SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NY 14853-3801

TECHNICAL REPORT NO. 1110

January 1995

FINDING THE OVERALL MINIMUM CUT IN UNDIRECTED PLANAR GRAPHS WITH NON-NEGATIVE EDGE COSTS

by

László Ladányi[1]
and Paul D. Martin[2]

[1]Research was supported by a summer internship from AT&T Bell Laboratories, Holmdel and by NSF, AFOSR, and ONR through NSF Grant DMS-8920550 to Cornell University.
[2]Research was supported by a summer internship from NEC Research Institute and by NSF, AFOSR, and ONR through NSF grant DMS-8920550 to Cornell University.

FINDING THE OVERALL MINIMUM CUT IN UNDIRECTED PLANAR GRAPHS WITH NON-NEGATIVE EDGE COSTS

LÁSZLÓ LADÁNYI*† AND PAUL D. MARTIN*‡

Abstract. In this paper we consider the problem of finding the overall minimum cut in an undirected planar graph with non-negative edge lengths on n nodes. This problem has many applications in network optimization problems. One can solve this problem by solving a sequence of n-1 minimum s-t cut problems where s and t vary from one subproblem to the next. Taking a different approach, we show how to find the overall minimum cut in $O(n^{3/2})$ time by finding the minimum cycle in the topoligical dual graph. This time bound is achieved by recursively applying the Planar Graph Separator Theorem to the topological dual, and solving the minimum cycle problem on the resulting subgraphs.

Key words. planar graph, minimum cut, topoligical dual graph, r-division of a planar graph

AMS subject classifications. 05C38, 05C85, 68R10, 90B10

1. Introduction. The classical problem of finding a minimum s-t cut in a graph with non-negative edge costs for a given pair of nodes s and t has a wide range of applications (see, e.g., Ahuja, Magnanti and Orlin [2]). However, often one needs the overall minimum cut in the graph, i.e., when we have no specified source and sink but we want to find the cut that minimizes the value of the minimum s-t cut over all possible s-t pairs. Most existing algorithms solve this problem by solving a sequence of n-1 minimum s-t cut problems.

For general graphs Hao and Orlin [4] have shown how to choose the sequence of problems to be solved so that the worst-case running time of their algorithm is the same as it would take to solve a single minimum s-t cut problem.

For planar graphs there are much faster algorithms to solve a single minimum s-t cut problem than for general graphs (see Itai and Shiloach [5] for an $O(n\log(n))$ algorithm for s-t planar graphs; Reif [8] for an $O(n\log^2(n))$ algorithm for general planar graphs). These algorithms utilize the fact that finding a minimum s-t cut in a planar graph is equivalent to finding a minimum s-t cut-cycle in the dual of the planar graph. In the case of s-t planar graphs this can be further reduced to a shortest path computation.

Taking this thought one step further one can observe that finding the overall minimum cut in a planar graph is equivalent to finding the minimum cycle in its topological dual. We propose a recursive algorithm (similar to that of Shih, Wu and Kuo [9] which solves the problem in $O(n^{3/2}\log(n))$ time) based on the Planar Graph Separator Theorem of Lipton and Tarjan [6] to find the minimum cycle in $O(n^{3/2}\sqrt{\log(n)})$ time. The bottleneck of this algorithm is a sequence of shortest path computations. We show how to utilize the notion of r-division (introduced by Fredrickson [3]) to speed up this computation sequence and achieve $O(n^{3/2})$ worst-case running time.

2. Definitions. An *undirected graph* $G = (V, E)$ consists of a *node set* V and an *edge set* E. Each edge $e \in E$ connects two nodes of V (if the two nodes are identical

* SCHOOL OF OR&IE, CORNELL UNIVERSITY, ITHACA, NEW YORK 14853, USA.

† Research was supported by a summer internship from AT&T Bell Laboratories, Holmdel and by NSF, AFOSR, and ONR through NSF Grant DMS-8920550 to Cornell University

‡ Research was supported by a summer internship from NEC Research Institute and by NSF, AFOSR, and ONR through NSF grant DMS-8920550 to Cornell University

then e is a *loop*). Let $e = (u, v)$ denote that e connects u and v. From now on the term *graph* will mean undirected graph since in this paper we do not treat directed graphs. Let $c : E \to \Re_+$ be a *cost function* over the edge set of G. For an edge set $E' \subset E$ denote $c(E') = \sum_{e \in E'} c(e)$.

A *u-v path* between the nodes $u, v \in V$ is a sequence of distinct edges $p = e_1, \cdots, e_k$ such that $e_i = (v_{i-1}, v_i)$ for $i = 1, \cdots, k$ where $u = v_0$, $v = v_k$ and the v_i's are distinct nodes. If a u-v path exists for every distinct node pair $u, v \in V$, then we say G is *connected*.

A *cycle* is a closed path, i.e., $v_0 = v_k$.

A *cut* is a set of edges $C \subset E$ such that the graph $(V, E \backslash C)$ is not connected. A cut is *minimal* if no $C' \subset C$ is a cut.

An *overall minimum cut* is a cut that minimizes the function $c(C)$ over all possible cuts C. Note that there exists an overall minimum cut which is minimal.

A *planar graph* is a graph for which there exists an embedding of the edges into the plane such that no two edges cross.

The *dual of a planar graph* G is another planar graph $D(G) = (\mathcal{F}, D(E))$ that can be derived from G in the following way. Consider an embedding of G into the plane. G partitions the plane into *faces*. Each face is bordered by a cycle of edges and each edge borders exactly two faces. Denote $\mathcal{F}$ the set of faces. $D(e) = (F_1, F_2)$ is an edge of the dual graph if and only if F_1 and F_2 are the faces that e borders. Note that there is a one-to-one correspondece between the original and the dual edges so for the sake of simplicity we write e instead of $D(e)$ whenever it is clear from the context that we are talking about a dual edge.

In the problem of finding a minimum cost cut we can assume without loss of generality that G has neither loops nor parallel edges (since minimal cuts contain no loops and since parallel edges can be replaced by a single edge whose cost is the sum of of the costs of the parallel edges it replaced). Therefore if G has n nodes then it has $O(n)$ edges.

3. Minimal cuts and dual cycles. We will assume henceforth that G is a connected planar graph with non-negative edge costs. It is easy to see that there is a one-to-one correspondence between the minimal cuts in G and the cycles in $D(G)$; namely, if we have a minimal cut C in G then $\{D(e) : e \in C\}$ forms a cycle in $D(G)$, and conversely, if we have a cycle $D(C)$ in $D(G)$ then $\{e : D(e) \in D(C)\}$ forms a minimal cut in G. Therefore by defining edge lengths in the dual graph to be the same as the cost of the corresponding edge in G: $l(D(e)) = c(e)$, we get the following proposition:

PROPOSITION 3.1. *A minimum cut in a connected graph G has the same cost as a minimum length cycle in $D(G)$ and the cut that consists of the edges corresponding to the dual edges of a minimum length cycle in $D(G)$ will be a minimum cut.* If G is not connected then a minimum cut is an empty cut and we can discover it in linear time. Otherwise from Proposition 3.1 we can focus on finding a minimum length cycle in the dual graph. Note that $D(G)$ has the same number of edges as G, hence the dual graph has $O(n)$ edges and therefore $O(n)$ nodes.

Therefore from now on we consider the problem of finding the minimum length cycle in a planar graph on $O(n)$ nodes and $O(n)$ edges with non-negative edge lengths.

4. Finding the minimum length cycle in a planar graph with non-negative edge lengths. Shih, Wu and Kuo ([9]) solve this problem in $O(n^{3/2} \log(n))$ time using Matsumoto, Nishizeki and Saito's ([7]) maximum matching algorithm for planar graphs. In the following subsection we propose a different approach that

achieves a slightly better worst-case running time ($O(n^{3/2}\sqrt{\log(n)})$) and in the subsection after that we show how to improve the running time to $O(n^{3/2})$.

4.1. Basic algorithm. THEOREM 4.1 (PLANAR GRAPH SEPARATOR THEOREM [6]). *Let $G = (V, E)$ be a planar graph. Then V can be partitioned into three sets A, B and S such that: no edge joins a vertex in A with a vertex in B; $|A|, |B| \leq c_1|V|$; and $|S| \leq c_2\sqrt{|V|}$ where $c_1(< 1)$ and c_2 are two suitable positive constants.*

Remark: Trivially there exists a constant $0 < c_1' < 1$ (for $|V|$ sufficiently large) such that $|A \cup S|, |B \cup S| \leq c_1'|V|$.

THEOREM 4.2. *Let $G = (V, E)$ be a planar graph with non-negative edge lengths. Then a minimum cycle in G can be found in $O(n^{3/2}\sqrt{\log(n)})$ time.*

Proof. Apply the planar graph separator theorem to G. Create a disjoint partition of the egdes (E^A, E^B) which is *compatible with the separator* S; that is, the endpoints of the edges in E^A are in $A \cup S$ and the endpoints of the edges in E^B are in $B \cup S$. Such a partition can be obtained by putting an edge into E^A (resp. E^B) if at least one end of the edge lies in A (resp. B), otherwise (i.e., when both endpoints are in S) put the edge arbitrarily into one of the edge sets.

Any cycle in G contains either edges only from E^A, edges only from E^B or edges from both E^A and E^B. If a cycle contains edges both from E^A and E^B then there must be a node v on the cycle that is incident to an edge from E^A and to an edge from E^B, therefore v must be in S. (We say that such a cycle *crosses* (E^A, E^B) *at* v.) Thus a minimum cycle of G can be found by recursively finding the minimum cycles in the edge-induced subgraphs generated by E^A and E^B, finding the minimum cycle that crosses (E^A, E^B) and choosing the minimum of these.

For each $v \in S$ a minimum cycle that crosses (E^A, E^B) at v can be found via shortest path computation as follows. Split v into two nodes v^A and v^B, and replace every edge $(v, u) \in E^A$ with an edge (v^A, u) of the same length. Likewise replace every edge $(v, u) \in E^B$ with an edge (v^B, u) of the same length. Then a shortest path from v^A to v^B in this new graph will contain the same edges as a minimum cycle we were looking for. Applying this procedure to every $v \in S$ and choosing the minimum of all these cycles we are able to find the minimum cycle that crosses (E^A, E^B).

Note that splitting v may produce a non-planar graph. However, we can use a planar shortest path algorithm to find the shortest v^A-v^B path by removing all edges incident to v^B (thus restoring planarity), finding the shortest path from v^A to each neighbor of v^B, and then adding the last edge to all those shortest paths.

Let $T(n)$ be the running time of the recursive algorithm on a planar graph with n nodes. Since $|S| = O(\sqrt{n})$ and since a shortest path computation in a planar graph takes $O(n\sqrt{\log(n)})$ time using Fredrickson's algorithm [3], we can find a minimum cycle that crosses (E^A, E^B) in $O(n^{3/2}\sqrt{\log(n)})$ time. Then we have the following recursion for $T(n)$

$$T(n) = T(n_A) + T(n_B) + O(n^{3/2}\sqrt{\log(n)})$$

where n_A (resp. n_B) is the number of nodes in $A \cup S$ (resp. $B \cup S$). Since $n_A, n_B \leq c_1'n$, it follows inductively that $T(n) = O(n^{3/2}\sqrt{\log(n)})$.
□

4.2. Refined algorithm. As we have seen in the previous subsection the bottleneck of the algorithm is the shortest path computation, which takes $O(n^{3/2}\sqrt{\log(n)})$

time. Now we improve this to $O(n^{3/2})$ time. In order to do so we need the notion of an *r-division*.

DEFINITION 4.3. An *r-division* of a planar graph G with n nodes is a collection of edge-induced subgraphs $G_1, \ldots, G_k$ (where $k = c\frac{n}{r}$ for a suitable constant c) that are induced by partitioning the edges of G into $O(n/r)$ disjoint sets $E_1, \ldots, E_k$ such that the following properties hold:

1. Each subgraph G_i contains $O(r)$ edges (thus $O(r)$ nodes).
2. The number of boundary nodes (A *boundary node* is a node which is in more than one subgraph.) in each G_i is $O(\sqrt{r})$.

Note that at first glance this definition appears to differ from that in Frederickson [3]; however, it is essentially the same and does not materially alter any of his proofs.

LEMMA 4.4 (LEMMA 2. IN [3]). *Given a planar graph G with n nodes, an r-division can be obtained in $O(n \log(n))$ time.*

We remark that the proof of this lemma makes repeated use of Lipton and Tarjan's Planar Separator Theorem [6]. Moreover, given a partition A, B and S as in Theorem 4.1, it is possible to construct an r-division which is *compatible with the separator S*; that is, the node set of each G_i is contained either in $A \cup S$ or in $B \cup S$. (Use the given separator as the first in the sequence of separators used to create the r-division.)

DEFINITION 4.5. Given a graph G and an r-division $G_1, \ldots, G_k$ the *substitute graph* W_i of G_i is the complete graph on the boundary nodes of G_i with edge lengths $l_i(u, v) = \{$*the length of the shortest path between u and v within G_i*$\}$. The *skeletal graph* of G is $W_G = \bigcup W_i$. Note that in W_G there will be parallel edges if a pair of boundary nodes u and v both appear in the same two components G_i, G_j.

LEMMA 4.6. *Let $G = (V, E)$ be a planar graph with non-negative edge lengths, and A, B and S a planar separator of G as in Theorem 4.1. Then we can find a partition of the edges (E^A, E^B) which is compatible with the separator and a minimum cycle in G that crosses the edge partition in $O(n^{3/2})$ time.*

Proof. First we create an r-division $G_1, \ldots, G_k$ which is compatible with S. Create E^A and E^B as follows. For $i = 1, \ldots, k$ if G_i contains a node of A then put all edges in G_i into E^A; if G_i contains a node of B then put all edges in G_i into E^B; otherwise (i.e. if all nodes of G_i are in S) put all edges of G_i arbitrarily into one of E^A or E^B. Since our r-division is compatible with S, G_i can not contain nodes both from A and B, thus (E^A, E^B) is an edge partition which is compatible with S.

Now we build the skeletal graph W_G of G corresponding to our r-division. Partition the edges of W_G into two disjoint sets E'^A and E'^B such that if an edge e' of W_G represents a shortest path in subgraph G_i whose edges belong to E^A then put e' into E'^A and otherwise put it into E'^B.

Note that if $v \in S$ is not a boundary node then the edges incident to v are either all in E^A or all in E^B. Therefore there is no cycle that crosses (E^A, E^B) at v. For every boundary node $v \in S$ a minimum cycle that crosses (E^A, E^B) at v can be found via a shortest path computation as follows. Form the modified skeletal graph W_v from W_G by splitting v into v^A and v^B, and replace every edge $(v, u) \in E'^A$ with an edge (v^A, u) of the same length. Likewise replace every edge $(v, u) \in E'^B$ with an edge (v^B, u) of the same length. In this new graph W_v the edges of a shortest path from v^A to v^B correspond to the edges in a cycle that crosses (E^A, E^B) at v (by expanding the edges to the shortest paths they represent in their individual subgraphs).

Consider a cycle that crosses (E^A, E^B) at v. We can divide this cycle into path segments such that every segment goes from one boundary node to another in the same region and contains edges only from that region. If we replace each segment

with the shortest path in the region between the corresponding pair of boundary nodes then we get another cycle that crosses (E^A, E^B) at v (since edges in a region are either all in E^A or all in E^B) and which is no longer than the original cycle. The edges in E'^A and E'^B corresponding to these new segments form a path from v^A to v^B in the W_v. Since the path our algorithm produces is a shortest v^A to v^B path in W_v, the corresponding cycle is a minimum cycle which crosses (E^A, E^B) at v.

Taking the minimum of these cycles for each boundary node $v \in S$ we get a minimum cycle which crosses (E^A, E^B).

Lemma 4.4 says that we can compute the r-division in $O(n\log(n))$ time, which dominates the time required to partition the edges into E^A and E^B. Since each G_i is planar and contains $O(\sqrt{r})$ boundary nodes, $O(r)$ nodes and $O(r)$ edges, it follows that the substitute graph W_i can be computed in $O(r^{3/2}\sqrt{\log(r)})$ time using $O(\sqrt{r})$ planar shortest path computations. Creating all the substitute graphs takes $O(\frac{n}{r}\ r^{3/2}\sqrt{\log(r)}) = O(n\sqrt{r\log(r)})$ time. Combining these to get the skeletal graph W_G takes $O(n)$ time, since we have to touch every edge in W_G once and there are $O(n/r)$ substitute graphs, each having $O(r)$ edges (a complete graph on $O(\sqrt{r})$ nodes). This time is dominated by the previous bound; thus in creating W_G we spend $O(n\sqrt{r\log(r)})$ time.

The skeletal graph is the union of $O(n/r)$ graphs each of which is a complete graph on $O(\sqrt{r})$ nodes, thus W_G has $O(n/\sqrt{r})$ nodes and $O(n)$ edges. Therefore partitioning the edges of W_G into E'^A and E'^B can be done in $O(n)$ time. We can create W_v in constant time (by maintaining the adjacency lists in two partitions corresponding to E'^A and E'^B) and finally, one shortest path computation runs in $O(\frac{n}{\sqrt{r}}\log(\frac{n}{\sqrt{r}}) + n)$ time using a heap implementation of Dijkstra's algorithm (see [1]).

Therefore the total running time of the algorithm is

$$O\left(n\sqrt{r\log(r)} + \sqrt{n}\left(\frac{n}{\sqrt{r}}\log(\frac{n}{\sqrt{r}}) + n\right)\right).$$

The term $n^{3/2}$ in this expression doesn't depend on r; thus with this algorithm we can't get a better worst-case bound than $O(n^{3/2})$, which can be achieved, e.g., with the choice of $r = \sqrt{n}$.

□

THEOREM 4.7. *Let $G = (V, E)$ be a planar graph with non-negative edge lengths. Then a minimum cycle in G can be found in $O(n^{3/2})$ time.*

Proof. As in Theorem 4.2 we evaluate $T(n)$ recursively. Here we have

$$T(n) = T(n_A) + T(n_B) + O(n^{3/2}),$$

which resolves to $T(n) = O(n^{3/2})$.

□

Remark: Although our algorithm requires an r-division at each stage of the recursion, and this seems to be a significant overhead, we can always use the fragments of the r-division of the previous stage as a starting point, and thereby greatly reduce the amount of time required to compute the subsequent r-divisions.

5. Remark. We have recently learned that an algorithm similar to our unrefined algorithm was independently devised by D. Hartvigsen and G. Fredrickson, and it was presented on the ORSA/TIMS Phoenix 1993 meeting.

REFERENCES

[1] A. AHO, J. HOPCROFT AND J. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] R. K. AHUJA, T. L. MAGNANTI AND J. B. ORLIN, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[3] G. N. FREDRICKSON *Fast Algorithms for Shortest Paths in Planar Graphs, with Applications*, SIAM J. on Computing 16 (1987), pp. 1004–1022.

[4] J. HAO AND J. B. ORLIN, *A Faster Algorithm for Finding the Minimum Cut in a Graph*, SODA 1992, pp. 165–174.

[5] A. ITAI AND Y. SHILOACH, *Maximum Flow in Planar Networks*, SIAM J. on Computing 8 (1979), pp. 135–150.

[6] R. J. LIPTON AND R. E. TARJAN, *A Separator Theorem for Planar Graphs*, SIAM J. Applied Math. 36 (1979), pp. 177–189.

[7] K. MATSUMOTO, T. NISHIZEKI AND N. SAITO, *Planar Multicommodity Flows, Maximum Matchings and Negative Cycles*, SIAM J. on Computing 15 (1986), pp. 495–510.

[8] J. H. REIF, *Minimum s-t Cut of a Planar Undirected Network in $O(n \log^2(n))$ Time*, SIAM J. on Computing 12 (1983), pp. 71–81.

[9] W. SHIH, S. WU AND Y. S. KUO, *Unifying Maximum Cut and Minimum Cut of a Planar Graph*, IEEE Transactions on Computers 39 (1990), pp. 694–697.