SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

TECHNICAL REPORT NO. 833

December 1988
(revised April 1989)

ON MAXIMUM FLOWS IN POLYHEDRAL DOMAINS

By

Joseph S. B. Mitchell

To Appear: *Journal of Computer and System Sciences*
A preliminary version of this paper appears in the *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry*, Champaign-Urbana, IL, June 6-8, 1988, pp. 341-351.

ON MAXIMUM FLOWS IN POLYHEDRAL DOMAINS†

Joseph S. B. Mitchell‡

School of Operations Research and Industrial Engineering
Cornell University
Ithaca, NY 14853

Abstract

We introduce a new class of problems concerned with the computation of maximum flows through two-dimensional polyhedral domains. Given a polyhedral space (e.g., a simple polygon with holes), we want to find the maximum "flow" from a source edge to a sink edge. Flow is defined to be a divergence-free vector field on the interior of the domain, and capacity constraints are specified by giving the maximum magnitude of the flow vector at any point. The problem is the natural extension to the continuous domain of the discrete problem of finding maximum flows through a capacitated network. For this problem, Strang proved that max flow equals min cut; we address the problem of *constructing* min cuts and max flows. We give polynomial-time algorithms for maximum flow from a source edge to a sink edge through a simple polygon with uniform capacity constraint (with or without holes), maximum flow through a simple polygon from many sources to many sinks, and maximum flow through weighted polygonal regions. Central to our methodology is the intimate connection between the max-flow problem and its dual, the min-cut problem. We show how the continuous Dijkstra paradigm of solving shortest paths problems corresponds to a continuous version of the uppermost path algorithm for computation of maximum flow in a planar network.

Key Words: maximum flow, minimum cut, shortest paths, continuous Dijkstra, shortest path maps, Voronoi diagrams, computational geometry, duality

---

† A preliminary version of this paper appeared in the Proceedings of the Fourth Annual ACM Symposium on Computational Geometry, Urbana-Champaign, June 6-8, 1988.

‡ Partially supported by NSF Grants IRI-8710858 and ECSE-8857642 and by a grant from Hughes Research Laboratories.

1. Introduction

Computing maximum flows in a capacitated network has been a very important problem in combinatorial optimization, and many efficient polynomial-time algorithms exist [EK, Ga, GT, IS, etc.]. It is natural to ask how to generalize the maximum flow problem to a continuous (two-dimensional) domain. When generalized to the continuum, the max flow problem becomes that of computing an optimal two-dimensional *vector field* (in contrast to the problem of computing an optimal flow in a network, which assigns a single non-negative flow value to each arc). Gomory and Hu have addressed the problem of flows in continua from the point of view of approximating them with a discrete network (see [Hu]). Strang [St] and Iri [Ir] have shown duality results that max flow equals min cut for flows in continua. Since the min cut problem is, as we will see, an instance of a geometric shortest path problem, Papadimitriou [Pa] has recently suggested that examining algorithms to compute maximum flows in continua may suggest new methods of attacking the shortest path problem in weighted regions (see [MP]). It is the purpose of this paper to examine the complexity of computing maximum flows in the continuum and to analyze the duality relationship between the maximum flow problem and the shortest path problem.

Several applications motivate our problem. First, consider the problem of moving fleets of parts through a domain. Parts can be moved at a certain speed, and they must avoid holes (obstacles) in the domain. If the parts are small, then the problem can be approximated by a continuum of "fluid" which must flow through the domain. The maximum flow gives us the rate at which parts can be moved through the domain. There are applications of this problem to routing flows of vehicles through factory floors and to routing fleets of ships through enemy radar stations or to routing planes through mountain passes.

Another application can be seen in routing wires on a circuit board among components (obstacles). The value of the maximum flow is an approximation to the number of wires that can be routed between terminals. Given a maximum flow field, we can route the wires along the "streamlines" of the vector field. Still another application is in the automatic determination of the amount of flow that can be sent through a part that is designed on a CAD system. For instance, we may be designing a particular type of valve or pipe fitting, and we need to know how much fluid will be able to flow through the part.

2. Notation and Problem Formulation

We begin with a precise statement of our problem, following the notation of Strang [St].

We are given a compact connected *domain* Ω, which we assume to be a *polyhedral* domain in two dimensions, described by a total of n_Ω vertices. Specifically, Ω is a simple polygon with holes, represented as a list of vertices comprising its outer boundary (say, in clockwise order), together with a list of h holes, each given by an ordered list of vertices.

Let $\Gamma = \partial\Omega$ denote the boundary of Ω. There is an open subset, $\Gamma_s \subset \Gamma$, of the boundary through which flow can enter Ω, and there is an open subset, $\Gamma_t \subset \Gamma$, disjoint from Γ_s, through which flow can exit Ω. We assume that Γ_s and Γ_t each contain a finite number of connected components. Thus, each connected component will consist of either a polygonal path or a polygonal cycle. We assume without loss of generality that the endpoints of any polygonal path component of Γ_i are vertices of Ω. Γ_s (resp., Γ_t) is naturally called the set of *sources* (resp., *sinks*). We abuse notation slightly by referring to the connected components of Γ_s (resp., Γ_t) as the "sources" (resp., the "sinks").

Let $\Gamma_w = \Gamma \setminus (\Gamma_s \cup \Gamma_t)$ be the portion of the boundary of Ω which is neither source nor sink. We think of Γ_w as the "walls" of the domain which cannot be penetrated by flow. Since we are assuming that Ω is bounded, its complement, Ω^c will have precisely one unbounded connected component; we let $\Gamma_\infty \subseteq \Gamma$ be the boundary of this component.

We are given a flow capacity constraint function, $c : \Omega \to \Re^+ \cup \{+\infty\}$. We require that the magnitude of the flow vector at point $p \in \Omega$ is bounded by $c(p) \geq 0$. We consider only the case in which c is piecewise-constant over a given polygonal subdivision of Ω of size n_c. We then define $n = n_\Omega + n_c$ to be the total combinatorial complexity of the problem statement, including the number of vertices in the descriptions of Ω (n_Ω) and the subdivision (n_c). In most of our discussion, c will be constant over Ω, in which case we can take $c = 1$ without loss of generality. (An alternative specification of our problem would have Ω always simply connected (no holes), since we could "simulate" a hole by making $c = 0$ over the polygon that defines the hole.)

The *max flow* problem is to compute a *vector* field $\sigma : \Omega \to \Re^2$, a *flow*, that solves the following program:

$$\begin{aligned} \textit{Maximize} \quad & \mu = \int_{\Gamma_t} \sigma \cdot \mathbf{n}\, ds \\ \textit{subject to} \quad & |\sigma| \leq c \quad \text{in } \Omega, \\ & \sigma \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_w, \\ & \text{div } \sigma = 0 \quad \text{in } \Omega. \end{aligned}$$

Here, $\mathbf{n}$ is the unit normal vector to Γ which is outward pointing, and μ is said to be the *value* of flow σ.

We are assuming that there are no source or sink points internal to Ω. Hence, the statement of flow conservation is that the field σ is divergence-free. Note that flow conservation implies that for any feasible σ, $\mu = \int_{\Gamma_t} \sigma \cdot \mathbf{n}\, ds = -\int_{\Gamma_s} \sigma \cdot \mathbf{n}\, ds$; i.e., "flow in equals flow out".

The flow across any simple curve, $\mathcal{C}$, is obtained by integrating $\int_{\mathcal{C}} \sigma \cdot \mathbf{n}\, ds$ along the curve (where $\mathbf{n}$ is the unit normal to $\mathcal{C}$). We will refer to any σ^* that solves the above program as a *maximum flow field*, or a *max flow*, for short. We let μ^* denote the value of an optimal flow σ^*. Note that in general there may be many possible maximum flow fields σ^*.

The specification of our problem here is slightly different from that of Strang [St]. In addition to the bounds on the magnitude of σ and the conservation of flow, Strang requires that $\sigma \cdot \mathbf{n} = \lambda f$ on Γ, and he maximizes λ. The real-valued function f describes the intensity of sources/sinks on the boundary of Ω. Our formulation can be seen to be equivalent to a special case of Strang's.

In order to illustrate some of these concepts, we refer to Figure 2.1. We show a domain Ω consisting of a simple polygon with two holes. The set of sources, Γ_s, consists of two connected components, one having a single line segment and the other having two segments. The set of sinks consists of a single edge on one of the polygonal holes. We assume that the capacity function $c(p)$ is everywhere constant (without loss of generality, it equals 1). We indicate a maximum flow field by showing a set of regions with small "arrows" to denote the vector σ. Everywhere outside of these regions the flow field is zero. The value μ^* of the maximum flow is $\mu_1 + \mu_2$.

A *cut* is defined as a (not necessarily connected) subset S of Ω such that S contains all sources and no sinks (i.e., $\Gamma_s \subseteq S$ and $S \cap \Gamma_t = \emptyset$). We will denote by γ that portion of the boundary (∂S) of S which "separates" Γ_s from Γ_t. We make the definition of γ more precise as follows: $\gamma = [\partial S \setminus \Gamma_w] \setminus [\Gamma_s \cap \partial(int(S))]$, where $int(S)$ denotes the interior of set S, and $\partial(X)$ denotes the boundary of set X. The rationale for this definition is that we want to measure only that portion of the boundary of S which cuts through the inside of Ω (hence, we subtract the set Γ_w), but it may be that the set S contains portions of Γ_s without containing any neighboring points of $int(\Omega)$. We need to subtract those portions of the boundary of S which do have interior points of S nearby; hence, we subtract the set $\Gamma_s \cap \partial(int(S))$. For example, if $S = \Gamma_s$, we get $\gamma = \Gamma_s$, while if $S = \Omega \setminus \Gamma_t$, we get $\gamma = \Gamma_t$.

Remark: An equivalent way to define the set γ would be to append to S a set K_s of

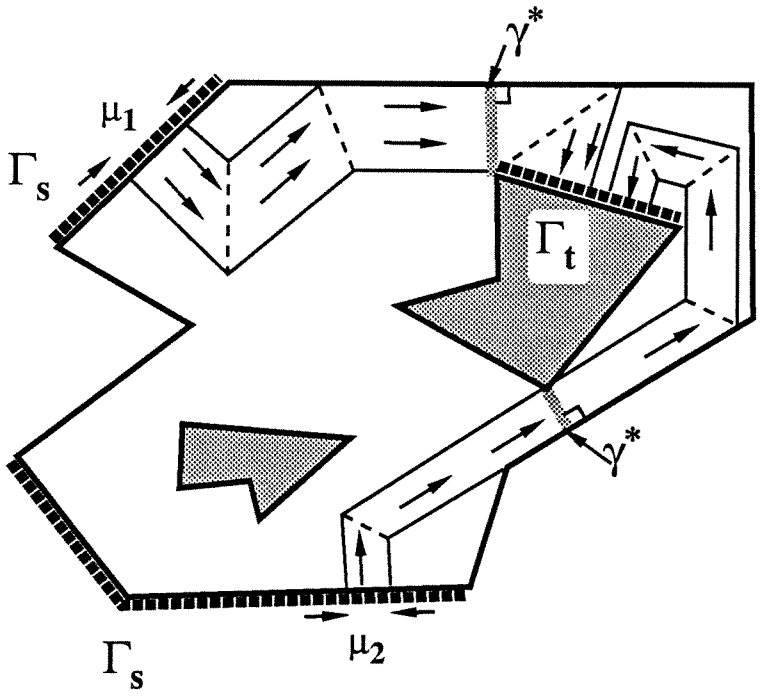

Figure 2.1. Example of a maximum flow field.

"sources" just outside Ω which are adjacent to the sources Γ_s (think of K_s as very thin little strips along side each portion of Γ_s), take the boundary of the set $S \cup K_s$, intersect this with Ω, and then subtract Γ_w. Then, $\gamma = [\partial(S \cup K_s) \setminus \Gamma_w] \cap \Omega$.

Frequently, we will speak of γ (rather than S) as the "cut", since it is a set of paths which separate Ω into two or more connected components, with sources and sinks being in different components. The capacity, $C(S)$, of a cut S is the line integral $\int_\gamma c \, ds$. Any cut γ^* which minimizes $\int_\gamma c \, ds$ over all cuts γ is called a *minimum capacity cut*, or a *min cut*, for short. For the example in Figure 2.1, the min cut is shown as a heavy gray line.

If Γ_s and Γ_t each consist of only one connected component, then γ^* will be either a single simple path or a single simple cycle. Specifically, if Γ_s and Γ_t lie in the same connected component of Γ, then γ^* will be a simple path; otherwise, γ^* will be a cycle. (This is true since, if S contains more than one connected component, then $C(S)$ can be reduced by deleting all of those components of S which do not border Γ_s. See Lemma 5.1.) If $c = 1$, then the capacity of a cut is simply the length of γ, while for arbitrary piecewise-constant functions c, the capacity is the "weighted length" of γ. This suggests that minimum cuts

are intimately related to shortest paths through weighted regions (see [MP]).

In fact, the max flow and min cut problems are "dual" to one another. It is not hard to see that "weak duality" must hold, namely, that the value of any feasible flow field cannot exceed the capacity of any cut. (This is just a statement of the classical divergence theorem.) Strang [St] has proved a max-flow/min-cut theorem, which is a statement of "strong duality": the value of the maximum flow, μ^*, is equal to the capacity, $C(S^*)$, of a minimum capacity cut, S^*. Strang, however, offers no algorithm for computing either the min cut or the max flow. Iri [Ir] has worked on similar results in higher dimensions.

3. Summary of Results

We are interested in the computational complexity of computing the *value*, μ^*, of a max flow (which is equivalent to computing the length, $|\gamma^*|$, of a min cut) and of *constructing* max flow fields σ. We give polynomial-time algorithms for both of these problems, under a variety of different assumptions on Ω, Γ_s, Γ_t, and c. Our results are summarized below.

1). We consider the problem of computing max flow through a uniformly capacitated ($c = 1$) simple polygon, from a single source edge to a single sink edge. This problem is solved in time $O(n \log n)$. We generalize to the case of many source and many sink edges within the same time bound.

2). We consider the generalization to the case in which Ω is a (multiply-connected) simple polygon with h holes, and there are many sources and sinks on the outer boundary Γ_∞. We prove a lower bound of $\Omega(n + h \log h)$ on computing the max flow field. We define a new type of Voronoi diagram based on the $0/1/\infty$ weighted region path metric, and show how it can be used to map the max flow problem in Ω into a simple network flow problem. By a "continuous-Dijkstra" type method, we obtain an $O(nh+n \log n)$ algorithm for computing the necessary Voronoi diagram, and thereby to solve the max flow problem within the same time bound. Our algorithm can be interpreted as a continuous version of the uppermost path algorithm for max flow in planar graphs.

3). We solve the general case of uniformly capacitated domains in which the source and sink sets can be arbitrary (with many source and sink edges lying on boundaries of many different holes).

4). We consider the more general case in which the capacity constraint function c is piecewise constant. We relate the maximum flow problem to the shortest path problem in weighted regions [MP], and thereby get a polynomial-time algorithm for max flows.

4. Basic Facts About Flow Fields

A divergence-free vector field is one that is "conservative" in that for any small subregion of $int(\Omega)$, the net flow in or out is zero. If σ is piecewise-constant over a subdivision, then it is clearly divergence-free within any region over which it is constant. In order to verify that the field is conservative, then, we need only check that flow is conserved at the boundaries of the subdivision. Flow will be conserved at a boundary if and only if the (constant) vectors in the regions on either side have the same dot product with a unit vector normal to the boundary. In the case that the magnitudes of the flow vectors are the same on either side of the boundary, this implies that the boundary forms an angular bisector between the flow vectors on either side. Refer to Figure 4.1.

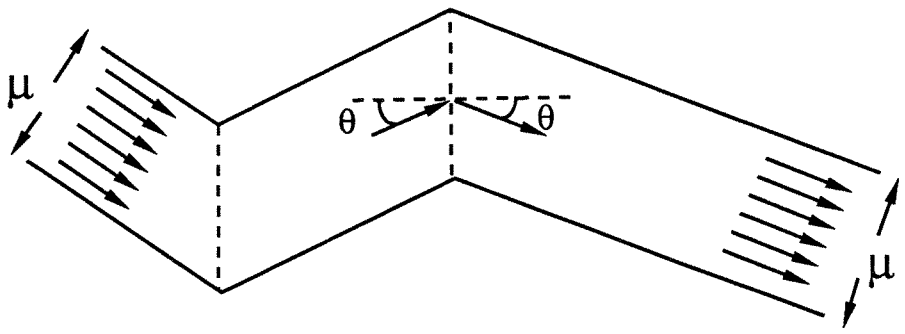

Figure 4.1. A conservative piecewise-constant flow field.

Another important case of a conservative vector field is that in which the vector σ at any point has some constant magnitude and is oriented to be tangent to a circle centered on some fixed origin. Specifically, if $\hat{\mathbf{r}}$ and $\hat{\theta}$ are the unit vectors in a polar coordinate frame, and C is any constant, then the field $\sigma = C\hat{\theta}$ is conservative (as can be checked by taking its divergence). We refer to such fields as *radial* about the origin. Refer to Figure 4.2. Note that such fields are not piecewise-constant, even though the vector magnitudes are, since the direction of the vector σ varies.

It is possible to compose more complex conservative vector fields by concatenating trapezoidal and circular sector regions, as shown in Figure 4.3. The field is defined to be zero everywhere outside of the trapezoids and sectors. Within each trapezoid the field is constant, with magnitude C and orientation parallel to the parallel sides of the trapezoid. Within each sector, the field is radial, with magnitude C. The crucial property

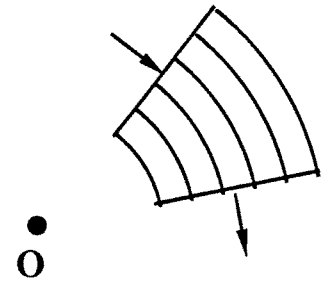

Figure 4.2. A radial flow field.

of the subdivision which makes the resulting field conservative is the manner in which the trapezoids and sectors are put together, which must be done in such a way that flow is conserved across boundaries.

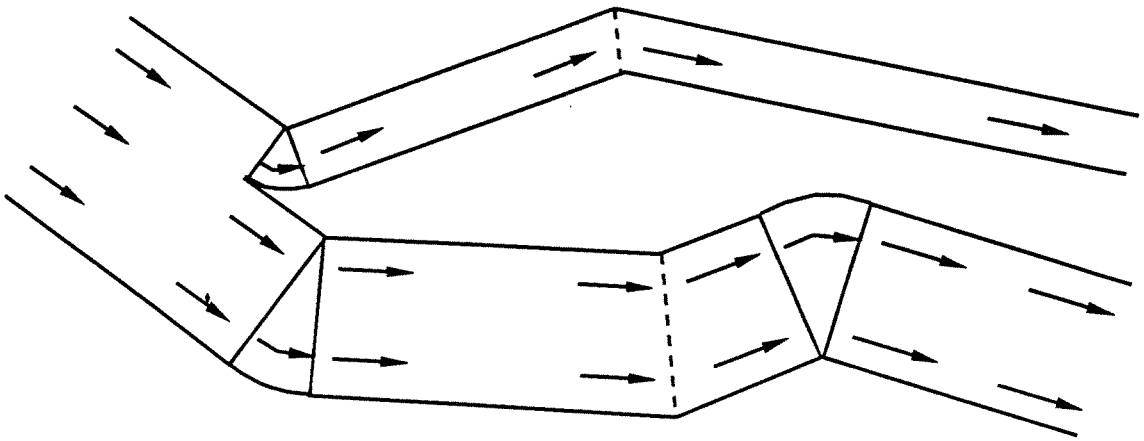

Figure 4.3. "Plumbing": combining constant and radial flows.

Figure 4.3 suggests the metaphor of referring to the construction of these flow fields as "plumbing". Indeed, the trapezoids can be thought of as sections of "pipe" or "tubes", while the sectors serve as "elbows". We can depict a flow field by showing the "streamlines", which are defined by the paths that a particle would take if released within a force field given by σ.

In the problems addressed in this paper, the capacity function is piecewise-constant (usually it is the constant 1). It is not hard to see that it suffices to consider only those flow fields whose flow vector is everywhere at its upper bound c. Also, while radial flows

arise naturally in the continuous version of the uppermost path algorithm (Section 7), we will see constructively that it suffices to consider only piecewise-constant flow fields (i.e., that radial flows can be replaced by "straight" ones).

As far as representing a piecewise-constant maximum flow field, we can simply use any standard data structure (e.g., [GS]) for representing a planar subdivision, with each cell in the subdivision labelled by the value of σ within the cell. We will see that there exists an optimal piecewise-constant flow field for which the size of the corresponding planar subdivision is only linear ($O(n)$).

Another simple property of optimal flow fields that was noted in [St] is the following: The maximum flow field σ^* must be normal to the minimum cut γ^*. This follows from the max flow-min cut theorem of Strang ([St], page 126).

5. Flows in a Simple Polygon

Consider first the case in which there is precisely one source edge, Γ_s, and one sink edge, Γ_t, and we are working in a uniformly capacitated ($c = 1$) simple polygon Ω (without holes). Then, the set Γ_w consists of two connected components, T ("top") and B ("bottom"). (In the clockwise ordering of Γ, T appears after Γ_t and before Γ_s.) Refer to Figure 5.1. Note that either T or B may be a single point.

The Min Cut Problem

Let us begin with a statement of a fact that holds even if Ω has holes.

Lemma 5.1 *If there is only one source and one sink and both lie on the same component of Γ, then there is a min cut set S consisting of only one connected component.*

Proof: Let S be an optimal cut. We can assume that S is closed. (If S is not closed, then note that the cut given by the closure of S is also optimal.) Then, ∂S must contain Γ_s. If S has more than one connected component, then there must be one component, say S', whose boundary includes Γ_s. Consider any other component S''. There is no reason to include S'' in the cut S, since its boundary must be disjoint from Γ_s, and it can only make the capacity of the cut larger. Thus, we could replace S by S' and still have a valid cut, whose capacity is at least as small as that of S. ∎

The lemma tells us that the problem of finding a min cut (in the special case of a single source and single sink) is simply the problem of finding a shortest path within Ω from B to T. In the case of no holes, it is easy to see that the shortest path will be a single

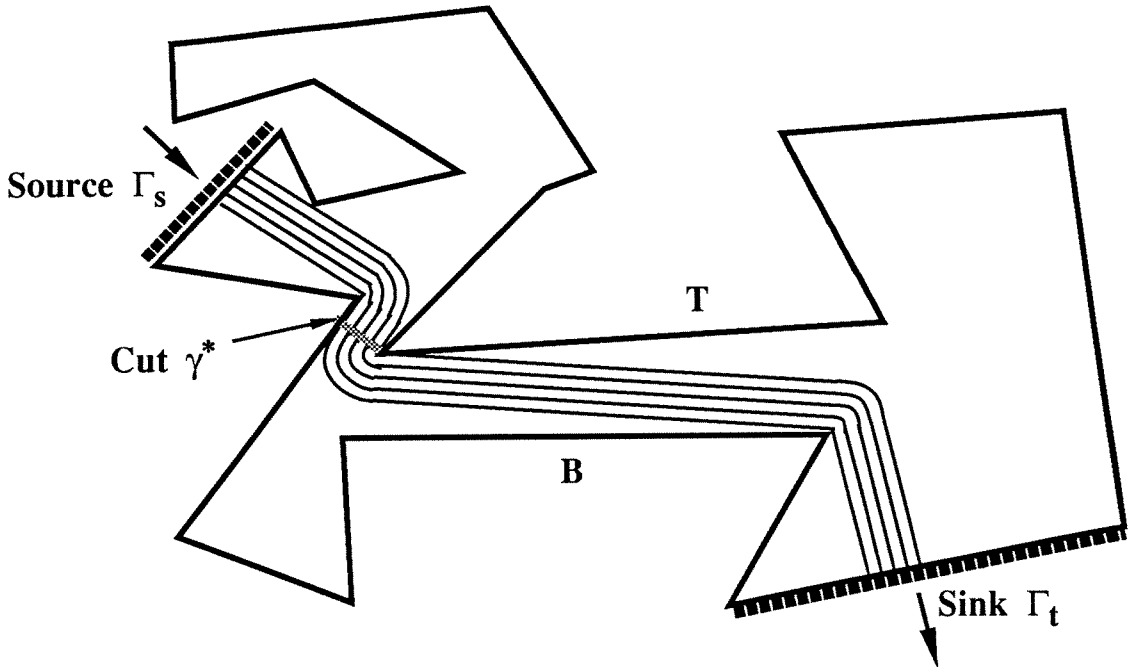

Figure 5.1. Maximum flow in a simple polygon.

line segment. (The path would not "bend" at a vertex, since each vertex is in fact part of either B or T.) In the context of the $0/1/\infty$ weighted region problem of [GMMN,Mi2], we can consider the exterior of the polygon to be an obstacle (a weight ∞ region), each of the two simple chains B and T to be zero-weight regions, the interior of Ω and Γ_s and Γ_t to be weight 1, and the start and goal points to be arbitrary points of B and T. The results of [GMMN,Mi2] tell us that the min cut can be found in time $O(n^2)$, which is obvious also from the fact that the min cut is a single segment joining B and T.

Using the Voronoi Diagram

We can do better by observing that the min cut is closely related to the Voronoi diagram of Ω. The notion of a Voronoi diagram of a set of "source" objects (e.g., vertices, segments, and curves) is a natural extension of the definition of a Voronoi diagram of a set of points. The (generalized) Voronoi diagram of a set X of (open) line segments and vertices is the locus of all points which are equidistant to two or more of the segments and vertices. The Voronoi diagram of a simple polygon (possibly with holes) is obtained by taking X to be the set of edges and vertices of the polygon and results in an *internal* and an *external* Voronoi diagram. A closely related notion for a polygon is that of the *medial axis transform*

(or *skeleton*), which is the locus of points which are the centers of circles that touch two or more distinct points of the boundary of the polygon. Thus, every point of the medial axis is also a point of the Voronoi diagram. It is known that Voronoi diagrams and medial axis transforms can be computed in time $\Theta(n \log n)$ for polygons with n vertices [Ya]. See also [Ki,Le] for more details on Voronoi diagrams and medial axis transforms for polygonal regions.

For our region Ω, we will let $VD(\Omega)$ denote the (usual) internal Voronoi diagram. In order to avoid confusion, we will refer to "sources" used in the specification of a Voronoi diagram as *Voronoi-sources*, or *V-sources* for short, to distinguish them from the notion of "sources" (Γ_s) used in the specification of the max flow problem.

For our flow problem in polygon Ω, it is not true that there must exist a Voronoi edge in $VD(\Omega)$ which will correspond to a min cut: consider a narrow rectangle whose source and sink are on the longer sides. It is true that the min cut in the simple case of a single source and single sink will correspond to some Voronoi edge in the Voronoi diagram of B and T, so we could just search it for a Voronoi edge for which the corresponding line segment joining B and T lies within Ω and is shortest among all such segments. This appraoch however does not generalize easily to the case of many sources and sinks. Instead, we will see that the min cut corresponds to an edge in an appropriately defined skeleton of a "perforated polygon with flaps", and this framework will generalize to include other cases as well.

The basic problem with the diagram $VD(\Omega)$ is that it includes the effects of "interaction" between all pairs of boundary elements, including the sources and sinks, and does not include the interactions between some pairs of boundary elements which may constrict the flow into a sink or out of a source. Our solution will be to "perforate" the boundary of the original polygon at the sources and sinks, and then to attach "flaps" (Riemann sheets) at the perforations. The purpose of the flaps is to allow us to "propagate" the Voronoi diagram through the source/sink openings into the exterior of the region, without involving other edges of the polygon. Physically, we are allowing flow to enter and leave through sources/sinks without regards to how the flow might have been able to the get through Ω^c to get to the sources or away from the sinks; the flow might have come from "out of thin air" and then disappear "into thin air". We need some definitions to make our discussion more precise.

Let Ω be a polygonal region in $\Re^2$, possibly with holes. Let $\Gamma_P \subset \Gamma$ be an (open)

subset of Ω's boundary. Γ_P is referred to as a *perforation* of Ω. Assume that Γ_P has only finitely many (i_P) connected components; in particular, assume that the endpoints of each connected component of Γ_P is one of the n_Ω vertices of Ω. We let $\Gamma_Q = \Gamma \setminus \Gamma_P$ (in the special case of a single source and single sink, $\Gamma_Q = B \cup T$). Then the *perforated polygon* $\mathcal{P}(\Omega, \Gamma_P)$ is defined to be Ω augmented by a set of Riemann sheets as follows: we attach a Riemann sheet F_π for each connected component π of Γ_P, attaching it to the base sheet $\Re^2$ (which contains the original region Ω) along π. We refer to the sheets F_π as *flaps*. Thus, for each point (x, y) in the plane, there exists a set of $i_P + 1$ copies of the point, one in the base sheet $\Re^2$, and one in each of the i_P flaps.

We define a notion of *visibility* on $\mathcal{P}(\Omega, \Gamma_P)$ as follows: If points p and q both lie in Ω on the base sheet $\Re^2$, then p and q are visible if and only if $\overline{pq} \subset \Omega$ (this is the usual notion of visibility in a region); if p and q are both on flap F_π, then they are always visible; if p and q lie on different flaps, then they are never visible; and if p lies on flap F_π and q lies in Ω on the base sheet, then p and q are visible if and only if $\overline{pq} \cap \pi \neq \emptyset$ and $\overline{qq'} \subset \Omega$, where q' is the first point where the ray from q towards p intersects π. Given the notion of visibility between any two points of $\mathcal{P}$, we can define a natural distance function $d_{\mathcal{P}}(p, q)$ to be the Euclidean distance from p to q if p and q are visible, and $+\infty$ otherwise.

Remark: The purpose of the Riemann sheets is to allow us to investigate the following multi-valued function: Let $f(p)$ be the distance from a point $p = (x, y)$ to the closest point q of some component of Γ_Q such that the ray from q towards p crosses the perforation Γ_P before exiting Ω. Then, $f(p)$ is multi-valued, since it acquires different values according to which component π of Γ_P is crossed first by the ray from q towards p.

We define the *skeleton* $\mathcal{S}(\Omega, \Gamma_P)$ of the perforated polygon $\mathcal{P}(\Omega, \Gamma_P)$ to be the locus of points $p \in \mathcal{P}$ such that p is equidistant (according to $d_{\mathcal{P}}$) from two or more connected components of Γ_Q. An example is given in Figure 5.2. The structure of the skeleton is given by the following lemma.

Lemma 5.2 *The skeleton $\mathcal{S}(\Omega, \Gamma_P)$ of a perforated polygon (without holes) of size n is an embedding of a forest whose edges consist of a union of $O(n)$ straight line segments and arcs of parabolas. Furthermore, two edges of $\mathcal{S}$ that meet at a node of degree two do so smoothly (with continuous slope).*

Proof: The structure of the edges of $\mathcal{S}(\Omega, \Gamma_P)$ follows from elementary analytic geometry: we get line segment bisectors between two edges and parabolic bisectors between a

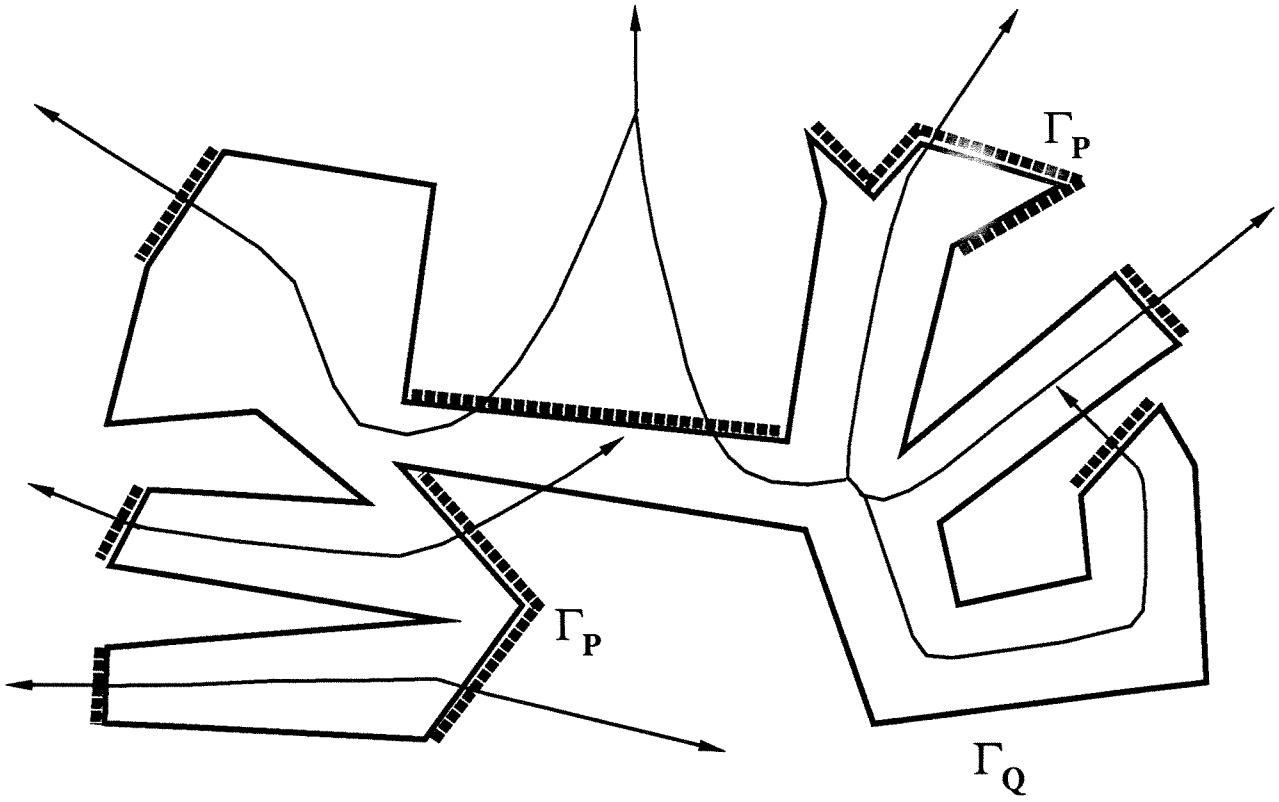

Figure 5.2. Example of a skeleton $\mathcal{S}(\Omega, \Gamma_P)$.

vertex and an edge. (See [Ya] for similar proofs in the unperforated case.) The fact that $\mathcal{S}(\Omega, \Gamma_P)$ is a forest follows from the fact that there are no holes in Ω: If there were a cycle, then some portion of Γ_Q would have to lie inside of it. ∎

In order to avoid confusion between edges of the polygon Ω and "edges" of the forest represented by $\mathcal{S}(\Omega, \Gamma_P)$, we will refer to the latter as *skeletal edges*. The skeleton $\mathcal{S}$ of a perforated polygon is an embedding of a forest on a set of Riemann sheets (flaps). It is not hard to see that each flap has at least one skeletal edge on it that goes off to infinity along a ray. We can truncate these skeletal edges to make the embedding finite, and thereby speak of "leaves" of the forest. (Alternatively, we could speak of leaves "at infinity" to represent the ends of the infinite skeletal edges.) Our goal is to compute the skeleton $\mathcal{S}(\Omega, \Gamma_P)$ in time $O(n \log n)$.

Lemma 5.3 *The skeleton $\mathcal{S}$ of the perforated polygon (Ω, Γ_P) without holes can be computed in time $O(n \log n)$.*

Proof: We begin by constructing the Voronoi diagram $VD(\Omega)$ in time $O(n \log n)$ (e.g., by using the algorithm of Yap [Ya]). We will assume for simplicity that the components of Γ_P are single edges. (If not, then removing the vertices from a component of Γ_P will break it into single edges, and will not fundamentally change the problem.) Let edge e be one of the edges of Γ_P, let $V(e)$ be the Voronoi cell of e within Ω, and let F_e be the flap corresponding to e. In the original computation of $VD(\Omega)$, the edge e was included as a V-source. We now remove e as a V-source, deleting the Voronoi edges (V-edges) that make up the boundaries of each cell $V(e)$. Now, we imagine what happens as we allow the Voronoi diagram to "propagate" into the cell $V(e)$ and then through the opening e into the flap F_e. Basically, we are subdividing $V(e)$ and F_e according to the set of V-sources which were adjacent to e in the original diagram $VD(\Omega)$. Refer to Figure 5.3.

First, we claim that the medial axis portion of the Voronoi diagram $VD(\Omega)$ that lies in $\Omega \setminus [\cup_{e \in \Gamma_P} V(e)]$ yields the skeleton $\mathcal{S}$ within that region. This follows trivially from the fact that the medial axis is the locus of centers of circles within Ω that touch two or more points on the boundary Γ and that these touch points will not be part of the perforation Γ_P if we restrict our attention to $\Omega \setminus [\cup_{e \in \Gamma_P} V(e)]$.

For each edge e of Γ_P, we consider the V-edges $\mathcal{E}(e)$ of $VD(\Omega)$ that make up the boundary of cell $V(e)$ and that are bisectors between e and segments or vertices of Γ_Q. (So, $\mathcal{E}(e)$ does not include those V-edges that bisect e from other edges of Γ_P.) Associated with each V-edge $\overline{uv} \in \mathcal{E}(e)$, there is a line subsegment $\overline{u'v'}$ or a vertex w of Γ_Q. Let $W(e)$ be the set of all such subsegments and vertices. Add the endpoints of e to set $W(e)$ (if they are not already in $W(e)$).

Now, compute the Voronoi diagram, $VD(W(e))$, of $W(e)$, treating the connected components of $W(e)$ as V-sources. This can be done in time $O(|W(e)| \log |W(e)|)$ by Yap's algorithm [Ya]. We claim that we can obtain from $VD(W(e))$ the portion of the skeleton $\mathcal{S}$ that lies within $V(e)$ and extends into the flap F_e.

Let p be a point on the skeleton $\mathcal{S}(\Omega, \Gamma_P)$, and let q_1 and q_2 be two distinct $d_{\mathcal{P}}$-closest points of Γ_Q from p. Then, q_1 and q_2 are both visible from p, according to our notion of visibility on $\mathcal{P}$. If p lies in the base sheet and $p \in V(e)$, then q_1 and q_2 must either be vertices in the set $W(e)$ or lie on segments of $W(e)$ (since $\overline{pq_i}$ must intersect the boundary of $V(e)$ at some point of a V-edge from set $\mathcal{E}(e)$). Thus, p must be on a V-edge of $VD(W(e))$. Conversely, if p lies on a V-edge of $VD(W(e))$ and $p \in V(e)$, then p must lie on the skeleton $\mathcal{S}$.

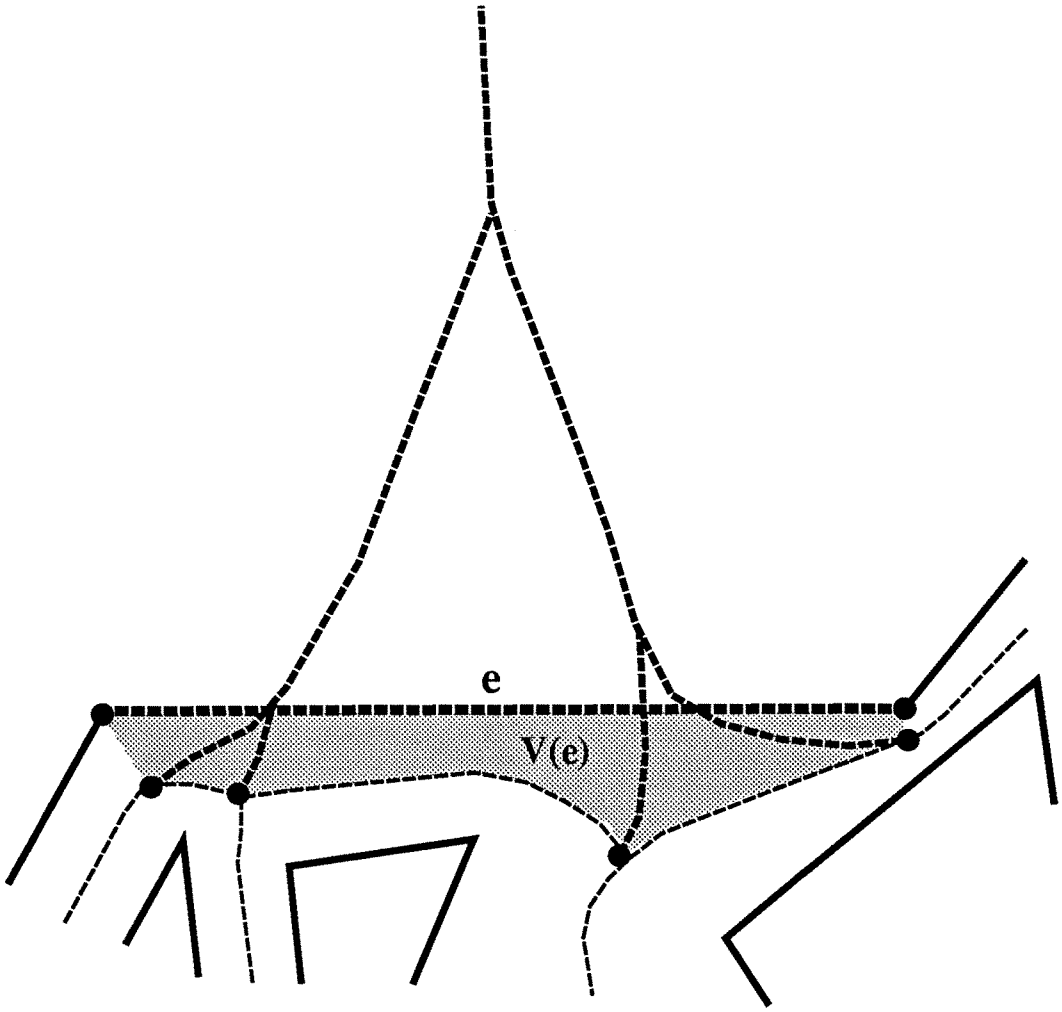

Figure 5.3. Propagating the Voronoi diagram through edge e.

If p is on the flap F_e, then again q_1 and q_2 must lie on elements of $W(e)$, since $\overline{pq_i}$ must intersect e (by definition of visibility between p and q_i) and so must also intersect an element of $\mathcal{E}(e)$. This proves that $\mathcal{S} \cap F_e$ is a subset of the diagram $VD(W(e))$. To prove the converse, consider cutting the diagram $VD(W(e))$ along the segment e. The portion that we cut off is a tree, τ, since no elements of $W(e)$ can lie in the halfplane (not containing $V(e)$) defined by the line through edge e and hence there can be no cycles of $VD(W(e))$ in that halfplane. We claim that if we think of τ as embedded in the sheet F_e, then $\tau = \mathcal{S} \cap F_e$. Simply note that if p is a point on τ, then there must be two points q_1 and q_2 on distinct elements of $W(e)$ such that p is equidistant from q_1 and q_2, and the segments $\overline{pq_i}$ must intersect e. This shows that $p \in \mathcal{S}$, so we are done.

By doing the above construction for each e of Γ_P, we can construct the skeleton $\mathcal{S}$ within each cell $V(e)$ and on each flap F_e. The total construction time is $O(n \log n)$, since the sum of the sizes of $W(e)$ is linear (by the linearity of the size of the Voronoi diagram $VD(\Omega)$). ∎

Note that the complications involved in attaching flaps and propagating the skeleton through perforations can be avoided if the perforations all lie on the boundary of the convex hull of Ω. Note also that it should be possible to build $\mathcal{S}$ in a single pass, rather than by cutting and pasting Voronoi diagrams together as we have done. Recently, Seidel [Se] has adapted Fortune's sweepline algorithm [Fo] to the case of *constrained* Voronoi diagrams, suggesting that a similar technique should work here; however, it is still important in Seidel's algorithm to maintain a distinction among many Riemann sheets.

In the general case of arbitrary Γ_P note that we can potentially simplify the problem slightly by doing the following preprocessing: Pull each component of Γ_P "taut" within Ω, replacing each by its shortest path within Ω. (Shortest paths can be computed efficiently by the algorithm of [GH].) Let Γ'_P be the new sections of boundary, and let Ω' be the resulting new region. (Note that $\Omega' \subseteq \Omega$.) Refer to Figure 5.4, where the set $\Omega \setminus \Omega'$ is shown shaded. The interior of Ω' may have many components; we solve the skeleton problem within each component separately, since the max flow problem can be decomposed into a set of separate problems, one problem per component. Within a component of $int(\Omega')$, the components of Γ'_P are concave chains.

Having addressed the problem of constructing skeletons, we return attention now to the use of the skeleton for solving the flow problem. In the special case that Γ_s and Γ_t are each single edges of Ω, the skeleton will be a simple *path*, and the min cut will be a line segment (connecting B and T) that crosses the skeleton $\mathcal{S}$ at a point p whose $d_{\mathcal{P}}$-distance to Γ_Q is minimized. (In fact, the cut γ^* will be orthogonal to $\mathcal{S}$ at p.) Since each skeletal edge is either a line segment or a parabolic arc, it is easy to find the min cut in $O(n)$ time by searching the skeletal edges.

Lemma 5.4 *Assume that Ω is a simple polygon without holes and that Γ_s and Γ_t each consist of a single (open) edge in the boundary Γ. Then the min cut γ^* consists of a single line segment joining B and T, and γ^* can be computed in time $O(n \log n)$.*

Constructing the Flow Field

In the special case of single source and single sink, the skeleton is a simple directed path

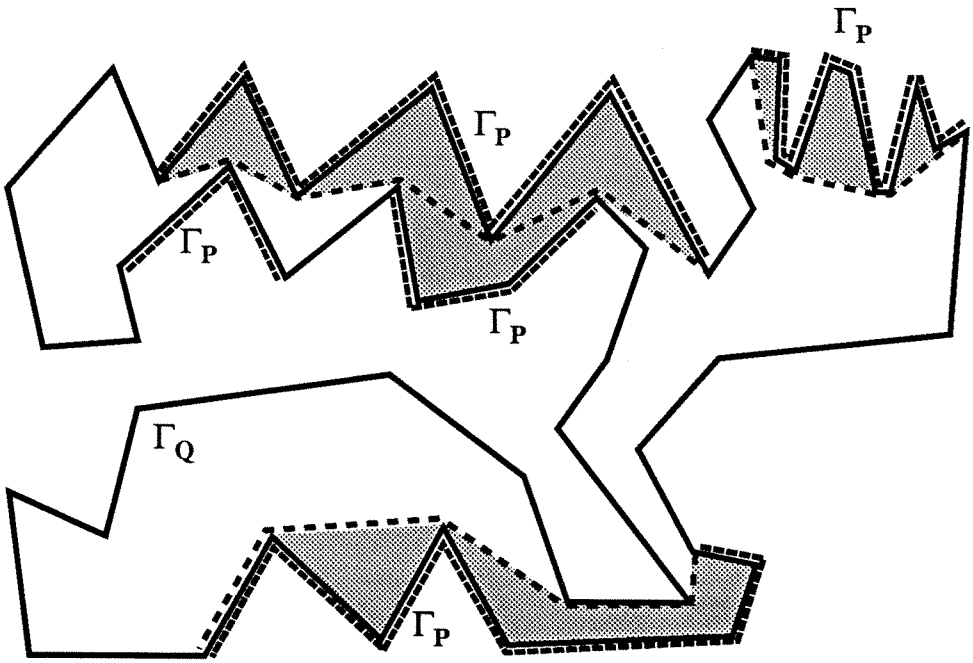

Figure 5.4. Pulling Γ_P taut.

which crosses the boundary Γ only at the source and sink edges Γ_s and Γ_t. The skeleton gives us a means of transforming the continuous flow problem in a polygon into a trivial network flow problem. Capacities can be assigned in the obvious way: a skeletal edges's capacity is simply twice the minimum distance from a point along the skeletal edge to the set Γ_w. There is a unique path through the skeleton from source leaf to sink leaf, and the minimum capacity along it is the value of the max flow problem. One can actually construct a flow field σ^* of maximum value based on the path from source to sink. The result is the following.

Theorem 5.5 *Given a simple polygonal region without holes, and given a source and a sink edge, one can find a min cut γ^* and a max flow field σ^* in time $O(n)$ once the skeleton of the region is computed (which can be done in time $O(n \log n)$). Furthermore, within the same time bounds, one can find an optimal field σ^* which is piecewise constant over a polygonal subdivision of linear size.*

Proof: Lemmas 5.3 and 5.4 addressed the issues of building the skeleton and finding a min cut within the claimed time bounds. We show how to convert an optimal flow on the capacitated skeleton $\mathcal{S}$ into an optimal flow field through Ω. One simple approach

is to take the flow field whose streamlines are defined by the set of paths of a particle at distance x from the skeleton $\mathcal{S}$, for each $x \leq \frac{1}{2}\mu^*$. It is not hard to see that this field can be stored in a linear-size data structure and is conservative, since it can be written $\sigma(p) = (\partial f(p)/\partial y, -\partial f(p)/\partial x)$, for $p \in \Omega$ with $f(p) \leq \frac{1}{2}\mu^*$ (and $\sigma(p) = (0,0)$ for $f(p) > \frac{1}{2}\mu^*$), where $f(p)$ is the Euclidean distance from p to $\mathcal{S}$. However, we give below a simple construction which yields a field which is piecewise constant over a polygonal subdivision. Refer to Figure 5.5.

Along skeletal edges a of $\mathcal{S}$ that are straight line segments, we simply build a rectangular region, centered along a, whose width is μ^*. Within the region, we assign σ^* to be unit vectors parallel to a, oriented with the direction from source to sink.

A parabolic arc $a = \overline{u_1 u_2}$ of $\mathcal{S}$ is a bisector between some vertex v and some edge e of Ω. If a includes the midpoint, m, of the segment $\overline{vv_\perp}$ (where $v_\perp$ is the projection of v onto the line containing e), then we split arc a in two at point m. We can therefore concentrate on the case illustrated in Figure 5.5. Basically, we construct two rectangular "pipes", each of width μ^*, centered on the tangent lines at either end of the arc. Where the two "pipes" meet we form a junction, which will be a bisector between the center lines of the two "pipes". Thus, when we assign σ^* to be unit vectors parallel to the center lines within each region, we get a conservative flow field. The fact that this construction is possible follows from the fact that the two pipes can fit inside the path of a maximum-radius circle that moves from u_1 to u_2 along arc a. The fact that the construction "fits together" in a conservative way from one arc to the next is a consequence of the smoothness of the skeleton. The construction builds only a constant-size polygonal subdivision for each skeletal edge, and hence we get the claimed space complexity. ∎

Note the close similarity between the simple max flow problem considered here and the motion planning problem of moving a disk through a simple polygonal region. The length of the min cut corresponds to the maximum diameter disk that can be moved through the polygon, entering at the source edge, and leaving at the sink edge (assuming that there are additional Riemann sheets attached at the source and sink edges). Use of the skeleton to compute flows relates to the familiar "retraction method" of motion planning. An optimal flow field can be specified by the streamlines that one would obtain by tracing the paths of each point of the disk during its motion through the polygon (assuming that we do not turn the disk during its motion).

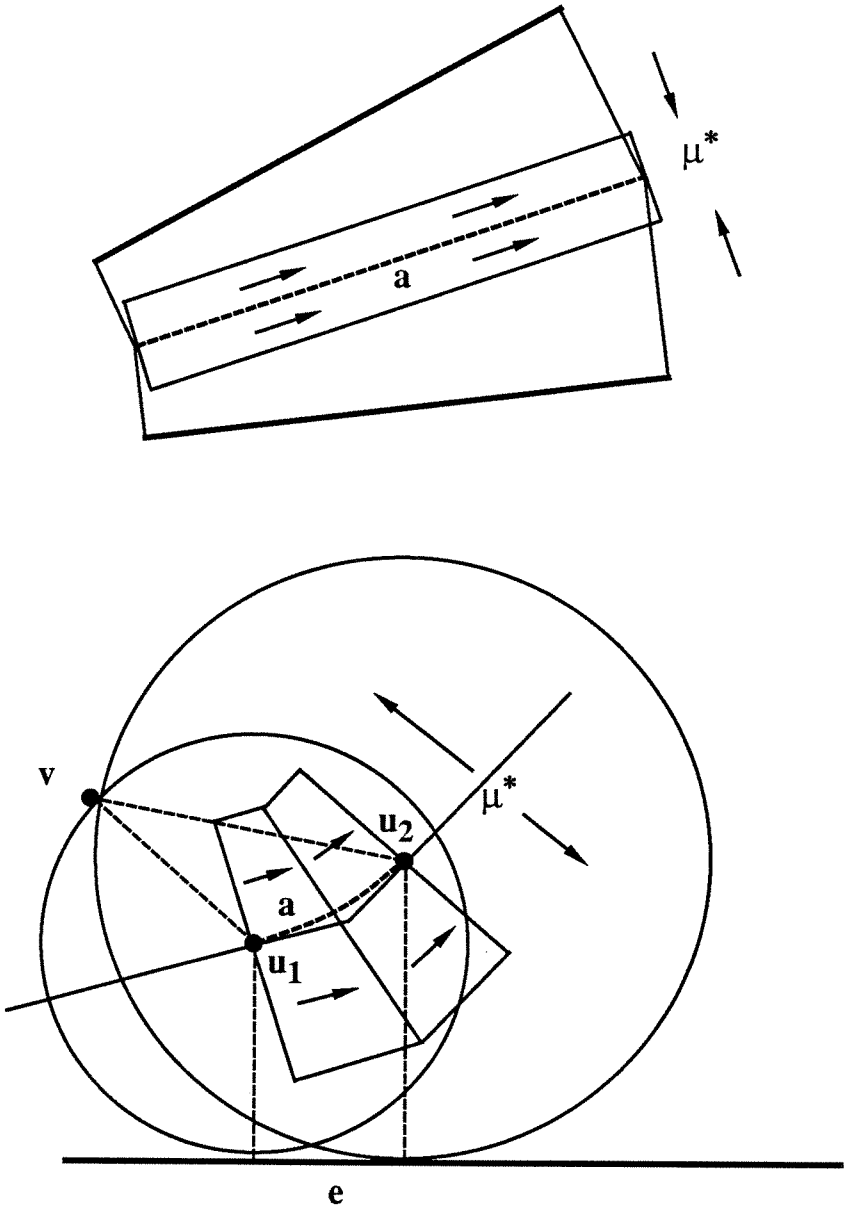

Figure 5.5. Constructing a flow field from a skeleton.

Many Sources and Many Sinks

Another interesting flow problem for simple polygonal domains is that in which there are multiple sources and multiple sinks; in other words, Γ_s and/or Γ_t consist of several connected components. Let us assume for now that no two connected components of Γ_s (resp., Γ_t) lie in consecutive order about the boundary $\Gamma_\infty = \Gamma$. In particular, this will

imply that the number of sources equals the number of sinks. This assumption will be lifted when we address the more general problem (in which there are holes in Ω) in the next section.

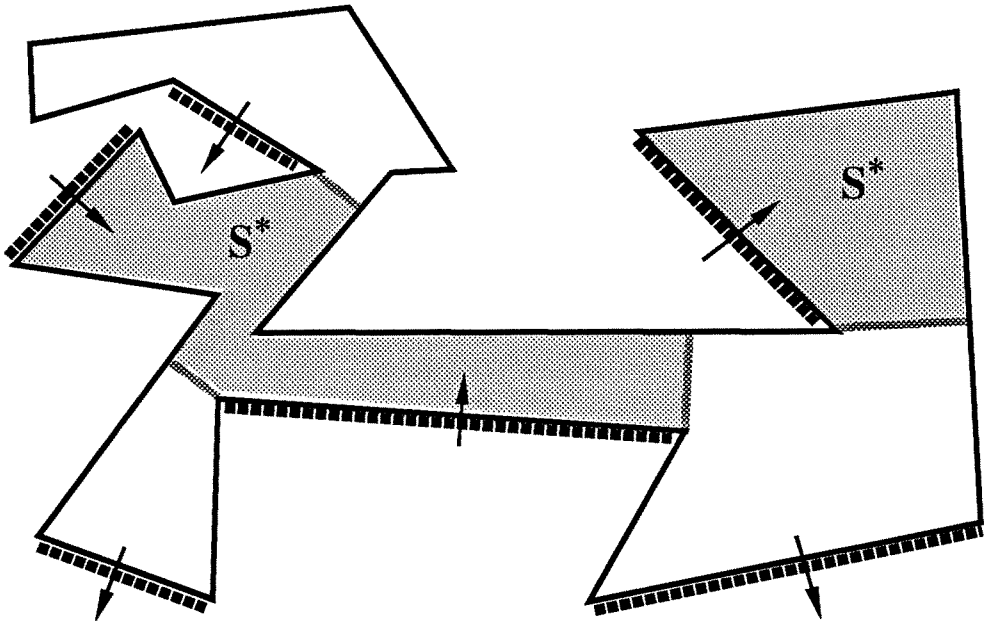

Figure 5.6. A min cut for many sources and sinks.

It is no longer the case that γ^* must be connected; γ^* may consist of several disjoint line segments, as in Figure 5.6 (where an example is shown with 3 source edges and 3 sink edges). It is not obvious how one could find a min cut by applying a shortest path algorithm.

We see in the figure that the optimal cut S^* (the shaded region) consists of two connected components, and γ^* consists of four paths (line segments). In general, S^* will be fully describable by a partitioning of the sources into classes each bordering a connected component of S^*. Given such a partitioning, then, we can define a set of shortest path problems (according to the $0/1/\infty$ weighted region distance [GMMN,Mi2]) which will find a min cut consistent with the partition. But among the exponentially many possible partitions, how does one pick the partition whose cut length will be minimum?

We look at the problem in another way. Consider the skeleton $\mathcal{S}(\Omega, \Gamma_P)$ of the perforated simple polygon ($\Gamma_P = \Gamma_s \cup \Gamma_t$). To each skeletal edge a, assign a capacity equal to twice the minimum distance from a point on a to the set Γ_w. Then, by placing source/sink

nodes accordingly at the leaves of $\mathcal{S}$, we get a network flow problem on a capacitated forest of size $O(n)$ with many sources and sinks. Any cut in the network flow problem corresponds to a cut in the continuous flow problem on Ω, and, furthermore, any min cut γ^* for the continuous problem corresponds to some cut in the network flow problem (since the capacity of skeletal edges corresponds to the length of a locally optimal path in the $0/1/\infty$ weighted region problem). Thus, in order to solve the continuous flow problem, we are faced with solving a network flow problem on a forest. Such flow problems can be solved by a straightforward method in time $O(n)$, as we now see.

Lemma 5.6 *Given a forest with n nodes, capacitated edges, and with many sources and sinks at the leaves, a maximum flow can be computed in time $O(n)$.*

Proof: Let $c(u,v)$ be the capacity of edge (u,v). Any leaf which is not a source or a sink can be deleted, along with the unique edge joining it to the forest. If u is a source (sink) adjacent to a node u' of degree two (which is adjacent to node $v \neq u$), then we can replace edges (u,u') and (u',v) with a single new edge (u,v) whose capacity is $\min\{c(u,u'), c(u',v)\}$. If u_1 and u_2 are both sources (sinks) adjacent to a non-leaf node v, then we can merge u_1 and u_2 into a single source (sink) node u, placing a capacity on edge (u,v) equal to $c(u_1,v) + c(u_2,v)$. Finally, if u_1 is a source adjacent to v and u_2 is a sink adjacent to v, then we can route an amount $x = \min\{c(u_1,v), c(u_2,v)\}$ of flow from u_1 to v to u_2, and then decrease by x the capacities of (u_1,v) and (u_2,v), allowing one of the two edges to be deleted. The result of applying these operations to a forest will be to reduce the problem to the trivial one in which every connected component is either a single node or a single edge. Thus, it is easy to see that this process solves the original maximum flow problem and can be done in linear time. ∎

Once the flow problem is solved on the skeletal network, a construction similar to that in the proof of Theorem 5.5 shows that an optimal piecewise-constant flow field can be constructed in time and space $O(n)$. Since $\mathcal{S}$ can now have nodes with degree greater than two, we must modify the construction slightly at such nodes of $\mathcal{S}$. We omit the details here since a more general version of flow construction will be given in Section 7. This yields the following.

Theorem 5.7 *If there are many sources and sinks in alternating order on the boundary of a uniformly-capacitated simple polygon Ω, the max flow problem can be solved in time $O(n)$, once the skeleton has been computed, which takes time $O(n \log n)$. Furthermore,*

σ^* can be constructed to be piecewise-constant (consisting only of unit vectors and zero vectors) over a polygonal subdivision of size $O(n)$.

6. Simple Polygons with Holes

Assume now that the domain Ω is a simple polygon with h holes, and that the capacity is uniform ($c = 1$). We begin by assuming that Γ_s and Γ_t are single source and sink edges that lie in the same connected component of Γ; we assume that they belong to Γ_∞ (other cases are handled similarly). Then the set $\Gamma_\infty \cap \Gamma_w$ consists of two connected components, T ("top") and B ("bottom"). Refer to Figure 6.1.

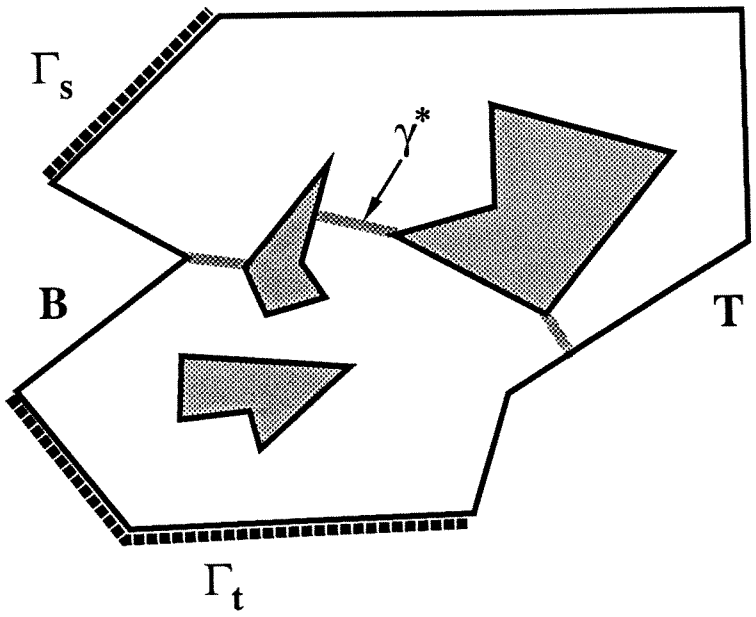

Figure 6.1. A min cut for a simple polygon with holes.

Theorem 6.1 *The min cut problem for a multiply connected polyhedral region Ω with Γ_s and Γ_t each consisting of a single (open) edge of Γ_∞ can be stated as the following 0/1/∞ weighted region shortest path problem:*

Min Cut Problem: Find a shortest weighted path from any point $s \in B$ to any point $t \in T$, where weights are assigned as follows: Ω^c gets weight ∞; Γ_w gets weight 0; and Γ_s, Γ_t, and $int(\Omega)$ get weight 1. The (weighted) length of the shortest path is the value of the minimum cut, and γ^* is given by the restriction of the path to $int(\Omega) \cup \Gamma_s \cup \Gamma_t$.

Proof: By Lemma 5.1, the min cut set S^* will consist of a single connected component, possibly with holes. Let $\partial_\infty S^*$ be the component of ∂S^* which is the boundary of the unbounded component of $\Re^2 \setminus S^*$ (i.e., $\partial_\infty S^*$ is the outer boundary of S^*). Since $\Gamma_s \subseteq \partial_\infty S^*$, there must exist a point $s \in B$ and a point $t \in T$ such that $s, t \in \partial_\infty S^*$. Points s and t split the outer boundary $\partial_\infty S^*$ into two pieces; let P be the piece that does not include Γ_s. P will be a path from s to t. Now, the length of $\gamma^* = \partial(S^* \cup \Omega^c) \setminus \Gamma_w$ is precisely the length of the portion of the path P that does not coincide with Γ_w. The $0/1/\infty$ weighted region shortest path problem minimizes this path length, since it treats travel along Γ_w as "free". ∎

Corollary 6.2 *The min cut for the problem stated in Theorem 6.1 can be computed in time and space $O(n^2)$.*

Proof: This follows from the algorithms of [GMMN,Mi2] which basically build a "critical graph" which is guaranteed to contain an optimal path. Actually, these algorithms yield potentially better (output-sensitive) bounds that can be written in terms of the size of the visibility graph of Ω. ∎

We will see in Section 7 that a slight improvement in the bounds given above are possible. In the case that Γ_s and Γ_t are single edges that lie on *different* connected components of Γ, we can solve the min cut problem in time $O(n^2)$ by solving a constant number of shortest path and shortest enclosing cycle problems in $0/1/\infty$ weighted regions, but we omit the discussion here since the method does not generalize to many sources and sinks, a case which will be addressed at the end of this section.

A Lower Bound

We now give a lower bound to the complexity of computing max flows in uniformly capacitated simple polygons with h holes. The min cut problem, as we have seen, is a shortest path problem among $0/1/\infty$ weighted regions with h zero-weighted regions (the boundaries of the holes).

Theorem 6.3 *The min cut problem has a lower bound of $\Omega(n + h \log h)$ (to produce a min cut path γ^*).*

Proof: We use a simple reduction from sorting. Given integers $i_1, \ldots, i_h$, we construct a $0/1/\infty$ weighted region problem by building tiny square holes with zero-weighted boundaries centered on the points $(i_j, 0)$, and placing a start point to the left of the smallest i_j

and a goal point to the right of the largest i_j. (Clearly the construction takes time $O(h)$.) Then, any algorithm which must output the shortest path from start to goal, will visit the squares in sorted order. ∎

To show a lower bound on constructing max flow fields, we will show that we can use *any* optimal field σ to sort h integers in time $O(h)$. Note that there may be uncountably many possible optimal flow fields. We make only the following assumption about the representation of σ^*: we assume that it is stored in a way such that the value of σ^* at any vertex of Ω can be obtained in constant time; that is, we assume that every vertex of Ω is labelled with the value of σ^* there. (Actually, the value of σ at a vertex may be zero since the vertex lies on the boundary Γ. Technically, then, we want to associate with a vertex the value of σ^* in some arbitrarily small neighborhood of the vertex.)

Theorem 6.4 *If Ω is a uniformly capacitated simple polygon with h holes, a lower bound on computing a representation of a maximum flow field σ is $\Omega(n + h \log h)$. (This assumes that the representation of σ allows constant-time evaluations at vertices of Ω.)*

Proof: For integers $i_1,\ldots,i_h$, let i_s (resp., i_l) be the smallest (resp., largest) integer. Map the points i_j onto the parabola defined by the function $y(x) = (x - (i_s - 1))^2$ (let the resulting points be p_j). Let $r = i_l - i_s + 1$, and place a point O at location $(i_s - 1, -r^2)$. Draw a ray from O through each p_j, and define a set of holes, H_j, which are the segments of these rays between p_j and the boundary of a large rectangle R'. (R' has sides parallel to the coordinate axes, and it should be at least large enough to enclose O and all of the p_j's.) Enclose this construction in a slightly larger rectangle, R. Let the bottom edge of R be a source edge, and let the top edge be a sink edge. Let Ω be the rectangle R with holes H_j, and let $c = 1$. See Figure 6.2.

In this max flow problem we see that the min cut γ^* visits the holes H_j in the sorted order of the i_j's. Furthermore, from simply knowing the values of $\sigma^*(p_j)$ for each j, we can read off the sorted order in $O(h)$ time: the vector $\sigma^*(p_j)$ must be a unit vector whose slope equals that of the ray from O through $p_{j'}$, where j' is the index of the integer $i_{j'}$ which immediately precedes i_j in the sorted order of the integers. Thus, we can walk through the sorted order by making h inquiries on σ^* at vertices of Ω. ∎

Many Sources and Many Sinks

What happens when there are many sources and sinks? As in the case without holes, it is not obvious how to find the min cut, since there are many possible topologies of cut sets.

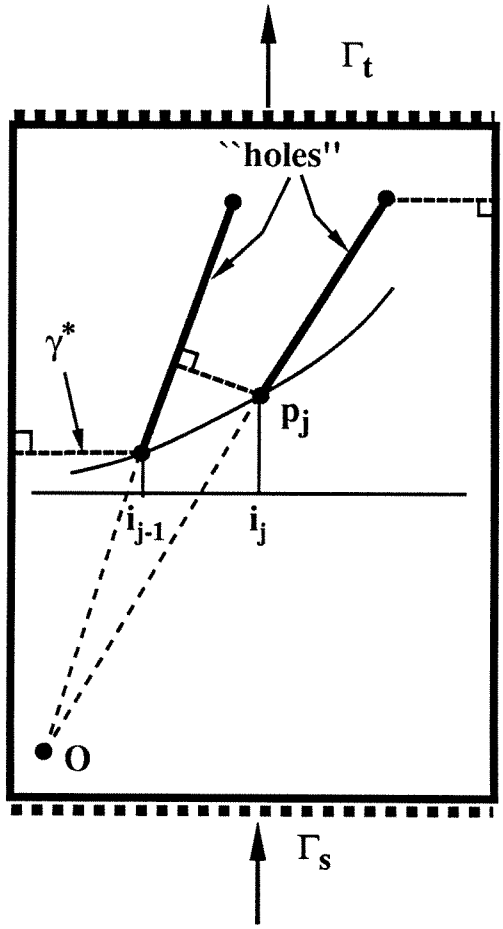

Figure 6.2. Lower bound construction.

There is no single shortest path problem whose solution yields a min cut.

Our approach will be to map the geometric problem into a corresponding network flow problem on a planar graph. We consider first the case in which all sources and sinks lie in the same connected component of Γ. (The case in which sources and sinks lie on different connected components of Γ will be addressed in the next subsection.) Without loss of generality, we can assume that $\Gamma_s, \Gamma_t \subset \Gamma_\infty$.

A first attempt to solve the problem might be to define a network based on the generalized Voronoi diagram $VD(\Omega)$ of the region Ω with holes, assigning a capacity to a Voronoi edge according to minimum separation between the objects defining the edge. It is not hard to see that this network does not work in general. The problem is that the min cut will not necessarily correspond to a path (or set of paths and cycles) in the Delaunay

diagram.

Instead, we define a new kind of skeleton for a perforated polygon with holes. The basic idea is to compute the skeleton using the metric which assigns a weight of zero to holes (so that we are charged only for motion within Ω). First, we make the following simplification: assume that the sources and sinks lie in alternating order about the outer boundary Γ_∞. This is without loss of generality, since, if two connected components of Γ_s lie adjacent on Γ_∞ (with no sink between them), then we can consider the (connected) portion of Γ_∞ that separates them to be a "hole" that lies very close to the outer boundary of Ω, and connect the two sources into one component. See Figure 6.3.

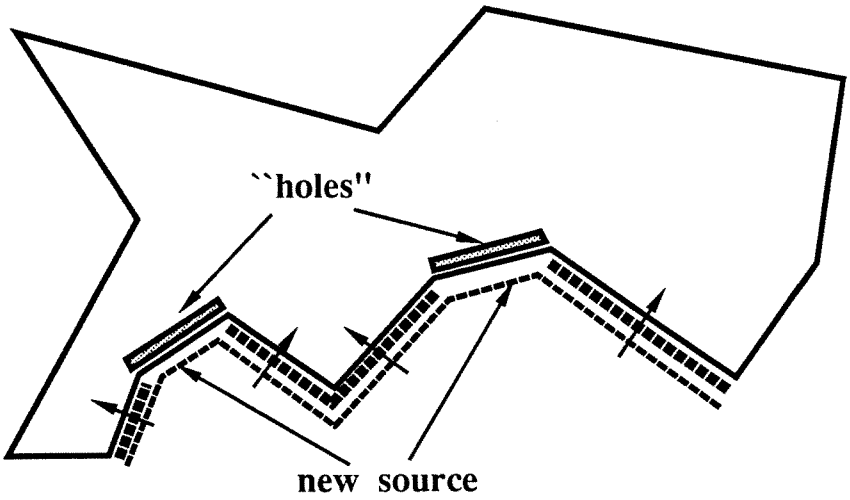

Figure 6.3. Joining adjacent sources yields alternating pattern.

We now define more formally the modified skeleton of a perforated polygon $\mathcal{P}(\Omega, \Gamma_P)$ with a set $\mathcal{H}$ of holes. We define visibility between two points of $\mathcal{P}(\Omega, \Gamma_P)$ exactly as we did before in the case without holes. We define a distance function $d'_{\mathcal{P}}(p, q)$ as follows. Construct a graph whose nodes correspond to p, q, and the holes $\mathcal{H}$. Connect two nodes by an edge if and only if there exists a pair of points, one on each of the two objects defining the nodes, which are visible to one another. Define the length of the resulting edge as the minimum Euclidean distance between all such pairs of visible points. Now define $d'_{\mathcal{P}}(p, q)$ to be the length of the shortest path in this graph from node p to node q ($d'_{\mathcal{P}}(p, q) = \infty$ if no path exists). Another way to view this definition is to think of the holes as having weight 0, and then applying the 0/1 weighted region distance. The graph we just defined

is closely related to the critical graph used in [GMMN,Mi2] for the 0/1 weighted region problem.

We define the *modified skeleton* $\mathcal{S}'(\Omega, \Gamma_P)$ of $\mathcal{P}(\Omega, \Gamma_P)$ (*with respect to holes* $\mathcal{H}$) as the locus of points $p \in \mathcal{P}$ that are equidistant (according to $d'_{\mathcal{P}}$) from two or more connected components of $\Gamma_Q = \Gamma_\infty \setminus \Gamma_P$. Figure 6.4 shows an example of a skeleton.

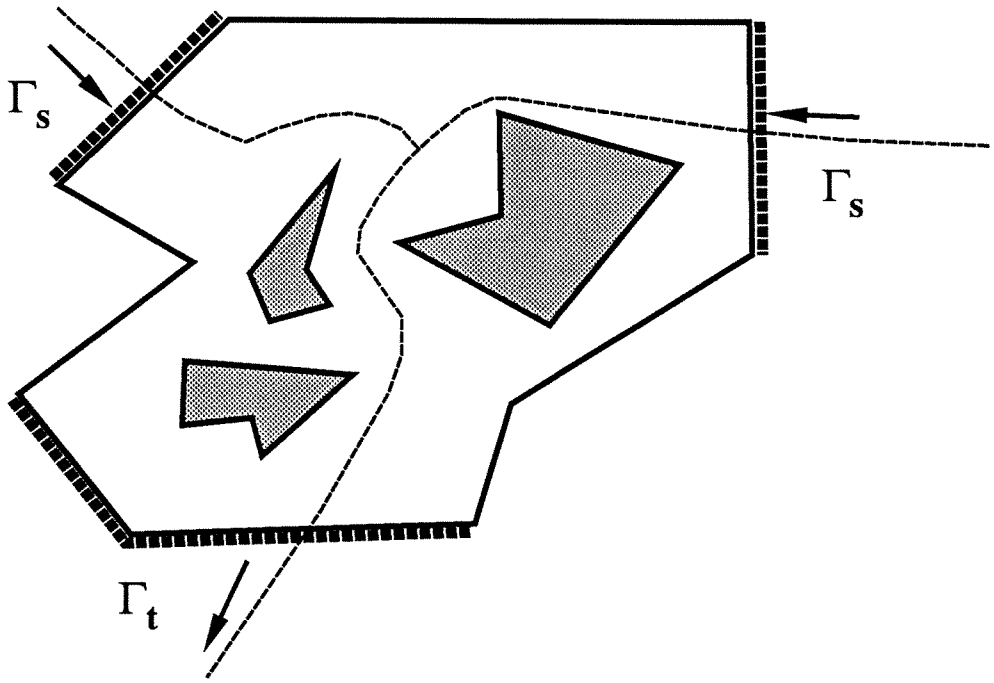

Figure 6.4. Modified skeleton of a perforated polygon with holes.

Lemma 6.5 *The modified skeleton* $\mathcal{S}'(\Omega, \Gamma_P)$ *of a perforated polygon (with holes) of size* n *is an embedding of a forest whose edges consist of a union of* $O(n)$ *straight line segments, parabolic arcs, and hyperbolic arcs.*

Proof: The structure of the edges of $\mathcal{S}'(\Omega, \Gamma_P)$ follows from elementary analytic geometry: we get line segment bisectors between two edges, parabolic bisectors between a vertex and an edge, and hyperbolic arcs between two vertices (from different holes). See [Ya] for similar proofs. The fact that $\mathcal{S}'(\Omega, \Gamma_P)$ has no cycles follows from the fact that the connected components of Γ_Q all lie on the outer boundary Γ_∞. ∎

Remark: We can think of the modified skeleton in terms of the following "grass fire" analogy. Imagine that the holes (or their boundaries) have been doused with an extremely

flammable fluid such that as soon as any point of a hole's boundary is ignited, the entire hole instantly is engulfed in flames. Now, start a grass fire on the portion Γ_Q of the outer boundary of the perforated polygon. Whenever the flame reaches a hole, it spreads throughout it instantly. Those points of $\mathcal{P}$ where walls of flame collide define the modified skeleton.

In the next section, we will show how to compute the modified skeleton $\mathcal{S}'$ of a perforated polygon with holes using a "continuous Dijkstra" algorithm, which can be interpreted as a continuous version of the uppermost path algorithm for max flow in planar graphs [FF].

We can associate capacities with the arcs of $\mathcal{S}'$ in the obvious way: the capacity of arc e is equal to twice the minimum $d'_{\mathcal{P}}$-distance from e to the components of Γ_Q defining the arc. We place source and sink nodes at the leaves of $\mathcal{S}'$ according to the original flow problem. Thus, we are again left with the task of solving a max flow problem on a forest which has many sources and sinks at the leaves.

Theorem 6.6 *The max flow problem in Ω (with holes) with many sources and sinks on the outer boundary can be formulated as a max flow problem in the (capacitated) modified skeleton $\mathcal{S}'$, which can be solved in time $O(n)$ once $\mathcal{S}'$ is known.*

Proof: Since it is easy to see that any cut in the network flow problem on the skeleton corresponds to a cut in the continuous flow problem on Ω, the main thing to argue is that any min cut γ^* for the continuous problem must correspond to *some* cut in the capacitated forest given by the skeleton. This follows from the fact that if p lies on edge a of the skeleton $\mathcal{S}'$ and is the cause of a bottleneck along a, then there are two points q_1 and q_2 on different components of Γ_Q which are equidistant (according to $d'_{\mathcal{P}}$) from p and such that the $d'_{\mathcal{P}}$-optimal path from q_1 to p to q_2 is a $d'_{\mathcal{P}}$-optimal path between the two components of Γ_Q containing q_1 and q_2. Basically, the modified skeleton captures the relevant set of shortest paths according to the $0/1/\infty$ weighted region problem obtained when putting weight 0 on holes.

Applying Lemma 5.6, we get the claimed time bound for solving the network flow problem. The min cut for the geometric flow problem can be constructed readily from the min cut found in the solution to the network flow problem on $\mathcal{S}'$. A max flow field σ can be constructed from the network solution by a method similar to that described in Section 5 for the case of a simple polygon without holes. (More will be said on the issue of construction in Section 7.) ∎

The More General Problem

A more difficult problem is the general case in which there are many sources and/or sinks on many different components of Γ. We know of no way to generalize the above discussion to include this case, as it seems to be difficult to define an appropriate capacitated "skeleton" that captures the geometric flow problem. Instead, we present a straightforward method of converting a geometric flow problem into a network flow problem on a planar graph. (The method also applies, of course, to the various simpler cases we have described so far; however, it is much less efficient in such cases than the methods we have already described.)

First, associate $0/1/\infty$ weights with regions in the plane: (1). Ω^c gets weight $+\infty$; (2). $int(\Omega)$, Γ_s, and Γ_t get weight 1; (3). Γ_w gets weight 0. Now construct the critical graph within region Ω (see [GMMN]).

The *critical graph* of a polygonal region joins a pair of visible vertices p and q if and only if there is a line orthogonal to $\overline{pq}$ supporting the obstacle at p and also supporting the obstacle at q. The critical graph joins a vertex p to a point p' on line segment $\overline{qr}$ if and only if p' is the foot of the perpendicular dropped from p onto the line containing $\overline{qr}$ and p is visible from p'. In general, the critical graph may not be planar.) It is shown in [GMMN] that any path which is shortest, or locally shortest, (according to the weighted region metric) must lie on the critical graph. Thus, the set γ^* must be a set of disjoint (simple) paths and (simple) cycles in the critical graph.

Let $\mathcal{C}$ be the set of (closed) cells in the arrangement $\mathcal{A}$ defined by the line segments of Γ together with those that make up the critical graph. We define a planar graph $\mathcal{G}$ whose nodes correspond to cells $C \in \mathcal{C}$, and whose edges join nodes that correspond to cells that share a common face (of dimension one). Thus, $\mathcal{G}$ is the planar dual of the arrangement $\mathcal{A}$ defined by the critical graph line segments. Associate with each edge of $\mathcal{G}$ a capacity equal to the length of the face shared by the corresponding cells. We now augment $\mathcal{G}$ by adding a node corresponding to each edge e of Γ_s and Γ_t, and connecting each such node to the (unique) node corresponding to the cell $C \in \mathcal{C}$ which has e as a face. Refer to Figure 6.5. These newly added nodes serve as sources and sinks in $\mathcal{G}$ according to whether they correspond to source or sink edges in the original problem. The end result is that we have a planar graph $\mathcal{G}$ which has many source and sink nodes. The size of $\mathcal{G}$ is simply the number of cells in $\mathcal{A}$. Since the critical graph has worst-case size $O(n^2)$, $\mathcal{G}$ has at most $O(n^4)$ nodes and edges. (Unfortunately, it is not hard to construct examples which achieve this worst-case bound.)

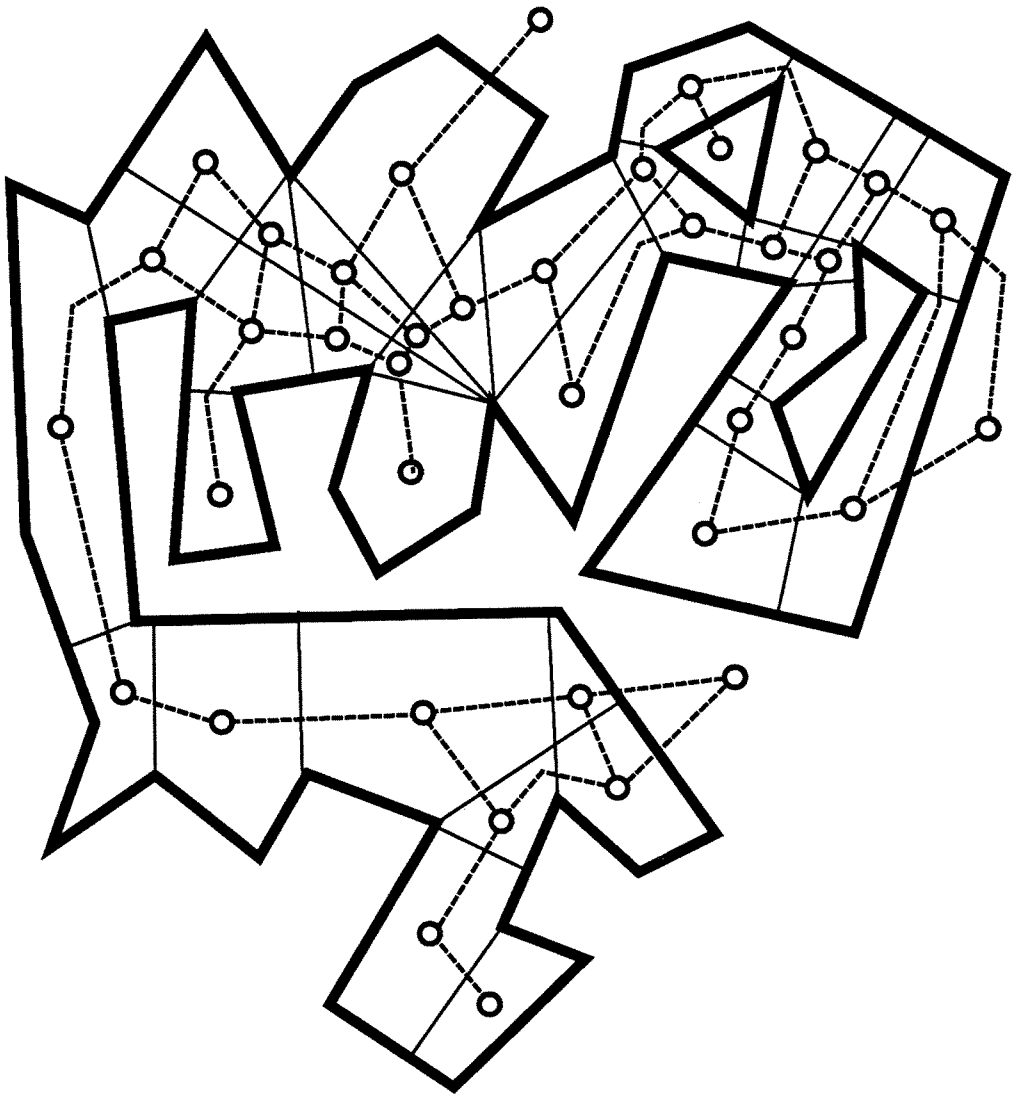

Figure 6.5. Constructing the planar graph $\mathcal{G}$.

Theorem 6.7 *The solution to the general max flow problem in a uniformly capacitated polyhedral region in two dimensions, with arbitrary sources and sinks, can be obtained by solving the max flow problem in the graph $\mathcal{G}$ defined above.*

Proof: A cut in the max flow problem on graph $\mathcal{G}$ corresponds to a set Σ of nodes which contains all source nodes while not containing any sink node. Consider the set $\mathcal{E}$ of all edges of $\mathcal{G}$ that join nodes of Σ to nodes in Σ^c. Each edge $e \in \mathcal{E}$ corresponds to a one-dimensional face in $\mathcal{A}$. We claim that the set of all such segments defines γ for a valid cut S in the geometric flow problem. This is an immediate consequence of the fact that $\mathcal{G}$

was defined as the planar dual of the critical graph arrangement. Thus, a valid cut in the graph problem corresponds to a valid cut in the geometric problem.

Furthermore, for each locally optimal cut in the geometric problem, there is a corresponding cut in the graph problem (since all locally optimal paths and cycles are present within the critical graph). Thus, when we solve the graph problem, we obtain a minimum cut in the graph problem which corresponds to a minimum cut in the geometric problem, and the cut γ^* is readily constructed from the data in the solution to the graph problem. Constructing an optimal flow field for the geometric problem once an optimal solution to the graph problem is known is straightforward, since the graph solution gives us for each cell of $\mathcal{A}$ exactly how much flow is to enter and leave each face of the cell. ∎

Corollary 6.8 *The general max flow problem in a uniformly capacitated polyhedral region in two dimensions can be solved in time* $O(n^8 \log n)$.

Proof: Simply apply the algorithm of [ST] to the network flow problem described above. ∎

7. Continuous Uppermost Path Algorithm

In this section we establish a direct correspondance between the continuous Dijkstra methodology of searching for shortest paths and a continuous version of the uppermost path algorithm for maximum flow in a planar graph [FF].

The uppermost path algorithm applies to a capacitated planar network in which the source and sink nodes are both on the outer face. The idea is simply to select the "uppermost" path from source to sink, push as much flow as possible through it, delete the resulting saturated edges, and repeat the process. In this way, augmenting paths are selected according to an ordering from the "top" of the graph to the "bottom". The uppermost path algorithm can be implemented to run in time $O(n \log n)$ [IS].

For the case of shortest paths among obstacles in the plane, [RS] have given an $O(nh + n \log n)$ algorithm, where h is the number of obstacles. Their algorithm actually computes for a single source point a Shortest Path Map (SPM), which is a planar subdivision linear in size that gives shortest path information from the source to every other point in the plane. This is an improvement over the $O(n^2)$ algorithms of [AAGHI,We] in the case $h << n$. We can achieve similar bounds for the problem of maximum flows in a uniformly weighted polygonal region with h holes. The idea is to apply continuous Dijkstra (which takes on the form of a "continuous uppermost path" algorithm for this problem).

The idea behind continuous Dijkstra is to propagate a wavefront from a V-source (or more generally, a set of V-sources, $\mathcal{B}$), maintaining a structure which describes the shortest path map (or more generally, the generalized Voronoi diagram) for all points whose distance from the source is less than the current event distance (see [Mi1]). Events occur at those discrete times when the combinatorial structure of the wavefront changes.

Our goal is to compute the modified skeleton $\mathcal{S}'$ for a perforated polygon with holes for the case in which $\Gamma_P \subset \Gamma_\infty$. We will be abusing notation by equating $\Gamma_Q = \Gamma_\infty \setminus \Gamma_P$ with the set of connected components of Γ_Q. Let us restate this problem in a slightly different setting:

Problem: Given a simple polygon Ω with a set of holes $\mathcal{H}$, and given a set of open edges Γ_P on the outer boundary Γ_∞ of Ω, compute the Voronoi diagram of the set of V-sources given by $\Gamma_Q = \Gamma_\infty \setminus \Gamma_P$ with respect to the distance function d_Ω which puts a weight of 0 on holes and requires paths to stay within Ω.

Although the problem is not stated to include the effects of attaching flaps, a solution to the above problem can be used to construct the modified skeleton for a perforated polygon with holes by techniques similar to those outlined in Section 5.

We call the Voronoi diagram requested in the problem above the "weighted Voronoi diagram" (WVD), since it can be defined alternatively as follows. For each hole $H \in \mathcal{H}$, let $d_\Omega(H, \Gamma_Q)$ be the distance from H to the set Γ_Q, where distance is measured according to the weighted region metric which assigns a weight of 0 to holes. For connected components $\gamma_Q \in \Gamma_Q$, $d(\gamma_Q, \Gamma_Q) = 0$. Now, the Voronoi diagram requested in the problem defined above is precisely the same as the problem of computing the *weighted* Voronoi diagram (WVD) of the set $\Gamma_Q \cup \mathcal{H}$, where weights are assigned by the function $d_\Omega(\cdot, \Gamma_Q)$ (and weights are additive): subdivide the region Ω into cells $C(\alpha)$ corresponding to objects $\alpha \in \Gamma_Q \cup \mathcal{H}$ such that, if p lies in the cell $C(\beta)$, then $\beta \in \Gamma_Q \cup \mathcal{H}$ is the object minimizing $d_\Omega(\alpha, \Gamma_Q) + d_\Omega(p, \alpha)$.

Actually, we will be constructing a refinement of WVD in which each cell $C(\alpha)$ is further subdivided according to the vertices and open segments that define each object $\alpha \in \Gamma_Q \cup \mathcal{H}$. We will still refer to the subdivision as the WVD, and note that it still has size $O(n)$.

The continuous Dijkstra method to compute WVD will need to determine the effect of propagating wavefronts from the V-sources Γ_Q throughout Ω, treating the holes as zero-weight regions. This means that as soon as a hole is hit by a wavefront, its entire boundary

instantaneously becomes a new V-source to continue propagation. Our algorithm keeps track of what happens each time the wavefront hits a new hole. We use a "merge curve" between the set of V-sources Γ_Q and the unhit holes to keep track of what hole will be hit next. The merge curve is simply the locus of points which are equidistant between set Γ_Q and the set of unhit holes. It is important in telling us when the next collision will occur with a hole. Our algorithm maintains the (weighted) Voronoi diagram (WVD) of $\mathcal{O}$, the set of objects that have already been hit by the wavefront, and the (usual) Voronoi diagram of $\mathcal{R}$, the set of objects remaining to be hit. We outline our algorithm as follows.

Continuous Dijkstra Algorithm

0). *[Initialization]* Compute the (usual) Voronoi diagram, VD, of V-sources Γ_Q within Ω. Initialize $\mathcal{O} = \Gamma_Q$, $\mathcal{R} = \mathcal{H}$, and WVD($\mathcal{O}$)=VD. Compute the usual (outside) Voronoi diagrams, Vor(H), of each hole $H \in \mathcal{H}$. Also compute the Voronoi diagram of the set $\mathcal{H}$, and initialize Vor($\mathcal{R}$) to be this diagram.

1). *[Find merge curve]* Compute the *merge curve set* $\mathcal{M}$ between WVD($\mathcal{O}$) and Vor($\mathcal{R}$). Associate with each arc of $\mathcal{M}$ an *event distance* equal to the minimum distance between the portions of the two objects that the arc bisects and an *event segment* joining the closest points on the two objects. Enqueue all of the arcs of $\mathcal{M}$ with their event distances and event segments in a priority queue, EVENT-QUEUE.

2). *[Main Loop]* If EVENT-QUEUE is empty, then STOP; otherwise, pop the EVENT-QUEUE. Let H be the next hole to be hit. Add H to $\mathcal{O}$, label H with its distance from Γ_Q ($= d(u) + |\overline{uv}|$), and delete H from $\mathcal{R}$.

3). Update WVD($\mathcal{O}$), Vor($\mathcal{R}$), and $\mathcal{M}$. Go to 2).

We now give some more details to explain the above algorithm.

0). In time $O(n \log n)$ we compute the (usual) Voronoi diagram, VD, of V-sources Γ_Q within Ω, ignoring the extension of the diagram into the flaps, and ignoring the holes in Ω. Do this by the techniques of Section 5, beginning with the usual Voronoi diagram of the interior of Ω, and then removing the edges Γ_P. In total time $O(n \log n)$ (and space $O(n)$) we compute the usual (outside) Voronoi diagrams, Vor(H), of each hole $H \in \mathcal{H}$. (Use, for example, the algorithm of [Ya].) These diagrams consider each vertex and (open) edge of each hole H to be V-sources, and they give a subdivision of H^c for each H. We also compute the Voronoi diagram of the set $\mathcal{H}$, and initialize Vor($\mathcal{R}$) to be this diagram.

1). We define the *merge curve set* $\mathcal{M}$ between WVD($\mathcal{O}$) and Vor($\mathcal{R}$) to be the locus of points which are equidistant between set Γ_Q and set $\mathcal{R}$.

Lemma 7.1 *Computing $\mathcal{M}$ can be done in time $O(n \log n)$.*

Proof: Find the closest pair of objects (vertices and line segments) between set $\mathcal{R} = \mathcal{H}$ and set $\mathcal{O} = \Gamma_Q$. (This can be done in time $O(n \log n)$.) Use the midpoint of the segment joining the closest pair as a "starter" to build (in time $O(|M_1|)$) a merge curve (cycle) M_1. (This is done by the standard Shamos-Hoey scan [SH].) If all of the holes lie within the region bounded by M_1, then we are done. (Keeping track of what is in or out of the cycle is easy to do as we construct the merge curves, since we are traversing a path in the planar diagram Vor($\mathcal{H}$).) Otherwise, we find another starter between the remaining (unenclosed) holes and set $\mathcal{O}$, and trace out another merge curve M_2. This continues until all holes are surrounded by some merge curve. The resulting set of curves is $\mathcal{M}$. ∎

Lemma 7.2 *The set $\mathcal{M}$ will consist of line segments, parabolic arcs, and hyperbolic arcs.*

Proof: Each arc is a portion of the bisector between a vertex/edge of Γ_Q and a vertex/edge of some hole. (Hyperbolic arcs will not be present in the original merge curves, but will appear later when some of the holes have been hit and therefore labelled with their distances from Γ_Q.) ∎

We can therefore associate with each arc a distance from Γ_Q at which the wavefront will first hit the corresponding element (vertex or edge) of some unhit hole.

2). If the EVENT-QUEUE is empty, then we are done, since all holes have been incorporated into WVD. Otherwise, in the spirit of Dijkstra's algorithm, we pop the EVENT-QUEUE to determine which hole is to be "permanently labelled" next. Let $M \in \mathcal{M}$ be the merge curve containing the event arc and let the corresponding event segment be $\overline{uv}$, where $v \in H$ and H is the next hole to be hit by the wavefront. (Point $u \in \mathcal{O}$ belongs either to Γ_Q or to some hole which has already been hit.) We add H to $\mathcal{O}$, label H with the appropriate distance from Γ_Q $(= d_\Omega(u, \Gamma_Q) + |\overline{uv}|)$, and delete H from $\mathcal{R}$.

3). Using the event point $v \in H$ as a "starter", we can update the diagram WVD($\mathcal{O}$) by a Shamos-Hoey scan through the current WVD diagram and the (external) Voronoi diagram of the hole H. (This is done in time linear in the number of changes to the diagram.) Updating Vor($\mathcal{R}$) (in order to remove H) is straightforward in time $O(n)$. Now, we update the merge curve set $\mathcal{M}$:

Lemma 7.3 *If H had been the only hole surrounded by M, then no further updating is necessary, since any other merge curves still accurately reflect when the next collision will be with holes that they surround.*

Proof: If p is a point on some other merge curve that surrounds holes that have not yet been hit, then we claim that p will not be affected by the recent change in the WVD due to hitting H: If the new shortest path to p were to use H, this would imply that the path would have to cross M, which is impossible. ∎

If other unhit holes had also been surrounded by M, we can obtain a new merge curve, M', to replace M which surrounds the set $\mathcal{H}'_M$ of all of the holes that M did, except for H: Look at one of the other holes, H', surrounded by M which is adjacent to H in $\mathrm{Vor}(\mathcal{R})$. We can begin our construction of M' at any point along the bisector between H and H', and then proceed by a Shamos-Hoey scan to trace out the locus of all points p such that $d_\Omega(p, \Gamma_Q) - d_\Omega(H, \Gamma_Q) = d(p, \mathcal{H}'_M)$. (Note that $\mathcal{M}$ is not, in general, the bisector between Γ_Q and $\mathcal{R}$, but it is the bisector between the current position of the wavefront and the set $\mathcal{R}$ of unhit holes.)

Initialization takes time $O(n \log n)$, and at each of h iterations we have performed a linear amount of work; thus, we get the claimed complexity of $O(nh + n \log n)$ (and linear space).

Theorem 7.4 *We can solve the Voronoi diagram Problem stated above in time $O(nh + n \log n)$ and space $O(n)$. Then, within these same time and space bounds, we can compute the modified skeleton of a perforated polygon with holes.*

Corollary 7.5 *The min cut and max flow problems for a region Ω with h holes, and for all sources and sinks on the outer boundary, can be solved in time $O(nh + n \log n)$.*

Note the close analogy between the continuous process described above and the operation of the uppermost path algorithm in a planar network. Consider the case in which Γ_P consists of two (open) edges (a source edge and a sink edge), so that we can compare the geometric problem with the network flow problem with one source and one sink on the outer face. Then, we can run the continuous Dijkstra algorithm, starting with V-source T (the "top" boundary chain of Γ_Q), and stopping when we first hit the bottom B. The events in the algorithm correspond to instants when an edge becomes saturated in the application of the uppermost path algorithm to the planar network flow problem. At an event, a hole is hit by the wavefront, and we must examine the capacity of the "augmenting

path" that exists between the current position of the wavefront and the next hole to be hit. The capacity of the augmenting path is simply the distance from where the wavefront is now to where it will be at the next collision with a hole. The set of all wavefronts between collisions make up the streamlines defining the augmenting path. This step corresponds to finding the uppermost path from source to sink in the planar network. Basically, we are "filling" the region Ω with streamlines, from top T to bottom B, until we can fill no more.

The continuous Dijkstra algorithm provides a means of constructing the (weighted) Voronoi diagram (WVD) that one gets by considering Γ_Q to be the V-sources in the weighted region problem that assigns a weight of 0 to the holes (and weight 1 to $int(\Omega) \cup \Gamma_P$ and weight ∞ to Ω^c). From the WVD we can get the modified skeleton (which will be the Voronoi boundary between different components of Γ_Q), and then apply the methods of the preceding section to compute the value, $\mu^* = |\gamma^*|$, of the max flow. We now show that the WVD also gives us a direct method to find an optimal flow field.

Given the WVD, we can obtain an optimal flow field σ^* as follows. For each $p \in \Omega$ such that $f(p) := d_\Omega(p, \Gamma_Q) \leq \frac{1}{2}|\gamma^*|$ and such that there is a unique shortest path from Γ_Q to p, let $\sigma_1(p) = \partial f(p)/\partial y$ and $\sigma_2(p) = -\partial f(p)/\partial x$. Assign $\sigma(p) = 0$ if $f(p) > \frac{1}{2}|\gamma^*|$. (We exercise some care here, since the gradient of f does not exist everywhere; it may not exist along the boundaries of cells in the WVD, but these have only measure zero.) Strang refers to f as a *stream function.* The basic idea is that the level sets of the scalar field f will serve as the streamlines of our flow. These streamlines are precisely the set of all positions of the wavefront during the propagation in continuous Dijkstra.

We claim that the above definition of σ works as an optimal flow field. First, note that σ is indeed divergence-free ($\partial\sigma_1/\partial x + \partial\sigma_2/\partial y = 0$). Next, note that along any shortest path from p to Γ_Q, the field is, by definition, orthogonal to the path. In particular, this is true along the min cut (γ^*); hence, the value of the flow is optimal. Since the size of the WVD is linear, this suffices to prove a linear space bound on a representation of an optimal flow σ^*. It also proves that one can construct an optimal flow within the time complexity of building the WVD.

Note, however, that the vector field defined above is not piecewise-constant, and the boundaries of the WVD are not necessarily line segments. In fact, the streamlines corresponding to the field defined above consists of arcs of circles (centered at vertices) and straight line segments (parallel to edges of Ω), and the boundaries of cells of the WVD consist of straight line segments and parabolic and hyperbolic arcs. It is possible, however,

to "straighten out" the streamlines in a systematic way by applying constructions similar to those given in the proof of Theorem 5.5 within each cell of the WVD, and thereby we obtain again the result that there is a piecewise-constant optimal flow field defined on a polygonal subdivision of size $O(n)$. Note also that some of the streamlines form closed loops, which do not contribute to the net flow from source to sink. These loops correspond to "homogeneous solutions" which can be subtracted without affecting the optimality of σ.

In conclusion, we have in fact given a constructive proof of the following theorem:

Theorem 7.3 *For a maximum flow problem with capacity function $c = 1$, there exists an optimal piecewise-constant flow field σ^* with $|\sigma^*| \in \{0, 1\}$ and such that the region in which $|\sigma^*| = 1$ is a polyhedral region with $O(n)$ vertices.*

8. Flow Through Weighted Regions

We can generalize our results to the case in which c is piecewise-constant over a polygonal subdivision. The higher the value of c, the greater the magnitude of the flow vector that is permitted in the region. We allow c to be 0 or ∞. We can think of the weight assigned to a region as a measure of the *permeability* of the region. Another interpretation might be that c models a third dimension; for example, c might measure the depth of a streambed.

We concentrate here on the case in which there is a single source edge Γ_s and a single sink edge Γ_t, both on the outer boundary Γ_∞. More general cases can be handled by methods similar to those outlined earlier for the uniformly capacitated case. (In particular, the algorithm of [MP] can compute Voronoi diagrams among weighted regions, and can thus be used to compute skeletons similar to those defined in the uniformly capacitated case.)

The min cut problem is to find a path γ from B to T which minimizes the weighted Euclidean length, $\int_\gamma c\ ds$. This problem has been solved in polynomial time $O(n^7 L)$ to yield an ϵ-optimal path [MP]. (Here, L is the precision of the problem instance, including the number of bits in ϵ.) Note that since the min cut problem is solved only to within ϵ-optimality, so will our max flow problem. In particular, the path produced by the algorithm of [MP] can be as much as $(1 + \epsilon)$ times the length of the true shortest path from B to T. Thus, if $\mu = |\gamma_\epsilon|$ is the length of the path produced, we will not be able to push an amount of flow μ through the domain; we will have to be satisfied with trying to get only $\mu/(1 + \epsilon)$ through it.

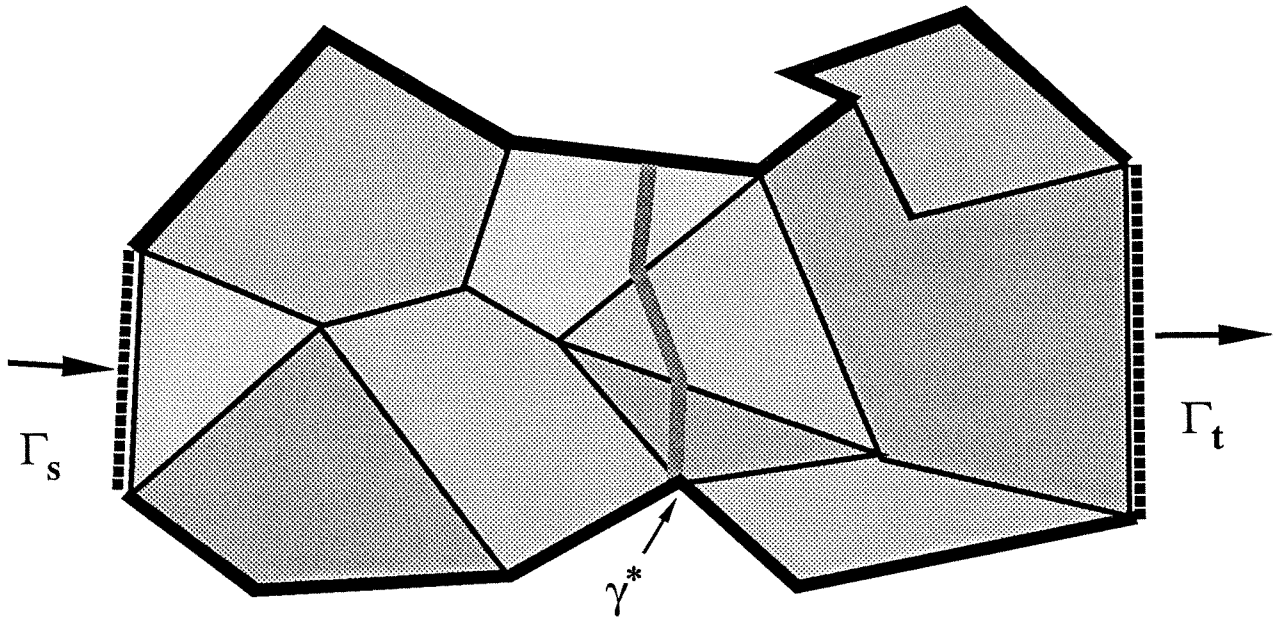

Figure 8.1. A min cut in a weighted polygonal domain.

We can characterize the local behavior of min cuts and streamlines of max flows. The min cut γ^* obeys Snell's Law of Refraction every time it passes through the interior of an edge of the subdivision [MP]. The streamlines of a conservative flow field must obey the following law as they pass through an edge: $\alpha \cos\theta = \alpha' \cos\theta'$, where α and α' are values of c on either side of some boundary edge of the subdivision, and θ and θ' are the angles of "incidence" and "refraction" that the streamlines make with respect to the normal to the boundary edge. See Figure 8.2. We can interpret this law as an "orthogonal" version of Snell's Law, and can see that it follows also from the fact that streamlines are orthogonal to min cuts.

In order to construct an ϵ-optimal flow field σ^*, we can proceed as follows. First, when running the algorithm of [MP], we actually end up finding a tree of $O(n^3)$ polygonal paths (one per event) from B (which is taken to be the "source" in the shortest path algorithm) to vertices and "event points" of the subdivision, each path having at most $O(n^2)$ bend points at edges or vertices of the original subdivision of Ω. We also end up with a subdivision (Shortest Path Map) of Ω which gives for each point $p \in \Omega$ the "root" of p in an ϵ-shortest path from B to p. The root is either the last vertex or the last "critical edge" (see [MP]) in the path from B to p; we concentrate here on the case of a vertex root. The set of all

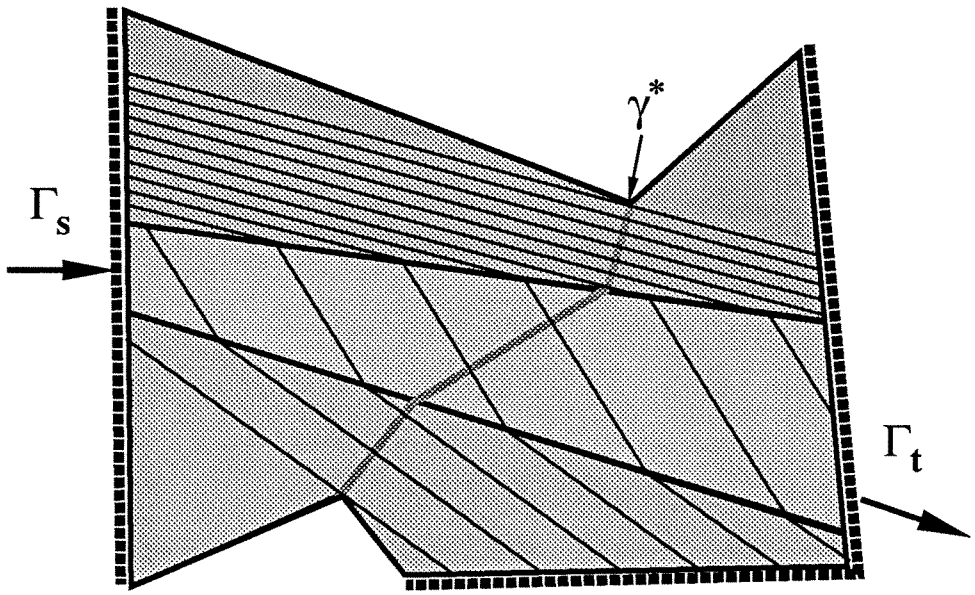

Figure 8.2. Streamlines through weighted regions.

points p with a common root r is a "cell" C, and C is partitioned by the set P of event paths which pass through r. We could define σ as $(\partial f(p)/\partial y, -\partial f(p)/\partial x)$, for points $p \in C$ with $f(p) \leq |\gamma_\epsilon|/(1+\epsilon)$, where the function $f(p)$ measures the (weighted) distance from r to p plus the (weighted) distance from B to r (i.e., $f(p)$ is the (weighted) distance from B to p). However, in order to get a piecewise-constant field over a polygonal subdivision, we can apply a technique similar to that given in the construction for Theorem 5.5. Basically, we build a field within C whose vectors σ are defined in terms of the gradient of $f(p)$ (exactly as above) for those points p along paths of the discrete set P. Then, in order to extend the definition of σ into the regions of C that lie between consecutive paths of P, we simply follow the streamlines of the field that are implied by the values of σ on P, crossing into regions of different c values according to the local optimality criterion for streamlines described above (and depicted in Figure 8.2). Where the streamlines corresponding to two consecutive paths of P meet one another inside cell C, we get a polygonal boundary (an "elbow" in the "pipe" of parallel streamlines). Thus, we end up with a refinement of the cell C according to these new polygonal boundaries and the set of paths P. Within each region of this new subdivision, we define the field to be at maximum magnitude and parallel to the streamlines in that region. In this way, we can obtain a representation of an ϵ-optimal flow which is piecewise-constant over a polygonal subdivision within the

same time and space bounds that the algorithm of [MP] requires to solve the shortest path problem. (We note the close similarity between this method of using the shortest path tree to construct max flows, and that of Hassin [Ha], who showed a similar construction for the case of max flow in (s,t)-planar graphs.) While this shows a polynomial bound, it clearly leaves open the question of finding a simple and practical algorithm.

10. Conclusion

We have addressed the problem of computing min cuts and max flows in the continuum from the point of view of algorithmic complexity. Several other extensions and open questions are suggested by our research.

1). Our results can be applied to get an algorithm for maximum *integer* flows in which one wishes to route the maximum number of wires from source edges to sink edges, with the restriction that no two wires can lie closer than a unit distance from each other. In the simple case of a single source and single sink, all we have to do is round down (to the nearest integer) the lengths of the edges in the critical graph (used in solving $0/1/\infty$ path problems) before computing a min cut. Construction of a flow field then gives an optimal routing for the wires.

2). We could consider allowing point sources inside Ω. Another generalization would be to allow the capacity function c to also depend on the *direction* of flow at a point. A generalization of possible relevance to pipe routing applications would be to restrict the streamlines of the flow to have a bounded radius of curvature.

3). Consider the problem in which one wants to minimize the area of Ω for which $\sigma \neq 0$. We then get the *min-area max flow* problem. (Strang [St] refers to these as "optimal design" problems.) Another criterion might be to find a max flow field that minimizes the length of the longest streamline.

4). Given a triangulated simple polygon Ω, can we find the min cut in linear time?

5). Can one achieve an $o(n^2)$ algorithm for finding max flows in a simple polygon with holes? At the time of this writing, our best algorithm for the general case is $O(nh+n\log n)$ (the same bound that is known for the shortest path problem obstacles [RS]).

6). For the general form of the flow problem in which there are many sources and sinks on different holes, we suspect that there is a more efficient solution than the naive one given here.

7). Can an arbitrary shortest path problem among 0/1 weighted regions be mapped into a max flow problem (at a cost of at most $O(n\log n)$)? A similar question arises for

arbitrarily weighted regions.

8). What connection is there between shortest paths in 0/1 weighted regions and shortest paths in $1/\infty$ weighted regions? Are these problems equally hard?

9). How does one generalize the flow problem to higher dimensions? Min cuts in three dimensions correspond to minimum area surfaces spanned by a closed Jordan curve (Plateau's problem). Such problems are hard to solve in general. It may be possible to solve the special case in three dimensions for orthohedral spaces under the restriction that the flow field be parallel to one of the coordinate axes at all times.

10). An interesting variation on the max flow problem is the one that restricts the orientations of σ. For instance, we may be interested in routing wires which must everywhere be parallel to either the x- or the y-axis (or, more generally, constrained to be from among a given set of orientations). Our results can be extended to yield quadratic-time algorithms for these cases by using the L_∞ metric in place of the Euclidean metric. We expect that the results of [Mi3] or of [CKV] yield an $O(n \log^2 n)$ or $O(n \log n)$ algorithm for this case (and more generally, for any set of fixed orientations).

Finally, our research suggests that further investigations should be conducted into the mapping between discrete graph algorithms and their continuous analogues in geometry.

Acknowledgement

I am very grateful to the referees for a careful reading, for many helpful suggestions, and for correcting some bugs in the original draft. I would like to thank Christos Papadimitriou for suggesting this class of problems, and for helpful discussions. I would also like to thank Esther Arkin and Samir Khuller for many useful discussions on the problems of max flows in simple polygons. The author is partially supported by NSF Grants IRI-8710858 and ECSE-8857642 and by a grant from Hughes Research Laboratories, Malibu, CA. Equipment used in the preparation of this report was furnished by the Cornell Computational Optimization Project.

11. Bibliography

[AAGHI] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility of Disjoint Polygons", *Algorithmica*, **1**, pp. 49-63, 1986.

[CKV] K. Clarkson, S. Kapoor, and P. Vaidya, "Rectilinear Shortest Paths through Polygonal Obstacles in $O(n \log^2 n)$ Time", *Proc. Third Annual ACM Conference on Computational Geometry*, Waterloo, Ontario, 1987, pp. 251-257.

[EK] J. Edmonds and R.M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", *J. ACM*, **19**, No. 2 (1972), pp. 248-64.

[FF] L.R. Ford Jr. and D.R. Fulkerson, "Maximal Flow Through a Network", *Canadian Journal of Mathematics*, **8** (1956) pp. 399-404.

[Fo] S.J. Fortune, "A Sweepline Algorithm for Voronoi Diagrams", *Algorithmica* **2** (1987), 153-174.

[Ga] Z. Galil, "A New Algorithm for the Maximal Flow Problem", *Proc. 19th Symp. on Foundations of Computer Science*, IEEE (October 1978), pp. 231-45.

[GMMN] L. Gewali, A. Meng, J.S.B. Mitchell, and S. Ntafos, "Path Planning in $0/1/\infty$ Weighted Regions With Applications", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, June 6-8, 1988, pp. 266-278.

[GH] L.J. Guibas and J. Hershberger, "Optimal Shortest Path Queries in a Simple Polygon", *Proc. Third Annual ACM Conference on Computational Geometry*, Waterloo, Ontario, 1987, pp. 50-63.

[GS] L. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", *ACM Trans. Graphics* **4** (1985), 74-123.

[GT] A.V. Goldberg and R.E. Tarjan, "A New Approach to the Maximum Flow Problem", In *Proc. 18th ACM Symposium on Theory of Computing*, pp. 136-146, 1986.

[Ha] R. Hassin, "Maximum Flow in (s,t)-Planar Networks", *Information Proc. Letters* **13**, No. 3, 1981, p. 107.

[Hu] T.C. Hu, *Integer Programming and Network Flows*, Addison-Wesley, Reading, MA, 1969.

[Ir] Iri, *Survey of Mathematical Programming*, edited by A. Prékopa, North-Holland, Amsterdam, 1979.

[IS] A. Itai and Y. Shiloach, "Maximum Flow in Planar Networks", *SIAM J. Comput.*, **8**, No. 2, May 1979, pp. 135-50.

[Ki] D. Kirkpatrick, "Efficient Computation of Continuous Skeletons", *Proceedings of the 14th IEEE Symposium on Foundations of Computer Science*, pp. 18-27, 1979.

[Le] D.T. Lee, "Medial Axis Transformation of a Planar Shape", *IEEE Trans. Pattern Anal. Mach. Intel.*, 4 (1982), pp. 363-369.

[Mi1] J.S.B. Mitchell, "Planning Shortest Paths", PhD Thesis, Department of Operations Research, Stanford University, August, 1986. (Available as Research Report 561, Artificial Intelligence Series, No. 1, Hughes Research Laboratories, Malibu, CA.)

[Mi2] J.S.B. Mitchell, "Shortest Paths Among Obstacles, Zero-Cost Regions, and Roads", Technical Report No. 764, Department of Operations Research, Cornell University, 1987.

[Mi3] J.S.B. Mitchell, "Shortest Rectilinear Paths Among Obstacles", Technical Report No. 739, School of Operations Research and Industrial Engineering, Cornell University, April, 1987.

[Mi4] J.S.B. Mitchell, "On Maximum Flows in Polyhedral Domains (extended abstract)", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, June 6-8, 1988, pp. 341-351.

[MMP] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou, "The Discrete Geodesic Problem", *SIAM Journal on Computing*, **16**, No. 4, pp. 647-668, August, 1987.

[MP] J.S.B. Mitchell and C.H. Papadimitriou, "The Weighted Region Problem", Technical Report, Department of Operations Research, Stanford University, 1985. (Extended

abstract appears in: *Proc. Third Annual ACM Conference on Computational Geometry*, Waterloo, June 1987.) To appear: *J. of ACM.*

[Pa] C. H. Papadimitriou, "Shortest-Path Motion", *Proc. Foundations of Software Technology and Theoretical Computer Science*, New Delhi, India, December 18-20, 1986. In *Lecture Notes in Computer Science* (G. Goos and J. Hartmanis, eds.), Springer-Verlag.

[RS] J.H. Reif and J.A. Storer, "Shortest Paths in Euclidean Space with Polyhedral Obstacles", Technical Report CS-85-121, Computer Science Department, Brandeis University, April, 1985.

[Se] R. Seidel, "Constrained Delaunay Triangulations and Voronoi Diagrams with Obstacles", Manuscript, 1988.

[SH] M.I. Shamos and D. Hoey, "Closest-Point Problems", *Proc. 16th IEEE Symp. on Foundations of Computer Science*, 1975, pp. 151-162.

[ST] D.D.K. Sleator and R.E. Tarjan, "A Data Structure for Dynamic Trees", *J. Comput. System Sci.* **26** (1983) 362-391.

[St] G. Strang, "Maximal Flow Through A Domain", *Mathematical Programming*, **26** (1983), pp. 123-143.

[We] E. Welzl, "Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time", *Information Processing Letters*, Vol. 20 (1985), pp. 167-171.

[Ya] C.K. Yap, "An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments", *Discrete Comput. Geom.*, **2** (1987), pp. 365-393.