12-2017

An Alternative Approach to Counting Minimum (s; t)-cuts in Planar Graphs

Rachel E. Silva
res8493@rit.edu

An Alternative Approach to Counting Minimum (s, t)-cuts in Planar Graphs

by

Rachel E. Silva

THESIS

Presented to the Department of Computer Science
of the Golisano College of Computing and Information Sciences
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Computer Science

Rochester Institute of Technology

December 2017

Abstract

An Alternative Approach to Counting Minimum (s, t)-cuts in Planar Graphs

Rachel E. Silva, M.S.
Rochester Institute of Technology, 2017

Supervisor: Dr. Ivona Bezáková

Finding and counting minimum cuts in graphs can be useful in image processing and segmentation and in networking systems such as computer or road networks. Researchers have previously developed polynomial-time algorithms to count minimum cuts in planar graphs which utilize the relationship between maximum network flows and minimum cuts.

This thesis presents a polynomial-time algorithm to count the number of minimum (s, t)-cuts in a planar graph without first finding a maximum flow. The presented algorithm is dependent on the finding that (s, t)-cuts in a planar graph correlate to certain cycles found in the dual of that graph, which can be efficiently counted.

Table of Contents

Chapter 1

Introduction

Graph cuts are useful in many applications in imaging, vision, and networking. With a network of computers or roads, a minimum cut can determine the danger of disconnect between groups of computers or cities, respectively. In imaging, cuts are used as an energy minimization tool in a variety of instances, such as image restoration, image segmentation, and finding disparities or recreating object shapes from stereo inputs [BK04, BV06].

Image segmentation is a clear and simple example of the importance of minimum cuts in planar graphs. Consider an image as a grid of pixels with varying color intensities. A pixel can be directly connected to its neighboring pixels to its north, south, east, and west with each of these connections given a weight based on the similarity of the two pixels' intensities. Edges between similar pixels are given a higher weight than those of extremely different pixels, thus minimizing the total energy required to separate different groups of similar pixels, allowing for the segmentation of objects via a graph cut in an image [BK04]. The edge weights of the graph may be calculated from cost functions based on any combination of attributes of the pixels, not just on the color values or intensities between two adjacent pixels.

Calculating a minimum (s,t)-cut in an image, in order to segment a specific object of interest, may require a user to interactively choose the source, s, and sink, t, nodes. User interaction can be particularly useful in sensitive applications, such as object extraction in medical imaging, to ensure that the correct objects are segmented properly and to a users specifications. As there is not always a single minimum cut in a graph, the counting and sampling of cuts is also useful, particularly in such sensitive fields [BF12, UHCC14].

Ball and Provan introduced a method of counting minimum (s,t)-cuts in a directed planar graph, provided the source and sink vertices are located on the same face and provided that every vertex in the graph is on some path from s to t [BP83]. This polynomial-time algorithm depends on their result showing that there is a one-to-one mapping from the minimum (s,t)-cuts in the graph G to antichains in a processed version of the same graph, G'. An antichain is a set of vertices such that, for every vertex in the set, no predecessors of the vertex are in the set. [BP83] shows that, by adding an edge from s to t in G' across the face they share, the number of minimum cuts in G is the same as the number of unique paths between the two sections of the split face in the dual graph of G'.

Bezáková and Friedlander have since developed an algorithm to count minimum (s,t)-cuts in a directed planar graph, with no restriction on the positioning of s and t [BF12]. This algorithm does still require that each vertex in the original graph lies on some path from s to t. This algorithm is based upon the same concepts as that of [BP83], but rather than relying on

fortunate placement of the source and sink, it simulates the effect of having a shared face by instead manipulating a path between s and t.

[BP83] and [BF12] reduce the number of minimum cuts in a graph to the number of paths between certain faces in the dual of the residual graph of a maximum flow. Some details regarding these algorithm can be found in Section 2.1. The algorithm in this thesis introduces a method of counting the minimum (s,t)-cuts of a planar graph without first calculating the maximum flow. This method utilizes the relationship between a graph and its dual, as brought to light by [BP83], but allows for the cuts to be counted more directly and with less preprocessing of the graph.

Let $G = (V, E, w)$ be a undirected planar graph with a source s and sink t. Let s_d be a face of G adjacent to s and let t_d be a face of G adjacent to t. Given a path from s_d to t_d in G_d, it can be shown that a minimum cut will intersect the path at least once. This thesis explores the ways in which a cycle in the dual will interact with a right-most shortest path from s_d to t_d, allowing for the unique minimum cycles intersecting such a path to be counted.

Chapter 2

Preliminaries

Let $G = (V, E, w)$ be a *graph*, where V and E are the sets of vertices and edges of the graph, respectively, and w is the function mapping each of the edges in E to a positive edge weight. If G is a *directed graph*, $(u, v) \in E$ is an ordered pair of vertices, $u, v \in V$ such that (u, v) is directed from u to v. If G is *undirected*, $(u, v) \in E$ has no orientation and is identical to $(v, u) \in E$. Unless otherwise specified, all graphs discussed are multigraphs; they may contain two or more edges connecting the same two vertices as well as edges that connect a vertex to itself. A *path* in G is a sequence of adjacent vertices in V such that no vertices may repeat with the exception of the endpoints. A *cycle* in G is a path such that the beginning vertex is the same as the end vertex [Wes01]. A subset of a path p between vertices u and v in p may be denoted as $p(u, v)$. The magnitude of a path, $|p|$, is the sum of the weights of all of the edges of the path.

Let s, the source, and t, the sink, be two vertices in G such that $s \neq t$. An (s, t)-*cut* is a set of vertices $S \subset V$ such that $s \in S$ and $t \notin S$. The weight of a cut is the sum of all weights of the edges (u, v) where $u \in S$ and $v \notin S$. A minimum cut is such that the weight of the cut is no greater than the weight

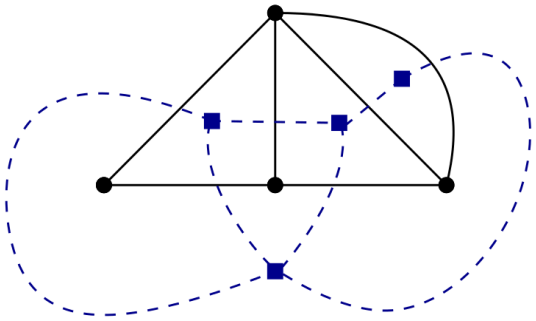


Figure 2.1: An example of a planar graph and its dual. Dual graphs are denoted by having dashed edges and square vertices. This convention will be used consistently through following visual examples.

of any other cut in the graph.

Given a directed graph $G = (V, E, w)$, where w has only non-negative weights, a *flow* is a function f that assigns a value $f(e)$ to each edge e. $f^+(v)$ is the sum of the values $f(e)$ for all edges e beginning at v, and $f^-(v)$ is the sum of $f(e)$ for all e ending in v. The total value of an (s,t)-flow is the sum of all $f(e)$ for all $e \in E$. Given a source vertex, s, and a sink vertex, t, a feasible (s,t)-flow is a flow that satisfies the capacity constraints imposed by the graphs weights, $0 \leq f(e) \leq w(e)$, and conservation constraints, $f^+(v) = f^-(v)$ for all $v \in V$ excluding s and t. A maximum (s,t)-flow in G is a feasible flow that has a value no less than any other (s,t)-flow in G. A *residual graph* of a flow models the available capacity of the edges. For every edge $e = (u,v) \in E$, excluding 0 weight edges, the residual graph contains two edges, $e_1 = (u,v)$ and $e_2 = (v,u)$, such that the weight of $e_1 = w(e) - f(e)$ and the weight of $e_2 = f(e)$. [FF56]

A graph G is *planar* if it can be embedded in a plane with no edges

intersecting. The plane is divided into faces, each separated by the edges of the graph. The *dual graph* of an undirected planar graph G, hereby expressed as $G_d = (V_d, E_d, w_d)$, can be created as follows. Each vertex in V_d corresponds to a unique face of G. For every edge (u, v) in E, there is a corresponding edge (w, z) in E_d such that w and z correspond to the faces in G that border (u, v). The weight of (w, z) is equal to that of (u, v). An example of a graph and its dual in Figure 2.1.

A planar graph may have multiple planar embeddings, which may have non-isomorphic dual graphs. Once a planar embedding of a graph is found, the graph must be represented in a way that will preserve the embedding itself; standard graph representation and data storage generally does not store or dictate the positions of the vertices. To maintain the embedding of the planar graph, an adapted adjacency list will be used, as described below, and pictured in Figure 2.2. This data struture, introduced, formalized by Heffter [Hef91], Edmonds [Edm60], and Youngs [You63], is further explained by Archdeacon [Arc97].

Every vertex in the graph will have an associated list of vertices that are adjacent to it. This adjacency list is sorted in the radial order in which the vertices appear around the vertex to which they are adjacent. For example, if vertex v is adjacent to the vertices $u_1, u_2, \ldots, u_n$, listed in that order, the edges $(v, u_1), (v, u_2), \ldots, (v, u_3)$ would be in clockwise order around v. The first vertex to be listed in the adjacency list can be chosen arbitrarily, as the list can be considered cyclically continuous, with the first vertex immediately

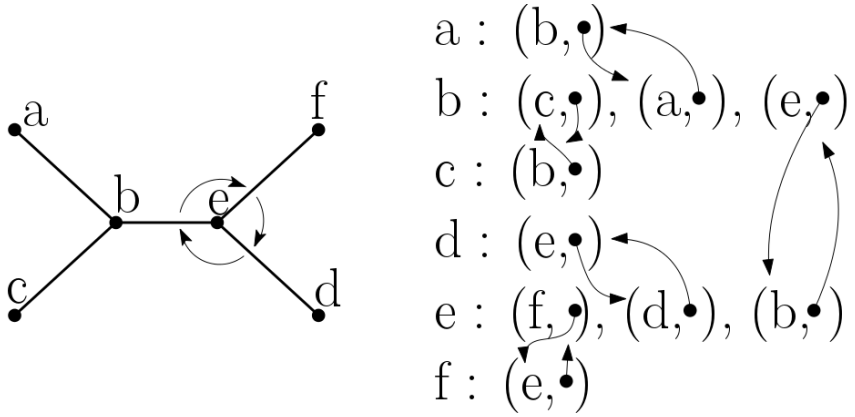


Figure 2.2: A planar graph embedding and its corresponding adjacency list representation

following the last. Additionally, to facilitate seamless traversal of the graph, the locations of each endpoint of an edge within an adjacency list contain a pointer to the opposite endpoint. If the vertices v and u were adjacent, the instance of u located in v's adjacency list would point to the location of v in u's list, and vice versa.

For the sake of simplifying the algorithm presented, the idea of a *right-most shortest* (u, v)*-path* will be introduced here. A right-most shortest (u, v)-path can be defined as a shortest path directed from u to v such that no other path of equal length can branch off of the path from a position clockwise to the direction of the path. In this case, the term right-most may be considered synonymous to being most clockwise. There may exist more than one right-most shortest (u, v)-path in a graph because vertices are listed cyclically and no one vertex adjacent to u may be considered more right than the others. However, once a second vertex along the (u, v)-path has been chosen, there is

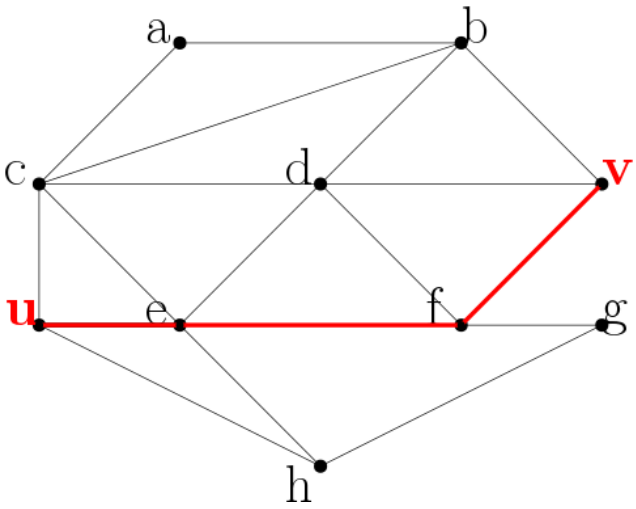


Figure 2.3: An example of right-most shortest (u, v)-path in a planar graph with uniform weight

at most one right-most shortest (u, v)-path with that second vertex. Unless otherwise specified, the right-most shortest path may be arbitrarily chosen. Figure 2.3 shows a planar graph with uniform edge weights and a right-most shortest path from u to v is highlighted. In this example, it is clear that the path u, e, d, v also has a weight of 3 but the edge (e, f) branches from that path further clockwise, without rotating past the previous edge in the path. The highlighted path, however, is not a unique right-most shortest path. The path u, c, d, v also has a weight of 3, and there are no (u, v)-paths of equal or shorter distance that split from the path anywhere but the beginning vertex. Because the listing of vertices is cyclic, this could not be considered "less right" than the path u, e, f, v.

2.1 Previous Work

Bezáková and Friedlander found that for every planar weighted graph G with s and t, there exists a graph $\widehat{G}$ such that there is a one-to-one mapping between the minimum (s,t)-cuts in G and the number of forward $(\widehat{t},\widehat{s})$-cuts in $\widehat{G}$. A forward cut is a set $S \subset V$ such that for any vertex v in S, all of its predecessors are also in S [BF12].

The process to create $\widehat{G}$ is as follows. A maximum flow must be found within G such that the direct flow edges form no cycles. The residual graph resulting from this process is what will be used moving forward. Because the residual graph contains the reverse edges of the flow, the focus shifts to paths and flows in the direction from t to s. Requiring that the direct flow edges are acyclic does not guarantee that the edges in the residual graph also have no cycles. Though there may be cycles in the residual graph, it is shown in [BF12] that no minimum cut will cut through a strongly connected component of a graph. To remove cycles and simplify the counting process, a new graph, $\widehat{G}$, is created by contracting all of the strongly connected components of the residual graph. That is, for each set of vertices in the graph that form a cycle, that set is contracted into a single vertex. The vertices corresponding to s and to t in G are $\widehat{s}$ and $\widehat{t}$, respectively, in $\widehat{G}$.

Once the graph $\widehat{G}$ is found, the forward $(\widehat{t},\widehat{s})$-cuts must be counted to find the minimum (s,t)-cuts in G. Let p be a path from $\widehat{t}$ to $\widehat{s}$ in $\widehat{G}$. For simplicity, consider that the path p is oriented horizontally from left to right. The author refers to above and below the horizontal path to be north and

south, respectively. Each edge e in p is duplicated, with a new edge, e', being drawn to the north of its original. In this process, each face north of p is divided into two or more faces; let the face south of each e' be f_e^{south}, and the face north of each e' be f_e^{north}. Let p' be the path formed by the newly created edges, and let G' be $\widehat{G}$ with the edges of p'. Let G'_d be the dual graph of G', with the edges corresponding to p' excluded from the graph. It is shown in [BF12] that the number of forward $(\widehat{t}, \widehat{s})$-cuts in $\widehat{G}$ is equal to the sum of the number of paths in G'_d from f_e^{south} to f_e^{north} for each edge e in p.

Chapter 3

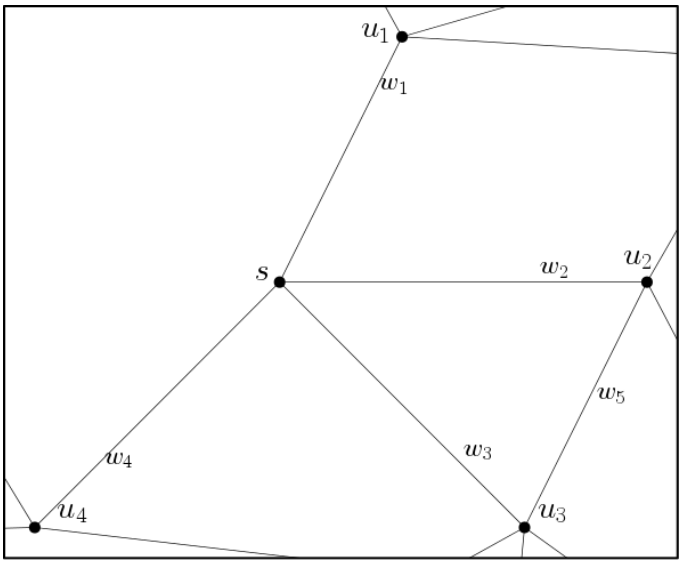
Constructing a Dual Graph

In this chapter, a dual graph will be constructed from a planar graph, which can be used to count the number of minimum (s, t)-cuts of the planar graph. To transfer the concept of the source and sink vertices to the dual graph, a face will be constructed around each of the two vertices in the original graph before the dual is constructed. Additionally, it will be shown that there is a one-to-one correlation between minimum (s, t)-cuts in a graph and cycles in its dual.

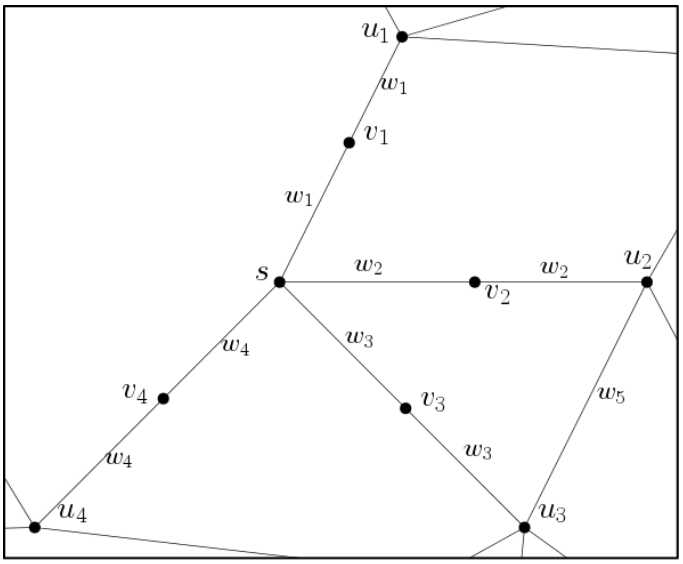
3.1 Creating s/t faces

This section discusses the correlation of minimum cuts in a planar graph G with cycles in a constructed dual graph. However, to find minimum (s, t)-cuts using a dual, there must be some correlation from s and t in G to vertices in the dual. The graph G' has been designed so that each of the vertices in G' corresponding to $s, t \in G$ has only one incident face; the dual of the graph G' will then have a single vertex corresponding to each of those faces, here denoted s_d and t_d. The construction of G' is demonstrated in Figure 3.1.

Let G be an undirected planar graph with vertices s and t.

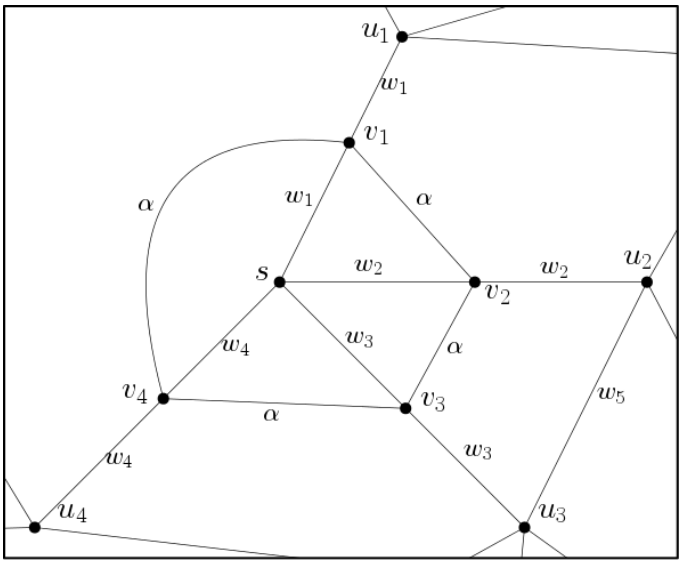


(a) A portion of G' containing s

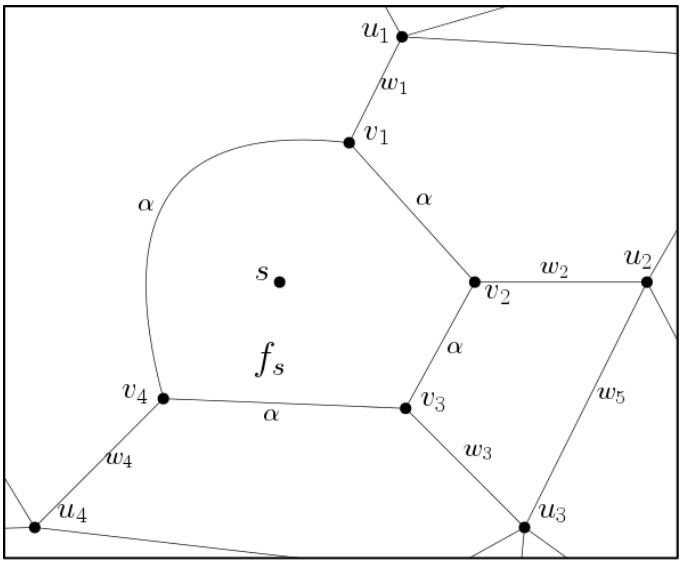


(b) Vertices are added to split each edge adjacent to s

Figure 3.1: An example of the steps to create f_s in G'



(c) The vertices $v_1, \ldots, v_i$ are connected in a clockwise direction



(d) All edges adjacent to s are removed

Figure 3.1: An example of the steps to create f_s in G' (cont.)

Initially, let G' be a copy of G. Let $u_1, u_2, \ldots, u_n$ be the vertices adjacent to s, in rotational order. In G', bisect each edge (u_i, s) with a new vertex, v_i, giving each new edge (u_i, v_i) and (v_i, s) the same weight as the former edge (u_i, s). For each vertex $v_i, i < n$, let (v_i, v_{i+1}) be an edge with a weight of α. Conceptually, α can be considered to be a weight of infinity, but to simplify future mathematical steps, the weight must be a finite number of sufficient size, such as the sum of all the edge weights in the graph plus 1. Let (v_n, v_1) be an edge with a weight of α. Delete all edges incident to s in G'. Repeat this process with respect to the vertex t. The single face in G' bordering s will be referred to as f_s, and the face bordering t, f_t.

Lemma 3.1. *The number of minimum (s,t)-cuts in G is equal to the number of minimum (s,t)-cuts in G'.*

Proof. Let S be the set of vertices in G' that border the face f_s, and T be the set of vertices that border f_t. The sets S and T are comparable to the vertices s and t in G with regards to minimum (s,t)-cuts. If the set S was contracted into a single vertex, as was the set T, the graph would be equivalent to G; each of the edges and vertices in G are still present in G'. Thus, every minimum (s,t)-cut in G has an equivalent (s,t)-cut in G', with S included in the cut set, and T excluded from the cut set. G' cannot have fewer (s,t)-cuts than G.

Each edge (u,v), $u, v \in S$ has a weight of α, and therefore the set S cannot not be divided by a minimum cut set; by design, α is a weight greater than any possible minimum cut weight. Likewise, the set of vertices bordering

the face f_t cannot be divided by a minimum cut set. Thus, the graph G' cannot have more minimum (s,t)-cuts than G has, merely by the introduction of these new edges in the graph. □

3.2 Minimum Cuts in G and its Dual

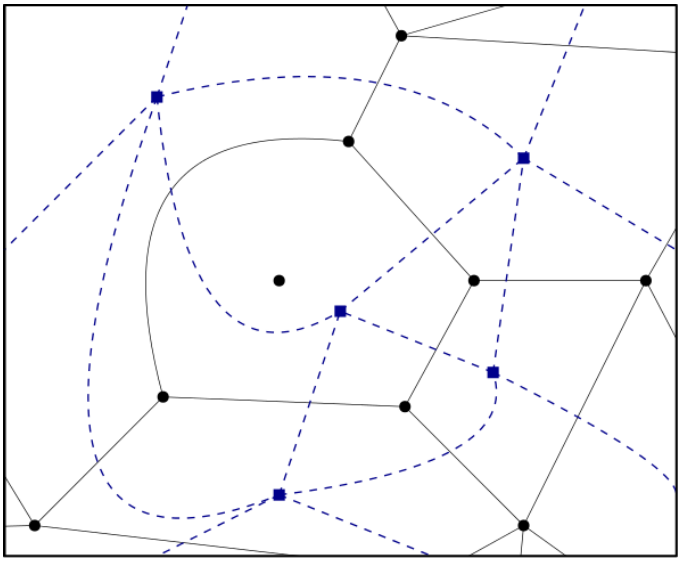
Construct G'_d to be the dual graph of G'. The vertices corresponding to the faces f_s and f_t will be s_d and t_d, respectively. Figure 3.2 shows the dual graph of the example in Figure 3.1.

For every (s,t)-cut in the graph G', a corresponding set of cycles in G'_d can be said to represent that cut. This corresponding set of cycles can be constructed by including each of the edges in G'_d that border the cut set of the (s,t)-cut, as in Figure 3.3. The weight of these cycles will be equal to the weight of the cut, as each of the cycles' edges will intersect an edge in G' with one endpoint in the cut set and the other outside of the cut set.

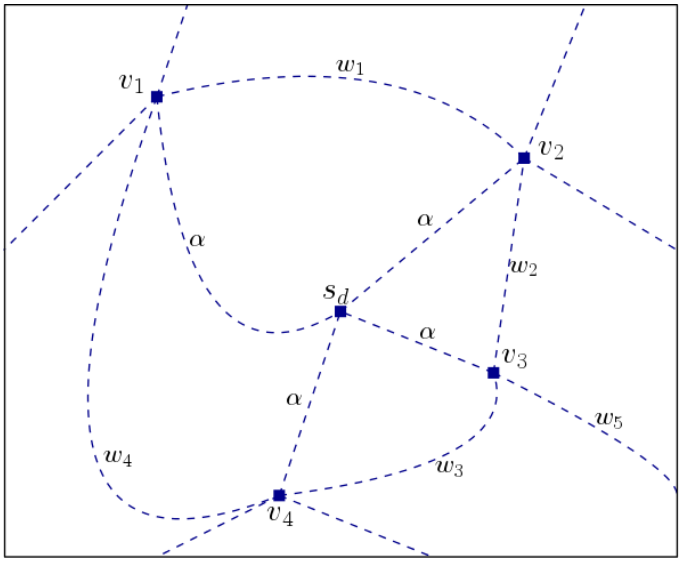
Lemma 3.2. *There is one-to-one correlation between (s,t)-cuts in G' and cut representing cycle sets in G_d .*

Proof. As each set of cycles representing a cut in G'_d directly separates the vertices of a cut set S from the rest of the graph, no two cycle sets in G'_d can represent the same cut set. Likewise, no two cut sets can correspond to the same set of cycles in G'_d. □

Lemma 3.3. *A minimum (s,t)-cut in G is represented by a single cycle in G'_d.*

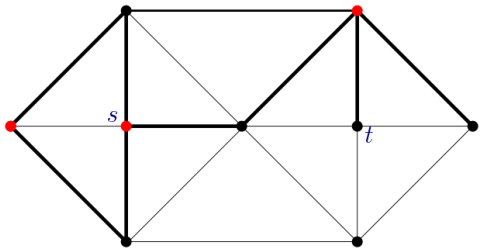


(a) G_d overlaid on G'

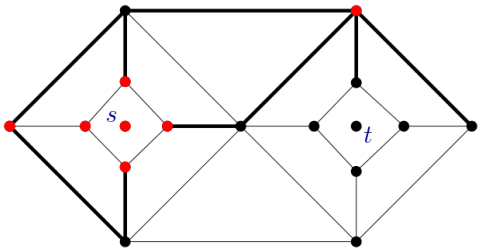


(b) G_d with the appropriate edge weights labeled. Recall that the weights of the edges are derived from the weights of the edges in G'

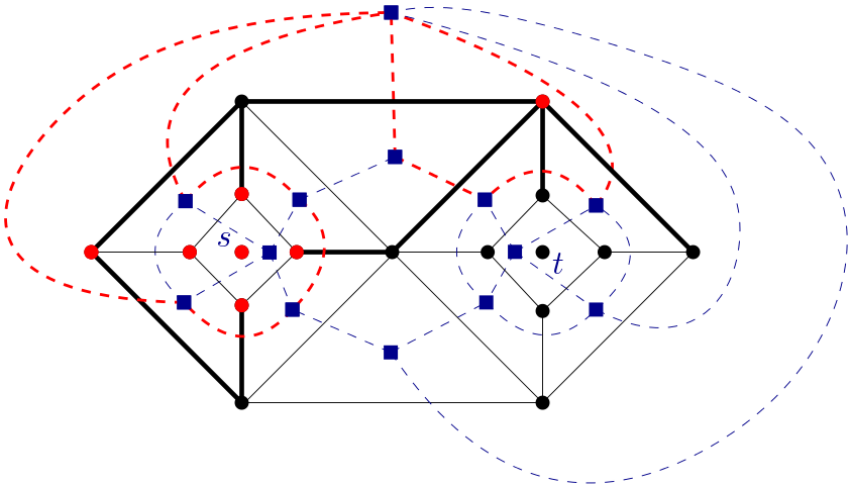
Figure 3.2: A portion of the dual of the graph G' constructed in 3.1



(a) A planar graph G with cut set S and its adjacent edges emphasized



(b) The graph G' from G, with the same cut set emphasized



(c) G'_d with the cycles corresponding to the cut set S emphasized

Figure 3.3: An (s, t)-cut in a graph overlaid on its dual

Proof. Let S be an (s,t)-cut set in G. Then S' will be the vertices in G' that correspond to S in G. That is, S' will contain all of the vertices in S as well as the new vertices introduced in G' that are adjacent to s. Let E' be the set of edges in G' such that $u \in S',\ v \notin S'$ and E'_d be the set of edges in G'_d that intersect with E'.

Since every edge directly connecting vertices in S' to vertices not in S' are intersected by an edge in E'_d, the edges in E'_d must form a set of cycles in G'_d. This is illustrated in Figure 3.3. The weight of the edges in E'_d is the weight of the cut it represents.

Suppose E'_d consists of multiple cycles $c_1, c_2, \ldots, c_n,\ n > 1$, as in Figure 3.3c. Let c_1 be a cycle such that s is contained on one side of the cycle and t is on the other. Consider that the set E'_d without the cycle c_n is also an (s,t)-cut and that it has a smaller weight than the cut defined by E'_d. Thus, a minimum (s,t)-cut in G' must be represented as a single cycle in G'_d; any set of multiple cycles representing a cut can be reduced in size (and weight) by removing any cycle in the set as long as one remains that separates s from t.

□

A single cycle in G'_d that corresponds to an (s,t)-cut in G' is a *separating cycle*, pictured in Figure 3.4. In G'_d, a separating cycle will create a boundary between s_d and t_d such that any (s_d,t_d)-path must share at least one vertex with the separating cycle. A minimum separating cycle is such that there is no separating cycle in the graph with a smaller weight.

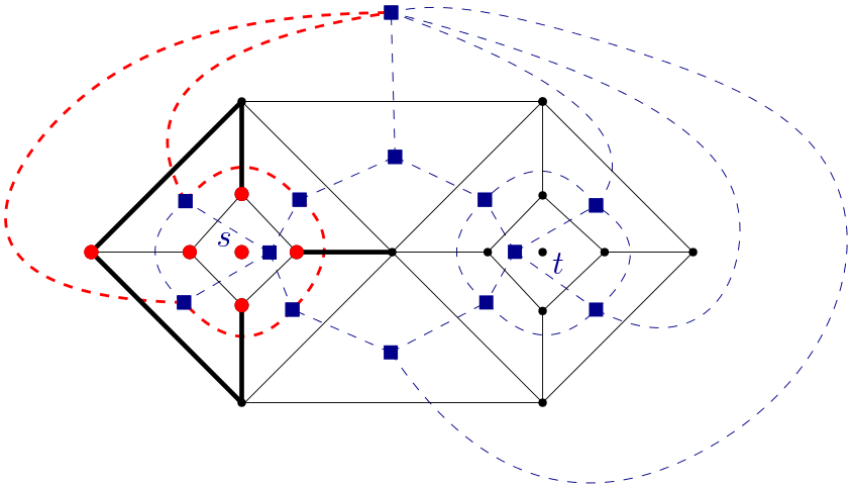


Figure 3.4: G'_d with a separating cycle based on Figure 3.3

Lemma 3.4. *Every minimum separating cycle in G'_d corresponds to a unique minimum (s,t)-cut in G.*

Proof. Let c be a minimum separating cycle in G'_d. There must be an (s,t)-cut that corresponds to c, as the edges of c separate s from t in G'. Since the weight of a separating cycle is equal to the weight of its corresponding (s,t)-cut and every (s,t)-cut has a corresponding separating cycle, the minimum weight of a separating cycle is the same as that of a minimum (s,t)-cut in G'. Therefore, as c has the smallest possible weight of a separating cycle in G'_d, it must correspond to a minimum (s,t)-cut in G'. By Lemma 3.2, c corresponds to a unique minimum (s,t)-cut in G'.

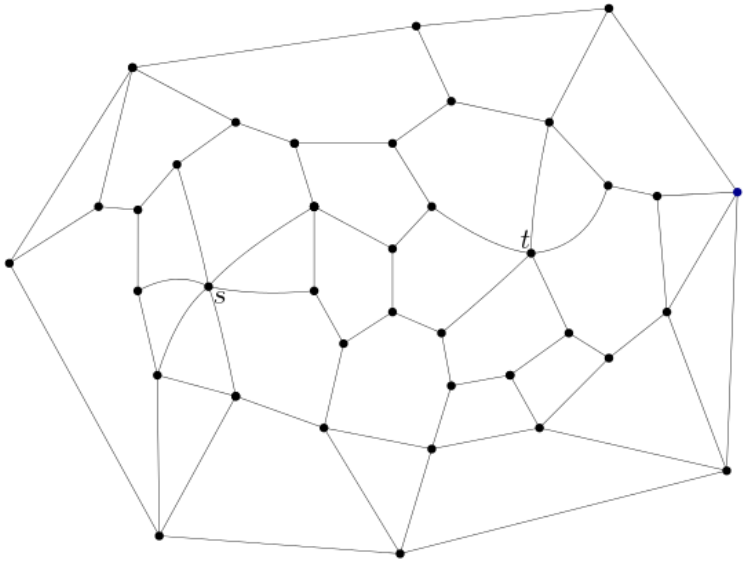
□

Chapter 4

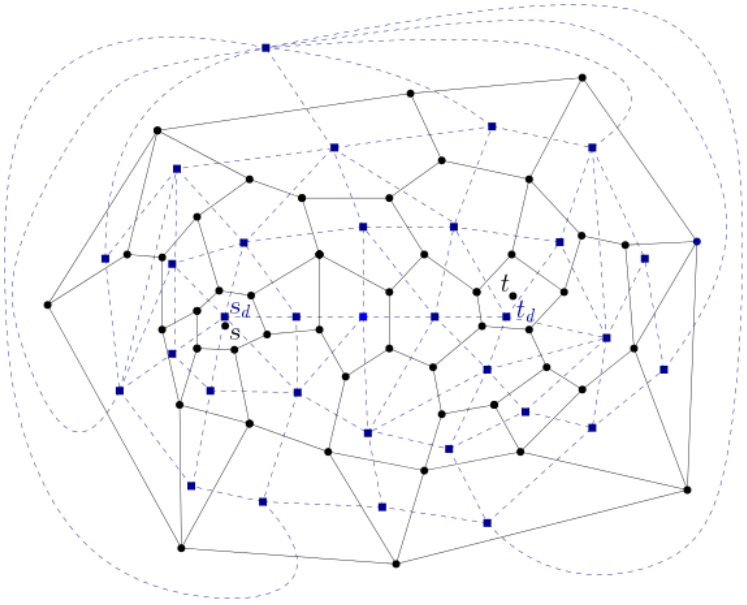
Relating (s, t)-cuts to the Dual

This chapter analyzes separating cycles in the dual of a graph G' and the behavior of minimum separating cycles, which correspond to minimum cuts in G'. Specifically, the ways in which a minimum separating cycle will interact with a right-most shortest path between the source and the sink of a dual graph are analyzed.

Let G be a planar undirected graph, and G'_d be the dual of G', as constructed in Chapter 3. Let p be a right-most shortest path from s_d to t_d, as defined in Chapter 2. For simplicity, consider p to be represented horizontally, with s_d as the leftmost vertex in p and t_d as the right-most vertex in p. This is shown in Figure 4.1c. With this orientation in mind, vertices adjacent to p will be referred to as *above* p if the vertex is to the left of the path, and as *below* p if the vertex is to the right of the path. Likewise, an edge incident to the path p can be considered above or below the path based on its orientation. The relationship of vertices below and above the path p are illustrated in Figure 4.2. A vertex may be considered both above and below p if there exist two edges incident to the vertex such that one is incident to the path from above, and the other from below.

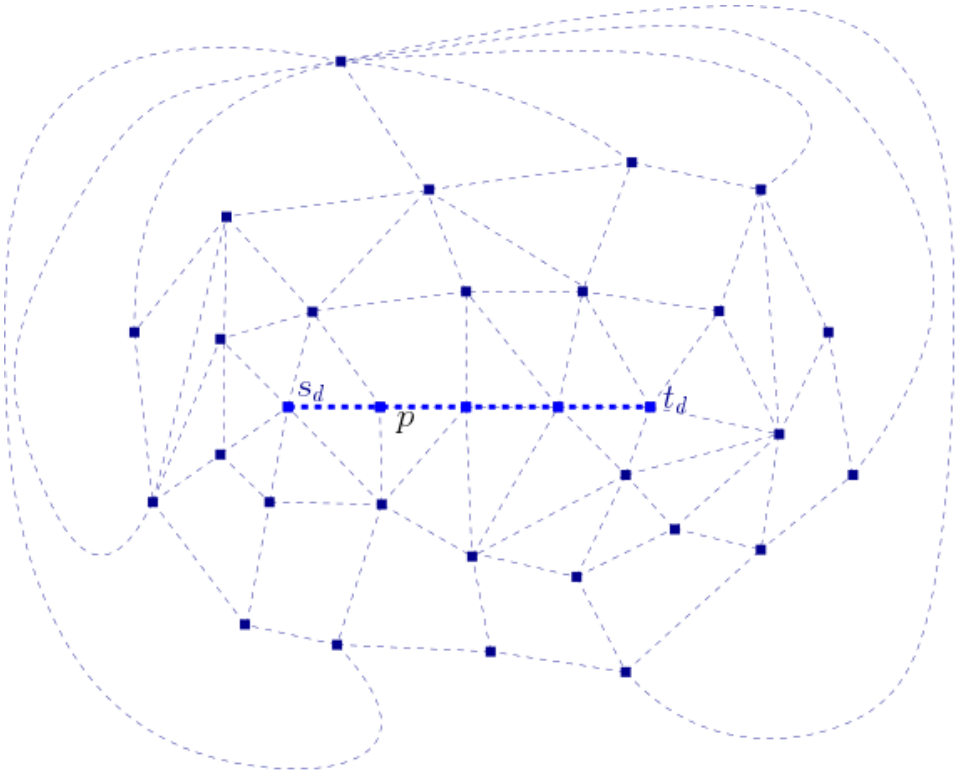


(a) A planar graph G



(b) The graphs G' and G_d overlaid on one another

Figure 4.1: Construction of the dual G_d



(c) G_d with the right-most shortest path p

Figure 4.1: Construction of the dual G_d(cont.)

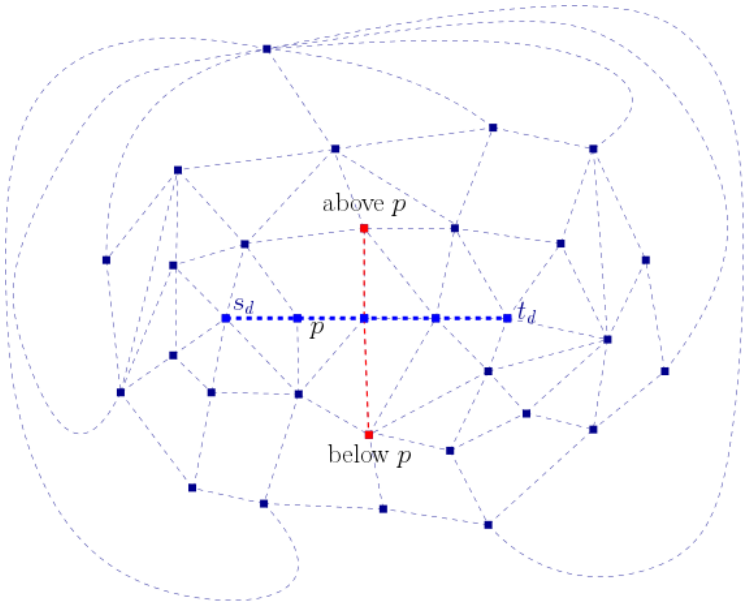


Figure 4.2: Vertices above and below a path, p

Let c be a minimum (s_d,t_d)-separating cycle. Then c may share edges with p, as well as contain edges that are incident to p that are not a part of the path. The cycle may *intersect* the path completely at some point, meaning there exists some subpath of the cycle c, $q = v_1, v_2, \ldots, v_n$, such that v_1 is adjacent to p from above, v_n is adjacent to p from below, and the internal vertices, $v_2, \ldots v_{n-1}$, are in p. The cycle may also share a subset of vertices with p without intersecting the path by containing some subpath of c, $q = v_1, v_2, \ldots, v_n$, such that either both v_1 and v_n are adjacent to p from above or both are adjacent to p from below while the internal vertices of the path, $v_2, \ldots, v_{n-1}$, are in p. Paths intersecting p and sharing vertices without intersecting p are shown in Figure 4.3.

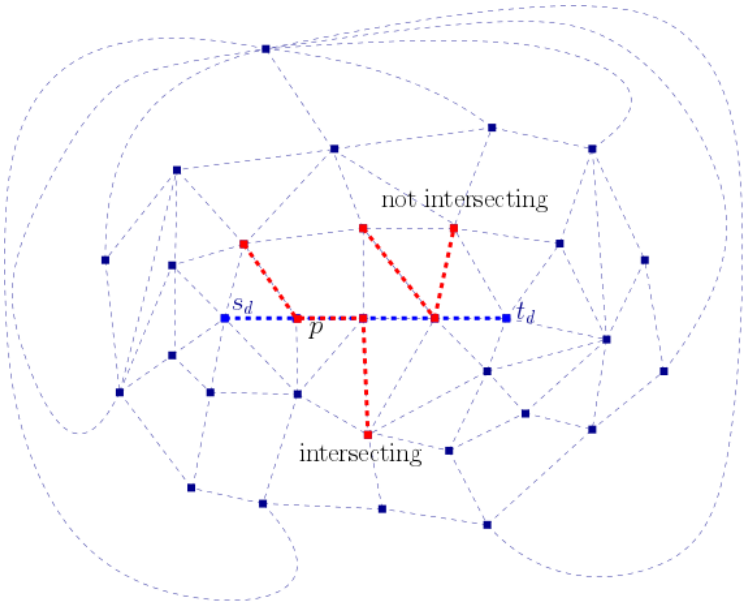


Figure 4.3: Example of paths intersecting or not intersecting p

Lemma 4.1. *A minimum (s_d,t_d)-separating cycle intersects a right-most shortest (s_d,t_d)-path at least once.*

Proof. Let c be a minimum (s_d,t_d)-separating cycle and p a right-most shortest (s_d,t_d)-path in G'_d. Then c must intersect p. If this were not the case, all of p, including s_d and t_d, would be contained within one side of the cycle c, and so c would not separate s_d from t_d. □

Consider a path q in G'_d such that only the end points of the path are vertices in p. If both ends of the path q are incident to p from below, the path is considered to be *entirely below* p.

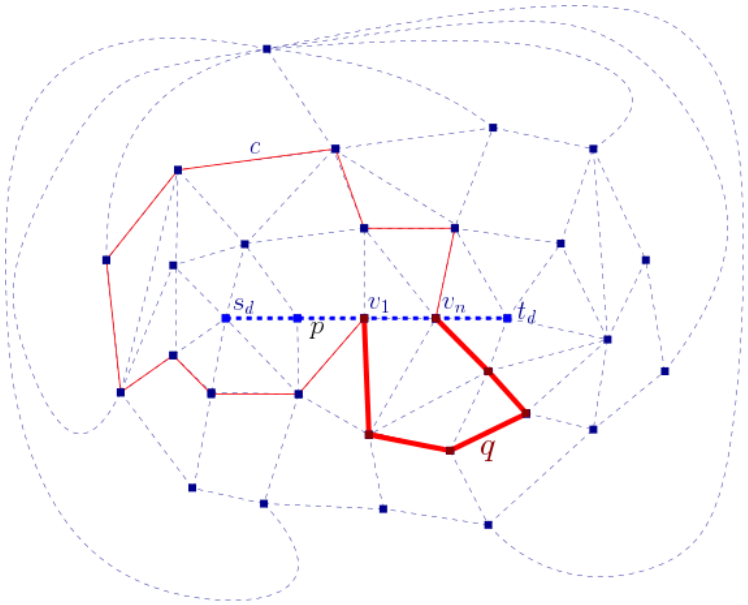


Figure 4.4: An example cycle c with a subpath of c, $c(v_1, v_n)$, entirely below p

Lemma 4.2. *A minimum (s_d,t_d)-separating cycle c cannot contain any path entirely below a right-most shortest (s_d,t_d)-path p.*

Proof. Let c be a minimum (s_d,t_d)-separating cycle and p a right-most shortest (s_d,t_d)-path in G'_d. Suppose q is a path in c from v_1 to v_n that is entirely below p, as in Figure 4.4. Since the subpaths of p, $p(s_d, v_1)$ and $p(v_n$,$t_d)$, in combination with q, create a (s_d,t_d)-path that uses the same first edge as p and is more right than p, q must be longer than the segment $p(v_1, v_n)$, otherwise p would not be a right-most shortest path. Thus, there exists a shorter separating cycle than c, if q is replaced by $p(v_1, v_n)$, a contradiction. □

Lemma 4.3. *A minimum separating cycle c cannot contain two edges incident to a right-most shortest (s_d,t_d)-path from below.*

Proof. Let c be a minimum (s_d,t_d)-separating cycle and p a right-most shortest (s_d,t_d)-path in G'_d. Let u_p, v_p be vertices in p such that u_p comes before v_p in p. Suppose c contains edges (u_b,u_p) and (v_b,v_p) such that both edges are incident from below.

Case 1. *There is a path $q = u_p, u_b, \ldots, v_b, v_p$ in c; u_p, v_p may be the same vertex.*

By Lemma 4.2, there must be at least one internal vertex in q that is contained within p otherwise q would be entirely below p. Further, there must be some subpath of q, $q(u_p, x_p)$, such that (x_a, x_p) is an edge incident to p from above and no internal vertices of $q(u_p, x_p)$ are contained in p. Then the subpath of p, $p(x_p, u_p)$, and the path $q(u_p, x_p)$ together would create a cycle separating s_d and t_d.

The cycle $p(x_p, u_p) + q(u_p, x_p)$ must be shorter than the cycle c.

Case 1.1. *x_p comes before u_p in p.*

Consider the paths $p(u_p, v_p)$, and $q(x_p, v_p)$, as well as the path r, a subset of c from u_p to v_p such that its internal vertices are distinct from those in q. See Figure 4.5. Then $c = q + r$ and p contains a path $p(x_p, u_p) + p(u_p, v_p)$. Since p is a right-most shortest (s_d,t_d)-path, $|r| \geq |p(u_p, v_p)|$, and $|q(x_p, v_p)| \geq$

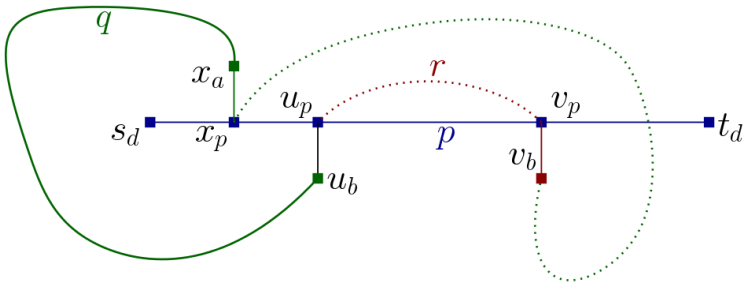


Figure 4.5: A basic example of the paths p,q, and r as described in Lemma 4.3, Case 1.1. Dotted lines indicate that the path may have vertices in p not pictured

$|p(x_p, u_p) + p(u_p, v_p)|$. Then,

$$\begin{aligned}
|c| &= |q| + |r| \\
&= |q(u_p, x_p)| + |q(x_p, v_p)| + |r| \\
&\geq |q(u_p, x_p)| + |p(x_p, u_p)| + |p(u_p, v_p)| + |p(u_p, v_p)| \\
&> |q(u_p, x_p)| + |p(x_p, u_p)|
\end{aligned}$$

Thus the new cycle, $p(x_p, u_p) + q(u_p, x_p)$ is shorter than c. This contradicts the initial proposition that c is a minimum separating cycle, therefore q cannot be a subpath of c.

Case 1.2. *x_p is between u_p and v_p in p.*

Consider the paths $p(x_p, v_p)$ and $q(x_p, v_p)$ as well as the path r, a subset of c from u_p to v_p such that its internal vertices are distinct from those in q. Then $c = q + r$ and p contains a path $p(u_p, x_p) + p(x_p, v_p)$. See Figure **??**. Since p is a right-most shortest (s_d,t_d)-path, then $|r| \geq |p(u_p, x_p)| + |p(x_p, v_p)|$ and $|q(x_p, v_p)| \geq |p(x_p, v_p)|$.

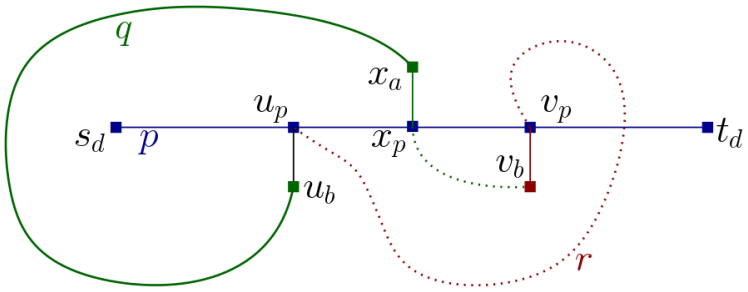


Figure 4.6: A basic example of the paths p,q, and r as described in Lemma 4.3, Case 1.2. Dotted lines indicate that the path may have vertices in p not pictured

Then,

$$\begin{aligned}
|c| &= |q| + |r| \\
&= |q(u_p, x_p)| + |q(x_p, v_p)| + |r| \\
&\geq |q(u_p, x_p)| + |p(x_p, v_p)| + |p(u_p, x_p)| + |p(x_p, v_p)| \\
&> |q(u_p, x_p)| + |p(x_p, u_p)|
\end{aligned}$$

Thus the new cycle, $p(x_p, u_p) + q(u_p, x_p)$ is shorter than c. This contradicts the initial proposition that c is a minimum separating cycle, therefore q cannot be a subpath of c.

Case 1.3. *x_p is between v_p and t_d in p.*

Consider the paths $q(v_p, x_p)$, $p(u_p, v_p)$ and $p(v_p, x_p)$ as well as the path r, a subset of c from u_p to v_p such that its internal vertices are distinct from those in q. In this case, $c = q + r$, and $p(x_p, u_p) = p(u_p, v_p) + p(v_p, x_p)$. See Figure 1.3. Since p is a shortest (s_d,t_d)-path, $r \geq p(u_p, v_p)$ and $q(v_p, x_p) \geq$

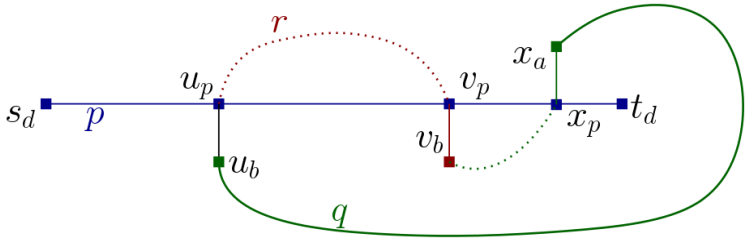


Figure 4.7: A basic example of the paths p,q, and r as described in Lemma 4.3, Case 1.3. Dotted lines indicate that the path may have vertices in p not pictured

$p(v_p, x_p)$. However, since p is a right-most shortest (s_d,t_d)-path, and $q(v_p, x_p)$ is more right than $p(v_p, x_p)$, $q(v_p, x_p)$ must be strictly longer than $p(v_p, x_p)$. Then,

$$\begin{aligned}
|c| &= |q| + |r| \\
&= |q(u_p, x_p)| + |q(x_p, v_p)| + |r| \\
&\geq |q(u_p, x_p)| + |q(x_p, v_p)| + |p(u_p, v_p)| \\
&> |q(u_p, x_p)| + |p(x_p, v_p)| + |p(u_p, v_p)| = |q(u_p, x_p)| + |p(u_p, x_p)|
\end{aligned}$$

Thus the new cycle, $p(x_p, u_p) + q(u_p, x_p)$ is shorter than c. This contradicts the initial proposition that c is a minimum separating cycle, therefore q cannot be a subpath of c.

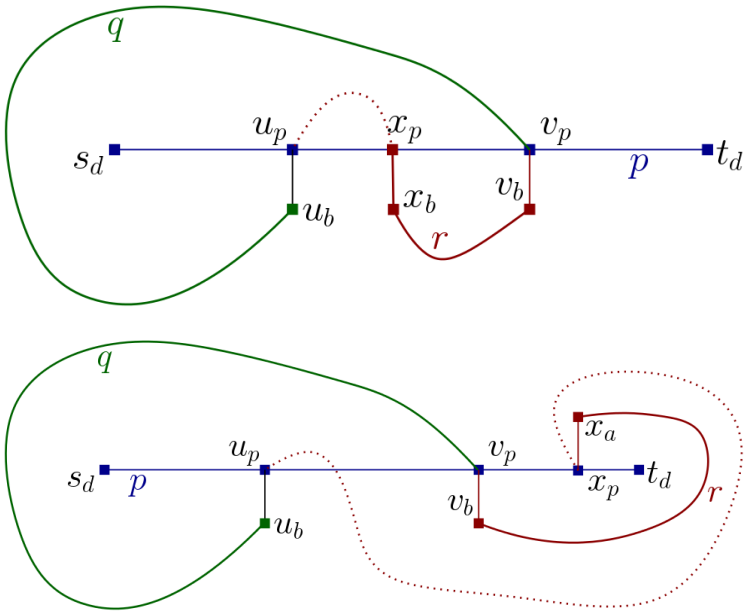


Figure 4.8: Examples of the paths p,q, and r as described in Lemma 4.3, Case 2. Dotted lines indicate that the path may have vertices in p not pictured

Case 2. *There is a path* $q = u_p, u_b, \ldots, v_p, v_b$ *in* c

Then there must also be a path $r = v_p, v_b, \ldots, u_p$ in the cycle c. Consider that the path q will create a boundary, separating the vertex v_b from any vertices in p from s_d to v_p from above, shown in Figure 4.8. Then the path r can either continue on to include a vertex in p from below starting from u_p to the end of the path p, or r can first approach the path p from above, between v_p to the end of the path.

Case 2.1. r *contains a path* $v_p, v_b, \ldots, x_b, x_p$ *such that* $x_p \in p$ *and* (x_b, x_p) *is adjacent to* p *from below with no internal vertices in* p.

In this case r cannot be a part of a minimum separating cycle, namely

c, given that it contains a path entirely below p, as shown in Lemma 4.2.

Case 2.2. *r contains a path $v_p, v_b, \ldots, x_a, x_p$ such that $x_p \in p$ and (x_a, x_p) is adjacent to p from above with no internal vertices in p.*

For this to be the case, x_p must be between v_p and t_d on the path p. However, the path r is to the right of the path p, as the edge (v_p, v_b) is incident to the path from below. Then the path $v_p, v_b, \ldots, x_a, x_p \in r$ cannot be shorter than or equal to the path $v_p, \ldots, x_p \in p$. Then replacing that section of r with the path $v_p, \ldots, x_p \in p$ would create a shorter separating cycle than c, which contradicts the beginning conditions.

Therefore, in any case, no minimum separating cycles may contain two or more edges incident to p from below.

□

The lemmas above demonstrate that the behavior of a minimum (s_d,t_d)-separating cycle with a right-most shortest (s_d,t_d)-path can be quite restricted. The following theorems concisely describe the conditions imposed on a minimum (s_d,t_d)-separating cycle.

Theorem 4.4. *A minimum separating cycle c must intersect a right-most shortest path p exactly once.*

Proof. Let c be a minimum separating cycle between s_d and t_d, and p be a right-most shortest (s_d,t_d)-path in G'_d. By Lemma 4.1, its shown that c must intersect p at least once. Lemma 4.3 shows that a minimum separating cycle c

cannot be incident to p from below more than once, and thus cannot intersect p more than once. Therefore, c will intersect p exactly once. □

Theorem 4.5. *Any cycle c that intersects a (s_d,t_d)-path exactly once and does not contain s_d or t_d is a separating cycle.*

Proof. Consider a cycle c that intersects a (s_d,t_d)-path, p, exactly once such that c does not contain s_d or t_d. Then the vertices in p that are before the intersection are contained within one side of the cycle, and the vertices in p after the intersection are on the other side. Thus s_d and t_d are separated by c, and c is an (s_d,t_d)-separating cycle. □

Chapter 5

Counting Minimum (s, t)-cuts

This chapter will discuss the counting of minimum separating cycles in the dual of a graph and by extension the counting of minimum cuts in said graph. As Chapter 3 has concluded, the number of minimum (s, t)-cuts in a graph G is equal to the number of minimum separating cycles in G'_d. Chapter 4 concludes that every minimum separating cycle will have exactly one edge incident to a right-most shortest (s_d,t_d)-path from below. The method of counting minimum separating cycles will break down simply to counting a series of minimum paths between vertices on the right-most shortest (s_d,t_d)-path of the dual.

Let G be a planar undirected graph, and G'_d be the dual of G', as constructed in Chapter 3. Let p be the right-most shortest (s_d,t_d) path in G'_d. For simplicity, consider p as represented horizontally, with s_d as the leftmost vertex in p and t_d as the right-most, with the vertices $v_1, v_2, \ldots, v_n$ between them, sequentially. To simplify the counting method, the path p will be split into two sets of vertices with the edges connected to each vertex of the path split between them, as shown in Figure 5.1. For each of the vertices v_i in p, not including s_d and t_d, create a vertex u_i, such that u_i is incident to the

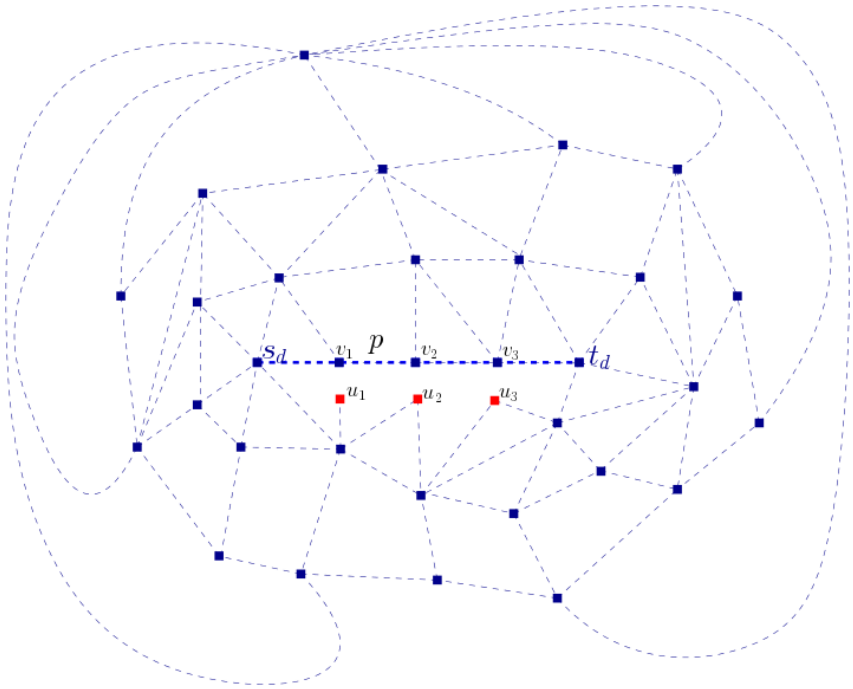


Figure 5.1: The transformation of G'_d from Figure 4.1 to $\widehat{G}$. p and the vertices new to $\widehat{G}$ are highlighted

edges of v_i that are below the path p. Remove all of these edges from v_i, i.e. ensure v_i has only edges above p and those edges along p. Let $\widehat{G}$ be the graph containing the altered path p, and the set of new vertices and edges.

A (v_i, u_i) path in $\widehat{G}$ is equivalent to a separating cycle in G'_d, such that the cycle intersects the path at v_i. Creating the graph $\widehat{G}$ allows for greater control in filtering out cycles in G'_d that could never be minimum separating cycles, such as cycles that touch below a right-most shortest path more than once, or that never intersect the path at all. The minimum separating cycles of G'_d can be counted by counting the minimum paths between each pair of vertices (v_i, u_j) in $\widehat{G}$. The vertices s_d and t_d need not be considered in any of the pairings, as all of the edges incident to the two vertices have a significantly large weight of α and as such will not contribute to a minimum separating cycle.

Algorithm 1 Counting Minimum (s,t)-cuts

```
G ← undirected planar graph with positive weights
Ĝ ← constructed from G in Chapter 5
count_total ← 0
weight_min ← α
for all v_i, u_i ∈ Ĝ do
    count_i ← number of shortest (v_i, u_i)-paths in Ĝ
    weight_i ← weight of the shortest(v_i, u_i)-paths in Ĝ
    if weight_i < weight_min then
        weight_min ← weight_i
        count_total ← count_i
    else
        if weight_i = weight_min then
            count_total ← count_total + count_i
        end if
    end if
end for
```

Algorithm 1 describes the method for counting the minimum (s_d,t_d)-separating cycles of G'_d, and thus for counting the minimum (s,t)-cuts in G.

The total number of pairs of vertices considered in this process is d, where d is the number of vertices in the path p. The run-time of the counting process is $\mathcal{O}(dR)$, where R is the run-time of the path counting algorithm used.

Any algorithm for finding or counting minimum paths can be effectively used, such as an implementation of Dijkstra's algorithm that counts the number of minimum paths found. Dijkstra's algorithm will find the shortest distance between any two vertices by applying the concept that a u,v-segment of a shortest path will need to be a shortest u,v-segment [Dij59]. A simple implementation will process each of the shortest paths between two

vertices in $\mathcal{O}(|V|^2)$. Using a fibonacci heap, the running time can be reduced to $\mathcal{O}(|E| + |V| \log(|V|))$ [CLRS09]. With this implementation, the number of minimum cuts can be counted in $\mathcal{O}(d(|E| + |V| \log(|V|)))$.

Chapter 6

Conclusion

The proposed algorithm has a polynomial runtime; when using even a simple algorithm to find shortest paths, such as Dijkstra's, the worst case run-time would be $\mathcal{O}(d(|E| + |V| \log(|V|)))$, where d is the number of vertices in a shortest right-most path in the dual of the graph. This can be further simplified to $\mathcal{O}(d|V| \log(|V|))$ in simple planar graphs, as the number of edges in a simple planar graph is bounded by a constant multiple of $|V|$ [Eul58]. While the algorithm is not restricted to simple graphs, and allows for counting minimum cuts in multigraphs, any multigraph in this context can be converted into a simple graph. Consider that any looping edge (u, u) will not contribute to a minimum cut weight, as u cannot be in both the cut set, and outside of the cut set, and so (u, u) can be preemptively removed from the graph G. Any duplicate edges e and f with endpoints (u, v) in the planar graph G, can be combined into a single edge x such that the weight of x is $w(e) + w(f)$, as any cutset that includes u but not v would take the weight of both edges e and f. Finally, d is bounded by the number of vertices in G, simplifying the algorithms running time to $\mathcal{O}(|V|^2 \lg(|V|))$

Although it runs in polynomial time, this algorithm does not run faster

than previous counting algorithms; [BF12] had a runtime of $|\mathcal{V}| + |\mathcal{V}| \log(|\mathcal{V}|)$ where d is the length of a shortest (s, t)-path in the original graph. However, this algorithm is simpler, both computationally and conceptually. This algorithm significantly reduces the number of processing steps performed on a graph before any counting can be done. With visual examples, it becomes clear that cuts in a planar graph are related to cycles in its dual.

While the algorithm has only been outlined for the case of undirected planar graphs, the same technique may be applicable to directed planar graphs with some further research. Additional consideration must be made with regards to cuts in directed graphs, as only the weights of edges directed out of the cut set S are included in the total weight of the cut. This suggests that the dual of the directed graph used in this case must have additional 0 weight edges directed opposite to each of the already existing dual edges, to allow for all of the cuts to be counted. The nature of directed graphs may also require additional steps in the counting process, perhaps counting not just paths from v_i to u_i in $\widehat{G}$ as described by Chapter 5, but also the paths from u_i to v_i. The property that an edge can only be incident to below the shortest right-most path once will also have to be revisited to determine any difference between the edges directed into the path and edges directed out of the path.

This algorithm most likely is not applicable to planar graphs with negative edge weights, or edge weights with a value of 0, as Lemma 4.3 would no longer hold.

Additionally, future work may be considered to apply a similar concept

to finding minimum cuts in a grid-like graph with 8-point connectivity, to expand its application. To consider such an application, a concept of a dual-like graph for 8-point connectivity must first be established.

Appendix

Glossary of Symbols

General Symbols

G	a graph
V	the set of vertices of a graph G
v	commonly used to indicate a single vertex
u	commonly used to indicate a single vertex
E	the set of edges of a graph G
e	commonly used to indicate a single edge
w	the set of positive weights of a graph G
s	the source vertex
t	the sink vertex
α	a sufficiently large value/weight
f	a face of a planar graph
p	a path
$p(u, v)$	a subset of a path p from vertex u to vertex v
$\|p\|$	the length of a path p, i.e. the sum of the edge weights of p
$\|V\|$	the number of vertices in a vertex set V

Symbol Modifiers to be used on a graph (i.e. G) or its components (e.g., v or s)

G_d	the dual of graph G
G'	a modified version of G
	In 3.1, G' is a version of G with faces replacing the s and t vertices
$\widehat{G}$	A graph derived from G

Index

Bibliography

[Arc97] D. Archdeacon. Topological graph theory - a survey. 115, 09 1997.

[BF12] I. Bezáková and A. J. Friedlander. Counting and sampling minimum (s,t)-cuts in weighted planar graphs in polynomial time. *Theoretical Computer Science*, 417:2–11, 2012.

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept 2004.

[BP83] M. O. Ball and J. S. Provan. Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13:253–278, 1983.

[BV06] Y. Boykov and O. Veksler. *Graph Cuts in Vision and Graphics: Theories and Applications*, pages 79–96. Springer US, Boston, MA, 2006.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[Edm60] J. Edmonds. A combinatorial representation for poyhedral surfaces (abstract). *"Notices of the American Mathematical Society"*, 7:646, 1960.

[Eul58] L. Euler. Elementa doctrinae solidorum. *Novi Commentarii academiae scientiarum Petropolitanae*, 4:109–140, 1758.

[FF56] L. R. Ford, Jr. and D. R. Fulkerson. Canadian journal of mathematics. 8:399–404, 1956.

[Hef91] L. Heffter. Ueber das problem der nachbargebiete. *Mathematische Annalen*, 38:477–508, 1891.

[UHCC14] B. Uzkent, M. J. Hoffman, E. Cherry, and N. Cahill. 3-d mri cardiac segmentation using graph cuts. In *Image and Signal Processing Workshop (WNYISPW), 2014 IEEE Western New York*, pages 47–51, Nov 2014.

[Wes01] D. B. West. *Introduction to Graph Theory.* Pearson, 2 edition, 2001.

[You63] J. W. T. Youngs. Minimal imbeddings and the genus of a graph. *Journal of Mathematics and Mechanics*, 12(2):303–315, 1963.