Contiguous minimum single-source-multi-sink cuts in weighted planar graphs

Ivona Bezáková and Zachary Langley

Rochester Institute of Technology, Rochester, NY, USA
{ib,zbl9222}@cs.rit.edu

Abstract. We present a fast algorithm for uniform sampling of contiguous minimum cuts separating a source vertex from a set of sink vertices in a weighted undirected planar graph with n vertices embedded in the plane. The algorithm takes $O(n)$ time per sample, after an initial $O(n^3)$ preprocessing time during which the algorithm computes the number of all such contiguous minimum cuts. Contiguous cuts (that is, cuts where a naturally defined boundary around the cut set forms a simply connected planar region) have applications in computer vision and medical imaging [6, 14].

1 Introduction

Graph cuts have become a popular tool in computer vision over the past decade, see, e. g., [7, 6, 14]. The goal of image segmentation is to partition a given image into meaningful segments, for example, to isolate an object in the foreground from the background, or to find the boundary of an organ on an ultrasound image.

The image is represented by a graph of pixels (the vertices), edges connect neighboring pixels, and edge weights represent (dis)similarity between the endpoints. In the simplest scenario a user selects a point in the object (the source) and a point in the background (the sink) and a minimum cut between the source and the sink is used to isolate the object from the background.

However, thin objects such as blood vessels are often hard to isolate when using a cut between only two points, see, e. g., [14]. This is because the minimum cut might be clustered around the selected point in the object – for example, opting to sever the point from the rest of the blood vessel – instead of forming a needle-like shape with numerous cut edges of small weight. To avoid this problem, the user may select additional points (seeds) in the object and/or the background, and keep selecting such points until the desired segmentation is achieved – this is known as interactive image segmentation. However, with multiple seeds the cut might consist of several planar regions – for example, regions around the seeds, severing the blood vessel multiple times – instead of a desired single region. A natural solution to this problem is to enforce "contiguity" of the cut; similar concepts are known as a "connectivity prior" [14] and a "topology preserving cut" [6, 15].

A cut separating the source vertices S from the sink vertices T is a set of vertices containing every vertex in S and no vertex from T – we refer to these cuts as (S,T)-cuts. An (S,T)-cut is minimum if the sum of the weights of edges connecting a vertex in the cut set to a vertex outside the cut set is the smallest possible across all (S,T)-cuts. For planar graphs embedded in the plane, we consider a cut to be contiguous if a region formed by connecting the neighborhoods around each cut vertex along edges and through faces shared by these vertices is simply connected, see Figure 1 and the formal definition in Section 2.

We present an $O(n)$ algorithm that produces a uniformly random contiguous minimum cut separating a single source vertex from a set of sink vertices in a positively weighted undirected planar graph embedded in the plane. The algorithm uses $O(n^3)$ preprocessing time during which it computes the number of all contiguous minimum (s,T)-cuts for a source s and a set of sink vertices T. Note that there could be exponentially many such cuts. Optimization problems with multiple optimum solutions have been recognized as drawbacks in computer vision, see, e. g., [10]. In such cases, random sampling can be used to gather various statistical data on the solutions, or the user can be given a choice between several randomly generated solutions.

We note that heuristics and approximation algorithms have been proposed for finding regions of various connectivity requirements [14, 15]; however, to the best of our knowledge, our work is the first polynomial-time provably correct exact algorithm for such a problem.

The earliest works considering the problem of counting minimum cuts in a graph date back to the 1980's. Ball and Provan [1] showed that for a single source and a single sink, the problem reduces to the problem of counting maximal antichains in a poset. In particular, this poset is the directed acyclic graph obtained by finding an acyclic maximum flow, constructing the corresponding residual graph, and contracting each strongly connected component into a single vertex. This implies that the problem is $\#P$-complete for general graphs [13]. Recently, building on [1], a polynomial-time algorithm was developed for the single-source-single-sink variant for planar graphs [2], using, as the first step, the same reduction to the maximal antichains. However, the reduction can not be applied in the contiguous multi-sink case, as the contractions can "bypass" vertices lying in the region defined by the contracted component.

We present a novel reduction that preserves the contiguity of the cuts by selectively contracting certain edges within the strongly connected components, as well as edges that connect vertices from different strongly connected components. This yields a planar directed acyclic multi-graph in which we need to count antichains satisfying a contiguity requirement – we call them contiguous forward cuts. Additionally, we present a new contiguity variant of the cut-cycle duality, where we represent contiguous forward cuts as special kinds of tours of the dual graph – we refer to them as non-crossing. This notion is similar to the so-called non-self-crossing cycle, but with the restriction that the tour forms a star-like shape with respect to every face.

Then we form an acyclic subgraph of the dual graph by "cutting" the primal graph along a tree connecting the source and sink vertices. We decompose each tour into paths in this subgraph where every pair of consecutive paths is joined by a single edge in the dual graph. Moreover, the paths can be sampled independently and are guaranteed to not cross. This allows us to use dynamic programming to obtain the final count of all non-crossing tours. While the proof of correctness is quite involved, the final algorithm is reasonably simple, as summarized in Algorithms 1 and 2.

For completeness, we mention recent works dealing with maximum flows and minimum cuts in planar graphs. Borradaile and Klein [3] gave an $O(n \log n)$ algorithm for the single-source-single-sink acyclic maximum flow. Borradaile, Sankowski, and Wulff-Nilsen [5] produce a minimum single-source-single-sink cut for any source-sink pair in time proportional to the size of the cut, after an initial $O(n \operatorname{polylog} n)$ preprocessing time. Italiano, Nussbaum, Sankowski, and Wulff-Nilsen [11] give algorithms for undirected planar graphs that break the $O(n \log n)$ time barrier. Recently, Borradaile, Klein, Mozes, Nussbaum, and Wulff-Nilsen [4] gave an $O(n \log^3 n)$ algorithm for maximum flow from multiple sources to multiple sinks. While all these algorithms are very ingenious, as far as we know, none of them produce the respective cut counts (or samples).

Finally, we remark that we do not know of any polynomial-time algorithms counting or sampling all minimum single-source multi-sink cuts in planar graphs (i. e., not just contiguous cuts). Similarly, the problem is open for simple cuts (i. e., cuts where the graph induced by the cut vertices is connected). In both cases Ball and Provan's reduction can be applied but it is unclear how to count the corresponding sets of antichains. In the case of general cuts, an antichain might correspond to a set of several tours, not just one. In the case of simple cuts, an antichain corresponds to a cycle and our dynamic programming technique does not guarantee to not repeat vertices across different path segments. The case of contiguous cuts with multiple sources and multiple sinks is also open.

The paper is organized as follows. Section 2 contains preliminaries, Section 3 describes how to reduce the problem to the problem of counting contiguous forward cuts in a planar directed acyclic multi-graph, Section 4 describes the representation of contiguous forward cuts via non-crossing tours in the dual graph, and Section 5 describes the main dynamic programming algorithm, followed by the sampling procedure. The proofs are omitted due to space constraints.

2 Preliminaries

Let $G = (V, E, w)$ be a weighted undirected connected planar graph with edge weights $w : E \to \mathbf{R}^+$. Let $s \in V$ and $T \subseteq V$, $s \notin T$. Our objective is to count[1] all minimum (s, T)-cuts of G where an (s, T)-*cut* is a set of vertices $C \subseteq V$ such that $s \in C$ and $T \cap C = \emptyset$. The value of the cut C is the sum of all

[1] We first develop the counting algorithm – the sampling part will be discussed in Section 5.

edge weights of edges leading out of C – formally, $\sum_{(u,v)\in E: u\in C, v\notin C} w(u,v)$. A *minimum* (s,T)*-cut* has the smallest possible value of all (s,T)-cuts.

Given a directed graph $H = (V_H, E_H)$, we say that a cut $C \subseteq V_H$ is a *forward-cut* if there is no edge leading into C, i. e., there is no edge (u,v) such that $u \notin C$ and $v \in C$. A forward-cut C is a *forward-*(A,b)*-cut* where $A \subseteq V_H$, $b \in V_H$ and $b \notin A$, if $A \subseteq C$ and $b \notin C$.

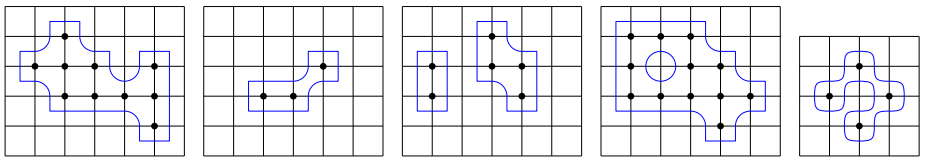
A *flow network* is a directed graph $G = (V, E, c)$ where $c : E \to \mathbf{R}^+$ defines non-negative *edge capacities*. Let $s, t \in V$, $s \neq t$, be two vertices called the *source* and the *sink*, respectively. A *flow* from s to t is a function $f : E \to \mathbf{R}_0^+$ satisfying the following properties: (1) *capacity constraint:* $f(e) \leq c(e)$ for every $e \in E$, and (2) *flow conservation:* $\sum_{u:\ (u,v)\in E} f(u,v) = \sum_{u:\ (v,u)\in E} f(v,u)$ for every $v \in V \setminus \{s,t\}$. The *value of the flow* f is the sum of the values of flow edges out of s minus the sum of the values of the flow edges into s, i. e., $\sum_{u:\ (s,u)\in E} f(s,u) - \sum_{u:\ (u,s)\in E} f(u,s)$. 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). A flow is said to be acyclic if the set of edges with positive flow value $\{e \in E \mid f(e) > 0\}$ does not contain a directed cycle.

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: (1) for every $e = (u,v) \in E$ with $f(e) < c(e)$, the set E_f contains a *forward edge* $e = (u,v)$ with weight $w_f(e) = c(e) - f(e)$, and (2) for every $e = (u,v) \in E$ with $f(e) > 0$, the set E_f contains a *backward edge* $e' = (v,u)$ with weight $w_f(e') = f(e)$.

The following theorem describes Ball and Provan's reduction.

Theorem 1 ([1, 2]). *Let $G = (V, E, w)$ be a connected positively weighted undirected graph and let $s, t \in V$, $s \neq t$. Let $G' = (V, E', c)$ be a flow network obtained by including, for every edge $(u,v) \in V$, two directed edges (u,v) and (v,u) in E' with capacities $c(u,v) = c(v,u) = w(u,v)$. Let f be an acyclic maximum s-t flow in G' and let G'_f be the corresponding residual graph. Let $H = (V_H, E_H)$ be the graph obtained from G'_f by contracting each strongly connected component into a single vertex, omitting duplicate and self-loop edges, and ignoring the edge weights. Hence, vertices of H are sets of vertices of G – let $\hat{s}$ and $\hat{t}$ be the vertices in V_H containing s and t respectively. Then, the set of minimum (s,t)-cuts in G is in bijection with the set of forward-$(\hat{t},\hat{s})$-cuts in H by mapping a forward-$(\hat{t},\hat{s})$-cut $C_H \subseteq V_H$ to the (s,t)-cut $C = \cup_{x\notin C_H} x$. Moreover, if G is connected, then H, viewed as an undirected graph, is connected, and if G is planar, then H is planar. The graph H can be obtained in time $O(|V|^3 + |E|^2)$ and time $O(|V| \log |V|)$ if G is planar.*

Next we define contiguous cuts that give rise to a contiguous region in the plane – a concept useful for many segmentation applications, see, e. g., [6, 14] for a discussion of several contiguity concepts. For a planar (directed or not) graph $G = (V, E)$ embedded in the plane and a set of vertices $C \subset V$, we define $R(C)$, a set of points in the plane, as follows. We start with the union of all faces that contain a vertex from C on their boundary. Then, for every vertex not in C, we remove an ε-neighborhood around this vertex. Finally, for every edge between



(a): cut set C_1 (b): cut set C_2 (c): cut set C_3 (d): cut set C_4 (e)

Fig. 1. (a)-(d): Contiguous vs. noncontiguous cuts: four possible cut sets (of the 6×7 grid graph). The highlighted vertices are in the respective cut sets $C_1, \ldots, C_4$. The shown boundaries bound the corresponding point sets $R(C_1), \ldots, R(C_4)$. Cut sets C_1 and C_2 are contiguous, cut sets C_3 and C_4 are not. (e): Contiguous cuts vs. non-self-crossing cycles: not a contiguous cut, yet the cut set can be bounded by a non-self-crossing cycle.

two vertices that are both not in C, we remove an ε-neighborhood around this edge. (By ε-neighborhood we mean the set of points in the plane with distance $\leq \varepsilon$ from the vertex or edge. We choose ε so that the ε-neighborhood does not intersect with non-adjacent edges or contain other vertices in the planar drawing of G.) We say that C is *contiguous* if $R(C)$ forms a simply connected region in the plane, i. e., if the boundary of $R(C)$ splits the plane into exactly two regions. See Figure 1(a)-(d). Informally, in the grid graph the contiguity concept means a "corner-connected" region without holes.

To put contiguous cuts in perspective with the standard cut-cycle duality, we note that contiguous cuts are *not* dual with the so-called non-self-crossing cycles. Consider Figure 1(e) where a non-self-crossing cycle separates the cut vertices from the remaining vertices, yet the cut is not contiguous.

Finally, to simplify our language, for a directed planar graph embedded in the plane, we refer to the two faces neighboring an edge $e = (u, v)$ as the *left face of* e (when traversing e from u to v, this face is on the left) and the *right face of* e (the other face). We use the same terminology when describing regions bounded by directed paths/cycles. By a *clockwise traversal* of the boundary of a face (or a planar simply connected region) f we mean listing the edges on the boundary of f in the clockwise order as seen from the viewpoint of somebody standing *inside* f.

3 Reduction to contiguous forward cuts

In this section we present an algorithm that reduces the problem of counting all contiguous minimum (s, T)-cuts to the problem of counting all contiguous forward-$(\hat{T}, \hat{s})$-cuts in a planar directed acyclic (multi)graph. On the surface this statement seems analogous to Theorem 1: we can create a super-sink connected to every sink by an ∞-weighted edge and apply the original reduction. However, this can result in contracting a cycle consisting of edges that are not minimum-cut edges into a single vertex while "bypassing" the area inside the cycle. Hence,

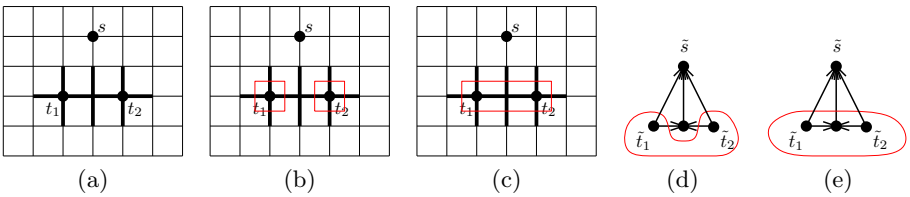


Fig. 2. Contiguous (s, T)-cuts in G vs. contiguous forward-$(\hat{T}, \hat{s})$-cuts in H. Figure (a) shows a source and two sinks, the highlighted edges are of weight 1, all other edges are of weight ∞. Figures (b) and (c) depict two possible minimum (s, T)-cuts (of weight 8) – (b) is not contiguous and (c) is contiguous. Figures (d) and (e) show the graph H and the forward cuts corresponding to the minimum cuts from figures (b) and (c). Both forward cuts are contiguous, even though the cut in figure (b) is not contiguous.

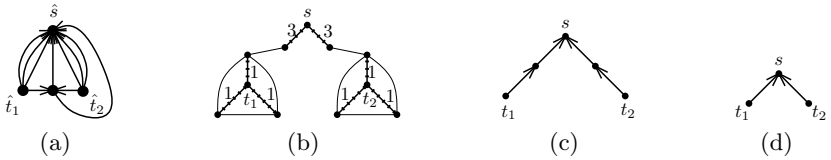


Fig. 3. Applying Algorithm 1: Figure (a) shows the result for the graph from Figure 2(a). Figure (b) shows another possible input graph, the highlighted edges are of weights 1 or 3, the other edges are of weight ∞. Figures (c) and (d) depict the corresponding graphs after applying Ball and Provan's reduction (there are 4 minimum $(s, \{t_1, t_2\})$-cuts and thus 4 corresponding forward cuts) and Algorithm 1 (there is only one forward cut corresponding to the single contiguous minimum $(s, \{t_1, t_2\})$-cut).

noncontiguous cuts become contiguous forward cuts, as demonstrated in Figure 2. To avoid such problems, we designed a new reduction (Algorithm 1) that selectively contracts and removes edges to preserve contiguity.

Theorem 2. *Let $G = (V, E, w)$ be a connected undirected planar graph embedded in the plane, $w > 0$. Let $s \in V$ and $T \subseteq V$, $s \notin T$. Algorithm 1 decides whether there exists a contiguous minimum (s, T)-cut in G. If yes, it constructs a directed acyclic (multi)graph $H' = (V'_H, E'_H)$ embedded in the plane, a vertex $\hat{s} \in V'_H$, and a set of vertices $\hat{T} \subseteq V'_H$ such that the set of all contiguous minimum (s, T)-cuts in G is in bijection with the set of all contiguous forward-$(\hat{T}, \hat{s})$-cuts in H'. The algorithm runs in time $O(|V|^3)$. Moreover, $\hat{T}$ is the set of vertices of indegree 0 in H' and $\hat{s}$ is the only vertex of outdegree 0 in H'.*

What happens when we apply Algorithm 1 to the graph from Figure 2? We will keep two edges from the middle vertex to $\hat{s}$, as shown in Figure 3(a), preventing the "illegal" contiguous forward cut from Figure 2(d). A graph where Algorithm 1 needs to deal with self-loops is shown in Figure 3(b)-(d). Figure 4 sketches the individual self-loop cases on which the proof of Theorem 2 is based.

Algorithm 1 Reduction to contiguous forward cuts

Add an extra vertex, τ, to G, connect it with ∞-weight edges to the vertices in T, and replace every edge by two directed edges of the same weight. Let $G_f = (V_\tau, E_f, w_f)$ be the residual graph of an acyclic s-τ maximum flow f of the new graph. Let $\hat{T}$ be the set of vertices of G that belong to the same strongly connected component of G_f as τ.
Remove τ and its adjacent edges from G_f, and ignore the edge weights, obtaining H'_0 (embedded in the plane analogously to G).
Let $e_1, e_2, \ldots, e_\ell \in E_f$ be the edges that belong to any strongly connected component of H'_0.
for $i = 1, \ldots, \ell$ **do**
  if e_i does not form a self-loop in H'_{i-1} **then**
    Get H'_i from H'_{i-1} by contracting the edge e_i (if it has not been removed earlier).
  else
    Let (a, a) be the self-loop formed by e_i in H'_{i-1}.
    if $s \notin a$ **then**
      The self-loop splits the plane into two regions: let R be the region that does not contain s.
      if all vertices in $\hat{T}$ are in R **then**
        Get H'_i from H'_{i-1} by removing the self-loop (a, a). See Figure 4(a).
      else
        Get H'_i from H'_{i-1} by contracting all vertices in R into a and remove all (a, a) self-loops. See Figure 4(b)-(c).
    else
      if the two regions bounded by the self-loop each contain a vertex in $\hat{T}$ **then**
        Return "no contiguous minimum cuts". See Figure 4(d).
      else
        Let R be the region that contains the vertices in $\hat{T}$.
        Get H'_i from H'_{i-1} by contracting all vertices outside R into a and remove all (a, a) self-loops. See Figure 4(e).
For every $t \in \hat{T}$, contract all predecessors of t into t and omit self-loops.
Return $H' := H'_\ell$ and its planar embedding, $\hat{T}$, and the vertex $\hat{s}$ containing s.

4 Non-crossing tours

In this section we classify contiguous forward-$(\hat{T}, \hat{s})$-cuts as certain types of *tours* (i. e., cycles that are allowed to repeat vertices) in the dual planar graph. While these tours may revisit faces (i. e., vertices in the dual graph), they cannot "self-cross," as defined below.

First, recall the standard definition of a directed planar dual. For a directed planar (multi)graph $H = (V_H, E_H)$ embedded in the plane, we define the *dual (multi)graph* $H_D = (V_D, E_D)$ as follows: V_D is the set of all faces of H and for every edge $e \in E_H$ we include an edge from f_1 to f_2 in E_D where f_1 and f_2 are the left and the right face of e, respectively.

Next we define a "non-crossing" tour and the "inside" and the "outside" regions defined by the tour.

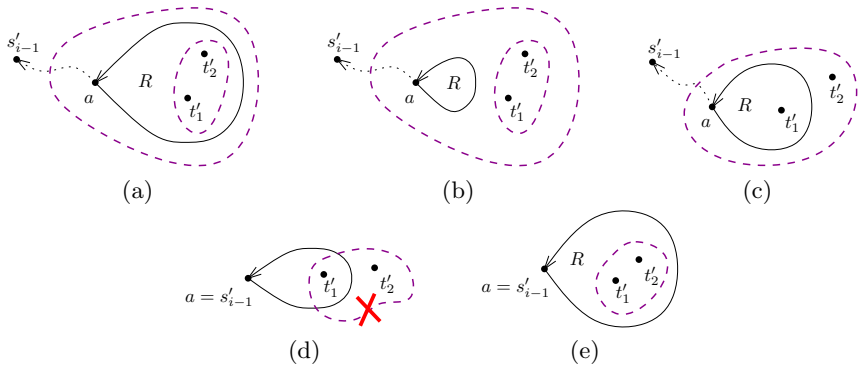


Fig. 4. Demonstrating the cases in Algorithm 1, possible contiguous cuts are dashed.

Definition 1. *Let $H' = (V'_H, E'_H)$ be a connected planar directed acyclic (multi) graph embedded in the plane and let $H'_D = (V'_D, E'_D)$ be its dual graph. Let $d_1, d_2, \ldots, d_\ell$, $d_i \in E'_D$ for $i \in \{1, \ldots, \ell\}$, be a tour in H'_D. Let $f_1, \ldots, f_\ell$ be the faces visited by the tour, i. e., the edge d_i goes from f_i to f_{i+1} (where $f_{\ell+1} = f_1$) and let e_i be the edge that gave rise to the edge d_i in the dual graph. We say that the tour is* non-crossing *if the following holds for every face f visited by the tour. Let $f_{j_1}, f_{j_2}, \ldots, f_{j_p}$ for $1 \leq j_1 \leq j_2 \leq \cdots \leq j_p \leq \ell$ be all the faces on the tour equal to f. Then, the edges $e_{j_1-1}, e_{j_1}, e_{j_2-1}, e_{j_2}, \ldots, e_{j_p-1}, e_{j_p}$ must appear in this order when clockwise traversing the boundary of f (where $e_0 := e_\ell$). See Figure 5.*

The inside region *defined by the tour consists of all the starting endpoints of the edges e_i, $i \in \{1, \ldots, \ell\}$, and all their predecessors. The* outside region *contains all the other vertices of H'.*

Notice that a non-crossing tour can be drawn in the plane in a "non-self-crossing way". Notice also that drawing a tour in a non-self-crossing way does *not* imply that the tour is non-crossing, as demonstrated in Figure 5(c).

Lemma 1. *Let $H'_D = (V'_D, E'_D)$ be the dual of the graph H' from Theorem 2. Then, the set of all contiguous forward-$(\hat{T}, \hat{s})$-cuts of H' is bijection with the set of all non-crossing tours in H'_D such that the inside region defined by the tour contains all vertices from $\hat{T}$ and the outside region contains $\hat{s}$.*

5 Counting and sampling contiguous minimum (s, T)-cuts

In this section we prove the main theorem of the paper:

Theorem 3. *Let $G = (V, E, w)$ be a connected undirected planar graph with edge weights $w : E \to \mathbf{R}^+$, embedded in the plane. Let $s \in V$ and $T \subset V$, $s \notin T$. The number of contiguous minimum (s, T)-cuts of G can be computed in time*

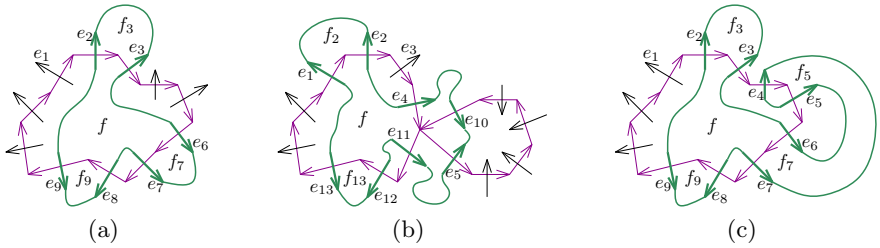


Fig. 5. Illustrating Definition 1 (a non-crossing tour): Face f (the curve-shaped shaped region) is visited by the tour three times in figure (a): $f = f_3 = f_7 = f_9$, four times in figure (b): $f = f_2 = f_5 = f_{11} = f_{13}$, and four times in figure (c): $f = f_3 = f_5 = f_7 = f_9$. The edges e_i appear on the boundary of f in this order: (a) $e_2, e_3, e_6, e_7, e_8, e_9$ – the clockwise order (with respect to the tour), i. e., the tour could be non-crossing (depending on the other faces on the tour); (b) $e_1, e_2, e_4, e_{10}, e_5, e_{11}, e_{12}, e_{13}$, and (c) $e_2, e_3, e_6, e_5, e_4, e_7, e_8, e_9$ – the tours (b) and (c) are definitely not non-crossing.

$O(|V|^3)$. *A uniformly random contiguous minimum (s,T)-cut can be produced in additional linear time.*

By Lemma 1, we know that it suffices to count all non-crossing tours separating $\hat{s}$ from $\hat{T}$ in the dual graph H'_D. Even though counting cycles or tours in planar graphs tends to be $\#P$-complete [8, 9, 12], we show that the problem of counting non-crossing tours in H'_D can be solved in polynomial time. In particular, we decompose the tour into paths (that cannot repeat vertices) and then we count (or sample) each path type separately. The counting algorithm combines the paths using dynamic programming.

Before we state the decomposition lemma, we define a "restricted dual" graph that, unlike the dual graph H'_D, will be guaranteed to be acyclic. The definition will use a "tree" of edges in H' that connects $\hat{T}$ to $\hat{s}$.

Observation 1 *Let $H' = (V'_H, E'_H)$ be a planar directed acyclic (multi)graph, let $\hat{s} \in V'_H$ be the only vertex of outdegree 0, and let $\hat{T} \subseteq V'_H$, $\hat{s} \notin \hat{T}$, be the set of vertices of indegree 0. There exists a set of edges $A \subseteq E'_H$ such that for every $\hat{t} \in \hat{T}$ there is a unique directed path from $\hat{t}$ to $\hat{s}$ using only the edges from A, and every edge in A is on the path from $\hat{t}$ to $\hat{s}$ for some $\hat{t} \in \hat{T}$. Moreover, A can be constructed in time $O(|\hat{T}||V'_H|)$.*

Definition 2. *Let H' (embedded in the plane), $\hat{s}$, $\hat{T}$, and A be as in Observation 1. We define the* restricted dual (multi)graph *$H'_d = (V'_d, E'_d)$ of H' as follows: V'_d is the set of all faces of H', and, for every edge $e \in E'_H \setminus A$, we include an edge from f_1 to f_2, where f_1 and f_2 are the left and right faces of e, respectively.*

Lemma 2. *The graph $H'_d = (V'_d, E'_d)$ from Definition 2 is acyclic.*

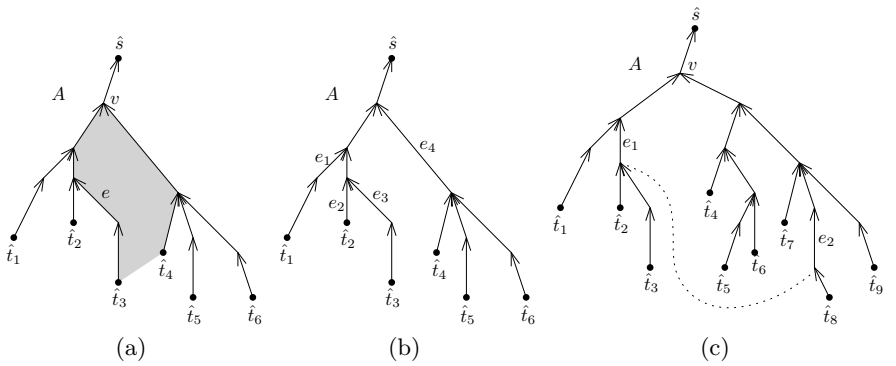


Fig. 6. A-induced order and wedges: (a) The A-induced order of vertices in $\hat{T}$ is $\hat{t}_1, \ldots, \hat{t}_6$. The wedge between $\hat{t}_3$ and $\hat{t}_4$ is highlighted – it is defined by the $\hat{t}_3$-v path (its left path) and the $\hat{t}_4$-v path (its right path). The wedge is on the right of the edge e; the wedge on the left of e is between $\hat{t}_2$ and $\hat{t}_3$. (b) The edges e_1, e_2, e_3, e_4 are listed in the A-induced order. (c) Edge e_1 is five wedges apart from e_2, since the region defined by the e_1-v path, the e_2-v path and the dotted curve contains 4 vertices from $\hat{T}$.

We need some additional terminology before stating the decomposition lemma.

Suppose we reverse the edges of A and perform a depth-first traversal from $\hat{s}$, going through the neighbors of the current vertex in the counterclockwise order, starting from the edge we used to get to the vertex. Let $\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_k$ be the order in which we visited the vertices in $\hat{T}$ (we refer to this order as the *A-induced order of vertices in $\hat{T}$*). Let $\hat{t}_{k+1} := \hat{t}_1$. For every $i \in \{1, \ldots, k\}$ we define a *wedge* as follows. Let v be the first common successor of $\hat{t}_i$ and $\hat{t}_{i+1}$ when restricted only to edges in A. The path from $\hat{t}_i$ to v, using only edges in A, forms the *left path* of the wedge; similarly, the path from $\hat{t}_{i+1}$ to v, using only edges in A, forms the *right path*. Every edge $e \in A$ is in two wedges: the *left* and the *right wedge of* e, for which e lies on the right and the left path, respectively. See Figure 6(a).

In addition to the A-induced order of vertices in $\hat{T}$, we also define the A-induced order of pairwise independent edges in A as follows. Let $e_1, e_2, \ldots, e_q \in A$, where for every $i \neq j$, e_i is not a successor of e_j in A. Suppose that we perform the same depth-first traversal of A as described in the previous paragraph. Then, if we visit the edges $e_1, e_2, \ldots, e_q$ in this order (or its cyclic rotation), we say that the edges are ordered in the *A-induced order of edges*, see Figure 6(b).

We say that edge $e_1 \in A$ is j *wedges apart* from $e_2 \in A$ if there are $j-1$ intermediate vertices from $\hat{T}$ between e_1 and e_2. More precisely, let v be the first common successor of e_1, e_2 when using only edges in A. Connect the starting vertices of e_1 and e_2 by a curve in the plane so that the curve does not touch any of the edges in A. Then, consider the region on the right of the e_1-v path in A, on the left of the e_2-v path in A, and bounded by the curve; if it contains exactly $j-1$ vertices from $\hat{T}$, we say that e_1 is j wedges apart from e_2, see Figure 6(c).

The following lemma describes how to decompose a non-crossing tour that separates $\hat{s}$ from $\hat{T}$ into paths in the restricted dual graph H'_d (and edges in H'_D

Algorithm 2 Counting non-crossing tours in the dual graph H'_D that contain all vertices in $\hat{T}$ in the inside region and $\hat{s}$ in the outside region

1: **for** every $e_1, e_2 \in A$ **do**
2: compute $\xi[e_1, e_2]$, the number of paths in H'_d from the right face of e_1 to the left face of e_2 (in linear time, since H'_d is acyclic)
3: let p be a path from one of the vertices in $\hat{T}$ to $\hat{s}$, using only edges in A
4: **for** every $e_1 \in p$ **do**
5: **for** every $e_2 \in A$ such that e_1 is exactly 1 wedge apart from e_2 in A **do**
6: let $a[e_1, e_2] = \xi[e_1, e_2]$
7: **for** $j = 2$ to $|\hat{T}| - 1$ **do**
8: **for** every $e_2 \in A$ such that e_1 is exactly j wedges apart from e_2 in A **do**
9: let

$$a[e_1, e_2] := \sum_{e' \in p'} a[e_1, e']\xi[e', e_2],$$

where p' is the left path of e_2's left wedge
10: **for** every $e \in p$ **do**
11: let $a[e, e] := \sum_{e' \in p'} a[e, e']\xi[e', e]$, where p' is the left path of e's left wedge
12: return $\sum_{e \in p} a[e, e]$

that connect the paths to form the tour). In essence, the tour is cut into path segments by the tree A.

Lemma 3. *Under the assumptions of Definition 2, let X be a tour in H'_D, the dual graph of H'. The tour X is non-crossing and separates $\hat{s}$ from $\hat{T}$ if and only if all of the following conditions hold: (1) there exist edges $e_1, e_2, \ldots, e_q \in A$ such that for every $i, j \in \{1, \ldots, q\}$, $i \neq j$, e_i is not a successor of e_j in A, (2) $e_1, \ldots, e_q$ is the A-induced order of these edges, (3) for every $\hat{t} \in \hat{T}$ there exists i such that e_i is on the path from $\hat{t}$ to $\hat{s}$ in A, (4) there exists a path p_i in H'_d from the right face of e_i to the left face of e_{i+1} (let $e_{q+1} := e_1$), and (5) $X = p_1, d_1, p_2, d_2, \ldots, p_q, d_q$, where d_i is the dual of the edge e_i.*

Lemma 3 yields a dynamic programming algorithm (Algorithm 2) for counting all non-crossing tours separating $\hat{s}$ from $\hat{T}$. By gradually increasing the wedge distance, it counts walks (paths with repeated vertices) in the dual graph H'_D starting and ending in faces both bordering an edge in A. In particular, $a[e_1, e_2]$ is the number of walks starting with the right face of e_1 and ending with the left face of e_2. For e_1, e_2 at distance 1, $a[e_1, e_2]$ can be computed by a topological traversal of H'_d. For larger distances, the computation goes through an intermediate edge e' on the left wedge of e_2, "responsible" for separating $\hat{s}$ from the $\hat{t}_i$ on this wedge (see step 9). Correctness of Algorithm 2, along with Theorem 2 and Lemma 1 imply the counting claim of Theorem 3.

For the sampling part, we first choose e on the path p proportionally to $a[e, e]$. Then we choose e' proportionally to $a[e, e']\xi[e', e]$, etc., getting the edges $e_1, e_2, \ldots, e_q$ from Lemma 3. Then we independently sample each path p_i (see [2] for the details of this step), obtaining a non-crossing tour and the corresponding contiguous minimum (s, T)-cut.

We conclude the paper with two remarks. First, the running time bound $O(n^3)$ is tight. This can be seen by forming a graph with three paths of length $n/3$ starting at the same vertex s and ending at t_1, t_2, t_3, respectively (except for s, the paths are disjoint). Second, the $O(n^3)$ preprocessing time might seem prohibitively large for larger data sets. We note that in practice the graph H', formed by contracting a typically very sizable set of edges, is likely going to be significantly smaller than the original graph. Combining this with a faster network flow algorithm [4], the overall running time becomes much more practical.

References

1. Ball, M.O., Provan, J.S.: Calculating bounds on reachability and connectnedness 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. Theor. Comp. Sci. 417, 2–11 (2012)
3. Borradaile, G., Klein, P.N.: An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. J. ACM 56(2) (2009)
4. Borradaile, G., Klein, P.N., Mozes, S., Nussbaum, Y., Wulff-Nilsen, C.: Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In: Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS). pp. 170–179 (2011)
5. Borradaile, G., Sankowski, P., Wulff-Nilsen, C.: Min st-cut oracle for planar graphs with near-linear preprocessing time. In: Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS). pp. 601–610 (2010)
6. Boykov, Y., Veksler, O.: Graph cuts in vision and graphics: Theories and applications (2006), in Handbook of Mathematical Models in Computer Vision, edited by N. Paragios, Y. Chen and O. Faugeras, Springer
7. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. 23(11), 1222–1239 (2001)
8. Creed, P.: Counting and sampling problems on Eulerian graphs (2010), Ph.D. Dissertation, University of Edinburgh
9. Ge, Q., Štefankovič, D.: The complexity of counting Eulerian tours in 4-regular graphs. Algorithmica 63(3), 588–601 (2012)
10. Grady, L.: Minimal surfaces extend shortest path segmentation methods to 3D. IEEE Trans. Pattern Anal. Mach. Intell. 32(2), 321–334 (2010)
11. 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 ACM Symposium on Theory of Computing (STOC). pp. 313–322 (2011)
12. Liskiewicz, M., Ogihara, M., Toda, S.: The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. Theoretical Computer Science 304(1-3), 129–156 (2003)
13. Provan, J.S., Ball, M.O.: The complexity of counting cuts and of computing the probability that a graph is connected. SIAM J. Comput. 12(4), 777–788 (1983)
14. Vicente, S., Kolmogorov, V., Rother, C.: Graph cut based image segmentation with connectivity priors. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
15. Zeng, Y., Samaras, D., Chen, W., Peng, Q.: Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in n-d images. Computer Vision Image Understanding 112, 81–90 (2008)