniformly at random in $O(n \log n)$ time per sample after running our algorithm to count minimum (s, t)-cuts in G once.*

Acknowledgments The authors would like to thank the anonymous referees for their careful reading and helpful suggestions on improving the quality of this paper. The first author was partially supported by the National Science Foundation under Grant No. CCF 1054779; the second author was partially supported 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; and the third author was partially supported by the National Science Foundation under Grants No. CCF 1065106 and CCF 09-15519. Portions of this work were done while the second and third authors were students at the University of Illinois at Urbana-Champaign. A preliminary version of this paper was presented at the 29th Annual Symposium on Computational Geometry.

References

1. Ball, M.O., Provan, S.J.: Calculating bounds on reachability and connectedness in stochastic networks. Networks **13**, 253–278 (1983)
2. Bezáková, I., Friedlander, A.J.: Counting and sampling minimum (s, t)-cuts in weighted planar graphs in polynomial time. Theoret. Comput. Sci. **417**, 2–11 (2012)
3. Borradaile, G., Klein, P.: An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. J. ACM **56**(2), 1–30 (2000)
4. Borradaile, G., Klein, P.: An $O(n \log n)$-time algorithm for maximum st-flow in a directed planar graph. In: Proceedings of 17th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 524–533 (2006)
5. Borradaile, G., Kenyon-Mathieu, C., Klein, P.N.: A polynomial-time approximation scheme for Steiner tree in planar graphs. In: Proceedings of 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1285–1294 (2007)
6. Borradaile, G., Kenyon-Mathieu, C., Klein, P.N.: Steiner tree in planar graphs: an $O(n \log n)$ approximation scheme with singly-exponential dependence on epsilon. In: Proceedings of the 10th Workshop on Algorithms and Data Structures, pp. 275–286 (2007)
7. Borradaile, G., Demaine, E.D., Tazari, S.: Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. In: Proceedings of the 26th International Symposium on Theoretical Aspects Computer Science. Leibniz International Proceedings in Informatics, vol. 3, pp. 171–182. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2009). http://drops.dagstuhl.de/opus/volltexte/2009/1835
8. Boykov, Y., Veksler, O.: Graph cuts in vision and graphics: theories and applications. In: Paragios, N., Chen, Y., Faugeras, O. (eds.) Handbook of Mathematical Models in Computer Vision, pp. 79–96. Springer, New York (2006)
9. Cabello, S., Chambers, E.W.: Multiple source shortest paths in a genus g graph. In: Proceedings of 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 89–97 (2007)
10. Cabello, S., Mohar, B.: Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. Discrete Comput. Geom. **37**, 213–235 (2007)
11. Chalermsook, P., Fakcharoenphol, J., Nanongkai, D.: A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. In: Proceedings of 15th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 828–829 (2004)
12. Chambers, E.W., Eppstein, D.: Flows in one-crossing-minor-free graphs. In: Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC 2010). Lecture Notes in Computer Science, pp. 241–252. Springer, Berlin (2010)
13. Chambers, E.W., Colin de Verdière, É., Erickson, J., Lazarus, F., Whittlesey, K.: Splitting (complicated) surfaces is hard. Comput. Geom. Theory Appl. **41**(1–2), 94–110 (2008)
14. Chambers, E.W., Erickson, J., Nayyeri, A.: Minimum cuts and shortest homologous cycles. In: Proceedings of the 25th Annual Symposium on Computer Geometry, pp. 377–385 (2009)
15. Chambers, E.W., Erickson, J., Nayyeri, A.: Homology flows, cohomology cuts. SIAM J. Comput. **41**(6), 1605–1634 (2012)
16. Colbourn, C.J.: Combinatorial aspects of network reliability. Ann. Oper. Res. **33**, 1–15 (1991)
17. Colin de Verdière, É.: Topological algorithms for graphs on surfaces. Habilitation thesis (2012). http://www.di.ens.fr/~colin/textes/12hdr.pdf
18. Demaine, E.D., Hajiaghayi, M., Mohar, B.: Approximation algorithms via contraction decomposition. In: Proceedings of the 18th Annual ACM-SIAM Symposium Discrete Algorithms, pp. 278–287 (2007)

19. Edelsbrunner, H., Harer, J.: Computational Topology. An Introduction. American Mathematical Society, Providence, RI (2010)
20. Eppstein, D.: Subgraph isomorphism in planar graphs and related problems. J. Graph Algorithms Appl. **3**(3), 1–27 (1999)
21. Eppstein, D.: Diameter and treewidth in minor-closed graph families. Algorithmica **27**, 275–291 (2000)
22. Eppstein, D.: Dynamic generators of topologically embedded graphs. In: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 599–608 (2003)
23. Erickson, J., Nayyeri, A.: Computing replacement paths in surface graphs. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1347–1354 (2011)
24. Erickson, J., Nayyeri, A.: Minimum cuts and shortest non-separating cycles via homology covers. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1166–1176 (2011)
25. Erickson, J., Fox, K., Nayyeri, A.: Global minimum cuts in surface embedded graphs. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1309–1318 (2012)
26. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**, pp. 399–404 (1956). First published as Research Memorandum RM-1400. The RAND Corporation, Santa Monica, CA, November 19, 1954
27. Fox, K.: Shortest non-trivial cycles in directed and undirected surface graphs. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 352–364 (2013)
28. Frederickson, G.N.: Fast algorithms for shortest paths in planar graphs with applications. SIAM J. Comput. **16**(6), 1004–1004 (1987)
29. Grohe, M.: Isomorphism testing for embeddable graphs through definability. In: Proceedings of the 32nd ACM Symposium on Theory of Computation, pp. 63–72 (2000)
30. Hagerup, T., Katajainen, J., Nishimura, N., Ragde, P.: Characterizing multiterminal flow networks and computing flows in networks of small treewidth. J. Comput. Syst. Sci. **57**(3), 366–375 (1998)
31. Hassin, R., Johnson, D.B.: An $O(n \log^2 n)$ algorithm for maximum flow in undirected planar networks. SIAM J. Comput. **14**(3), 612–624 (1985)
32. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
33. Henzinger, M.R., Klein, P., Rao, S., Subramanian, S.: Faster shortest-path algorithms for planar graphs. J. Comput. Syst. Sci. **55**(1), 3–23 (1997)
34. Hopcroft, J.E., Wong, J.K.: Linear time algorithm for isomorphism of planar graphs (preliminary report). In: Proceedings of the 6th Annual ACM Symposium on Theory of Computation, pp. 172–184 (1974)
35. Itai, A., Shiloach, Y.: Maximum flow in planar networks. SIAM J. Comput. **8**, 135–150 (1979)
36. Italiano, G.F., Nussbaum, Y., Sankowski, P., Wulff-Nilsen, C.: Improved algorithms for min cut and max flow in undirected planar graphs. In: Proceedings of the 43rd Annuals of ACM Symposium on Theory of Computation, pp. 313–322 (2011)
37. Jerrum, M.: Random generation of combinatiorial structures from a uniform distribution. In: Brauer, W. (ed.) Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 194, pp. 290–299. Springer, Berlin (1985)
38. Karger, D.R.: 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, pp. 11–17. ACM, New York (1995)
39. Kawarabayashi, K., Mohar, B., Reed, B.: A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In: Proceedings of the 49th IEEE Symposium on Foundation of Computer Science, pp. 771–780 (2008)
40. Klein, P.: Multiple-source shortest paths in planar graphs. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 146–155 (2005)
41. Klein, P., Mozes, S., Weimann, O.: Shortest paths in directed planar graphs with negative lengths: a linear-space $O(n \log^2 n)$-time algorithm. ACM Trans. Algorithms **6**(2), 1–18 (2010)
42. Kutz, M.: Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time. In: Proceedings of 22nd Annual Symposium on Computer Geometry, pp. 430–438 (2006)
43. Łącki, J., Sankowski, P.: Min-cuts and shortest cycles in planar graphs in $O(n \log \log n)$ time. In: Proceedings of 19th Annual European Symposium on Algorithms. Lecture Notes Computer Science, no. 6942, pp. 155–166. Springer, Berlin (2011)

44. Lipton, R.J., Rose, D.J., Tarjan, R.E.: Generalized nested dissection. SIAM J. Numer. Anal. **16**, 346–358 (1979)
45. Mareš, M.: Two linear time algorithms for MST on minor closed graph classes. Arch. Math. **40**(3), 315–320 (2004)
46. Miller, G.L.: Isomorphism testing for graphs of bounded genus. In: Proceedings of the 12th Annual ACM Symposiums on Theory Computation, pp. 225–235 (1980)
47. Mohar, B., Thomassen, C.: Graphs on Surfaces. Johns Hopkins University Press, Baltimore, MD (2001)
48. Mozes, S., Wulff-Nilsen, C.: Shortest paths in planar graphs with real lengths in $O(n \log^2 n/\log\log n)$ time. In: Proceedings of the 18th Annuals of European Symposiums on Algorithms. Lecture Notes in Computer Science, no. 6347, pp. 206–217. Springer, Berlin (2010)
49. Munkres, J.R.: Topology, 2nd edn. Prentice-Hall, Englewood Cliffs (2000)
50. Nagamoch, H., Sun, Z., Ibaraki, T.: Counting the number of minimum cuts in undirected multigraphs. IEEE Trans. Reliab. **40**, 610–614 (1991)
51. Orlin, J.B.: Max flows in $O(nm)$ time, or better. In: Proceedings of 45th Annual ACM Symposium on Theory of Computation, pp. 765–774 (2013)
52. Patel, V.: Determining edge expansion and other connectivity measures of graphs of bounded genus. In: Proceedings of 18th Annual European Symposiums on Algorithms, ESA'10, pp. 561–572. Springer, Berlin (2010)
53. Pe'er, D.: On minimum spanning trees. Master's Thesis, Hebrew University (1998). http://www.math.ias.edu/avi/STUDENTS/dpthesis.pdf
54. Provan, S.J., Ball, M.O.: The complexity of counting cuts and of computing the probability that a graph is connected. SIAM J. Comput. **12**, 777–788 (1983)
55. Reif, J.: Minimum s-t cut of a planar undirected network in $O(n \log^2 n)$ time. SIAM J. Comput. **12**, 71–81 (1983)
56. Shamir, A.: A survey on mesh segmentation techniques. Comput. Graph. Forum **27**(6), 1539–1556 (2008)
57. Tazari, S., Müller-Hannemann, M.: Shortest paths in linear time on minor-closed graph classes, with an application to Steiner tree approximation. Discrete Appl. Math. **157**, 673–684 (2009)
58. Weihe, K.: Maximum (s,t)-flows in planar networks in $O(|V|\log|V|)$-time. J. Comput. Syst. Sci. **55**(3), 454–476 (1997)
59. White, A.: Orientable embeddings of Cayley graphs. Duke Math. J. **41**, 353–371 (1974)
60. Wulff-Nilsen, C.: Solving the replacement paths problem for planar directed graphs in $O(n \log n)$ time. In: Proceedings of 21st Annuals ACM-SIAM Symposiums on Discrete Algorithms, pp. 756–765 (2010)
61. Zomorodian, A.: Topology for Computing. Cambridge University Press, Cambridge (2005)