Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

OPTIMUM SOLUTION OF THE DISCRETE PLATEAU PROBLEM (Revised)

T. C. Hu
A. B. Kahng
G. Robins

December 1992
CSD-920006

Optimum Solution of the Discrete Plateau Problem

T. C. Hu, Andrew B. Kahng† and Gabriel Robins‡

UCSD Computer Science and Engineering Dept., La Jolla, CA 92093-0114 Tel. 619-534-3854
† UCLA Computer Science Dept., Los Angeles, CA 90024-1596 Tel. 310-206-7073
‡ Univ. of Virginia Computer Science Dept., Charlottesville, VA 22903-2442 Tel. 804-982-2207

Abstract

We give the first efficient, optimal solution to a well-studied discrete version of the "Plateau problem" on minimal surfaces. Our approach is based on the use of network flows to find minimum-cost *slabs*, which intuitively correspond to the notion of minimal "surfaces" of prescribed thickness. An efficient implementation has been used to exhibit minimal surface solutions for a variety of problem instances.

OR/MS Subject Classification:
Networks: flow algorithms, optimal minimum-surface solutions
Mathematics: discrete Plateau problem, optimal algorithm

Given a contour in three dimensions, the "Plateau problem" is to find the surface of minimum area that spans the contour. The Plateau problem is part of the extensive field of minimal surfaces, which originated with the development of the multidimensional calculus of variations [6] [9] [10]. While the study of minimal surfaces can be traced to Lagrange (1768), it was J. Plateau (1801-1883) who conducted the first extensive investigations, using wire loops and soap films to physically model minimal spanning surfaces [25]. Subsequently, many mathematicians of the nineteenth and twentieth centuries, including Riemann, Weierstrass, and Schwarz, contributed to the theory of minimal surfaces [27], culminating with the discovery of general analytic solutions by Douglas [7] and Radó [26] in the 1930's.

Practical applications of the Plateau problem abound. In orthopedic surgery or dentistry, the shaping of prostheses involves a minimum surface computation to improve contact and to reduce the risk of rejection or infection. Minimum surface computations also arise in the design of packaging for consumer goods; the analogous formulation in two dimensions corresponds to optimum path planning for robotics, or rapid deployment in military applications [20].

A surface has minimal area if and only if it has zero mean curvature at each point, but this characterization is non-constructive. The problem is subtle: (i) a minimal surface may self-intersect, (ii) a given contour can bound numerous different surfaces of distinct topologies, and (iii) a very slight modification to the boundary contour can cause an enormous change in the corresponding minimal surface topology [6] [8]. Finding a minimal surface spanned by a given contour typically entails solution of a system of partial differential equations. In many instances, analytic solutions are known to exist but remain virtually impossible to find; thus, solutions to specific cases have been individually discovered and proved over the last two centuries [9] [24] [28].

This paper gives a new, *constructive* approach which solves a well-studied class of discrete Plateau instances using network flow techniques. We generalize standard formulations in that we do not search for a minimal (zero-thickness) surface; rather, we seek a minimal *slab* having some prescribed positive *thickness* d (this informal terminology should evoke the picture of, e.g., a thick orange peel). Our algorithm obtains a minimum-cost slab having thickness everywhere of at least d, where cost is defined to be the total weighted volume of the slab with respect to an arbitrary weight function defined over $\Re^3$.

Our solution diverges from the usual finite-element based approach, and instead employs a more direct combinatorial technique involving network flows [11]. The crucial observation is that a minimum-cost slab which *spans* a set of locations (e.g., the set of locations on the given contour) is also a minimum-cost cut-set which *separates* two other locations. Given this observation, we efficiently obtain *optimal* minimum surface solutions by computing maximum flows to exploit the duality between spanning sets and separating sets [21] [22]. Several important features of our method are as follows:

1. First, we depart from traditional methods in allowing the minimum surface to possess a positive thickness; this yields added realism in that all physical objects will have such "dimension".

2. Second, all previous methods define the cost of the surface to be its area; we generalize the minimum surface computation within an arbitrarily weighted space (as opposed to a

uniformly weighted volume). This allows our method to produce solutions which trade off surface area in favor of occupying "cheaper" regions of the space (e.g., stronger or healthier regions within a bone). Again, this adds practicality to the approach.

3. Third, our approach guarantees a *globally optimal* solution to the discrete Plateau problem that we formally define below. In contrast, all previous methods involve variational techniques which can only guarantee to converge to locally optimal solutions. Moreover, a major attraction is that the algorithm gives the first *efficient*, polynomial-time solution to the discrete Plateau formulation below. Our algorithm can be implemented to run in $O(|N|^2)$ time where $|N|$ is the number of nodes in a discrete mesh representation of the space, and experimental results confirm that we can efficiently find minimal surface solutions.

4. Finally, other advantages include: (i) the restriction of our method to two dimensions provides the first efficient, optimal solution for the minimum-width path planning problem in arbitrarily weighted terrains [20]; (ii) the method extends to address Plateau's minimum-surface formulation in arbitrarily high dimension; and (iii) the intrinsic regularity and geometry of the underlying space yields a layered, bounded-degree network representation, resulting in possible added efficiency of our network-flow approach.

1 Problem Formulation

In its simplest form, the Plateau problem is as follows: given a Jordan curve Γ^* in $\Re^3$, find a surface D^* of minimum area having boundary Γ^*. This formulation is difficult to address due to its generality, and thus in the remainder of the paper we deal with the well-studied class of instances first described by Radó [26], for which (1) the orthogonal projection Γ of the given boundary Γ^* onto the xy-plane is simple (i.e., non self-intersecting), and (2) the solution admits a functional representation $z = f(x, y)$, where f is continuous and has domain equal to the subset of the xy-plane bounded by Γ. The first condition specifies that the planar projection of the boundary curve forms a simple closed loop, while the second condition specifies that the minimal surface solution can be projected onto the interior of this planar loop without "overlap".

Radó's two conditions give us the "restricted Plateau Problem" [26]: given a Jordan curve Γ^* in $\Re^3$ whose projection Γ onto the xy-plane is a deformed (i.e., homeomorphic to a) circle, find a surface D^* (having functional representation $z = f(x, y)$) of minimal area with boundary Γ^* (Figure 1). Note that in our discussion, we adopt the convention of using starred letters to denote three-dimensional objects (e.g., a contour Γ^*, a surface D^*), and unstarred letters to denote their respective projections (e.g., a boundary Γ, a region D); a glossary of notation is given at the end of the paper.

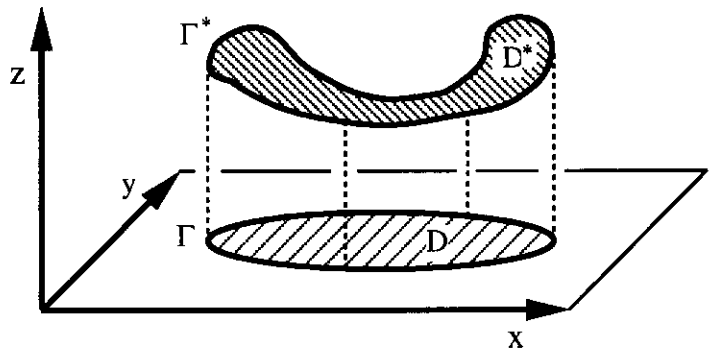

Figure 1: A surface D^* and its bounding contour Γ^*, as well as the corresponding projected region D and its boundary Γ.

A recent trend has been to solve instances of the Plateau problem empirically via numerical methods. Wilson [32] discretized the problem by approximating the minimal surface using triangulations. Other methods for the numerical solution of a restricted version of the Plateau problem were given by Greenspan [15] [16], who used a combination of difference and variational methods, and by Concus [4], who used a finite difference approach. A more general numerical method is the finite element scheme given by Hinata et al. [18]. More recent efforts include those of Tsuchiya, who gave methods for approximating minimal surfaces in parametric form, again using a finite element approach [29]. Tsuchiya's work is noteworthy for demonstrating convergence under certain conditions [30] [31], but this guaranteed convergence is not necessarily to the global optimum solution. It should also be noted that most of these previous works do not address the issue of computational efficiency; those methods which are "efficient" in the size of the discretized problem representation again cannot guarantee convergence to the global optimum solution.

We now formally develop the discrete Plateau problem formulation which satisfies Radó's conditions, and which we solve in Section 2 below. The development will focus on the duality between connection and separation which motivates our network flow solution.

Definition: A *region* is a simply connected, compact subset of $\Re^2$.

Given any three-dimensional point set $P^* \subset \Re^3$, its *projection* is the set of all points in the xy-plane with x and y coordinates equal to those of some point in P^*; i.e., $proj(P^*) = \{(x,y) \mid \exists z \ni (x,y,z) \in P^*\}$. We naturally extend this idea of projection to apply to any function $f(x,y)$ of two variables, by considering the function f to be the set/relation $\{(x,y,f(x,y)) \mid x,y \in \Re,$ where $f(x,y)$ is defined$\}$; thus, $proj(f)$ is simply the domain of the function f. With this in mind, we capture Radó's class of minimum-surface instances by defining a *boundary* to be a Jordan curve Γ in the plane, and by defining a *contour* Γ^* to be a three-dimensional embedding of a Jordan curve which has the required functional representation:

Definition: A *contour* Γ^* is the set of points $\{(x,y,f(x,y)) \in \Re^3 \mid (x,y) \in \Gamma\}$ where f is a continuous real function $f : \Gamma \rightarrow \Re$ over some boundary Γ.

By the Jordan curve theorem [5], any boundary Γ partitions the plane into three mutually disjoint sets: Γ itself; its interior $int(\Gamma)$; and its exterior $ext(\Gamma)$. Thus, a contour Γ^* is a three-dimensional embedding of a deformed circle, and the orthogonal projection of Γ^* onto the xy-plane is the boundary Γ of some region $D = int(\Gamma) \cup \Gamma$.

In view of the functional representation, any surface D^* that spans Γ^* will satisfy $proj(D^*) = D$, i.e.:

Definition: A *surface* D^* is the set of points $\{(x,y,f^*(x,y)) \in \Re^3 | (x,y) \in D\}$ where f^* is a continuous real function $f^* : D \rightarrow \Re$ defined over some region D in the xy-plane.

Any contour Γ^* induces an infinite family of distinct spanning surfaces. We may view the continuous surface function $f^* : D \rightarrow \Re$ as an extension of the contour function $f : \Gamma \rightarrow \Re$; i.e., $f^*(x,y) = f(x,y)$ for all $(x,y) \in \Gamma \subset D$ (recall Figure 1).

Given a surface D^*, define the *cylinder* $cyl(D^*)$ to be the set of all points directly above

or below D^*; in other words, $cyl(D^*) = \{(x, y, z) \mid (x, y) \in proj(D^*), z \in \Re\}$. We may extend the *cyl* function to contours: $cyl(\Gamma^*) = cyl(D)$, where as usual $D = proj(\Gamma^*) \cup int(proj(\Gamma^*))$. Intuitively, any surface D^* partitions $cyl(D^*)$ into three mutually disjoint subsets:

1. the points lying above D^*, denoted by $D_t^* = \{(x, y, z) \mid (x, y) \in proj(D^*), z > f^*(x, y)\}$;
2. the points of D^* itself, $\{(x, y, f^*(x, y)) \mid (x, y) \in proj(D^*)\}$; and
3. the points lying below D^*, denoted by $D_b^* = \{(x, y, z) \mid (x, y) \in proj(D^*), z < f^*(x, y)\}$.

In other words, the surface D^* *separates* D_b^* from D_t^*. In practice, we truncate both the top and the bottom of the cylinder $cyl(\Gamma^*)$ "far enough" above and below D^*, respectively, so that both D_t^* and D_b^* are bounded sets. We then define a weight function $w : cyl(\Gamma^*) \rightarrow \Re^+$ such that each point $s \in cyl(\Gamma^*)$ has a non-negative weight $w(s)$.

We now generalize our formulation to allow a prescribed non-zero thickness to the separating surface D^* (recall the orange peel suggested above). From this, we will establish the relationship between the concept of *d-separation* [14] [19] and this thickness-d requirement.

Definition: Given a contour Γ^*, a *d-separating slab* $\hat{D}^* \subset cyl(D^*)$ is a superset of some surface D^* with Γ^* as the bounding contour of D^*, such that any point of $D_t^* - \hat{D}^*$ is at distance d or greater from any point of $D_b^* - \hat{D}^*$.

This is illustrated in Figure 2. We say that $\hat{D}^*$ is a *minimal d-separating slab* if no subset of $\hat{D}^*$ satisfies the preceding definition. The *cost* of a slab is defined to be the integral of the weight function w over the volume of the slab. Because the weight function is non-negative and because we are interested in minimum-cost slabs, our discussion henceforth will refer only to minimal d-separating slabs. Given $d > 0$, the thickness-d Plateau problem is stated as follows:

Thickness-d Plateau Problem: Given a contour Γ^*, a weight function $w : cyl(\Gamma^*) \rightarrow \Re^+$, and a thickness $d > 0$, find a d-separating slab $\hat{D}^* \subset cyl(\Gamma^*)$ which has minimum total cost.

While the formulation specifies an arbitrary weight function that must be integrated over the volume of the slab to yield a total cost, in practical applications the space is often discretized

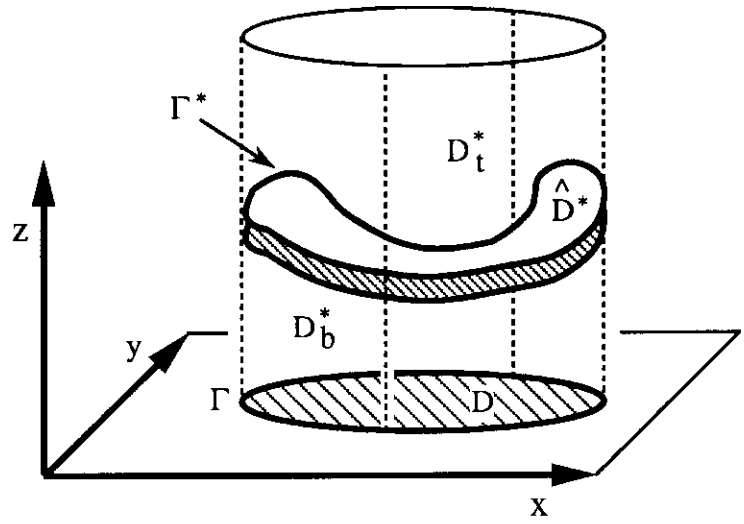

Figure 2: a d-separating slab $\hat{D}^*$ relative to a given contour Γ^*.

relative to a given fixed grid or sampling granularity. This is a standard assumption with numerical approaches to the Plateau problem (e.g., [18] [29] [32]), and in the present work we also adopt this assumption of a fixed grid representation. With such a discrete version of the thickness-d Plateau problem, the cost of a slab is naturally defined to be the sum of the weights of the grid points contained in it. The notion of d-separation also naturally extends to the discrete grid:

Definition: Given a cylinder S, a *discrete* d-separating slab $\tilde{D}^*$ in the gridded space $\tilde{S}$ is the set of gridpoints of $\tilde{S}$ contained in some d-separating slab $\hat{D}^*$ in S (Figure 3).

As in the continuous case, a discrete d-separating slab partitions the rest of the gridpoints into two sets, such that each gridpoint in one set is at least distance d away from any gridpoint in the other set. A discrete d-separating slab is minimal if no subset of it satisfies the preceding definition. We now have:

Discrete Plateau Problem: Given a weighted gridded space $\tilde{S}$ with boundary $B \subset \tilde{S}$, a contour Γ^* on the boundary of $\tilde{S}$, and a thickness $d > 0$, find a discrete d-separating slab $\tilde{D}^* \subseteq \tilde{S}$ which contains Γ^* and has minimum total cost.

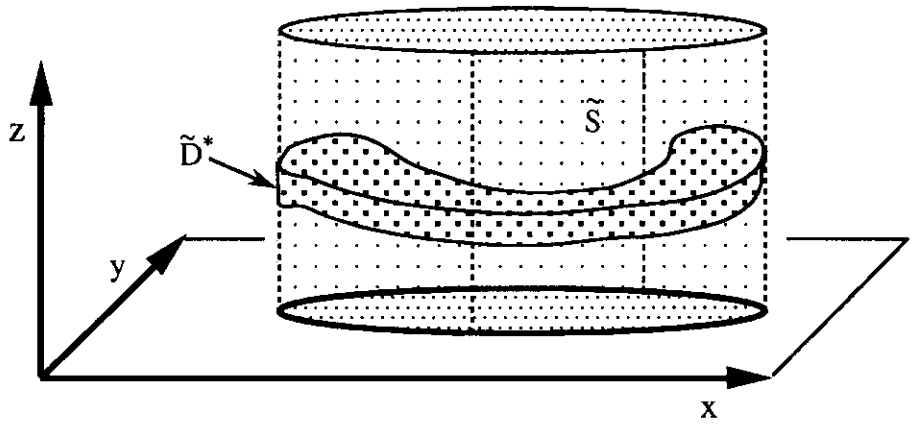

Figure 3: A discretized representation $\tilde{S}$ of a space S, and a discrete d-separating slab $\tilde{D}^*$ in $\tilde{S}$. Note that $\tilde{D}^*$ is the set of lattice points contained in the continuous d-separating slab $\hat{D}$ in S.

We now use a network flow approach to develop an efficient, optimal algorithm for the discrete Plateau problem. Note that the granularity of the grid is intrinsic to the problem instance, and that our method will optimally solve any discrete Plateau instance. Intuitively, if the granularity quantum of the grid is near zero, the solution of the discrete Plateau problem instance will also solve the corresponding continuous thickness-d Plateau problem instance.

2 A Solution Using Network Flow

To solve the discrete Plateau problem, we use ideas from network flows in continua [19] and exploit the duality between a minimum cut and a maximum flow. The overview of our solution is as follows:

1. Discretize the volume of the cylinder induced by the given contour (i.e., consider only the lattice points of $cyl(\Gamma^*)$ with respect to a given resolution).

2. Create a d-connected mesh network over $cyl(\Gamma^*)$ by connecting each lattice point to all other lattice points within distance d; this guarantees that any separating set of nodes will have a minimum thickness d (we use the obvious one-to-one correspondence between nodes of the network and lattice points of the cylinder).

3. Connect a source node s (sink node t) to all nodes on the surface of the cylinder that lie below (above) the contour.

4. Use a maximum flow algorithm to compute a maximum s-t flow in the resulting network.

5. A maximum s-t flow specifies a minimum cut through the cylinder which separates s from t, and this minimum cut corresponds to a minimal thickness-d slab that contains the given contour.

Before describing each of these steps in greater detail, we first review several key concepts from the theory of network flows [11] [23]. A *flow network* $\eta = (N, A, s, t, c, c')$ is a directed graph with node set N; a set of directed arcs $A \subseteq N \times N$; a distinguished source $s \in N$ and a distinguished sink $t \in N$; an *arc capacity* function $c : A \rightarrow \Re^+$ which specifies the capacity $c_{ij} \geq 0$ of each arc $a_{ij} \in A$; and a *node capacity* function $c' : N \rightarrow \Re^+$ which specifies the capacity $c'_i \geq 0$ of each node $n_i \in N$. (To handle undirected graphs, we may replace each undirected arc a_{ij} by two directed arcs a_{ij} and a_{ji}, each having capacity c_{ij}.)

A *flow* in η assigns to each arc a_{ij} a value ϕ_{ij} with the constraint that $0 \leq \phi_{ij} \leq c_{ij}$. An arc a_{ij} is *saturated* if $\phi_{ij} = c_{ij}$. We insist on *flow conservation* at every node except s and t, and we require that the flow through each node does not exceed the capacity of that node:

$$\sum_i \phi_{ij} \;=\; \sum_k \phi_{jk} \;\leq\; c'_j \qquad j \neq s, t$$

A node n_i is called *saturated* if $\sum_i \phi_{ij} = c'_j$. Since flow is conserved at every node, the total amount of flow from the source must be equal to the total flow into the sink, and we call this quantity the *value* Φ of the flow:

$$\Phi = \sum_i \phi_{si} \;=\; \sum_j \phi_{jt}$$

A flow with the maximum possible value is called a *maximum flow*. An s-t *cut* in a network is a set (N', A') of nodes $N' \subseteq N$ and arcs $A' \subseteq A$ such that every path from s to t uses at least one node of N' or at least one arc of A'. The *capacity* $c(N', A')$ of a cut is the sum of the

capacities of all nodes and arcs in the cut. A classical result of linear programming states that the maximum flow value is equal to the minimum cut capacity; this is the *max-flow min-cut* theorem [11]:

Theorem: Given a network $\eta = (N, A, s, t, c, c')$, the value of a maximum s-t flow is equal to the minimum capacity of any s-t cut. Moreover, the nodes and arcs of any minimum s-t cut are a subset of the saturated nodes and saturated arcs in some maximum s-t flow. □

Recall our earlier observation that any slab D^* will separate, or cut, D_t^* from D_b^*. In particular, a slab D^* with small cost will correspond to a cut between a node $s \in D_b^*$ and a node $t \in D_t^*$ with a small cost (capacity). Since a subset of the saturated nodes/arcs in some maximum s-t flow will yield this s-t cut, it is natural to derive the desired minimal slab via a maximum flow computation in an appropriately capacitated network.

Our first step towards this goal is to transform an instance of the discrete Plateau problem into an instance of network flow, by: (i) superimposing a discrete grid on $cyl(\Gamma^*)$, (ii) assigning capacities to nodes in the grid according to the weight function $w : cyl(\Gamma^*) \rightarrow \Re^+$, and (iii) converting the grid into a mesh network η by mapping gridpoints to capacitated nodes of η and then adding infinite-capacity arcs to join these nodes into a mesh.

To ensure that any s-t cut in the mesh created by (iii) will have the required thickness, we define the *d-neighborhood* of a node v to be the set of all nodes at distance d or less from v. We then connect each node to all nodes in its d-neighborhood with infinite-capacity arcs, where d is the prescribed slab thickness. An illustration of this construction for $d = 2$ is given in Figure 4.

Finally, we introduce a source node s and a sink node t, connecting them respectively to the nodes of the boundary $B \subset \tilde{S}$ lying "below" Γ^* and to the nodes of B lying "above" Γ^*. This forces any st-separating cut (which will correspond to the desired d-separating slab) to contain the given contour nodes Γ^* lying on the boundary B of the gridded space. In other words, we force the minimum slab to span the contour Γ^*. This completes the outline of our transformation; Figure 5 gives a high-level illustration of the construction.

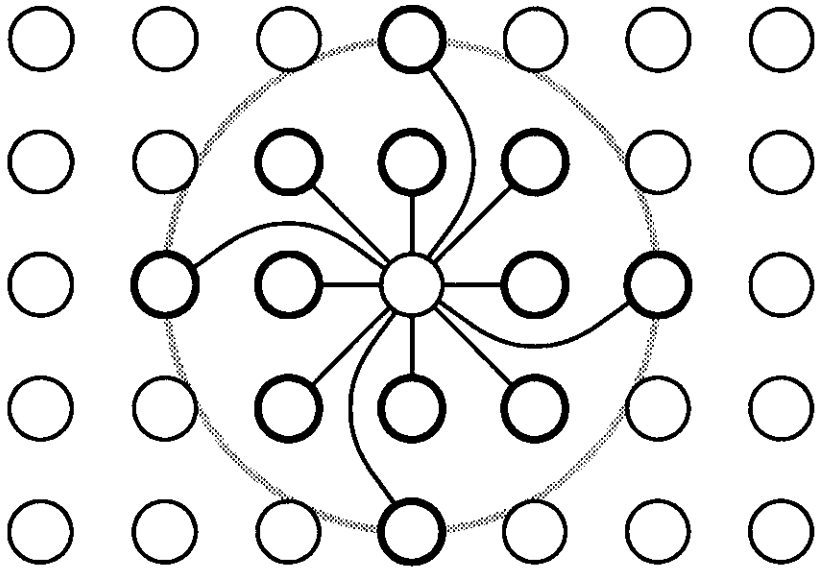

Figure 4: A node and its d-neighborhood; here $d = 2$.

The resulting *d-connected* network has two useful properties. First, a minimum s-t cutset in this network will consist only of nodes. This is because all arcs have infinite capacities, while there exist cuts with finite cost since all node capacities are finite. Second, any nodeset that cuts this network must correspond to the set of lattice points in a discrete d-separating slab; this property follows from the d-connectivity of the mesh.

Finally, observe that up to this point, we have converted a discrete Plateau problem instance to a maximum flow instance on an undirected, *node-capacitated* network. However, network flow algorithms typically assume that the input is an *arc-capacitated* network (with infinite node capacities). Therefore, in order to use a standard maximum flow algorithm on our network, we must transform an instance having both node and arc capacities into an equivalent arc-capacitated maximum flow instance. To accomplish this, we use the standard device of splitting each node $v \in N$ with weight w_v into two unweighted nodes v' and v'', then introducing a directed arc from v' to v'' with capacity w_v. Then, we transform each arc $(u, v) \in A$ of the original network into two infinite-capacity directed arcs (u'', v') and (v'', u'). Note that each arc (v', v'') of the resulting directed network will, when saturated, contribute the original node weight w_v to the minimum cut value. This transformation is illustrated in Figure 6.

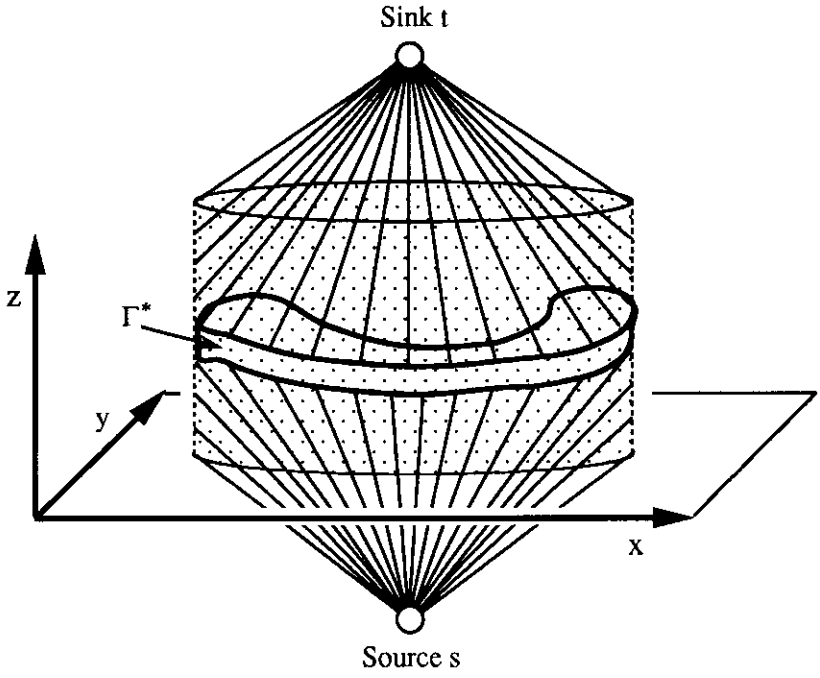

Figure 5: A discrete Plateau problem instance transformed into a network flow instance.

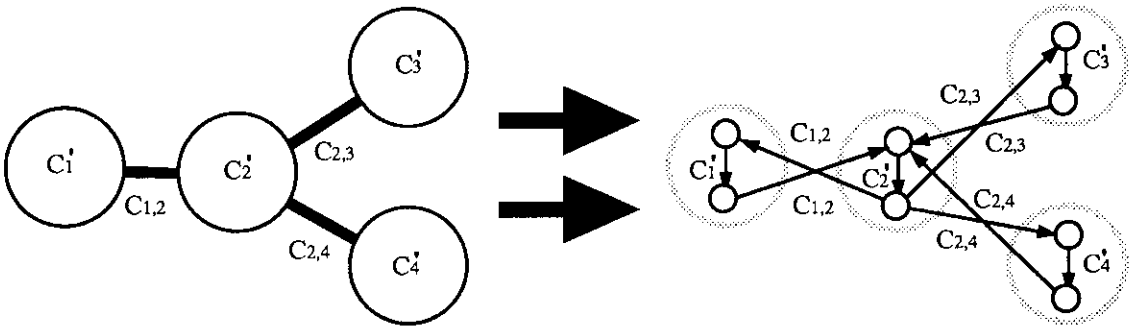

Figure 6: Transformation of a node- and arc-capacitated flow network to an arc-capacitated flow network.

The overall size of the network increases by only a constant factor via this last transformation, i.e., the final directed arc-capacitated network will have only $2|N|$ nodes and $|N|+2|A|$ arcs. This implies that the maximum flow computation in the transformed network will be asymptotically as fast as in the original node-capacitated network. A formal summary of our algorithm, which

we call the Disc_Plateau algorithm, is given in Figure 7.

Algorithm: Disc_Plateau
Input: contour Γ^* node weight function $w : cyl(\Gamma^*) \rightarrow \Re^+$ thickness $d > 0$ grid size g
Output: A minimal d-separating slab $\tilde{R}^*$ with boundary contour Γ^*
Create a d-connected mesh network G of grid size g over $cyl(\Gamma^*)$
Set node capacities of G according to weight function w
Set arc capacities of G to infinity
Set all boundary node capacities to ∞
Transform node-capacitated network G into arc-capacitated network η
Create source node s and sink node t in η
Connect s to boundary nodes $(x, y, z) \in cyl(\Gamma^*) \ni z < \Gamma^*(x, y)$
Connect t to boundary nodes $(x, y, z) \in cyl(\Gamma^*) \ni z > \Gamma^*(x, y)$
Set capacities of all arcs adjacent to s or t to ∞
Compute a maximum s-t flow in η
Output all nodes incident to arcs in a minimum cut of η

Figure 7: Algorithm Disc_Plateau finds a d-separating slab of minimum cost in an arbitrarily weighted discrete space, i.e., an optimal solution to the discrete Plateau problem. The time complexity of the algorithm is dominated by the maximum flow computation.

The max-flow min-cut theorem [11] and the existence of polynomial-time algorithms for maximum flow together imply the following:

Theorem: Algorithm Disc_Plateau outputs an optimal solution to the discrete Plateau problem in time polynomial in the size of the gridded space $\tilde{S}$. □

3 Correctness Issues

There exist cases where the minimal surface exits the cylinder $cyl(\Gamma^*)$ [2], and since our method is confined to yield an optimal solution *inside* $cyl(\Gamma^*)$, its solution will not correspond to the global optimum; a typical construction is given in Figure 8. However, in such examples, although the contour Γ^* projects to a deformed circle, the true minimal surface does not admit a functional representation $D^* = \{(x, y, f^*(x, y)) \in \Re^3 | (x, y) \in D\}$ as required by the second condition defining Radó's class, and thus our methodology does not claim to address such examples in the

first place.

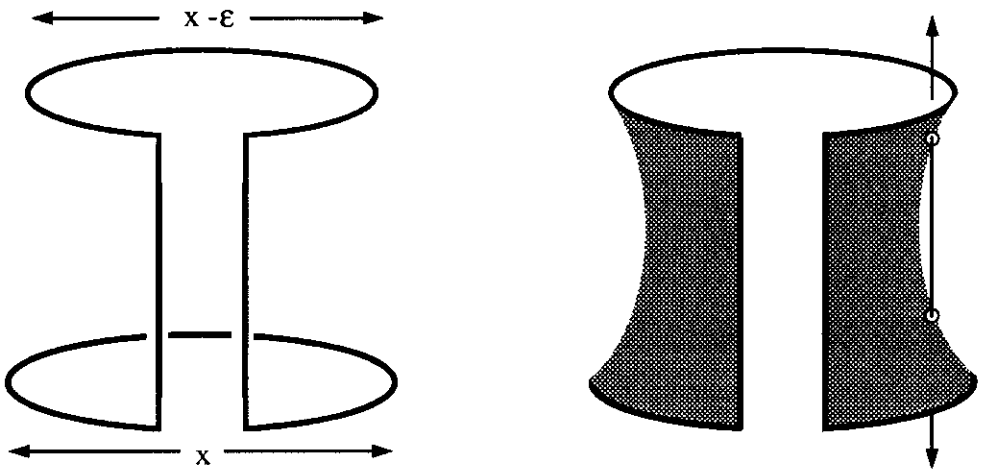

Figure 8: An example of a bounding contour Γ^* (left) where the minimal surface (right) exits the convex hull of $cyl(\Gamma^*)$. However, this instance is not in Radó's class since a vertical line intersects the surface more than once (right), contradicting the second condition (functional representation of D^*) defining Radó's class.

In fact, for a uniformly weighted space (which corresponds to Plateau's original minimum-surface formulation), the difficulty presented by D^* lying outside $cyl(\Gamma^*)$ can occur only when the contour's planar projection Γ is non-convex. However, in such a uniformly weighted space, no part of a minimal surface can lie outside the cylinder induced by the convex hull of the boundary contour, an observation which follows directly from the *minimum principle* that is usually applied in, e.g., the solution of Laplace's equation [3] [17]. The minimal principle states that if the values of a continuous function are held fixed at the domain's boundary, then the function can not attain a minimum or maximum anywhere in the interior of the domain, unless the function is constant over the entire domain (for example, if the temperature function on the boundary of a disk is held fixed and heat diffusion is allowed to occur until a thermal equilibrium steady state is reached, then both the minimum and maximum temperatures will occur on the disk's boundary). Thus, if we extend the search space to include the entire convex hull of the projection of the contour, i.e., we define the augmented cylinder induced by a contour Γ^* to be $cyl(\Gamma^*) = cyl(convex\_hull(proj(\Gamma^*)))$, our methodology applied within this augmented cylinder will yield globally optimal solutions.

The minimum principle also implies that a minimal surface in a uniformly weighted space can not extend up past the highest point on its bounding contour, or down past the lowest point on the contour. With regard to our solution of the discrete Plateau problem, the minimum principle thus implies that given a uniform weight function, "far enough" above (below) the bounding contour Γ^* (recall the discussion defining a cylinder in Section 1) may be achieved with a discrete cylinder that extends no higher (lower) than the highest (lowest) point on Γ^*.

4 A Practical Implementation

There are many algorithms for computing maximum flows in a network [11] [12] [19]. To demonstrate the viability of our approach, we have simply applied an existing implementation of the algorithm of Dinic [13]. Starting with an empty flow, the Dinic algorithm iteratively augments the flow in stages; the optimal flow solution is achieved when no flow augmentation is possible. Each stage starts with the existing flow, and attempts to "push" as much flow as possible along shortest paths from the source to the sink in a residual network wherein each arc has capacity equal to the difference between its original capacity and its current flow value. After the current flow has been thus augmented, newly saturated arcs are removed and the process iterates. Since there can be at most $|N| - 1$ such stages, each requiring time at most $O(|A| \cdot |N|)$, the total time complexity of the Dinic algorithm is $O(|A| \cdot |N|^2)$.

The time complexity of the Dinic algorithm is $O(|N|^3)$, where $|N|$ is the number of nodes in the discrete mesh representation of the space. In practice, more efficient flow algorithms are available. For example, by using the network flow algorithm of [1], we obtain the following:

Theorem: For a given prescribed slab thickness d, our algorithm solves the discrete Plateau problem in $O(|N|^2)$ time, where $|N|$ is the number of nodes in the gridded representation of the space.

Proof: The degree of each node in the mesh is bounded by d^3, so that $|A| = O(d^3 \cdot |N|)$. The network flow algorithm of [1] operates within time $O(|A| \cdot |N| \cdot \log(|A|/|N|))$. Assuming that d is a constant, the overall time complexity of our method is therefore $O(|N|^2)$. □

Our current implementation integrates ANSI C code to transform an arbitrary Plateau problem instance satisfying conditions (1) and (2) of the discrete Plateau problem formulation into a maximum-flow instance; we use the Fortran-77 Dinic code of [13] to compute the flow, and then invoke Mathematica [33] to draw the resulting surface. We have tested our implementation on several classes of problem instances, involving underlying spaces that are both uniformly weighted and non-uniformly weighted. Figure 9 shows a small example with boundary contour consisting of four diagonals on the faces of a cube, uniform node weights, and $d = 2$; the resulting saddle is optimal for the resolution used.

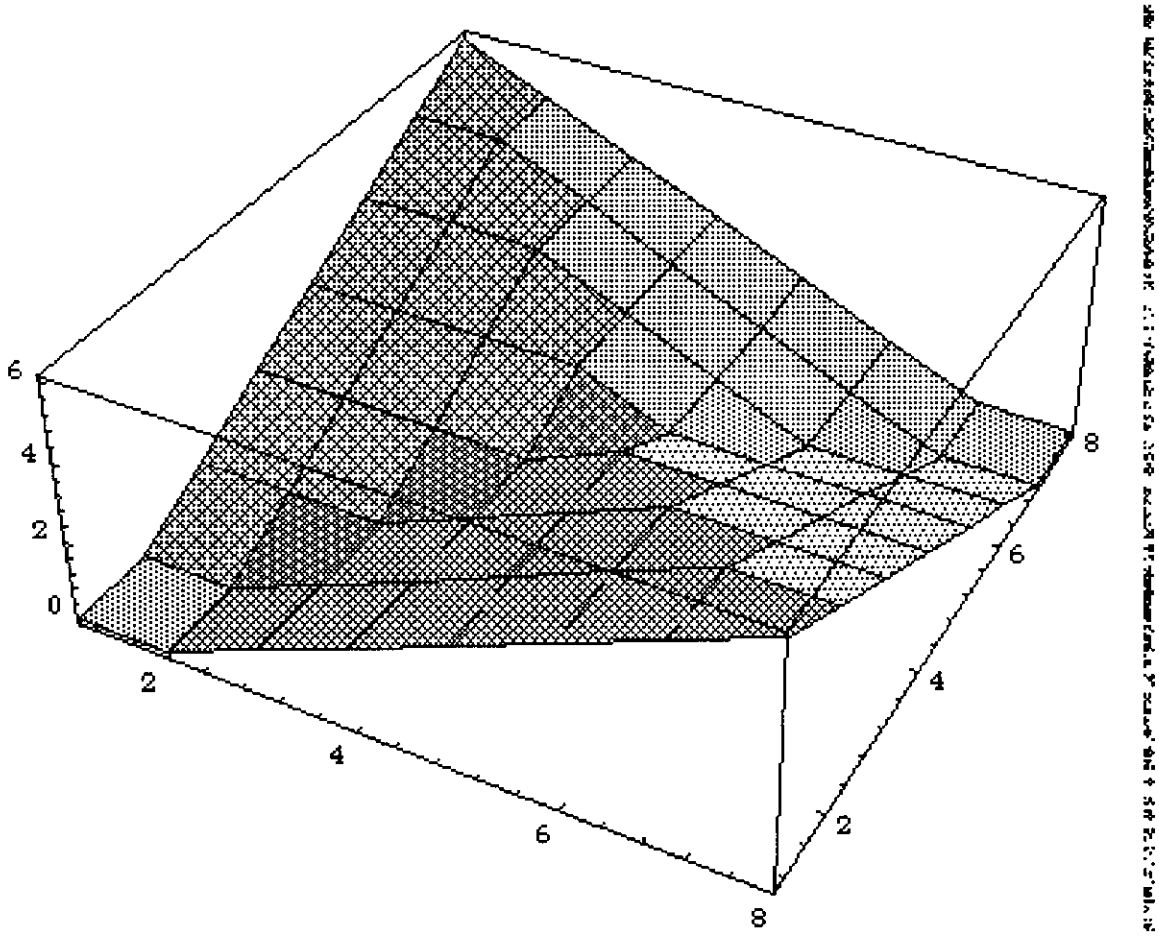

Figure 9: Minimal surface computation in a uniformly weighted space.

Based on our experimental results, we have concluded that our approach constitutes a viable

new method for solving the discrete Plateau problem in arbitrarily weighted spaces. Although Dinic's algorithm is certainly not the ideal maximum flow algorithm for a mesh topology, typical running times used to generate and solve our test cases range from only several seconds (for the example of Figure 9) to an hour on a low-end SUN-4 workstation with small RAM/swap space. Our runtimes show the expected clear dependencies on the mesh resolution, the minimum slab thickness d, and the memory/swap configuration available.

5 Conclusions

We have developed a polynomial-time combinatorial algorithm which gives optimal solutions to a well-studied class of instances of the discrete Plateau problem. Our method is based on the duality between connecting sets and separating sets, and relies on a maximum-flow computation which finds a minimum-cost d-separating slab of prescribed thickness d in an arbitrarily weighted space. The accuracy of the solution with respect to the continuous version of the problem depends on the grid resolution, which is a parameter intrinsic to the input. Our method generalizes to both lower- and higher-dimensional instances.

Chief among the future research goals is improvement of the time complexity of the network flow computation; substantial improvement is likely since the mesh is a highly regular, symmetric network that admits a concise representation. Additional research might also examine minimal surface computations using hierarchical approaches as a heuristic speedup; addressing the case where the prescribed contour does not necessarily lie on the boundary of its containing space is also of interest. Finally, we believe that our methodology can address a larger class of Plateau instances via the standard approach of decomposing a spanning surface into patches which may then be individually optimized.

6 Acknowledgments

Early discussions at IBM between Dr. Ralph E. Gomory and the first author are gratefully acknowledged. We also thank Dr. David Cantor, Professor Basil Gordon and Professor Sinai

Robins for numerous helpful comments on an earlier draft. Partial support for this work was provided by a Department of Defense Graduate Fellowship; by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050.

References

[1] R. K. AHUJA, J. B. ORLIN, AND R. E. TARJAN, *Improved Time Bounds for the Maximum Flow Problem*, Tech. Rep. CS-TR-118-87, Dept. of Computer Science, Princeton University, 1987.

[2] D. G. Cantor, CCR West, CA, private communication, Oct 1992.

[3] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, New York, 1978.

[4] P. CONCUS, *Numerical Solution of the Minimal Surface Equation*, Mathematics of Computation, 21 (1967), pp. 340–350.

[5] COURANT AND ROBBINS, *What is Mathematics? An Elementary Approach to Ideas and Methods*, Oxford University Press, London, England, 1941.

[6] R. COURANT, *Dirichlet's Principle, Conformal Mapping, and Minimal Surfaces*, Interscience Publishers, Inc., New York, 1950.

[7] J. DOUGLAS, *Solution of the Problem of Plateau*, Trans. Amer. Math. Soc., 33 (1931), pp. 263–321.

[8] ——, *The Most General Form of the Problem of Plateau*, Proc. Nat. Acad. Sci. USA, 24 (1938), pp. 360–364.

[9] A. T. FOMENKO, *The Plateau Problem, Part I: Historical Survey*, Gordon and Breach Science Publishers, Amsterdam, 1990.

[10] ——, *The Plateau Problem, Part II: The Present State of the Theory*, Gordon and Breach Science Publishers, Amsterdam, 1990.

[11] L. R. FORD AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1961.

[12] A. V. GOLDBERG, E. TARDOS, AND R. E. TARJAN, *Network Flow Algorithms.* manuscript, March 1989.

[13] D. GOLDFARB AND M. D. GRIGORIADIS, *A Computational Comparison of the Dinic and Network Simplex Methods for Maximum Flow*, Annals of Operation Research, 13 (1988), pp. 83–123.

[14] R. E. GOMORY, T. C. HU, AND J. M. YOHE, *R-Separating Sets*, Can. J. Math., XXVI (1974), pp. 1418–1429.

[15] D. GREENSPAN, *On Approximating Extremals of Functions. I: The Method and Examples for Boundary Value Problems*, ICC Bull., 4 (1965), pp. 99–120.

[16] ——, *On Approximating Extremals of Functions. II: Theory and Generalization Related to Boundary Value Problems for Nonlinear Differential Equations*, Intl. J. Engineering Sci., 5 (1967), pp. 571–588.

[17] R. HABERMAN, *Elementary Applied Partial Differential Equations*, Prentice Hall, New Jersey, 1987.

[18] M. HINATA, M. SHIMASAKI, AND T. KIYONO, *Numerical Solution of Plateau's Problem by a Finite Element Method*, Mathematics of Computation, 28 (1974), pp. 45–60.

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

[20] T. C. HU, A. B. KAHNG, AND G. ROBINS, *Optimal Robust Path Planning in General Environments*, Tech. Rep. CSD-910082, Computer Science Department, UCLA, December 1991.

[21] ——, *Optimal Solution of the Discrete Plateau Problem*, Tech. Rep. CSD-920006, Computer Science Department, UCLA, January 1992.

[22] ——, *Solution of the Discrete Plateau Problem*, Proc. of the National Academy of Sciences, 89 (1992), pp. 9235–9236.

[23] E. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt Rinehart and Winston, New York, 1976.

[24] R. OSSERMAN, *A Survey of Minimal Surfaces*, Van Nostrand Reinhold Company, New York, 1969.

[25] J. PLATEAU, *Statique expérimentale et théorique des liquides soumis aux seules forces molécularies*, Gathier - Villars, Paris, 1873.

[26] T. RADÓ, *On the Problem of Plateau*, Springer-Verlag, Berlin, 1933.

[27] M. STRUWE, *Plateau's Problem and the Calculus of Variations*, Princeton University Press, NJ, 1989.

[28] D. T. THI AND A. T. FOMENKO, *Minimal Surfaces, Stratified Multivarifolds, and the Plateau Problem*, American Mathematical Society, Providence, RI, 1991.

[29] T. TSUCHIYA, *On Two Methods for Approximating Minimal Surfaces in Parametric Form*, Mathematics of Computation, 46 (1986), pp. 517–529.

[30] ——, *Discrete Solution of the Plateau Problem and its Convergence*, Mathematics of Computation, 49 (1987), pp. 157–165.

[31] ——, *A Note on Discrete Solutions of the Plateau Problem*, Mathematics of Computation, 54 (1990), pp. 131–138.

[32] W. L. WILSON, *On Discrete Dirichlet and Plateau Problems*, Numerische Mathematik, 3 (1961), pp. 359–373.

[33] S. WOLFRAM, *Mathematica: A System for Doing Mathematics by Computer, Second Edition*, Addison-Wesley, Reading, MA, 1991.

Glossary

d - Prescribed minimum thickness (non-negative) of the solution surface.

P^* - A three-dimensional point set; i.e., $P \subset \Re^3$.

$proj(P)$ - Orthogonal projection of P onto the xy-plane.

$int(\Gamma)$ - Points interior to the Jordan curve Γ.

$ext(\Gamma)$ - Points exterior to the Jordan curve Γ.

Γ^* - A Jordan curve in $\Re^3$.

slab - A "surface" having a specified positive thickness d.

Γ - Orthogonal projection $proj(\Gamma)$ of Γ^* onto the xy-plane.

D - A (planar) region; also the interior $int(\Gamma)$ of Γ.

D^* - A minimal surface spanning the boundary Γ^*.

$cyl(D^*)$ - All points above and below D^*. i.e., $cyl(D^*) = \{(x, y, z) \mid (x, y) \in proj(D^*), z \in \Re\}$.

$cyl(\Gamma^*)$ - Extension of cyl to contours; i.e., $cyl(\Gamma^*) = cyl(int(proj(\Gamma^*)))$.

D_t^* - Set of points in $cyl(D^*)$ lying above D^*; i.e., $D_t^* = \{(x, y, z) \mid (x, y) \in proj(D^*), z > D^*(x, y)\}$.

D_b^* - Set of points in $cyl(D^*)$ lying below D^*; i.e., $D_b^* = \{(x, y, z) \mid (x, y) \in proj(D^*), z < D^*(x, y)\}$.

w - A non-negative weighting function; i.e., $w : cyl(\Gamma^*) \rightarrow \Re^+$.

$\hat{D}^*$ - A d-separating slab, which is also a superset of some surface D^*.

S - A space; i.e., a subset of $\Re^3$.

B - Boundary of a space.

$\tilde{S}$ - A gridded (discrete) space; i.e., a subset of $Z \times Z \times Z$.

$\tilde{D}^*$ - A discrete d-separating slab, which is also a subset of some d-separating surface $\hat{D}^*$.

N - A node set.

A - An arc set; i.e., $A \subseteq N \times N$.

G - A graph; i.e., $G = (N, A)$.

c - An arc weight function; i.e., $c : A \rightarrow \Re^+$.

c' - A node weight function; i.e., $c' : N \rightarrow \Re^+$.

s - A distinguished source node $s \in N$.

t - A distinguished sink node $t \in N$.

η - A capacitated flow network with node set N, arc set A, source s, sink t, arc capacity function c, and node capacity function c'.

n_i - A node $n_i \in N$ in the graph $G = (N, A)$.

a_{ij} - An arc $a_{ij} \in A$ connecting nodes n_i and n_j.

ϕ_{ij} - Flow value of arc a_{ij} (satisfying $0 \leq \phi_{ij} \leq c_{ij}$).

Φ - Value of the flow in the network; i.e., $\Phi = \sum_i \phi_{si} = \sum_j \phi_{jt}$.

(N', A') - An s-t cut in η; i.e., a set of nodes $N' \subseteq N$ and arcs $A' \subseteq A$ such that every path from s to t uses at least one node of N' or at least one arc of A'.

$c(N', A')$ - Capacity of the cut (N', A'); i.e., the sum of the capacities of the nodes in N' and the arcs in A'.

convex_hull(S) - The convex hull of S; i.e., all convex combinations of points in S.