Counting minimum (s,t)-cuts in weighted planar graphs in polynomial time

Ivona Bezáková and Adam J. Friedlander

[1] Rochester Institute of Technology, Rochester, NY, USA, ib@cs.rit.edu
[2] IBM, Poughkeepsie, NY, USA, afriedl@us.ibm.com

Abstract. We give an $O(nd+n\log n)$ algorithm computing the number of minimum (s,t)-cuts in weighted planar graphs, where n is the number of vertices and d is the length of the shortest s-t path in the corresponding unweighted graph. Previously, Ball and Provan gave a polynomial-time algorithm for unweighted graphs with both s and t lying on the outer face. Our results hold for all locations of s and t and weighted graphs, and have direct applications in image segmentation and other computer vision problems.

1 Introduction

Graph cuts play an important role in a number of computer vision algorithms. For example, in image segmentation, see e. g. [4, 5, 9], an image is represented by a graph with pixels as the vertices and an edge connects two pixels if they are neighboring and considered similar; the edge weights capture the similarity measure between the pixels. Naturally, the underlying graph tends to be planar, typically grid-like. One of the problems of image segmentation is to separate an object from the background. Many segmentation algorithms rely on finding a minimum cut between two positions, one from the object and one from the background (often provided as input from the user). The weight of the minimum cut corresponds to the least energy contour between the positions, viewing the edge weights as the strength of the connection between the respective pixels. Notice that the two chosen locations are very unlikely to be physically close to each other, hence our assumptions on planarity and arbitrary positions of the locations are well aligned with the image segmentation applications.

Counting problems are closely related to random sampling from the same universe [14]. In image segmentation the current algorithms might suffer from finding an "atypical" cut that does not represent the contour well. Ideally the user would get the opportunity to choose the best contour out of all possible minimum-cut-based segmentations. However, this might be infeasible because the number of minimum cuts between two chosen positions might be exponential. Having the ability to sample from several minimum cuts provides the user with the option to choose the best of these segmentations while keeping the running time reasonably small.

Minimum cuts are also related to network reliability problems where the vertices are individual computers in a network, edges are connections between

computers, and the edge weight captures the probability of connection failure. The number of minimum cuts between two end-points is useful in estimating the probability of disconnecting the network, see e. g., [1]. Ball and Provan [1] showed that, in case of unweighted (multi)graphs, the problem of counting all minimum (s,t)-cuts is polynomially reducible to the problem of counting all antichains in a poset. (An (s,t)-cut can be visualized as a set of edges that, if removed, disconnect the vertices s and t, see Section 2 for the formal definition.) Both problems are known to be #P-complete, as shown by the same authors in [18]. Nevertheless, they were able to devise a polynomial-time algorithm for counting all minimum (s,t)-cuts in planar graphs, under the assumption that both vertices s and t lie on the outer face. Different variants of the network reliability problem and its connection to minimum cuts were studied in a number of previous works, for example [2, 19, 17, 15, 6].

Our main contribution is an efficient polynomial-time algorithm for computing the number of minimum (s,t)-cuts in all weighted planar graphs and for all pairs of s and t (i. e., we do not impose assumptions on the locations of s and t). Recall that this is exactly the scenario of many vision applications. (For directed graphs we work under the natural and commonly assumed condition that all vertices are reachable from s and lead to t, see, e. g., [7]. Otherwise, the typical definition of cuts leads to pathological cases, as discussed in Section 4.) We extend the result of Ball and Provan to the case of weighted graphs, showing that this case also polynomially reduces to the problem of counting (unweighted) antichains in a poset. Our main result, summarized in Theorem 1 below, uses the reduction to devise a polynomial-time algorithm for counting minimum (s,t)-cuts for weighted planar graphs, for all possible pairs of s and t.

Theorem 1. *Let $G = (V, E, w)$ be any (directed) planar graph with edge weights $w : E \to \mathbf{R}^+$. Let $s, t \in V$, $s \neq t$ and assume that all vertices are reachable from s and lead to t. Then, there is an $O(nd + n \log n)$ algorithm for counting all minimum (s,t)-cuts in G, where $n = |V|$ and d is the smallest number of edges forming a path from s to t in G.*

When both s and t lie on the outer face, it is possible to connect them by an edge, splitting the outer face into two faces. The idea of [1] relies on the fact that the antichain problem can then be solved by counting the number of paths between the two new faces in the dual (directed) planar graph. However, planarity does not allow to add the (s,t) edge for arbitrary locations of s and t. We overcome this problem by showing that we can utilize one of the paths from s to t. Our proof of correctness is significantly more elaborate than the outer face case, yet the underlying algorithm is still relatively simple, as summarized in Algorithm 1.

For completeness, we review results studying the problem of finding one of the minimum (s,t)-cuts in a given weighted planar graph. Building on the work of Itai and Shiloach [12], Reif [20] developed an $O(n \log^2 n)$ divide-and-conquer algorithm for undirected graphs. Janiga and Koubek [13] designed an $O(n \log^2 n \log \log n)$ algorithm for directed planar graphs. The result of Bor-

Algorithm 1 Counting minimum-(s,t)-cuts in a weighted planar graph G

1: Compute a maximum s-t flow such that the directed flow edges do not form a cycle.
2: Construct $\hat{G}$ by contracting every strongly connected component of the residual graph, let $\hat{s}$ and $\hat{t}$ be the vertices of $\hat{G}$ corresponding to s and t, respectively.
3: Let p be a $\hat{t}$-$\hat{s}$ path in $\hat{G}$. Duplicate all edges of p, the new edges are on the left of p when traveling from $\hat{t}$ to $\hat{s}$.
4: Follow the new edges from $\hat{t}$ to $\hat{s}$. If the current edge shares the same face on the left as the previous edge, merge the edges into one edge by bypassing the middle vertex. This process results in an $\hat{t}$-$\hat{s}$ path p'.
5: Let G' be the graph G with the path p'. Construct a (directed) unweighted dual planar graph G'_d of G', omit edges that cross the edges of p'.
6: For every pair of vertices a, b in G'_d that correspond to faces that share an edge in p', compute the number of all a-b paths in G'_d, using an algorithm for directed acyclic graphs.
7: Return the sum of all numbers computed in step 6.

radaile and Klein [3] yields an $O(n \log n)$ algorithm for all planar graphs. The dual graph plays a central role in all these works.

This paper is organized as follows. We present preliminaries, graph terminology, and notation, in Section 2. We state the reduction result in Section 3 and we prove the main result, Theorem 1, in Section 4. Section 5 contains the proofs of the results from Section 3.

2 Preliminaries

We denote by $\mathbf{R}$, $\mathbf{R}^+$, and $\mathbf{R}_0^+$ the sets of all real numbers, positive real numbers, and nonnegative real numbers, respectively.

We work with directed graphs throughout the paper. The usual conversion of undirected graphs into directed graphs (for every undirected edge include two directed edges) provides corresponding algorithms for undirected graphs.

Let $G = (V, E, w)$ be a directed graph with positive edge weights $w : E \to \mathbf{R}^+$. Let $s, t \in V, s \neq t$ be two vertices. An *(s,t)-cut of G* is a set of vertices $S \subseteq V$ that contains s but not t. The *value of the cut S* is the sum of the edge weights of the edges going out of the set S, i. e., $\sum_{(u,v):u\in S, v\notin S} w(u,v)$. A *minimum (s,t)-cut* has the smallest possible value of all (s,t)-cuts.

Our objective is to *count* the number of all possible minimum (s,t)-cuts of an input graph G.

Minimum cuts are related to network flows. A *flow network* is a directed graph $G = (V, E, c)$ where $c : E \to \mathbf{R}^+$ defines non-negative *edge capacities*. Let $s, t \in V, s \neq t$ be two vertices called *source* and *sink*, respectively. A *flow* from s to t is a function $f : E \to \mathbf{R}_0^+$ satisfying the following properties:

- *capacity constraint:* $f(e) \leq c(e)$ for every $e \in E$, and
- *flow conservation:* $\sum_{u:(u,v)\in E} f(u,v) = \sum_{w:(v,w)\in E} f(v,w)$ for every $v \in V - \{s,t\}$.

The *value of the flow* f is the sum of the values of flow edges out of s minus the sum of the flow edges into s, i. e., $\sum_{w:(s,w)\in E} f(s,w) - \sum_{w:(w,s)\in E} f(w,s)$. A flow is said to be *maximum* if it has the largest possible value among all flows from s to t (we also refer to such flows as s-t-flows).

The *residual graph of the flow* f, denoted $G_f = (V, E_f, w_f)$, is a weighted directed graph where E_f contains the following two types of edges:

- for every $e \in E$ with $f(e) < c(e)$, the set E_f contains a *forward edge* e with weight $w_f(e) = c(u,v) - f(u,v)$, and
- for every $e = (u,v) \in E$ with $f(e) > 0$, the set E_f contains a *backward edge* $e' = (v,u)$ with weight $w_f(e') = f(e)$.

An *augmenting path* in a residual graph G_f is any path from s to t.

The following is a well-known Maximum-flow Minimum-cut Theorem by Ford and Fulkerson [8].

Theorem 2. *Let $G = (V, E, c)$ be a directed graph with positive edge weights and let $s, t \in V$. Then, the value of the minimum (s,t)-cut in G equals the maximum s-t-flow value in the flow network G.*

For more information about network flows, see, e. g., [16]. Most of our terminology and notation follows this reference.

3 Reduction to Forward-cuts

We give a polynomial reduction from the problem of counting minimum (s,t)-cuts in a positively weighted graph to the problem of counting antichains in a poset. A poset can be represented by a directed acyclic graph and an antichain is a set of pairwise unrelated vertices (i. e., no vertex has a predecessor in the set).

Instead of proving our results for antichains, we define a closely related notion that we call forward-cuts. A forward-cut contains the antichain elements and all their predecessors. Moreover, a forward-(s,t)-cut contains the vertex s but not t. The formal definition is summarized below.

Definition 1. *Let $G = (V, E)$ be a directed acyclic (multi)graph, and let $s \in V$ be a vertex in G of indegree 0 and $t \in V$ be a vertex in G of outdegree 0. Let S be a subset of the vertices V such that $s \in S$ and $t \notin S$. We say that S is a forward-(s,t)-cut of G if there is no edge $(u,v) \in E$ such that $v \in S$ and $u \notin S$.*

The reduction result is stated in the following theorem.

Theorem 3. *Let $G = (V, E, c)$ be a (directed) flow network with edge capacities $c : E \to \mathbf{R}^+$. Let $s \in V$ be the source and $t \in V$ be the sink. There exists a directed acyclic 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 $\tilde{G}$. Moreover, $|\tilde{V}| \leq |V|$, $|\tilde{E}| \leq 2|E|$, 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 it can be constructed in time $O(|V| \log |V|)$.*

The proof of the theorem is included in Section 5. In the next section we will deal with graphs where every vertex is reachable from s and leads to t. The following corollary will be used in the proof of Theorem 1.

Corollary 1. *Suppose that 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 $\tilde{G}$.*

4 Minimum Cuts in Planar Graphs

The following theorem states that we can count the number of minimum (s,t)-cuts in weighted planar graphs in polynomial-time. We impose a natural condition on the input graphs: we can get to every vertex from s and we can get to t from every vertex. Without this condition, vertices that do not influence connectivity of s and t may artificially increase the number of minimum (s,t)-cuts. For example for a graph on vertices $\{s,t,a,b,c\}$ with arcs $\{(s,a),(a,t),(a,b),(a,b)\}$ we have (s,t)-cuts $\{s\}$, $\{s,a\}$, $\{s,a,b\}$, $\{s,a,c\}$, and $\{s,a,b,c\}$. However, the "true" (s,t)-cuts are only $\{s\}$ and $\{s,a\}$.

Theorem 1. *Let $G=(V,E,w)$ be any (directed) planar graph with edge weights $w : E \to \mathbf{R}^+$. Let $s,t \in V, s \neq t$, be two of its vertices and assume that all vertices are reachable from s and t is reachable from every vertex. Then, there is an $O(nd + n\log n)$ algorithm for counting all minimum (s,t)-cuts in G, where $n = |V|$ and d is the smallest number of edges forming a path from s to t in G.*

Before we prove the theorem, it will be useful to observe that a simple dynamic programming idea can be used to count the number of all paths between two end-points in a given directed acyclic graph.

Proposition 1. *Let D be a directed acyclic graph and let $a,b \in V(D)$ be two of its vertices. The number of paths from a to b can be counted in linear time. Moreover, if D is a weighted graph where a path from a to b gets the weight of the product of its edge weights, we can compute the sum of the weights of all paths from a to b in linear time.*

Now we are ready to prove the main theorem of this section.

Proof (of Theorem 1). Let $\tilde{G}$, $\tilde{s}$, and $\tilde{t}$ be the graph, the source, and the sink from Theorem 3 applied to graph G. The theorem states that we need to count the number of forward-$(\tilde{t},\tilde{s})$-cuts in $\tilde{G}$. Suppose $\tilde{G}$ is already embedded in the plane (this can be done in linear time, see, e. g., [11]).

Let $p = (\tilde{t} = v_0, v_1, \ldots, v_k = \tilde{s})$ be a (directed) path from $\tilde{t}$ to $\tilde{s}$ in $\tilde{G}$. To simplify our language, let us redraw $\tilde{G}$ so that the path p goes horizontally from left to right. We duplicate every edge of the path, drawing the duplicate edges just above the original edges, thus the i-th duplicate edge e_i (drawn between vertices v_{i-1} and v_i) lies inside a face f_i. (Notice that we are allowing multi-edges.) We replace every consecutive block of the duplicate edges that lie inside

the same face by a single edge. Formally, if $e_i, e_{i+1}, \ldots, e_j$ lie inside the same face $f_i = f_{i+1} = \cdots = f_j$ (and if either edge e_{i-1} or edge e_{j+1} exist, neither lies in the face f_i), then the edges get replaced by a single edge from v_{i-1} to v_j that lies inside f_i. After this process we obtain a new path $p' = (\tilde{t} = v'_0, v'_1, \ldots, v'_{k'} = \tilde{s})$ where no two consecutive edges lie inside the same original face. We will refer to this new graph by G'. Notice that G' is a planar directed acyclic graph with $\tilde{t}$ and $\tilde{s}$ being the only vertices of in- and out-degree 0, respectively.

Thus, every original face bordered by an edge in p on the south got split into two or more faces in G'. More precisely, there the face got split into one or more "south" faces and exactly one "north" face. The "south" faces are in bijection with the e'_i edges and we refer to the "south" face corresponding to edge e'_i by f_i^{south}. There could be several e'_i edges bordering the same "north" face. We refer to the "north" face above the edge e'_i by f_i^{north}.

Next we construct a dual graph G'_d and its planar embedding as follows. The faces of G' become the vertices of G'_d and the edges will connect neighboring faces (with one exception, see below). For two neighboring faces f'_1 and f'_2 that share an edge $e' = (u'_1, u'_2)$, there is an edge from f'_1 to f'_2 in G'_d, drawn starting in f'_1, cutting across e', and ending in f'_2, if both of the following conditions are satisfied:

- edge e' is not on the path p',
- if G' is redrawn so that e' is vertical with u'_1 being the bottom end-point, then f'_1 is on the left of e' and f'_2 is on the right.

Notice that G'_d is a planar directed graph that allows multiple edges in case when two faces of G' neighbor in more than one edge.

We claim that G'_d is also acyclic. Suppose that G'_d contains a cycle going through faces $x_1, x_2, \ldots, x_z$ (for convenience let $x_{z+1} = x_1$). The edges in G'_d defining this cycle form a face in G'_d, let us call it X. By definition of edges in G'_d, for every pair of faces x_i, x_{i+1} (in G') there is an edge in G' shared by both faces that leads from inside of X to outside of X, and there are no other edges in G' crossing through the border of X. Let y_i be the end-point of the edge shared by x_i, x_{i+1} that lies inside X. Since G' is acyclic, following the predecessors of the y_i's, we must get to a vertex of indegree 0. Thus, $\tilde{t}$, the only vertex of indegree 0, lies inside X. Following the successors of the y_i's, we must get to $\tilde{s}$; therefore, $\tilde{s}$ lies outside of X. Then, the path p' needs to cut through the border of X and must go through one of the y_i's. But, by definition of G'_d, we did not include dual edges crossing through the edges of the path p', a contradiction.

Next we claim that every path in G'_d that starts at one of the "south" faces f_i^{south} and ends at the corresponding "north" face f_i^{north} uniquely corresponds to a forward-$(\tilde{t}, \tilde{s})$-cut in $\tilde{G}$, and, vice versa, every forward-$(\tilde{t}, \tilde{s})$-cut has a corresponding path from a "south" to the corresponding "north" face in G'_d.

Let q be a path from f_i^{south} to f_i^{north} in G'_d. Let us connect the end-points of q, forming a cycle q'. Notice that the edge connecting the end-points cuts through the path p'. The cycle q' splits the plane into two regions. Let T be the set of vertices of $\tilde{G}$ that lie in the same region as $\tilde{t}$. The path q cuts through at least one edge of $\tilde{G}$, the edge bordering the face f_i^{south} on the south. The

set T includes the left end-point of this edge and the complement of T includes the right end-point. Since $\tilde{s}$ is the only vertex with outdegree 0 and there is a vertex outside T, $\tilde{s}$ must lie outside of T as well. Thus, T is an $(\tilde{t}, \tilde{s})$-cut. By the definition of edges in G'_d, all edges cutting through q start in T and end outside T. By planarity, no other edges connect T with its complement, therefore, T is a forward-$(\tilde{t}, \tilde{s})$-cut. Thus, every "south-north" path defines a forward-$(\tilde{t}, \tilde{s})$-cut.

Vice versa, let T be a forward-$(\tilde{t}, \tilde{s})$-cut. Let (u, v) be such that $u \in T$ and $v \notin T$. We claim that on a face f adjacent to (u, v) there must be exactly one other edge (u', v') such that $u' \in T$ and $v' \notin T$.

First we show that there must be an edge (u', v') such that $u' \in T$ and $v' \notin T$. Let us follow the vertices on the face f, starting with v, going to u, etc. The existence of (u', v') follows by parity: we start in $v \notin T$, then visit $u \in T$, then there might be other vertices in T, but eventually we come back to v, a vertex outside of T. Thus, there must be a pair of consecutive vertices $u' \in T$ and $v' \notin T$. Since T is a forward-cut, the edge between u' and v' must go from u' to v'.

Next we show that (u, v) and (u', v') are the only two edges on the face f crossing the boundary of T. By contradiction, suppose there is another edge (u'', v'') leading out of T. Notice that the vertices u, u', u'' are not necessarily distinct but at least two of them are different (since every vertex has only two adjacent edges bordering the face and u, u', u'' are adjacent to three distinct edges (u, v), (u', v'), and (u'', v'')). Similarly, vertices v, v', v'' are not all the same vertex. Suppose we follow the edges on the face f in the cyclic order given by following its boundary (either clockwise or counterclockwise). By parity, we encounter an (u, v)-edge forward, followed by a (u, v) edge backward, followed by a (u, v) edge forward (followed, by parity, by a (u, v) edge backward, but we do not need this edge for our argument). Without loss of generality, assume the edges are encountered in order $\overrightarrow{(u, v)}$, $\overleftarrow{(v', u')}$, and $\overrightarrow{(u'', v'')}$. We know that $\tilde{t}$ leads to every vertex, in particular also u and u'. Consider the region defined by a $\tilde{t}-u$ path, a $\tilde{t}-u'$ path, and the $u, v, \ldots, v', u'$ part of the face f's boundary. We also know that every vertex, including v and v'' leads to $\tilde{s}$. Notice that $\tilde{s}$ does not lie strictly inside f since f is a face. Also notice that $\tilde{s}$ cannot lie on the paths defining the region since $\tilde{s}$ has outdegree 0. The last possibility to consider is if $\tilde{s}$ lies inside (but not on the paths) or outside (but not on the paths) of the region defined by the paths. Then, either v or v'' cannot get to $\tilde{s}$ since exactly one of v and v'' is inside the region. We obtained a contradiction with the existence of the third edge (u'', v'').

Therefore, we know that every face containing an edge cutting through T contains exactly two edges leading out of T. Such faces are neighboring by exactly the edges leading out of T, therefore, in the dual graph G'_d with edges between the "north" and "south" faces included, there is a cycle leading through all of the faces cutting through T. Since $\tilde{t}$ and $\tilde{s}$ are separated by the cycle, the path p' must cut through the cycle. Therefore, there is a "north" and "south" face pair on the cycle and the cycle can be represented by a "south" to "north" path in G'_d.

Therefore, we need to count the number of all paths starting at a "south" face and ending at the corresponding "north" face in G'_d. This can be done, by Proposition 1 with $a = f_i^{\text{south}}$ and $b = f_i^{\text{north}}$, in linear time. (Notice that G'_d could be a multi-graph. We can replace multiple edges by a single edge with edge weight equal to the duplicity of the original edges. The weighted path count corresponds to the path count in G'_d.) We need to count the paths for every "south" face f_i^{south} (this happens as many times as is the length of p', i. e., at most n times) and sum the returned values. The overall running time is $O(|p'|(|E| + |V|)) = O(dn)$ since for planar graphs $|E| = O(|V|)$. This running time includes the construction of the paths p, p', and the dual graph G'_d. Accounting for the running time from Theorem 3, we get an overall running time of $O(dn + n \log n)$. □

5 Proof of Theorem 3

We mentioned the connection between minimum cuts and maximum flows. We utilize the connection in our proofs where we work with network flows that are acyclic, as defined below.

Definition 2. *Let $G = (V, E, c)$ be a (directed) flow network with edge capacities $c : E \to \mathbf{R}^+$. Let $s \in V$ be the source and $t \in V$ be the sink. We say that an s-t flow $f : E \to \mathbf{R}_0^+$ is acyclic if the (directed) graph $F_f = (V, D_f)$, where D_f consists of edges in E that carry positive f-values (formally, $D_f = \{e \in E \mid f(e) > 0\}$), is acyclic. We call the graph F_f the flow graph of the flow f.*

Notice that the flow graph consists of the backward edges in the corresponding residual graph, reversed. The next claim, stated without proof, observes that there exists an acyclic maximum flow in every flow network.

Proposition 2. *There exists an acyclic maximum s-t flow in any flow network $G = (V, E, c)$. The acyclic flow can be found in time $O(T(G) + |E|^2)$, where $T(G)$ is the fastest maximum flow algorithm for G. For example, the push-relabel algorithm [10] yields $T(G) = O(|V|^3)$. The term $O(|E|^2)$ comes from sequential elimination of cycles in the flow. For planar graphs Borradaile and Klein's [3] maximum flow algorithm returns an acyclic maximum s-t flow in time $O(|V| \log |V|)$.*

In subsequent proofs we rely on the fact that every maximum flow can be obtained from a sequence of augmenting paths that do not use backward edges, as spelled out by the following lemma, stated without proof.

Lemma 1. *Let $f : E \to \mathbf{R}_0^+$ be an acyclic s-t flow in a flow network $G = (V, E, c)$ with source s and sink t. Then, f can be decomposed into augmenting paths $p_1, \ldots, p_d$, where $d \leq |E|$.*

Next we state that if a vertex lies in a minimum cut set, then the cut set must contain all of the vertex's predecessors.

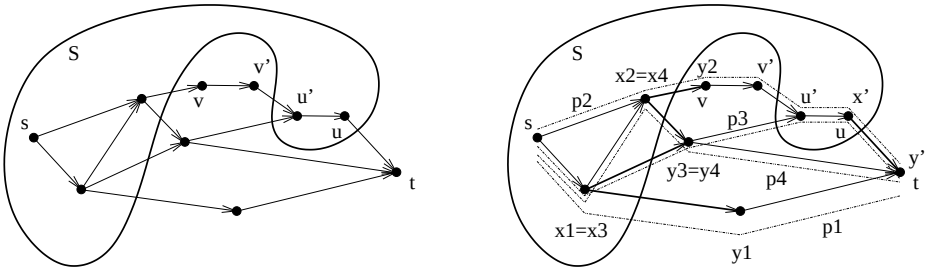

Fig. 1. Proof of Lemma 2: The left figure shows a graph F_f and an (s,t)-cut S such that there are vertices $u \in S$ and $v \notin S$, and there is a path from v to u. The right figure shows the graph F_f decomposed into paths p_1, p_2, p_3, p_4 and the cut-edges (x_i, y_i) and (x', y') (these edges are highlighted).

Lemma 2. *Let $f : E \to \mathbf{R}_0^+$ be an acyclic maximum s-t flow in a flow network $G = (V, E, c)$ with source s and sink t. Let S be a minimum (s,t)-cut in G. Then, if a vertex u is in S then every vertex v that precedes u in the flow graph F_f must be in S as well.*

Proof. By contradiction, assume that there are two vertices u, v such that $u \in S$, $v \notin S$ and there exists a path from v to u in F_f, see Figure 1. It follows that there is an edge (v', u') in F_f such that $u' \in S$ and $v' \notin S$. By Lemma 1, the flow f can be decomposed into d augmenting paths $p_1, \ldots, p_d$ where the path p_i carries flow of value $\phi_i > 0$. At least one of the paths contains the edge (v', u'), let p_j be such a path. For every $i \in \{1, \ldots, d\}$ there exists an edge (x_i, y_i) on the path p_i such that $x_i \in S$, $y_i \notin S$. Moreover, for the path p_j there exists another edge (x', y') besides (x_j, y_j) such that $x' \in S$ and $y' \notin S$.

The cut value is defined as the sum of the capacities of the edges (x, y) where $x \in S$ and $y \notin S$. Therefore,

$$\begin{aligned}\text{value}(S) &= \sum_{(x,y)\in E, x\in S, y\notin S} c(x,y) \geq \sum_{(x,y)\in E:\exists i: x=x_i, y=y_i \text{ or } x=x', y=y'} c(x,y) \\ &\geq \sum_{i=1}^{d} \phi_i + \phi_j = \text{value}(f) + \phi_j.\end{aligned}$$

Since $\phi_j > 0$, the value of the cut S is strictly greater than the value of the flow f. Therefore, by the Max-flow-min-cut Theorem (Theorem 2), S cannot be a minimum (s,t)-cut of G. □

Next we claim that no minimum cut cuts through strongly connected components of a residual graph corresponding to a maximum flow.

Lemma 3. *Let $f : E \to \mathbf{R}_0^+$ be an acyclic maximum s-t flow in a flow network $G = (V, E, c)$ with source s and sink t. Let G_f be the corresponding residual*

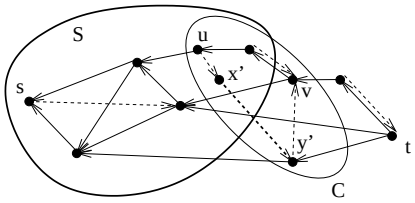

Fig. 2. Proof of Lemma 3: The figure shows a residual graph G_f and an (s,t)-cut S cutting through a strongly connected component C. Backward edges are depicted by straight arrows while the forward edges are dashed. The edge (x', y') is highlighted. For clarity we do not include residual capacities.

graph. If S is a minimum (s,t)-cut in G, then for each strongly connected component C in G_f, either $C \subset S$ or $C \cap S = \emptyset$.

Proof. By contradiction, suppose that there exists a strongly connected component C and a minimum (s,t)-cut S such that there exist vertices $u, v \in C$ such that $u \in S$ and $v \notin S$, see Figure 2. Since u, v belong to the same strongly connected component, there exists a path from u to v in G_f, as well as a path from v to u in G_f. By the definition of a residual graph, G_f contains two types of edges: forward and backward edges where the backward edges are simply the edges of f reversed. We will show that there must exist a forward edge (x', y') in G_f such that $x' \in S$ and $y' \notin S$. This will imply that S cannot be a minimum (s,t)-cut.

Consider a path from u to v in G_f. Since $u \in S$ and $v \notin S$, there must exist an edge (x', y') on this path such that $x' \in S$ and $y' \notin S$. If the edge (x', y') is a backward edge then its reverse (y', x') is a flow edge and, by Lemma 2, if S is a minimum (s,t)-cut such that $x' \in S$, then y' must also be in S. Therefore, (x', y') is a forward edge.

Now let us look at the value of the cut S. By the same argument as in the proof of Lemma 2, we get

$$\begin{aligned}\text{value}(S) = \sum_{(x,y)\in E, x\in S, y\notin S} c(x,y) &\geq \sum_{(x,y)\in E:\exists i: x=x_i, y=y_i \text{ or } x=x', y=y'} c(x,y) \\ &\geq \sum_{i=1}^{d} \phi_i + \phi' = \text{value}(f) + \phi',\end{aligned}$$

where $\phi' > 0$ is the residual capacity of the (forward) edge (x', y') in G_f. Notice that the edge (x', y') is distinct from every (x_i, y_i) since it is a forward edge. Therefore, the value of the cut S is strictly bigger than the value of the maximum flow f, a contradiction with the assumption that S is a minimum cut. □

Therefore, we define a contraction graph that contracts every strongly connected component. We obtain the following corollary of the last lemma.

Definition 3. *Let $G = (V, E)$ be a directed graph. We define the SCC-contraction graph of G, denoted $\widehat{G}$, to be the graph obtained from G by contracting each strongly connected component into a single vertex.*

Corollary 2. *Let $\widehat{G_f}$ be the SCC-contraction graph of the residual graph corresponding to an acyclic maximum s-t flow f in a flow network $G = (V, E, c)$. Suppose $\hat{s} \in V(\widehat{G_f})$ is the vertex obtained by contracting the strongly connected component containing s and $\hat{t} \in V(\widehat{G_f})$ is the vertex obtained by contracting the strongly connected component containing t. Moreover, we define a function α from the set of $(\hat{t}, \hat{s})$-cuts of $\widehat{G_f}$ to the set of (s, t)-cuts of G: for a $(\hat{t}, \hat{s})$-cut $\hat{T}$ in $\widehat{G_f}$, let $\alpha(\hat{T})$ contain all vertices v that belong to a strongly connected component $\hat{v} \notin \hat{T}$. Then,*

(i) the function α is injective, and
(ii) for every minimum (s, t)-cut S in G there exists a $(\hat{t}, \hat{s})$-cut $\hat{T}$ in $\widehat{G_f}$ such that $\alpha(\hat{T}) = S$.

Proof. To prove (i), let us first look at the (s, t)-cuts S for which there exists a $(\hat{t}, \hat{s})$-cut $\hat{T}$ such that $S = \alpha(\hat{T})$. By the definition of α it follows that any such S must satisfy the property that for every strongly connected component C of G_f, either $C \subset S$ or $C \cap S = \emptyset$.

Suppose that we have an S satisfying this property. Then we can uniquely construct $\hat{T}$ such that $S = \alpha(\hat{T})$: for every strongly connected component C of G_f, if $C \cap S = \emptyset$ then the vertex corresponding to C in $\widehat{G_f}$ is in $\hat{T}$ (and otherwise this vertex is not in $\hat{T}$). Since we can reconstruct $\hat{T}$ uniquely, α is injective.

To show part (ii), by Lemma 3, every minimum (s, t)-cut S satisfies the property that for every strongly connected component C, either $C \subset S$, or $C \cap S = \emptyset$. Therefore, there exists $\hat{T}$ (containing all strongly connected components not in S) such that $S = \alpha(\hat{T})$. □

The next two lemmas show that there is a bijection between minimum (s, t)-cuts in G and forward $(\hat{t}, \hat{s})$-cuts in the SCC-contraction graph. The bijection is given by the function α defined in the above corollary. The proofs of the lemmas, omitted due to space constraints, are similar to the proofs of Lemmas 2 and 3.

Lemma 4. *Under the assumptions of Corollary 2, suppose S is a minimum (s, t)-cut in G. Then $\alpha^{-1}(S)$ is a forward $(\hat{t}, \hat{s})$-cut in $\widehat{G_f}$.*

Lemma 5. *Under the assumptions of Corollary 2, let $\hat{T}$ be any forward-$(\hat{t}, \hat{s})$-cut in $\widehat{G_f}$. Then the (s, t)-cut $S = \alpha(\hat{T})$ is a minimum (s, t)-cut in G.*

The proof of Theorem 3, the main reduction theorem, follows from Proposition 2 and Lemmas 4 and 5.

Acknowledgments. We would like to thank the anonymous referees for many useful and insightful comments.

References

1. M. O. Ball, and J. S. Provan, *Calculating Bounds on Reachability and Connectnedness in Stochastic Networks*, Networks, Vol. 13, 253-278, 1983.
2. M. O. Ball, and J. S. Provan, *Computing Network Reliability in Time Polynomial in the Number of Cuts*, Operations Research, Vol. 32, No. 3, 516-526, 1984.
3. G. Borradaile, and P. Klein, *An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph*, Journal of the ACM, Vol. 56, Issue 2, 2009.
4. Y. Boykov, and V. Kolmogorov, *An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision*, IEEE Trans. Pattern Anal. Mach. Intell. 26(9): 1124-1137, 2004.
5. Y. Boykov, O. Veksler and R. Zabih, *Fast approximate energy minimisation via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29, 12221239, 2001.
6. C. J. Colbourn, *Combinatorial aspects of network reliability*, Annals of Operations Research, Vol. 33, Num. 1, 1-15, 2005.
7. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, 2001.
8. L. R. Ford, and D. R. Fulkerson, *Maximal flow through a network*, Canadian Journal of Mathematics 8: 399404, 1956.
9. D. Geman, and S. Geman, *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intell., 6, 721741, 1984.
10. A. V. Goldberg, and R. E. Tarjan, *A new approach to the maximum flow problem*, Journal of the ACM (JACM), Volume 35, Issue 4, 921-940, 1988.
11. J. Hopcroft, and R. E. Tarjan, *Efficient planarity testing*, Journal of the Association for Computing Machinery 21 (4): 549568, 1974.
12. A. Itai, and Y. Shiloach, *Maximum Flow in Planar Networks*, SIAM J. Comput. 8(2), 135-150, 1979.
13. L. Janiga, and V. Koubek, *Minimum Cut in Directed Planar Networks*, Kybernetika, Vol. 28, Num. 1, 37-49, 1992.
14. M. R. Jerrum, L. G. Valiant, and V. V. Vazirani, *Random generation of combinatorial structures from a uniform distribution*, Theoretical Computer Science, Vol. 43, Num. 2-3, 169-188, 1986.
15. D. R. Karger, *A Randomized Fully Polynomial Time Approximation Scheme for the All-Terminal Network Reliability Problem*, SIAM J. Comput. 29(2), 492-514, 1999.
16. J. Kleinberg and É. Tardos, *Algorithm Design*, Addison Wesley, 2005.
17. H. Nagamochi, Z. Sun, and T. Ibaraki, *Counting the number of minimum cuts in undirected multigraphs*, IEEE Transactions on Reliability, Vol. 40, Issue 5, 610-614, 1991.
18. J. S. Provan, and M. O. Ball, *The complexity of counting cuts and of computing probability that a graph is connected*, SIAM J. Comput., Vol. 12, No. 4, 777-788, 1983.
19. A. Ramanathan, and C. J. Colbourn, *Counting almost minimum cutsets with reliability applications*, Mathematical Programming, Vol. 39, Num. 3, 253-261, 1987.
20. J. H. Reif, *Minimum s-t cut of a planar undirected network in $O(n \log^2 n)$ time*, SIAM Journal on Computing, 12:71–81, 1983.