Final draft of paper that appeared in SIAM J. Comput. 23 (1994), 864-894

The Complexity of Multiterminal Cuts

E. Dahlhaus[1], D. S. Johnson[2], C. H. Papadimitriou[3], P. D. Seymour[4], and M. Yannakakis[2]

Abstract

In the Multiterminal Cut problem we are given an edge-weighted graph and a subset of the vertices called terminals, and asked for a minimum weight set of edges that separates each terminal from all the others. When the number k of terminals is two, this is simply the min-cut, max-flow problem, and can be solved in polynomial time. We show that the problem becomes NP-hard as soon as $k = 3$, but can be solved in polynomial time for planar graphs for any fixed k. The planar problem is NP-hard, however, if k is not fixed. We also describe a simple approximation algorithm for arbitrary graphs that is guaranteed to come within a factor of $2-2/k$ of the optimal cut weight.

1. Introduction

The *Multiterminal Cut* problem can be defined as follows: Given a graph $G = (V,E)$, a set $S = \{s_1, s_2, ..., s_k\}$ of k specified vertices or *terminals*, and a positive weight $w(e)$ for each edge $e \in E$, find a minimum weight set of edges $E' \subseteq E$ such that the removal of E' from E disconnects each terminal from all the others.

When $k = 2$ this problem reduces to the famous ''min-cut/max-flow'' problem, a problem of central significance in the field of combinatorial optimization due to its many applications and the fact that it can be solved in polynomial time (e.g., see [7,17,18,20]). The ''k-terminal cut'' problem for $k > 2$ has been a subject of discussion in the combinatorics community for years (closely-related variants were proposed as early as 1969 by T. C. Hu [17,p.150]). A variety of applications have been suggested, most having to do with the minimization of communication costs in parallel computing systems. In [23], Stone points out how the problem of assigning program modules to processors can be formulated in this framework. Other applications involve partitioning files among the nodes of a network, assigning users to base computers in a multicomputer environment, and partitioning the elements of a circuit into the subcircuits that will go on different chips. It is known that such problems can become NP-hard even for $k = 2$ if there is a

[1] University of Bonn. Current Address: Basser Department of Computer Science, University of Sidney, New South Wales 2006, Australia.
[2] AT&T Bell Laboratories, Murray Hill, NJ 07974
[3] Department of Computer Science and Engineering, University of California at San Diego
[4] Bell Communication Research, Morristown, NJ 07960
A preliminary version of this paper appeared as ''The complexity of multiway cuts'' in *Proc. 24th Annual ACM Symposium on Theory of Computing* (1992), pp. 241-251.

constraint imposed on the *size* of the components into which the graph is cut [9,10]. In this paper we ask whether the problem might be tractable without such a constraint (as it is for $k = 2$).

Our first results concern the planar case. The restriction to planar graphs, besides its basic graph-theoretic significance, has potential relevance in the circuit partitioning application.

Theorem 1.

(a) For $k = 3$, the planar Multiterminal Cut problem can be solved in time $O(n^3 \log n)$.

(b) For any fixed $k \geq 3$, the planar Multiterminal Cut problem is solvable in polynomial time.

The algorithms of Part (b) are, unfortunately, exponential in k. (Specifically, they are $O((4k)^k n^{2k-1} \log n)$.) That such exponential behavior is likely to be unavoidable follows from the next result.

Theorem 2. If k is not fixed, the Multiterminal Cut problem for planar graphs is NP-hard even if all edge weights are equal to 1.

For the Multiterminal Cut problem in arbitrary graphs, NP-hardness sets in much earlier.

Theorem 3. The Multiterminal Cut problem for arbitrary graphs is NP-hard for all fixed $k \geq 3$ even if all edge weights are equal to 1.

This theorem is proved using a ''gadget'' that has interesting properties on its own (as a counterexample to a conjecture about the possible submodularity of 3-Terminal Cut). The theorem's negative consequences are partially mitigated by technical lemmas that may yield substantial reductions in the sizes of instances encountered in practice.

Finally, we have the following two approximation results, one positive and one negative.

Theorem 4. There is an $O(knm \log(n^2/m))$ approximation algorithm for the Multiterminal Cut problem that for arbitrary graphs and arbitrary k is guaranteed to find cuts that are within $2(k-1)/k$ of optimal.

Theorem 5. For any fixed $k \geq 3$, the k-Terminal Cut problem is MAX SNP-hard (and hence cannot have a polynomial time approximation scheme unless P = NP [1,21]).

The results presented here can be contrasted to those of [13,16,22], which concern what might be called the *k-Cut* problem. In this problem we are given G, k, and w as above (but not S), and are asked merely for a minimum weight set of edges E' whose removal separates the graph into at least k nonempty connected components. Although this problem is NP-hard for arbitrary k, it is solvable in polynomial time for each fixed $k > 2$, even for arbitrary graphs [13]. The running time is $O(n^{k^2/2-k+11/2})$. Thus the k-Cut problem is significantly easier than the problem we study here. Note, however, that this may not hold true in the case of planar graphs. At present there is no known general method for exploiting planarity in the k-Cut problem, in contrast to our results for planar Multiterminal Cut. Thus for fixed $k \geq 6$, our planar Multiterminal Cut algorithm currently provides the best method for solving the planar k-Cut problem: Simply run our algorithm for all possible sets S of k terminals and take the least-weight solution found. A factor proportional to n^k is added to our running time, but the resulting time bound is still $O(n^{3k-1} \log n)$, which beats $O(n^{k^2/2-k+11/2})$ when $k \geq 6$. (For $k = 3$ and *unweighted* planar graphs, the $O(n^7)$ of the general k-Cut result has been beaten more directly, first by an $O(n^2)$ algorithm in [16], and subsequently by an $O(n \log n)$ algorithm in [15].) Reference [22] concerns approximation results for the k-Cut problem, showing that the bounds we obtain in Theorem 4 for Multiterminal Cut can be obtained for k-Cut directly, without having to apply our multiterminal result to all possible sets of k terminals.

To avoid bibliographic confusion, we should mention that, with the exception of Theorem 5, the results in the current paper were first announced in 1983 in an unpublished but widely circulated extended abstract [4]. The abstract has since been widely cited, both in the above-mentioned work on k-Cut, and in follow-up work on the Multiterminal Cut problem itself: In [2], Chopra and Rao observe, as we failed to do in our original abstract, that for trees and 2-trees, the general k-Terminal Cut problem can be solved in linear time by a straightforward dynamic programming algorithm. (This can be generalized to graphs of bounded tree-width for any fixed bound, by standard techniques.) The facets of the Multiterminal Cut polyhedron are studied in [2,3]. An interesting generalization of the Multiterminal Cut problem, about which we shall have more to say in our concluding section, is studied in [5,6]. The 1983 abstract did not contain our proofs; these are presented here for the first time. (The 1983 abstract also used the less-descriptive term *multiway* cut for what we now call a multiterminal cut. The new terminology was introduced in [3] and we adopt it here for added clarity.)

The paper is organized as follows. In Section 2 we cover the positive results for the planar case (Theorem 1a and 1b). The corresponding negative result for the planar case (Theorem 2) is covered in Section 3. Section 4 covers our results for general graphs (Theorems 3, 4, and 5 and associated technical lemmas). A concluding Section 5 discusses additional variants and generalizations of Multiterminal Cut to which our techniques can apply, and points out some of the remaining open problems in the area.

2. Algorithms for The Planar Case

Our main result for planar graphs (Theorem 1) says that for all fixed k, the Multiterminal Cut problem is solvable in polynomial time. This is in contrast to Theorem 3, which says that for arbitrary graphs, the problem is NP-hard for any fixed $k \geq 3$. The key advantage we gain from planarity lies in the existence of a planar dual to our given graph G. We will assume without loss of generality that our graph $G = (V,E)$ is connected and that we have fixed an embedding of it on the plane. We will use a superscript D to denote a dual object. Thus G^D is the dual graph of G. If F is a subset of the edges of G, F^D is the corresponding set of edges of G^D. (Note: F^D is *not* the dual of the subgraph (V,F) of G.)

We start with Theorem 1a and the case of $k = 3$, and then show how our proof techniques can be generalized to cover the case of general fixed k (Theorem 1b).

2.1. Planar 3-Terminal Cuts

A key concept in all that follows is the idea of an *isolating cut*. For a given terminal s_i, an *isolating cut* for s_i is any set of edges that cuts all paths between s_i and all the other terminals.

Note that a minimum weight isolating cut for s_i can be constructed by merging all the terminals other than s_i into a special vertex s_0, and then finding a minimum $s_i - s_0$ cut in the resulting graph by a standard 2-terminal minimum cut algorithm. Note also that any k-terminal cut induces isolating cuts for each of the k terminals. An optimal k-terminal cut need not induce optimal isolating cuts however, due to the savings that can be obtained when the induced isolating cuts share edges. As we shall see, when G is planar, the sharing of edges has a convenient interpretation in terms of paths in the dual graph G^D.

For the purpose of introducing some terminology, let us for simplicity first look at the dual in the case when there is no sharing of edges. From now on in this section, we shall assume $k = 3$. Figure 1 shows a graph G with a 3-terminal cut C, together with the duals G^D and C^D of each. The thicker edges in the figure are those of C and C^D, respectively. Note that the edges of C^D partition the geometric embedding of G^D into three regions. (In the case of Figure 1, two of these regions are single faces of G^D, but the other is the union of several faces.) Let us say that a vertex of G is *in* a given region if the face of G^D to which the vertex corresponds is part of that region. Then observe that in the figure, each of the terminals of G is in a separate region of C^D. This is clearly a general property: C is a 3-terminal cut of a graph G if and only if the terminals s_1, s_2, s_3 are in different regions of C^D. (The boundary of the region containing x_i is the dual of an isolating cut for x_i.) If C is an *optimal* 3-terminal cut, C^D has exactly three regions, each one containing a distinct terminal. Furthermore, removing any edge from C^D must merge two regions, as otherwise the corresponding edge of C is not needed in the cut.

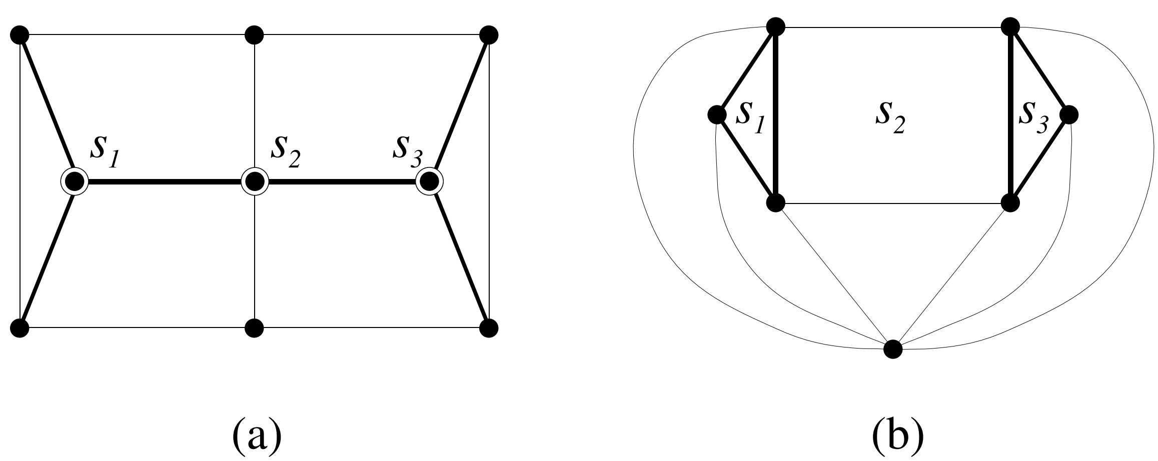


FIGURE 1. A planar 3-terminal cut (a) and its dual (b).

For a general instance of the 3-terminal cut problem, there are two topologically distinct possibilities for an optimal cut C^D. See Figure 2, where $\{i,j,k\} = \{1,2,3\}$.

Cut Type I. C^D consists of two edge-disjoint cycles. (See Figure 2a,b.) Note that the cycles may have one vertex in common and/or one cycle may lie inside the other, as in Figure 2b. They cannot have more than one vertex in common, however, as this would imply that C^D had more than three regions.

Cut Type II. Each pair of regions of C^D shares an edge. (See Figure 2c.)

For Type I cuts, the cut C consists of two edge-disjoint cuts (corresponding to the two

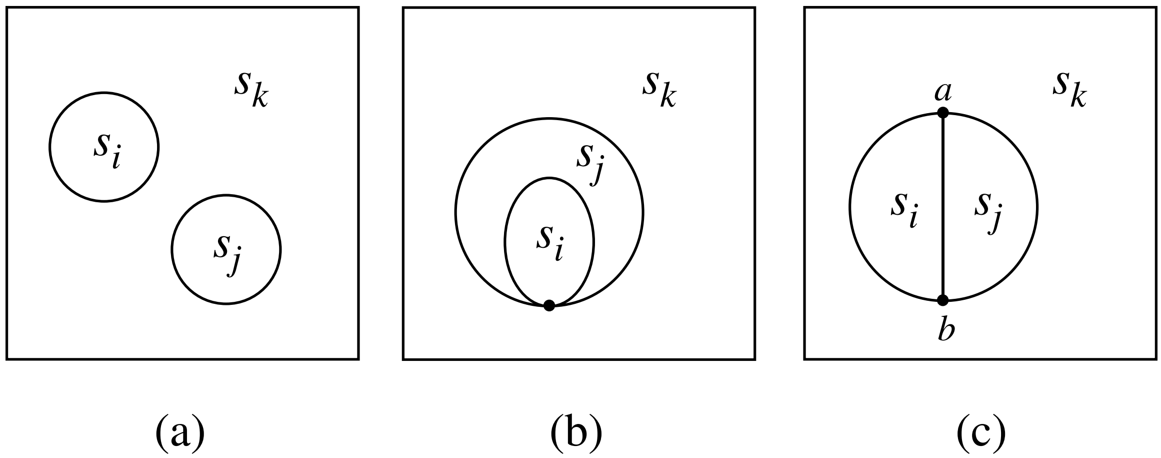


FIGURE 2. Types of 3-terminal cuts: Type 1 (a) and (b), Type 2 (c).

cycles of C^D), each isolating one of the three terminals from the other two. The notion of such an ''isolating cut'' will also be of use in treating the case of Type II cuts. We define it formally (and generalize it to arbitrary k) as follows.

We shall now describe how to find an optimal 3-terminal cut. We provide procedures that work for each type of cut. Each procedure either returns the best cut of the corresponding type, or else reports (correctly) that any optimal cut is of the other type.

Our procedure for Type I cuts is straightforward. We simply compute the three minimum weight isolating cuts for s_1, s_2, and s_3 respectively. Note that a minimum weight Type I cut must have weight at least as large as the sum of the weights of the two smallest of these three isolating cuts. If the two smallest are edge-disjoint, then their union is optimal among all 3-terminal cuts of Type I. If the two smallest are not edge-disjoint, then their union is a 3-terminal cut that has strictly smaller weight (since all edge weights are by assumption positive). Consequently, the best 3-terminal cut is not of Type I.

Our procedure for Type II cuts is significantly more complicated. Suppose we have an optimal 3-terminal cut that is of Type II. Look again at Figure 2c. The cycle that bounds each region corresponds to an isolating cut for the terminal contained in that region, but these isolating cuts are not necessarily optimal, as they overlap. Consider the two vertices that are of degree 3 in C^D and are labeled a and b in the figure. The following lemma allows us to fix one of the three paths connecting a and b in G^D.

Lemma 2.1. Suppose that the dual of an optimal 3-terminal cut C is of Type II and a and b are the two vertices of degree 3 in C^D. Let P be any shortest path from a to b in G^D. Then there is an optimal 3-terminal cut C_0 that is of Type II, has a and b as its two vertices of degree 3 in C_0^D, and such that P is one of the three paths that join a to b in C_0^D.

Proof. Consider all optimal 3-terminal cuts C_t such that C_t^D is of Type II with a and b as specified. Among these cuts, pick C_0 to be the cut C_t for which C_t^D contains the longest possible initial segment of P starting at a. We will show that C_0^D contains all of P.

Assume that P is *not* one of the three paths connecting a to b in C_0^D. As we traverse P from a to b, let x be the first vertex of P such that the edge leaving x in P is not in C_0^D. Let y be the first vertex of P after x which is in C_0^D. Note that it is possible that $x = a$ and/or $y = b$, or that x and y are consecutive vertices of P but the edge $\{x,y\}$ is not in C_0^D. Let Q_1, Q_2, Q_3 be the three $a-b$ paths in C_0^D. The path P initially follows one of these paths, say Q_1 without loss of generality, leaves it at vertex x, and then hits a path again at vertex y. We distinguish two cases depending on whether y is on the same path Q_1 as x (Figure 3a) or on a different path, say without loss of generality Q_2 (Figure 3b). In both cases the portion of P between x and y lies entirely in one region of C_0^D (by planarity), and partitions that region into two subregions. One of these two subregions contains the terminal that was contained in the original region, and the other contains no terminals at all.

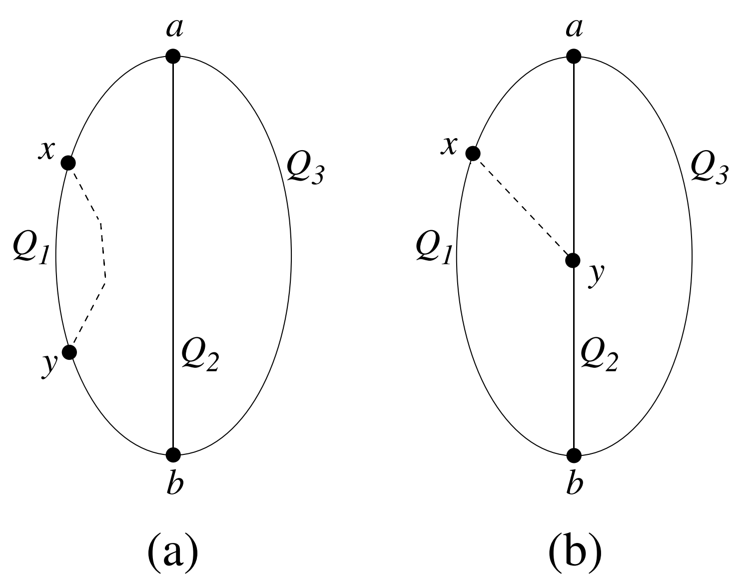


FIGURE 3. Possibilities for the $P[x,y]$ in the proof of Lemma 2.1.

Case 1. Vertices x and y are both on Q_1.

(This includes the cases when $x = a$ or $y = b$.) Let us denote the portion of a path Q between two of its vertices u and v as $Q[u,v]$. Let us assume without loss of generality that $P[x,y]$ lies in the region of C_0^D bounded by Q_1 and Q_2, as illustrated in Figure 3a.

Subcase 1.1. The subregion bounded by $Q_1[x,y]$ and $P[x,y]$ contains no terminal.

Consider the path from a to b that follows P (and Q_1) from a to x, follows Q_1 from x to y, and then follows P from y to b. It must be at least as long as P since P is a shortest path from a to b. Therefore $w(Q_1[x,y]) \geq w(P[x,y])$, where if F is a set of edges, we take $w(F)$ to be $\Sigma_{e\in F} w(e)$. If we modify C_0^D so that $Q_1[x,y]$ is replaced by $P[x,y]$, we will obtain a path Q' that is no longer than Q_1, agrees with a longer initial segment of P, and remains disjoint from Q_2 and Q_3 (except at their endpoints). Since the subregion bounded by $Q_1[x,y]$ and $P[x,y]$ contained no terminals, the union of Q' with Q_2 and Q_3 will still constitute the dual of a 3-terminal cut. This thus contradicts our definition of C_0, and so the subcase cannot apply.

Subcase 1.2. The subregion bounded by $Q_1[x,y]$ and $P[x,y]$ contains the terminal that lies in the region of C_0^D bounded by Q_1 and Q_2 (and hence the region bounded by Q_2 and the path consisting of $Q_1[a,x]$, $P[x,y]$, and $Q_1[y,b]$ contains no terminal).

In this case, we propose to modify C_0^D by deleting the path Q_2 and adding the path $P[x,y]$. Given the location of the terminal that was in the region of C_0^D bounded by Q_1 and Q_2, this will still be the dual of a 3-terminal cut. Thus all we must do now is argue that such a cut would violate the definition of C_0. First, observe that since P is a shortest path between its endpoints, all subpaths of P must themselves be shortest paths between their endpoints. In particular, we must have $w(P[x,y]) \le w(Q_1[y,b]) + w(Q_2) + w(Q_1[a,x])$. If either $x \ne a$ or $y \ne b$, we would then have $w(P[x,y]) < w(Q_2)$, since all edge weights are positive by definition. Thus our modified 3-terminal cut would be strictly lighter than C_0, contradicting its definition. Therefore we must have $x = a$ and $y = b$, and the current 3-terminal cut contains no edges from P. Note, however, that by our choice of P we have $w(P) \le w(Q_2)$, and so can replace Q_2 by P and obtain a new 3-terminal cut whose weight is at least as small. The new cut is hence also optimal, and contradicts our assumption that no optimal cut could contain a longer subpath of P (starting from a) than does the cut containing Q_2. Thus this subcase cannot hold either, and Case 1 is ruled out.

Case 2. Vertex x is on Q_1, $x \ne a$, and vertex y is on Q_2, $y \ne b$.

Subcase 2.1. The region bounded by $Q_1[a,x]$, $P[x,y]$, and $Q_2[a,y]$ contains no terminal.

In this case, replacing $Q_2[a,y]$ by $P[x,y]$ in C_0^D will yield the dual of a 3-terminal cut. We argue that this new 3-terminal cut must have strictly smaller weight than C_0, contradicting the definition of C_0. Consider the path from a to b consisting of $Q_2[a,y]$ and $P[y,b]$. It must be at least as long as P, so we must have $w(Q_2[a,y]) \ge w(P[a,y])$. Consequently, since $a \ne x$, we must have $w(Q_2[a,y]) > w(P[x,y])$, and the new cut indeed has smaller weight.

Subcase 2.2. The region bounded by $Q_1[x,b]$, $P[x,y]$, and $Q_2[y,b]$ contains no terminal.

In this case we replace $Q_2[y,b]$ by $P[x,y]$ and obtain a contradiction analogous to the one of Subcase 2.1.

Thus Case 2 as well as Case 1 is ruled out. Consequently, P is contained in C_0, and the optimal 3-terminal cut called for by Lemma 2.1 exists. □

In light of Lemma 2.1, our procedure for Type II cuts can work by repeatedly calling a subroutine, once for each potential pair a,b of degree-3 vertices in C^D. The subroutine either constructs a minimum weight 3-terminal cut C which is of Type II and has a and b as the two degree-3 vertices in C^D, or reports (correctly) that no minimum weight 3-terminal cut has that form. The subroutine proceeds as follows:

First, construct a shortest path P between a and b in G^D. By Lemma 2.1 we may assume that P is contained in C^D. Delete the edges in G corresponding to the edges of P, obtaining a new graph H. In the embedding of this new graph induced by our original embedding of G, all the regions of G corresponding to vertices on P in G^D are merged into a single region. This corresponds in H^D to coalescing all the vertices along the path P from a to b into a single vertex v_P. This coalescence turns C^D from a Type II cut into a Type I cut like the one in Figure 2b in which the two edge-disjoint cycles share a common vertex, in this case v_P. (The two cycles need not however be nested as they are in the figure; they can have disjoint interiors.)

We can now apply our previously described procedure for Type I cuts to H, obtaining a Type 1 cut C_H, or a report that the best 3-terminal cut for H is *not* of Type I. In the latter case, an optimal 3-terminal cut for G could not have been of Type II with the pair a,b as its degree-3 vertices, and we report this fact. In the former case, the cut C_H will, when augmented with the edges of G corresponding to the edges of P in G^D, be a 3-terminal cut for G. It cannot be an optimal cut, however, unless C_H^D has the desired form of two edge-disjoint cycles with v_P as a single common vertex. Otherwise the edges corresponding to P can be deleted and a valid (and lighter) 3-terminal cut for G will remain. Thus if C_H^D does not have the desired form, we once again report that no optimal 3-terminal cut for G is of Type II with a,b as its two degree-3 vertices.

Our overall algorithm for finding an optimal 3-terminal cut can thus proceed as follows:

Procedure 3-Terminal

1. Perform the Type I procedure on G.
   If a valid Type I cut is found, put it on the list of potential optima.
2. Construct the dual graph G^D and perform an all-pairs shortest path computation for G^D.
   For each pair a,b of vertices in G^D, do the following:
   - 2.1. Let P be the shortest path in G^D between a and b as constructed in step 2, and let H be the graph obtained from G by deleting the edges corresponding to edges of P.
   - 2.2. Perform the Type I procedure on H.
   - 2.3 Let v_P be the coalesced vertex in H^D corresponding to the path P. If a valid Type I cut C_H for H is found and has a dual consisting of two edge-disjoint cycles having v_P as their unique common vertex, do the following:
     - 2.3.1 Let C_G be the 3-terminal cut for G consisting of C_H together with the edges of G corresponding to the edges of P in G^D.
     - 2.3.2 Add C_G to the list of potential optima.
3. Output the lightest 3-terminal cut on the list of potential optima.

Theorem 1a. Given a planar graph G with specified terminals s_1, s_2, and s_3, Procedure 3-Terminal outputs an optimal 3-terminal cut, and can be implemented to run in time $O(n^3 \log n)$, where n is the number of vertices in G.

Proof. The fact that Procedure 3-Terminal outputs an optimal cut follows from the above discussion. To analyze the running time, note that the bulk of the time is spent in the all-pairs shortest path computation of Step 2 and the isolating cut computations needed for each of the $\binom{n}{2}$ invocations of the Type I procedure in Step 2.2. The all-pairs shortest path computations takes place in a planar graph, and so can be implemented to run in time $O(n^2)$ using the techniques of [8]. The isolating cut computations reduce as noted to 2-terminal minimum cut computations, and so can be performed using standard 2-terminal cut algorithms.

For planar graphs, such algorithms run in time $O(n \log n)$, again using techniques from [8]. Unfortunately, if we use the techniques we originally described for computing isolating cuts, the graphs to which the 2-terminal cut algorithm is applied will not necessarily be planar. (Recall that our original proposal was to apply the 2-terminal cut algorithm to graphs constructed from G

by coalescing pairs of terminals, and note that those pairs need not be adjacent in *G*.) Thus without a further idea, we might be forced to use a general algorithm, and the running time would grow to $O(n^2 \log n)$. (This can be obtained for instance by using the $O(nm\log(n^2/m))$ 2-terminal cut algorithm of [12] and taking advantage of the fact that although our graphs need not remain planar, they do remain sparse.) This would force our overall running time up to $O(n^4 \log n)$. Fortunately, we can get around this obstacle as follows.

Recall that our goal in the Type 1 procedure is to find the two lightest among the isolating cuts for s_1, s_2, and s_3. For $i \neq j \in \{1,2,3\}$, let $c(i)$ denote the weight of a minimum isolating cut for s_i and $c(i,j)$ denote the weight of a minimum (2-terminal) cut separating s_i from s_j. Clearly, both $c(i), c(j) \geq c(i,j)$. Now note that in any cut separating s_i from s_j, the third terminal will be disconnected from at least one of s_i, s_j, and therefore the cut must isolate either s_i or s_j. Thus either $c(i) \leq c(i,j)$ or $c(j) \leq c(i,j)$. Combining this with the previous inequality, it follows that $c(i,j) = \min\{c(i), c(j)\}$. We compute the best two isolating cuts as follows:

Compute a minimum 2-terminal cut $C_{1,2}$ separating s_1 from s_2 in *G*. This will be an isolating cut for one of the two terminals, say s_1 without loss of generality. At this point we have $c(1,2) = c(1) \leq c(2)$. Now compute a minimum 2-terminal cut $C_{2,3}$ separating s_2 (the non-isolated terminal) from s_3 in *G*. If this second cut isolates s_2 we have $c(2,3) = c(2) \leq c(3)$; if it isolates s_3 we have $c(2,3) = c(3) \leq c(2)$. In either case $C_{1,2}$ and $C_{2,3}$ are the two lightest isolating cuts as desired, and both were computed in the original planar graph *G*.

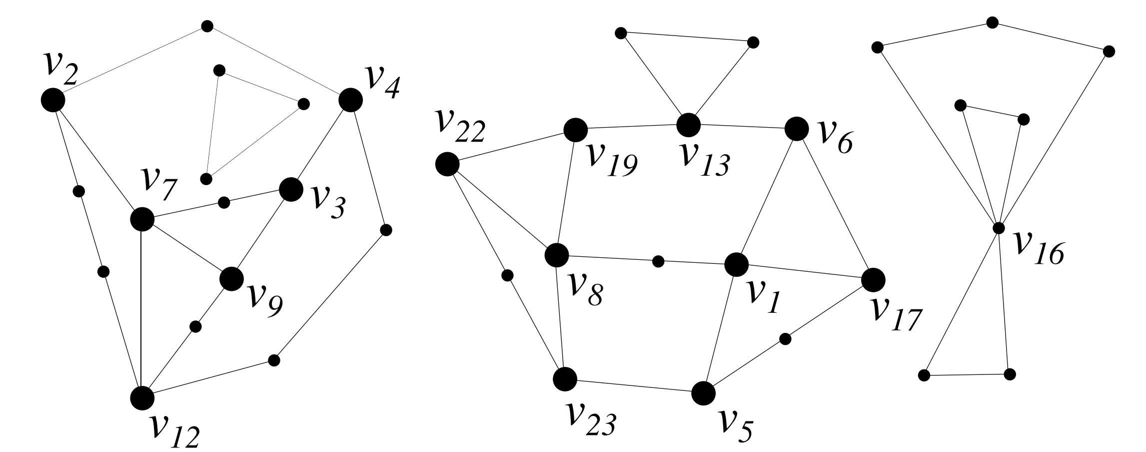
Step 2.2 thus involves $2\binom{n}{2} \approx n^2$ planar 2-terminal cut computations, for an overall time of $O(n^3 \log n)$. Since this is the dominant component of the running time, it also provides a bound on the overall running time of Procedure 3-Terminal, which thus obeys the claimed running time bound. □

2.2. Planar Multiterminal Cuts

In this section we turn to the case of *k*-terminal cuts where $k > 3$. The algorithm we present will work for all $k \geq 3$ and will have a running time that, although exponential in *k*, is polynomial whenever *k* is fixed. It can be viewed as a (major) generalization of the algorithm of the previous section for the $k = 3$ case. For our discussion here, it will be convenient to assume that no two subsets of edges have the same total weight. (We can make sure that the assumption is satisfied in various ways. For instance, if Δ is the weight of the the lightest edge and the edges are ordered $e_1, e_2, \ldots, e_m$, we could use the revised edge weights $w'(e_i) = w(e_i) + \Delta/2^i$.) The key consequence of the assumption is that optimal cuts, shortest paths, etc. are unique, so that we can refer to *the* optimal cut etc. A less desirable consequence is that the cost of doing additions and comparisons of edge weights may go up by a factor of *n*, given the large number of bits needed to represent them, but given that our main goal here is to show that running times are $O(n^{ck})$ for some *c*, a factor of *n* will not make a significant difference.

In the $k = 3$ case, we observed that the dual C^D of the optimal cut was a subgraph of G^D that partitioned the embedding of G^D into three regions, each containing a distinct terminal. We then reduced the problem to the computation of 2-terminal cuts and shortest paths by first guessing (i.e., trying all possibilities for) some information about C^D. In particular, we guessed the *topology* (whether the cut consisted of two edge-disjoint cycles or not) and (in the latter case) the identity of the two degree-3 nodes a and b in C^D.

For general $k \geq 3$, we follow the same approach. Assume as before that we have previously decided on some fixed embedding of G. Suppose C is the optimal k-terminal cut, and once again let C^D be the dual of C viewed as a subgraph of a predetermined planar embedding of G^D. Then C^D must partition the embedding of G^D into precisely k regions, each containing a distinct terminal. Our notion of a *topology* for C^D is derived as follows: Consider the connected components of C^D, and call such a component *complex* if it contains more than one vertex that has degree three or more in C^D. Let $C_1^D, C_2^D, \ldots, C_q^D$ be an enumeration of the complex components of C^D, and for each i, $1 \leq i \leq q$, let N_i be the set of vertices with degree three or more in C_i^D. (Note that we must have $q \leq k-1$, since every connected component of C^D must enclose at least one terminal, and the infinite region of C^D contains one terminal.) Let $N = \cup_{i=1}^{q} N_i$. The *topology* of C^D is simply the (unordered) partition of N given by the sets $N_1, N_2, \ldots, N_q$. See Figure 4 for an example of a C^D and its topology, consisting of two sets N_1 and N_2. In the figure the vertices of degree three or greater are highlighted. Note that the degree-6 vertex v_{16} does not participate in the topology because its connected component contains only one vertex of degree three or greater.



$$N_1 = \{v_2, v_3, v_4, v_7, v_9, v_{12}\} \quad N_2 = \{v_1, v_5, v_6, v_8, v_{13}, v_{17}, v_{19}, v_{22}, v_{23}\}$$

FIGURE 4. The dual C^D of an optimal cut C and its topology $\{N_1, N_2\}$.

Lemma 2.2. If G is a connected planar graph with n vertices, let C be an optimal k-terminal cut for G, and let C^D be the planar dual of C, viewed as a subgraph of G^D. Then the number of distinct possibilities for the topology of C^D is $O((2n)^{2k-4})$.

Proof. Let an *arc* of C^D be any maximal path, all of whose interior vertices have degree two in C^D. Note that a cycle of C^D may be an arc, so long as the cycle contains at most one vertex of degree three or greater. Consider the graph whose vertex set is N and whose edge set is the arcs of C^D with both endpoints in N. This graph is planar, and has q connected components. Let m be the number of arcs it contains, and $k' \le k$ the number of regions. By Euler's formula, we have $|N| - m + k \ge q + 1$. Since all vertices of this graph have degree three or more by definition, we have $m \ge 3|N|/2$. Thus $|N| \le 2(k - q - 1)$.

Now let V^D denote the set of vertices of G^D, and consider the set of sequences $x_1, x_2, \ldots, x_{2k-4}$, with each x_i being either a member of V^D or the special symbol ``$|$''. Such a sequence corresponds to a collection of subsets $S_1, S_2, \ldots$ of $V^D$ in a natural way: Let us pretend our sequence is augmented by placing one copy of the special symbol ``$|$'' at the beginning and one at the end. Then S_i is simply the set of vertices that occur in between the ith and $i+1$st occurrences of ``$|$'' in the sequence. Note that every topology can be represented in this way, since all we need is $|N| + q - 1$ symbols, where N and q are as above, and $|N| + q - 1 \le 2k - q - 3 \le 2k - 4$ by the remark at the end of the preceding paragraph. Thus the total number of topologies is bounded by the total number of such sequences, which is $(|V^D| + 1)^{2k-4}$. (Note that this is a gross overcount on the topologies, since a given topology will be represented many times, and most sequences will not represent topologies at all. It probably represents the correct order of magnitude, however; for $k = 3$ it is $O(|V^D|^2)$, and we did have to consider $\Theta(|V^D|^2)$ topologies in the $k = 3$ case.)

To complete the proof of the lemma, we must bound $|V^D|$ in terms of n, the number of vertices in the original graph G. But note that each vertex in V^D corresponds to a region in the embedding of G, and (again by Euler's formula) the number of such regions is at most $2n - 4$. Thus the total number of topologies is $O((2n)^{2k-4})$ as claimed. □

Our algorithm for the general k-terminal cut problem will work by considering in turn each of the possibilities for the topology of the optimal cut. For each we will invoke a subroutine analogous to those used in our 3-terminal cut algorithm. The subroutine will either output the shortest cut that has the given topology, or else report (correctly) that the optimal cut cannot have the given topology. In order to specify the subroutine, we will need to know some structural results relating the topology of the optimum cut and its topology.

So suppose we are given a topology $N_1, N_2, \ldots, N_q$. If this is the optimal topology, then C^D contains connected components $C_1^D, C_2^D, \ldots, C_{q'}^D$, $q' \ge q$, where for $1 \le i \le q$, N_i is the set of vertices with degree three or more in C_i^D, and for $q < i \le q'$, C_i^D contains at most one vertex with degree three or more. The analogue of Lemma 2.1 for this general k case is that for each C_i^D, $i \le q$, we can efficiently identify a subtree T_i^D of C_i^D that spans all the vertices of N_i. This was the case in Lemma 2.1, where for $N_1 = \{a,b\}$ (Figure 2c), we identified a path between the two degree-3 vertices a and b by doing a shortest path computation. For the general case, we shall need both shortest paths and minimum spanning trees.

For a given topology, the trees T_i^D, $i \le q$, are constructed as follows. We treat the sets N_i, $i \le q$, in turn. (Order is not important.) Given N_i, we construct an auxiliary weighted complete graph H_i with N_i as its vertex set and with the weight of the edge linking u and v being the length of the shortest path in G^D between u and v. Compute the minimum spanning tree $T[H_i]$ of H_i, and let T_i^D be the subgraph of G^D formed by replacing each edge in $T[H_i]$ by the corresponding shortest path in G^D. We shall call T_i^D the *minimum spanning tree* of N_i (although note that it may not even be a tree if C^D does not have the given topology). Figure 5 portrays the C^D of

Figure 4 with the trees T_i^D (chosen according to the correct topology) highlighted. (For future reference, the figure also indicates which terminal is contained in which region of C^D.) The next lemma establishes the key properties satisfied by T_i^D when C^D has the given topology.

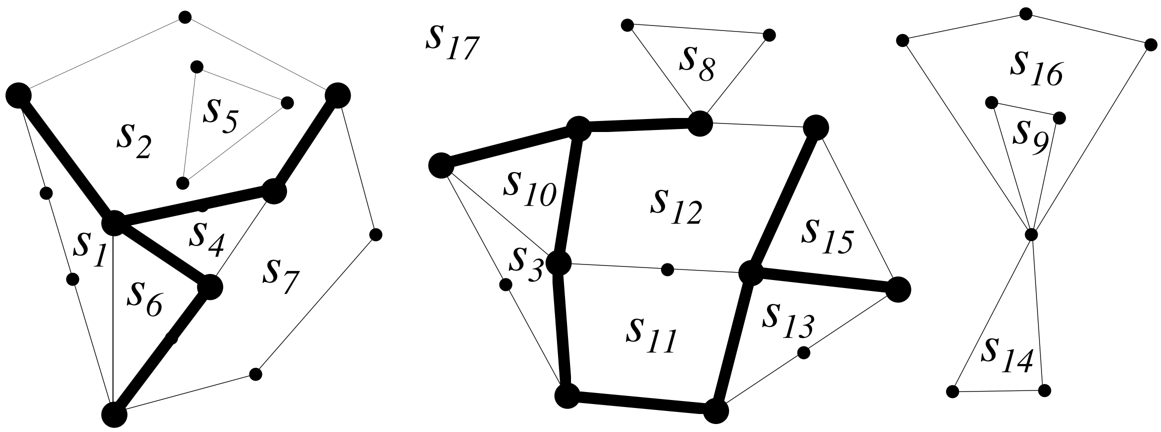


FIGURE 5. The dual C^D with the trees T_i^D highlighted and the locations of terminals indicated.

Lemma 2.3. Let C be the optimal k-terminal cut and C^D be its planar dual, viewed as a subgraph of G^D. Let C_i^D, $1 \le i \le q$, be the complex connected components of C^D, and let N_i be the set of vertices with degree three or greater in C_i^D, $1 \le i \le q$. Then (a) each C_i^D, $1 \le i \le q$, contains the minimum spanning tree T_i^D for N_i, and (b) no two of the paths in T_i^D corresponding to edges of $T[H_i]$ intersect except at a common endpoint (and hence T_i^D is indeed a tree).

Proof. Let us assume that the edges of $T[H_i]$ are labeled $F_1, F_2, \ldots, F_{|N_i|-1}$ in order of increasing weight. Thus if we let $T^j[H_i]$, $0 \le j < m$, be the graph consisting of the first j edges of $T[H_i]$, then for $0 \le j < m$, F_{j+1} is the shortest edge joining two different connected components (subtrees) of $T^j[H_i]$. Suppose $1 \le j \le m$, let $F_j = \{u,v\}$ and let S and T be the subtrees of $T^{j-1}[H_i]$ containing u and v respectively. Let P be the shortest path in G^D between u and v. Because of our assumption that no two sets of edges have the same aggregate weight, we can prove the desired properties of P directly, without the induction that was needed in the proof of Lemma 2.1. We shall prove three claims, from which the current lemma follows. Claims 2 and 3 correspond to parts (a) and (b) of the lemma, respectively. Claim 1 is used in the proof of each of the latter two.

Claim 1. P contains no vertex of N_i other than u and v.

Note that if P contained such a vertex w, then either w is not in S or w is not in T. If w is not in S, then the edge $\{u,w\}$ in H_i connects two different subtrees and has length at most $P[u,w]$, which is strictly less than the length of P. This contradicts our assumption about the ordering of edges in $T[H_i]$. A similar argument applies to the path $P[w,v]$ if w is not in T. This gets us part of the way toward proving that no two paths of T_i^D intersect, and will also be useful in Case 3 below.

Claim 2. P is entirely contained in C_i^D.

Suppose P is not entirely contained in C_i^D. As we traverse P from u to v, let x be the first vertex such that the edge leaving x is not included in C_i^D. Let y be the first vertex after x that is again included in C_i^D. As in the 3-terminal case, it is possible that $x = u$ and/or $y = v$, or that x and y are consecutive in P but $\{x,y\}$ in not in C_i^D. It is also possible that some portions of $P[x,y]$ hit components of C^D other than C_i^D. In any case, from planarity, $P[x,y]$ must be contained in one region of C_i^D, which it divides into two regions. Let B denote the boundary of that region, and let B_1 and B_2 be the two (closed) parts into which B is divided by $P[x,y]$. Call the region itself R_B. Then let R be the region of C^D (as opposed to just C_i^D) that is bounded by B together possibly with edges from connected components of C^D other than C_i^D, if such are contained in R_B. The region R is divided by $P[x,y]$ into (at least) two subregions R_1 and R_2. The boundary of R_i, $i \in \{1,2\}$ consists of B_i, $P[x,y]$, and possibly edges from other components. At least one of these two subregions of C^D contains no terminal; without loss of generality we may assume R_1 contains no terminal.

If we add $P[x,y]$ to C^D and remove a subpath of B_1 that contains no vertex of N_i as an interior point, the region R_1 will be merged with a single adjacent region. Since R_1 contains no terminal, the new subset of edges of G^D will still be the dual of a k-terminal cut. We will show that we can always find such a subpath of B_1 having higher weight than $P[x,y]$. This will mean that the replacement yields a k-terminal cut of lesser weight, contradicting the optimality of C. In the three cases below, we use ‘‘contains’’ as a shorthand for ‘‘contains as an interior vertex.’’

Case 1. B_1 contains no vertex of N_i.

This corresponds to Subcase 1.1 of Lemma 2.1 (see Figure 3a). Since P is a shortest path in G^D, $P[x,y]$ must be the shortest path between x and y in G^D. Thus it is shorter than B_1. Since the latter contains no vertex of N_i, we can replace all of B_1 by $P[x,y]$, thereby obtaining a shorter cut, in contradiction of the assumed optimality of C.

Case 2. B_1 contains vertices of N_i that are in different subtrees of $T^{j-1}[H_i]$.

This corresponds to Subcase 1.2 of Lemma 2.1 (see Figure 3a). Since B_1 contains vertices from N_i from different subtrees, it has two consecutive such vertices, say a and b. By our ordering of the edges F_h of $T[H_i]$, $w(P[x,y]) \le w(P) < w(B_1[a,b])$. Thus we can replace $B_1[a,b]$ by $P[x,y]$, obtaining a shorter cut and hence another contradiction.

Case 3. B_1 contains at least one vertex of N_i, and all such vertices are in the same subtree of $T^{j-1}[H_i]$.

This corresponds to Case 2 of Lemma 2.1 (Figure 3b). Suppose first that none of the vertices from N_i contained in B_1 are in the subtree S of $T_{j-1}[H_i]$ that contains u. Let z be the first vertex of N_i encountered as we traverse B_1 from x to y. Note that z cannot be x by Claim 1. Consider the path from u to z that follows P from u to x and then B_1 from x to z. It connects vertices of N_i in different subtrees, and so $w(P[u,x]) + w(B_1[x,z]) > w(P) > w([P[u,x]) + w(P[x,y])$. Consequently, we must have $w(B_1[x,z]) > w(P[x,y])$, and we can replace $B_1[x,z]$ by $P[x,y]$ for a shorter cut and a contradiction. An analogous argument holds for the case when none of the vertices from N_i contained in B_1 are in the subtree T containing v.

This exhausts the possibilities, so the path corresponding to F_j must be included in C_i^D.

Claim 3. P does not intersect any of the other paths in T_i^D corresponding to edges of $T[H_i]$ except at common endpoints.

Suppose P intersects another path P' of T_i^D at a vertex other than a common endpoint. We shall derive a contradiction, using the fact that Claims 1 and 2 hold for P' as well as for P. Of all the non-endpoint vertices the two paths share, let w be the closest one to u in P. Since all the edges of both P and P' are included in C_i^D by Claim 2, w must have degree three or greater in C_i^D, and hence by definition of *topology*, w must be included in N_i. This, however, contradicts Claim 1, which says that no interior vertices of P are in N_i. Thus Claim 3 holds, and the Lemma is proved. □

Lemma 2.3 treats the connected components C_i^D of C^D in isolation. Let us now look at how they interact. First note that the trees T_i^D are all vertex disjoint, since each is a subgraph of a different connected component of C^D. Thus they constitute a subforest of C^D. Let T^D represent this subforest, the union of all the edges in the T_i^D, and let T be the subset of edges in G whose planar dual is T^D. By Lemma 2.3, the optimal cut C for G will consist of T plus some additional edges, assuming we chose the correct topology for C. To find those additional edges, we delete T from G to obtain a new graph that we shall call $G[0]$. Assuming we have the correct topology, the optimal cut for $G[0]$ will be $C[0] = C - T$.

Let $G[0]^D$ be the embedded dual of $G[0]$ obtained by coalescing all the edges of each T_i^D in G^D to a single point. Note that $G[0]^D$ remains connected, and has lost none of the regions from G^D since only trees were coalesced. Assuming that we have the optimal topology, $C[0]^D$ will be a collection of $k-1$ simple cycles in $G[0]^D$, each of which is a biconnected component of $C[0]^D$. That is, a connected component of $C[0]^D$ may consist of more than one cycle, but all such cycles must share a single common vertex. (In particular, the coalesced vertices corresponding to the T_i^D will be such common vertices, although there may be others that were already present in C^D.) See Figure 6.

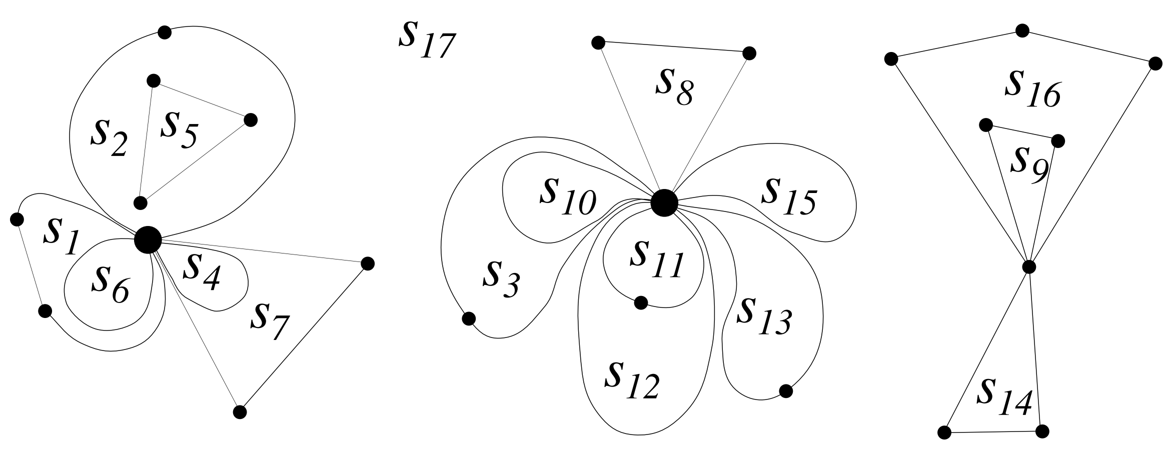


FIGURE 6. The graph $C[0]^D$ obtained by coalescing the spanning trees T_i^D in C^D.

Note that in the embedding of $G[0]^D$, some of the cycles of $C[0]^D$ will contain others. An innermost cycle will constitute the complete boundary of a region in $C[0]^D$, whereas other regions may have boundaries made up of the edges of several cycles. We shall show that if one treats the cycles in an appropriate ''inside out'' order, each of these cycles can be viewed as a minimum isolating cut in an appropriately constructed graph. Let us identify each region with the terminal s_i it contains, and construct a partial order $\leq$ on the terminals as follows: $s_i \leq s_j$ if and only if the outermost cycle bounding s_i's region is contained in the outermost cycle bounding s_j's region. (By convention, we assume that ''the outermost bounding cycle'' for the infinite region contains all other cycles, so that the terminal contained in the infinite region is > all other terminals.) Note that by this definition, the outermost bounding cycle for s_i must separate s_i from all terminals $s_j > s_i$.

Let π be an ordering of the terminals that is consistent with this partial order, i.e., if $s_i \leq s_j$, then $\pi(i) \leq \pi(j)$. For instance, for the $C[0]^D$ depicted in Figure 6, such an ordering would be $s_6, s_1, s_4, s_7, s_5, s_2, s_{10}, s_3, s_{11}, s_{12}, s_8, s_{15}, s_{13}, s_9, s_{16}, s_{14}, s_{17}$. We define a sequence of graphs and isolating cuts as follows. For $1 \leq i \leq k-1$, let A_i^D be the set of edges in $G[0]^D$ making up the outer boundary cycle for the region containing terminal $s_{\pi(i)}$. Let A_i be the corresponding set of edges in $G[0]$. Note that $C[0] = \cup_{i=1}^{k-1} A_i$. Now let $G[i]$ be the graph obtained from $G[0]$ by deleting the edges of $\cup_{j=1}^{i} A_j$.

Lemma 2.4. For $1 \leq i \leq k-1$, A_i is a minimum isolating cut for terminal $s_{\pi(i)}$ in graph $G[i-1]$.

Proof. Suppose not, and assume *j* is the lowest index for which the Lemma is violated. Let B_j be the minimum isolating cut for $s_{\pi(j)}$ in $G[j-1]$. Note that A_j is also a isolating cut for $s_{\pi(j)}$ in $G[j-1]$: by the minimality of *j*, all terminals $s_{\pi(i)}$, $i < j$, are already isolated in $G[j-1]$, and by our ordering of the terminals, all terminals of higher index are separated from $s_{\pi(j)}$ by the outermost bounding cycle for $s_{\pi(j)}$, i.e., A_j^D. Since the minimum weight isolating cut B_j is unique (by our assumption about edge weights) A_j does not equal it, $w(B_j) < w(A_j)$. But then the set $(C[0] - A_j) \cup B_j$ will have lower weight than $C[0]$. The set will also be a *k*-terminal cut for $G[0]$: Removing the edges of A_j^D from $C[0]$ will merge $s_{\pi(j)}$'s region with exactly one other, and the terminal for that region must have higher index, by our ordering of the terminals. But B_j separates $s_{\pi(j)}$ from all higher-indexed terminals. Thus $C[0]$ was *not* the optimum cut for $G[0]$, a contradiction. □

Given Lemmas 2.3 and 2.4, the following procedure will handle any given topology $N_1, N_2, \ldots, N_q$ appropriately, either outputting the minimum weight cut with that topology or reporting correctly that the optimal cut cannot have that topology. Assume that as a preprocessing step we have constructed our standard embedding of the dual graph G^D and computed all shortest paths between vertices of G^D.

Procedure CheckTopology $(N_1, N_2, \ldots, N_q)$

1. For $1 \leq i \leq q$, do the following.
   - 1.1 Construct the auxiliary graph H_i, the minimum spanning tree $T[H_i]$, and the subtree T_i^D.
   - 1.2. If any two paths in T_i^D corresponding to edges of $T[H_i]$ share a vertex other than a common endpoint, reject the topology.
2. If any two subtrees T_i^D share a common vertex, reject the topology.
   Otherwise, let T^D be the union of the edge sets T_i^D, let $G[0]$ be the graph obtained from G by deleting all edges in the dual set T.
3. Let $W^* = \infty$ (our initial estimate of the optimal cut weight).

For all possible permutations $s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(k)}$ of the terminals, do the following.

3.1. Set $C = T$.

3.2. For $1 \le i \le k-1$ do the following:

3.2.1. Find a minimum isolating cut A_i for $s_{\pi(i)}$ in $G[i-1]$.

3.2.2. Set $C = C \cup A_i$, and let $G[i]$ be the graph obtained by deleting the edges of A_i from $G[i-1]$.

3.3. If $w(C) < W^*$, set $W^* = w(C)$ and $C^* = C$ (the current best cut with this topology).

4. Output C^*.

It should be clear from the above discussion that this subroutine has the desired properties. Its running time is dominated by that for the minimum isolating cut computations occurring at Step 3.2.1. As discussed, each such computation can be performed using a standard 2-terminal minimum cut algorithm in time $O(n^2 \log n)$, assuming a machine model in which additions, subtractions, and comparisons take constant time. Given our proposed method for imposing the restriction that all sets of edges have unique weights, however, such a model is inappropriate. Even given the standard assumption that the original instance has edge weights that fit into a single computer word, our method for insuring the subset weight uniqueness restriction gives rise to weights whose binary representations involve $\Theta(n)$ bits. With such large numbers, the time bound for the isolating cut computations grows to $O(n^3 \log n)$. We perform $k-1$ such computations for each of the $k!$ permutations of the terminals, yielding total of $O(k^k)$ for each topology.

The overall algorithm then consists of performing Procedure CheckTopology for each possible topology, of which there are $O((2n)^{2k-4})$ by Lemma 2.2, and outputting the best cut found for any non-rejected topology. Thus we have our claimed result for general fixed k, stated here in slightly more precise form than given in the introduction:

Theorem 1b. Given a planar graph G with n vertices and k specified terminals, a minimum k-terminal cut can be constructed in time $O((4k)^k n^{2k-1} \log n)$.

Note that if we specialize this result to the previously considered case of $k = 3$, the time bound is $O(n^5 \log n)$, substantially larger than the $O(n^3 \log n)$ of Theorem 1a. This is a result of two factors: (1) A factor of n because we can't in general find the needed isolating cuts using planar 2-terminal cut algorithms, as we could when $k = 3$, and so have to use general 2-terminal cut algorithms. (2) A factor of n because we needed to operate with $\Theta(n)$-bit weights in order to insure that every subset of edges had a unique weight.

It may well be that more efficient algorithms can be derived by careful algorithmic design and analysis, but the bounds we have presented adequately fulfill our goal of showing that the k-terminal cut problem can be solved in polynomial time for fixed k and planar graphs. Moreover, as the NP-completeness result of the next section implies, it is likely that any algorithms for the general problem will have exponential (or at least super-polynomial) running times.

3. Complexity of Planar Multiterminal Cut for Arbitrary k

We saw in the previous section that the Multiterminal Cut problem is solvable in polynomial time for planar graphs and fixed k. In this and the next section we shall show that the problem becomes NP-hard if we allow either general graphs or general k. In this section we consider the case of planar graphs and general k. As is traditional for complexity results, we concentrate on a decision problem version of our problem. By proving it NP-complete for planar graphs, we imply that the corresponding optimization is NP-hard.

MULTITERMINAL CUT

INSTANCE: Graph $G = (V,E)$, subset $\{s_1, s_2, \ldots, s_k\} \subseteq V$ of terminals, for each edge e a positive integer weight $w(e)$, and a bound B.

QUESTION: Is there a subset $E' \subseteq E$ such that $w(E') \le B$ and $G' = (V, E - E')$ contains no path linking two distinct terminals?

We shall prove that MULTITERMINAL CUT is NP-complete, even if restricted to bounded-degree planar graphs with all edge weights equal to 1. Note that this is a stronger result than that quoted in the introduction, where for simplicity the question of degree bounds was not raised. We actually prove two separate results. The first allows weights to differ, but restricts vertex degrees to be 3 or less. (Note that the problem becomes trivial if vertex degrees must all be 2 or less.) The second result covers the equal-weight case, and follows by a slight modification of the proof of the first.

Theorem 2a. MULTITERMINAL CUT is NP-complete for planar graphs, even if edge weights are bounded and the maximum vertex degree is 3.

Proof. That planar MULTITERMINAL CUT is in NP is immediate. To complete the proof, we need to provide a polynomial transformation to it from some known NP-complete problem. The source problem we choose is PLANAR 3-SATISFIABILITY [9,19] (PLANAR 3-SAT for short). In the 3-SATISFIABILITY problem we are given a set $X = \{x_1, x_2, \ldots, x_n\}$ of variables and a collection $C = \{c_1, c_2, \ldots, c_m\}$ of 3-element *clauses*, i.e., subsets of the set of *literals* for X, where if x_i is a variable, the corresponding literals are x_i and $\overline{x}_i$. The question is whether there exists a satisfying truth assignment for X and C, where a *truth assignment* for X is a subset T of the literals for X that contains precisely one of $x_i, \overline{x}_i$ for each i, $1 \le i \le n$, and T satisfies C if for all clauses $c_j \in C$, $c_j \cap T \ne \phi$. (If $x_i \in T$, we say x_i is *true*; if $\overline{x}_i \in T$, we say x_i is *false*.) A natural graph to associate with this problem is the bipartite graph $G_{X,C}$ that has $X \cup C$ as its vertex set and has an edge between the vertices x_i and c_j if c_j contains either of the literals x_i or $\overline{x}_i$. PLANAR 3-SAT is 3-SATISFIABILITY restricted to those instances (X,C) for which $G_{X,C}$ is planar.

For our transformation to planar MULTITERMINAL CUT we shall actually use a restricted variant of PLANAR 3-SAT, one in which we allow clauses of size two as well as three, but require that each variable have degree exactly three in $G_{X,C}$, with one of its literals occurring in two clauses and the other in one. We can prove this variant NP-complete by a simple local-replacement transformation from ordinary PLANAR 3-SAT. Let X, C denote an instance of ordinary PLANAR 3-SAT, as specified above. First note that we may assume without loss of generality that both x_i and $\overline{x}_i$ are contained in $\cup_{j=1}^{m} c_j$ for every variable x_i: If $\overline{x}_i$ does not occur in any clause, we may assume without loss of generality that x_i goes in our truth assignment, and hence all the clauses containing x_i are always satisfied. Thus, we might as well delete those clauses (and the variable x_i) from our instance. Similarly if x_i is not contained in any clause, we might as well delete the variable x_i and all clauses containing $\overline{x}_i$. We may also assume that no clause contains both x_i and $\overline{x}_i$, as such a clause is always satisfied and so could also be deleted. Thus every variable occurs in at least two clauses, and all variable vertices in $G_{X,C}$ have degree at least two.

Our method for converting such an instance X,C to an instance X',C' of our restricted PLANAR 3-SAT variant involves replacing the old variables, adding some new clauses, and modifying the old clauses. First, we replace each variable x_i by a set of $degree(x_i)$ new variables $x_{i,1}, x_{i,2}, \ldots, x_{i,degree(x_i)}$ along with $degree(x_i)$ new clauses. Together, these form a length-$2degree(x_i)$ cycle in $G_{X',C'}$, as illustrated in Figure 7 for the case where $degree(x_i) = 5$. The added clauses insure that all the new variables must have the same truth value in any satisfying truth assignment. Note that we make this replacement even for variables with degrees two and three in $G_{X,C}$. These replacements together determine the structure of our new graph $G_{X',C'}$. The construction of X',C' is completed by modifying the old clauses so as to make the new instance consistent with the new graph. For each literal occurrence x_i ($\overline{x}_i$) in a clause c_j, there must be an edge in $G_{X',C'}$ between c_j and a new variable $x_{i,k}$. We replace the literal x_i ($\overline{x}_i$) by the new literal $x_{i,k}$ ($\overline{x}_{i,k}$). It is not difficult to verify that the new instance has the desired format and is satisfiable if and only if the original instance was.

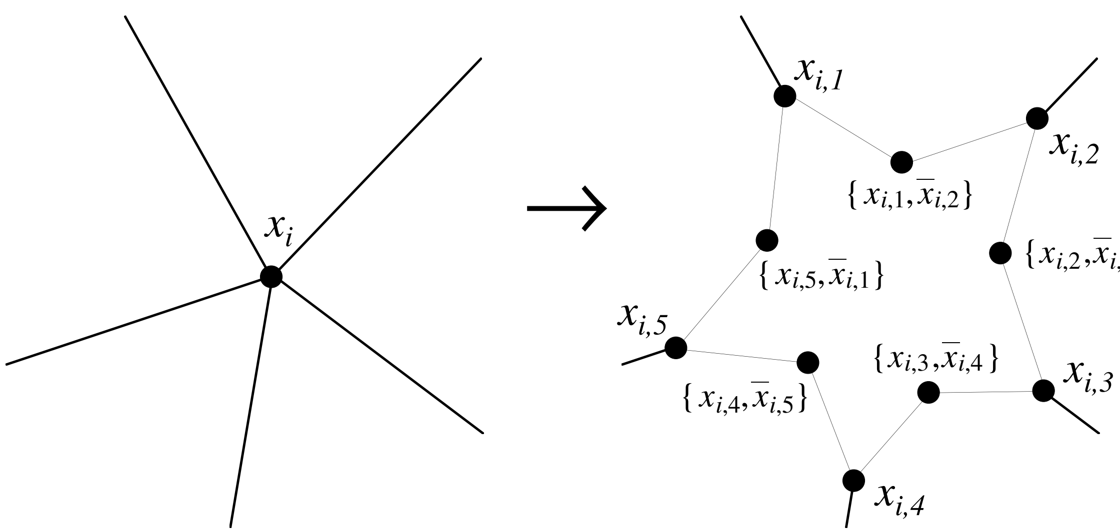


FIGURE 7. Reducing the number of occurrences per variable in PlANAR 3-SAT.

To complete our proof of Theorem 2a, we now show how to transform an instance X,C of restricted PLANAR 3-SAT into an instance of planar MULTITERMINAL CUT in which all weights are five or less and all vertices have degree three or less. We use the ‘‘component design’’ approach of [9]. Figure 8 shows the components we shall use to represent variables and clauses, with each edge labeled by its weight.

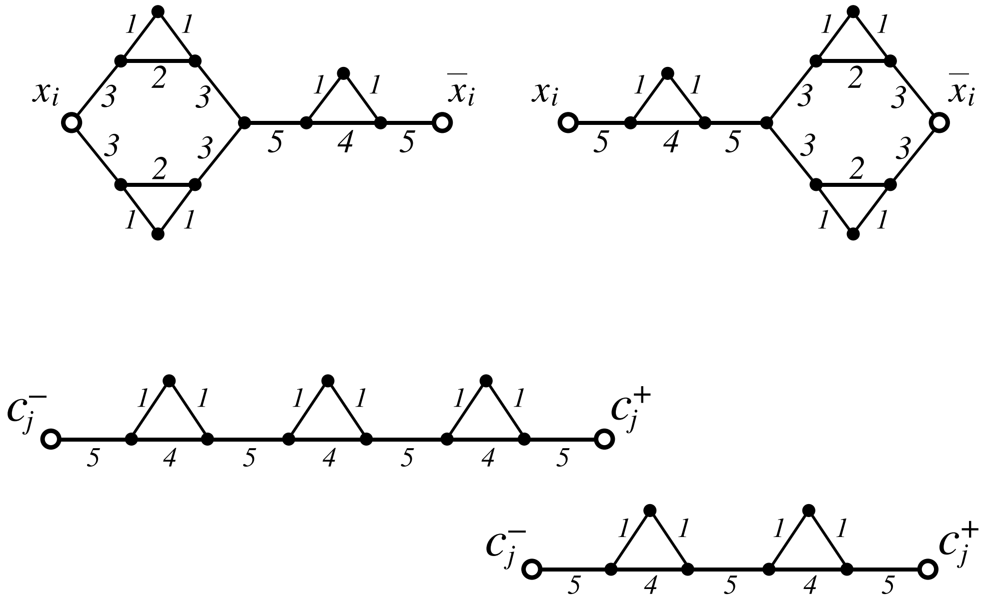


FIGURE 8. Variable and Clause Components in the proof of Theorem 2..

A variable x_i is represented by one of the two structures at the top of the figure, the first if the literal x_i occurs in two clauses, the second if $\overline{x}_i$ occurs in two clauses. (By our definition of restricted PLANAR 3-SAT one of these two possibilities must hold, and the other literal must occur exactly once.) The terminals in these structures are the vertices labeled x_i and $\overline{x}_i$. The *connector triangles* of these structures are the three triangles whose edge weights are 1,1,2 or 1,1,4. The *connector bases* are the weight-2 and weight-4 edges in these triangles, the *connector edges* are the weight-1 edges, and the *connector vertices* are the degree-2 vertices of the triangles. Each connector triangle, base, edge, and vertex will be thought of as representing the literal labeling the terminal nearest to it in the structure. Note that each literal is represented by precisely the same number of connector triangles as it has occurrences in clauses.

A clause c_j is represented by one of the two structures at the bottom of the figure, the first if $|c_j| = 3$, the second if $|c_j| = 2$. Here the terminals are the vertices labeled c_j^- and c_j^+. The *connector triangles* are the triangles whose edge weights are 1,1,4, and the *connector bases*, *connector edges*, and *connector vertices* are again the weight-4 edges, weight-1 edges, and degree-2 vertices of these triangles. Note that there are precisely as many connector triangles as there are literals in the clause. We shall assume that each connector is assigned to represent a distinct one of those literals

Note that all the structures of Figure 8 have maximum degree three. To complete our construction of the MULTITERMINAL CUT instance graph $G[X,C]$, we have only to hook together the clause and variable structures so as to maintain this property and yield a planar graph. This is done by adding *link* edges of weight 2 between variable connector vertices and clause connector vertices as indicated in Figure 9. (Each clause connector vertex is linked to a variable connector vertex that represents the same literal.) By the definition of restricted PLANAR 3-SAT, there will be precisely the right number of connector vertices for this to be done in a one-one fashion, with all connectors incident on precisely one weight-2 linking edge, implying that the maximum vertex degree in the resulting graph $G[X,C]$ is three. Moreover, it is not difficult to convince oneself that $G[X,C]$ is indeed planar, given that $G_{X,C}$ was.

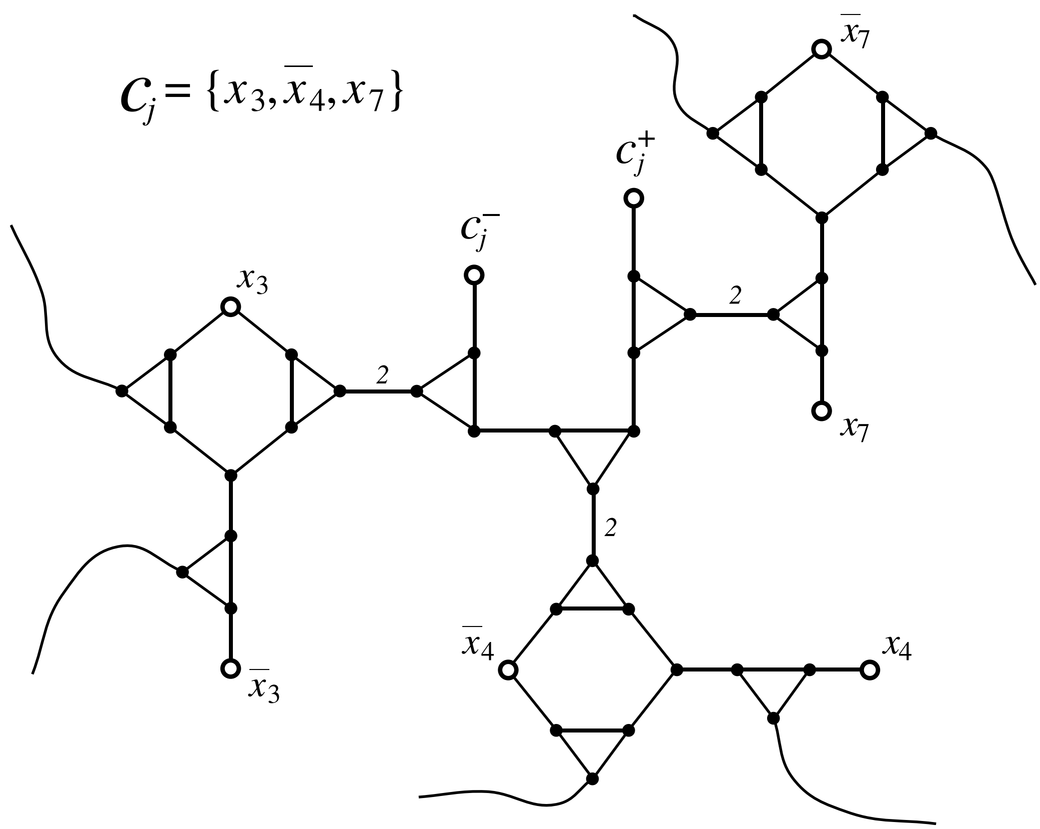


FIGURE 9. Hooking up the components in the proof of Theorem 2.

The only thing remaining to be specified is the upper bound B on the weight of the desired multiterminal cut. This is given by $B = 10n + 4m$, where n and m are the numbers of variables and clauses, respectively, in our restricted PLANAR 3-SAT instance.

The construction just described can clearly be accomplished in polynomial time. To complete the proof that it is a polynomial transformation, we need to show that the following two statements are equivalent.

(1) The PLANAR 3-SAT instance X,C has a satisfying truth assignment.

(2) There is a set E' of edges in $G[X,C]$ whose total weight is B or less and whose deletion would disconnect all $2n + 2m$ terminals from each other.

First suppose that the desired truth assignment T exists. Then the desired multiterminal cut is easy to construct. For each clause c_j pick a literal in $c_j \cap T$ (one must exist since T is a satisfying truth assignment) and delete all three edges of the corresponding connector triangle from the structure representing c_j in $G[X,C]$. For each variable, delete all three edges from each connector triangle representing the literal *not* in T. Finally delete each linking edge that is not adjacent to an already deleted connector triangle. Let E' be the set of deleted edges.

First, we note that E' has the desired cut properties. By our choice of when to delete link edges, we know that at least one endpoint of each remaining link edge must have degree one in the graph after E' is deleted. Thus no path between terminals can pass through a link edge. Consequently, if there is any path linking terminals, it must either link a pair of terminals $x_i, \bar{x}_i$ in the same variable structure, or a pair c_j^+, c_j^- in the same clause structure. In the former case, all such paths must pass through connector triangles representing both literals, and so must be broken, given that we deleted all connector triangles representing one of the literals. In the latter case, any such path must go through all the connector triangles of the c_j structure, and so must be broken because we deleted one of them.

We now claim that $w(E') = 10n + 4m = B$, and so E' obeys the weight bound and is the desired cut. In order to see this, let us for accounting purposes divide up the edges of $G[X,C]$ in a slightly different fashion. In particular, let us group each weight-2 link edge with the four weight-1 edges from connector triangles to which it is incident, and call the 5-edge ensemble a *link structure*. See Figure 10, where edges $\{a,c\}$, $\{b,c\}$, $\{c,d\}$, $\{d,e\}$, and $\{d,f\}$ constitute a link structure joining the structures for clause c_j and variable x_i. (Note that the base edges of the connector triangles, $\{a,b\}$ and $\{e,f\}$, are still viewed as being part of the corresponding clause and variable structures.)

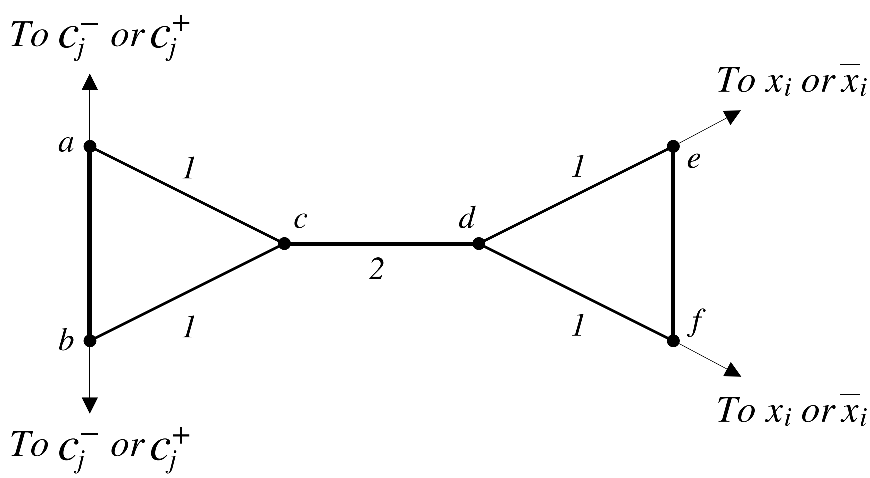


FIGURE 10. A typical connection between a clause and a variable component..

We claim that for any link structure, the total weight of deleted edges is exactly 2. This is clear if the weight-2 link edge is deleted, since that happens if and only if none of the weight-1 edges were deleted. Suppose one of the pairs of weight-1 edges was deleted. If it is the pair from the clause connector triangle, this means that the corresponding literal must be true. If it is from the variable connector triangle, this means that the corresponding literal must be false. Since both cannot happen simultaneously, only two of the weight-1 edges can be in E'. Thus the total weight of edges deleted from the link structure is two, and the total overall weight of edges deleted from link structures is twice the number of such structures, i.e., $6n$.

Now consider the clause and variable structures. We deleted one edge of weight 4 from each of the former, for a total weight of $4m$. From the latter, we deleted either one edge of weight

4 or two of weight 2, for a total weight of $4n$. This exhausts the possibilities, so we conclude that $w(E') = 6n + 4m + 4n = B$, as desired.

Conversely, suppose a set E' exists with the specified properties. We shall show that X, C has a satisfying truth assignment. We begin with a sequence of ''normal form'' lemmas. In what follows, we assume E' is a minimum weight set satisfying the specified properties.

Lemma 3.1. Suppose e is an edge incident on a degree-3 vertex v. If e has weight greater than or equal to the sum of the weights of the other two edges incident on v, we may assume that e is not in E'.

Proof. Suppose $e \in E'$, and consider the result of removing e from E' and replacing it by all the other edges incident on v and not already in E'. This will clearly not increase the weight of E'. It also cannot cause any two terminals to become linked by a path in the residual graph. If it did, that path would have to contain e. Since v now has degree one in the residual graph, v would consequently have to be one end of the path, and so would have to be a terminal. This is impossible since, as can be seen from Figures 7 and 8, no degree-3 vertex in our construction is a terminal. □

As a consequence of Lemma 3.1, we shall assume in what follows that none of the weight-5 edges in any clause structure are in E', and none of the weight-5 and weight-3 edges in any variable structure.

Lemma 3.2. If any edge in a cycle of $G[X, C]$ is in E', then at least two are.

Proof. If e is the only edge from a given cycle in E', then removing e from E' will decrease $w(E')$ without adding any new connections, contradicting our assumption that E' was a minimum weight cut. □

Lemma 3.3. At most one connector base (weight-4 edge) in any clause structure is in E'.

Proof. Suppose the structure for c_j had two weight-4 edges in E'. Let us consider the total clause structure, including all the connector edges, as in Figure 8. By Lemma 3.2, we may assume that E' contains at least one connector edge adjacent to each connector base it contains, and hence the total weight of edges from the clause structure that are in E' is at least 10. Consider the following modification to E': Remove one of the two connector bases and add in all the remaining connector edges (there can be at most four). This modification obviously does not increase the weight of E'. Nor can it create any new inter-terminal paths in the residual graph: Since all the connector edges are now deleted, such a created path could only link c_j^- and c_j^+, but those terminals remain separated since one of the weight-4 edges remains in E'. □

Lemma 3.4. For each variable x_i, E' either contains the connector base(s) representing the literal x_i or the base(s) representing the literal $\overline{x}_i$, but not both.

Proof. Let us consider the total variable structure, including all connector edges, as depicted in Figure 8. Since none weight-3 and weight-5 edges are in E' by Lemma 3.1, we must either delete the weight-4 base or both weight 2 bases if we are to disconnect terminal x_i from terminal $\overline{x}_i$. By Lemma 3.2, if we delete one of the weight-2 bases, we must delete both. So the only possibility we need to rule out is the situation in which we delete all three base connectors. Should we do this, we will also have to delete at least one connector edge from each connector triangle by Lemma 3.2. Thus the total weight of edges from the structure that would be in E' would be at least 11. Consider the following modification of E': remove the two weight-2 bases and add in all the remaining connector edges (there can be at most three). This modification cannot create any new paths in the residual graph, since x_i remains separated from $\overline{x}_i$ and no path involving a terminal other than these two can no traverse the structure. Moreover, it decreases the weight of E' (the total weight of edges from the x_i-structure that are in E' is now 10). Again we have a contradiction of the minimality of E'. □

Lemma 3.5. For each link structure, E' contains edges of weight totaling exactly two.

Proof. We first show that the total must be at least two. This will imply that it is exactly two, since by Lemmas 3.3 and 3.4, E' contains connector bases of total weight at least $4m + 4n$. This leaves at most $6n$ weight for the edges of link structures, and there are $3n$ such structures. So look again at the generic link structure pictured in Figure 10. By Lemmas 3.1, 3.3 and 3.4, the vertices a and b remain connected to clause terminals in the residual graph, and the vertices e and f remain connected to variable terminals. But this means that there will be a path in the residual graph from a variable terminal to a clause terminal unless we either delete the weight-2 link edge or two weight-1 edges from the same connector triangle. In either case the weight of the two deleted edges is at least two. □

We can now prove our claim that a satisfying truth assignment T must exist. For each variable x_i, put the literal whose connector base is *not* in E' into T. By Lemma 3.4, this is a valid truth assignment. We claim that it satisfies all the clauses. Consider a clause c_j, and let z be the literal corresponding to its broken connector base. By Lemma 3.3, a unique such z exists. Now consider the link structure joining the structure for c_j to the variable structure for z. Note that by our choice of z, the edge (a,b) is in E'. If z were not in T, then the edge (e,f) would also be in E'. Thus by Lemma 3.2, at least one connector edge from each connector triangle must be in E'. But note that if no other edges from the link structure are in E', there will still be a path from one of a,b to one of e,f within the structure. By Lemmas 3.1, 3.3, and 3.4, this would mean there is a path from a variable terminal to a clause terminal, a contradiction. Thus T is the desired truth assignment, and we have indeed constructed a polynomial transformation, and Theorem 2a is proved. □

Theorem 2b. MULTITERMINAL CUT is NP-complete for bounded degree planar graphs even if all weights are equal.

Proof. This is a fairly immediate corollary of the previous proof. Note that our construction would still have worked if we had replaced each edge of weight w by w parallel paths, each consisting of two weight-1 edges. Since no vertex in our construction had incident edges of total weight exceeding 11, this would yield a graph with maximum degree 11, and all edges with weight equal to 1. (The degree bound of 11 is not the best possible. Using a slight variant on the construction and considerably more complicated arguments, we believe it can be reduced at least to 6.) □

4. Complexity of Multiterminal Cut for Fixed $k \geq 3$ and Arbitrary Graphs

In this section we prove Theorem 4, which says that MULTITERMINAL CUT is NP-complete for all fixed $k \geq 3$ when arbitrary graphs are allowed. Note that it suffices to prove the problem NP-complete for $k = 3$, since the problem for higher values of k can trivially be derived from that for $k = 3$: simply add $k-3$ additional terminals with no edges incident on them. Thus we shall concentrate on the $k = 3$ case. In addition to proving this case NP-complete, we shall also present results that may be of assistance in coping with the NP-completeness. We present efficient algorithms that should allow us to reduce significantly the number of vertices and edges in 3-Terminal (and Multiterminal) Cut instances that arise in practice, and we show how to get within a factor of 4/3 of the optimal 3-terminal cut (and $2 - 2/k$ of the optimal k-terminal cut) with a relatively simple heuristic.

First, however, a brief digression into what at first seemed like a promising algorithmic approach to the 3-Terminal Cut problem, based on results on submodular set functions by Grötschel, Lovász, and Schrijver [14]. The key ''gadget'' in the NP-completeness proof we shall be presenting also serves as a counterexample to the applicability of this approach. (Indeed, before we discovered the gadget, we already had a proof that either 3-Terminal Cut was solvable in polynomial time by the submodular set function approach or it was NP-complete by a construction like the one given here.)

4.1. Submodular Set Functions: Algorithms and Counterexamples

In order to understand the results of [14], we first need some definitions. Let U be a finite set. A function f defined on the subsets of U is *submodular* if for any two subsets X and Y of U, $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$. Grötschel, Lovász, and Schrijver [14] show that if a submodular set function f can be computed in polynomial time, then f can also be *minimized*, i.e., set Y with $f(Y) = \min\{f(X) : X \subseteq U\}$ can be found, in polynomial time. (The algorithm involves an appropriate application of the ellipsoid method.)

A paradigmatic example of a submodular set function involves the usual (2-terminal) minimum cut problem. In this case, U is the set of nonterminal vertices $V - \{s_1, s_2\}$, and $f(X)$ is the total cost of the edges which have precisely one endpoint in the set $X \cup \{s_1\}$. The submodularity of this function is easy to verify, as is the fact that $\min\{f(X) : X \subseteq U\}$ is the weight of a minimum 2-terminal cut. We of course already know how to minimize this function f in polynomial time without resorting to the ellipsoid method, but the formulation is suggestive. Could it be possible that 3-terminal cuts might also be computable as minima of a submodular set function?

It is easy to define a set function for 3-terminal cuts that is analogous to the one above for the 2-terminal case. Let $U = V - \{s_1, s_2, s_3\}$ be the set of nonterminal vertices, and for any subset X of U, let $f(X)$ be the minimum cost of a 3-terminal cut that leaves no vertex in X connected to s_2 or s_3 (and each vertex in $U - X$ connected to one of the two). It is easy to see that the minimum value for this function equals the minimum weight for *any* 3-terminal cut. Moreover, we can use a polynomial-time algorithm for the 2-terminal cut problem to evaluate f in polynomial time: Given a subset X of U, find a minimum weight cut separating s_2 from s_3 in the graph obtained by deleting s_1 and all the vertices in X from G. Add to the weight of this cut the weight of all edges with precisely one endpoint in $X \cup \{s_1\}$. Therefore, if f were submodular (for all graphs), we could solve the 3-Terminal Cut problem in polynomial time.

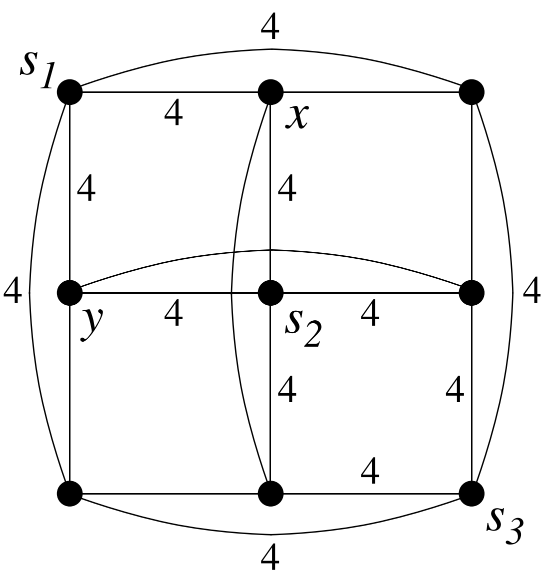


FIGURE 11. Graph C: Submodularity counterexample and NP-completeness gadget.

Unfortunately, this is not the case. Consider the 9-vertex graph C depicted in Figure 11. Note that in addition to the three terminals s_1, s_2, s_3, the graph contains two specified vertices x and y. The 12 edges incident on the terminals have weight 4, as indicated in the figure. The other 6 edges, unlabeled in the figure, have weight 1. Let c^* be the cost of an optimal 3-terminal cut for C. For each i,j, $1 \le i,j \le 3$, let an *i,j-cut* be a 3-terminal cut that leaves vertex x connected to s_i and vertex y connected to s_j, and let $c(i,j)$ be the cost of a minimum i,j-cut. The sets X and Y that cause f to violate submodularity are defined as follows:

Let X be the set of vertices connected to s_1 in an optimal $1,2$ cut. (Note that by definition of i,j-cut, x is in X and y is not.) Let Y be the set of vertices connected to s_1 in an optimal $2,1$ cut. (Note that y is in Y and x is not.) By definition, we have $f(X) = c(1,2)$ and $f(Y) = c(2,1)$. We also must have $f(X \cup Y) \ge c(1,1)$ and $f(X \cap Y) \ge \min\{c(2,3), c(3,2), c(2,2), c(3,3)\}$. Thus if f were to be submodular, we would need to have $c(1,2) + c(2,1) \ge c(1,1) + \min\{c(2,3), c(3,2), c(2,2), c(3,3)\}$. In light of the following lemma, however, this claim is false.

Lemma 4.1. For the graph C of Figure 11, the following properties hold:

(a) $c(1,2) = c(2,1) = c^*$,

(b) $c(i,j) \ge c^* + 1$ for all other pairs i,j, and

(c) $c(1,1) = c(2,2) = c^* + 1$.

Proof. As depicted in Figure 11, graph C has its vertices located at the nodes of a 3×3 grid. As an alternative naming convention for the vertices, let v_{ij} denote the vertex in the row i, column j, $1 \le i,j \le 3$. Thus the terminals are $s_1 = v_{11}$, $s_2 = v_{22}$, and $s_3 = v_{33}$, and the distinguished vertices x and y are v_{12} and v_{21}, respectively.

We first claim that $c(1,2) = c(2,1) = c^* = 27$. Note that the set consisting of the nine vertical edges in C is a weight-27 $1,2$ cut in which every vertex v_{ij} is left connected with the terminal s_i in its row, $1 \le i \le 3$. Similarly, the set consisting of the nine horizontal edges is a weight-27 $2,1$-cut in which every vertex v_{ij} is connected to the terminal s_j in its column. Could there be a 3-terminal cut of less weight? Note that each of the six nonterminal vertices v_{ij} is connected by weight-4 edges to both s_i and s_j. Thus in any 3-terminal cut at least six weight-4 edges must be deleted, one for each nonterminal v_{ij}. If the weight of the cut is to be 27 or less, no further weight-4 edges can be deleted, and so each nonterminal must remain connected to one terminal by a weight-4 edge. In particular, v_{ij} must remain connected either to s_i or s_j. Let us say that v_{ij} *belongs* to the terminal to which it remains connected.

Now consider the weight-1 edges. These all join nonterminal vertices, and together they form a cycle of length six with alternating vertical and horizontal edges: $v_{21}(y)$-v_{31}-v_{32}-$v_{12}(x)$-v_{13}-v_{23}-v_{21}. Suppose two consecutive edges of this cycle remain undeleted, say without loss of generality $\{v_{ij}, v_{ik}\}$ and $\{v_{ik}, v_{lk}\}$. Note that we must have $i \ne j$, $i \ne k$, $j \ne k$, $i \ne l$, and $l \ne k$. In order to rule out any inter-terminal paths, v_{ij}, v_{ik}, and v_{lk} must then all belong to the same terminal. That terminal must be either s_i or s_j, since those are the only two to which v_{ij} can belong. Similarly, it must be either s_i or s_k, since those are the only two to which v_{ik} can belong. Thus, since $j \ne k$, it must be i. But this is impossible, since v_{lk} can only belong to s_l or s_k, and neither of these can equal i, as already observed. Thus the cut must contain one of every two consecutive edges in the cycle of weight-1 edges, or at least three such edges, for a total weight of at least 27. The only way that precisely three can be chosen is if we take every other edge in the cycle, i.e., either the three vertical edges or the three horizontal ones, as was done in the horizontal and vertical cuts mentioned in the previous paragraph. We now can conclude that those cuts were optimal, and hence $c^* = c(1,2) = c(2,1) = 27$. Thus Part (a) of the Lemma holds.

For Part (b), we shall show that any cut other than the horizontal and vertical cuts mentioned above must have weight 28 or more. Assume there were such a cut of weight 27. As argued above, its intersection with the cycle of weight-1 edges is either the set of three vertical edges of the cycle or the set of three horizontal ones. By the symmetry of C, we may assume that it contains the three vertical edges. Since it is not the vertical cut, it must thus fail to contain at least one vertical weight-4 edge. For specificity (and again without loss of generality because of the symmetry of C), we may assume that the missing vertical weight-4 edge is $\{s_1, v_{21}\}$. But then, since the horizontal weight-1 edge $\{v_{21}, v_{23}\}$ is also not in the cut, the vertex v_{23} is connected to the terminal s_1 in the residual graph. This means that the cut must contain both the edge joining v_{23} to s_2 and the edge joining it to s_3. Since, as remarked earlier, the cut must contain at least one weight-4 edge incident on each of the six nonterminals, this yields a total of at least seven weight-4 edges, and so the cut will have weight at least 28, a contradiction. Thus Part (b) is proven.

To prove Part (c), we need to show that there exist $1,1$- and $2,2$- cuts of weight 28. It suffices (again by symmetry) to consider the $1,1$ case. A weight-28 $1,1$-cut is obtained by modifying the weight-27 cut consisting of all vertical edges. Instead of deleting the vertical weight-4 edge $\{y, s_1\}$, we delete the two horizontal edges incident on y: the weight-4 edge $\{y, s_2\}$ and the weight-1 edge $\{y, v_{23}\}$. It is easy to see that the result continues to be a 3-terminal cut, although it now has weight 28, and y as well as x is connected to s_1. Thus the cut is a $1,1$-cut of the desired weight, and Part (c) holds. □

4.2. The NP-Completeness of 3-Terminal Cut

We are now ready to prove Theorem 3, which we restate in terms of the $k = 3$ special case to which it reduces.

Theorem 3. If arbitrary graphs are allowed, MULTITERMINAL CUT for $k = 3$ (i.e., 3-TERMINAL CUT) is NP-complete even if all weights are equal to 1.

Proof. It is immediate that 3-TERMINAL CUT is in NP. We shall show that 3-TERMINAL CUT is NP-complete if weights 1 and 4 are allowed. This will suffice to prove the theorem, since a weight-4 edge could always be replaced by four parallel length-2 paths of weight-1 edges.

Our proof is by a polynomial transformation from the SIMPLE MAX CUT problem [9,10]. In SIMPLE MAX CUT, we are given a graph $G = (V,E)$ (without weights) and a number K, and are asked whether there is a partition of the vertices of G into two sets V_1 and V_2 such that there are at least K edges between V_1 and V_2. Given (G,K), we construct a corresponding instance (F,B) of 3-TERMINAL CUT as follows.

The graph F has three terminals s_1, s_2, s_3 and contains the vertices of G, but not the edges. For each edge $\{u,v\}$ of G, the graph F instead contains a copy of our ''gadget graph'' C, with the vertices s_1, s_2, s_3 of C identified with their named counterparts in F, and with x and y identified with u and v. The other vertices of different copies of C (i.e., the four nonterminals other than x and y) are distinct. Thus the total number of vertices in F is $3 + |V| + 4|E|$. We claim that G has a cut of size K or greater if and only if F has a 3-terminal cut of weight $B = 28|E| - K$ or less. Given that F and B can clearly be constructed in polynomial time, Theorem 3 will follow from this claim.

So suppose there is a cut V_1, V_2 for G of size $K' \geq K$. Consider the 3-terminal cut induced by the following assignment of the vertices of F to the terminals: First, vertices in V_1 are assigned to s_1 and vertices in V_2 are assigned to s_2. At this point, each copy of C has its x and y vertices assigned to one of s_1 or s_2, say s_i for x and s_j for y. Assign the remaining nonterminals of this copy of C to terminals according to a minimum weight i,j-cut for C. Note that the contribution to our 3-terminal cut from edges in this copy of C will thus be $c(i,j)$. In particular, if x and y were in the same set the contribution will by Lemma 4.1 be 28; if they were in different sets it will be 27. The overall weight thus becomes $28|E| - K' \leq 28|E| - K$, as desired.

Conversely, suppose a 3-terminal cut of size $B' \leq 28|E| - K$ exists. Let V_i be the set of vertices in V that are left connected to s_i, $i = 1,2,3$. For each edge $\{u,v\}$ of G with $u \in V_i$, $v \in V_j$, the cut removes edges of total weight at least $c(i,j)$ from the corresponding copy of C. By Lemma 4.1, $c(i,j) \geq 28$ unless $\{i,j\} = \{1,2\}$, and so there must be at least K edges between V_1 and V_2. Thus we can assign the vertices of V_3 to the two sets V_1 and V_2 arbitrarily and still obtain the desired cut for G. □

Note that the graph constructed in our proof of Theorem 3 does not have bounded vertex degrees. This is unavoidable, so long as we assume all edge weights are equal. If k is fixed, all edge weights are equal, and there is a bound d on vertex degree, then Multiterminal Cut can be solved in polynomial time! Observe that in this case the weight of a cut is simply the number of edges it contains, and an optimal cut can contain no more than kd edges (since the cut that simply breaks all the edges incident on each terminal is no bigger than this). But since k and d are fixed, kd is a constant independent of n. Consequently, we can use exhaustive search and still take time that is polynomially bounded in n.

If we remove the restriction to equal-weight edges, however, the fixed-k problem becomes NP-complete even if all vertex degrees are three or less. Simply replace each vertex v in our construction that has $degree(v) > 3$ by a cycle of $degree(v)$ vertices, each of which is adjacent to one of the former neighbors of v, and let the weights of the cycle edges be sufficiently high that they can't be chosen for an optimal cut.

4.3. Reducing the Instance Size

The results of Sections 4.1 and 4.2 effectively dash any hope of finding optimal k-terminal cuts, $k \geq 3$ efficiently by means of 2-terminal cut (i.e. max flow) algorithms. Such algorithms may still be useful, however, as we shall see in this and the next section. Recall that an *isolating cut* for a terminal s_i is a set of edges that separates s_i from the other terminals, and that minimum weight isolating cuts can be found in $O(nm\log(n^2/m))$ time by performing max flow computations in a modified graph. The following Lemma implies that the computation of minimum weight isolating cuts can be used to reduce the number of vertices in an instance. Suppose $G = (V,E)$ is a connected graph with specified terminals $s_1, s_2, \ldots, s_k$.

Lemma 4.2. Suppose $i \in \{1,2,...,k\}$, $k \geq 3$. Let E_i be a minimum weight isolating cut for terminal s_i, and let V_i be the set of vertices that remain connected to s_i when the edges of E_i are removed from G. Then there exists an optimal k-terminal cut for G that leaves all the vertices of V_i connected to s_i.

Proof. Without loss of generality, we may assume that $i = 1$. Suppose the Lemma were false for some $k \geq 3$, and let G be a minimal counterexample, i.e., a counterexample having the fewest vertices, and, among those counterexamples with that number of vertices, the fewest edges. Let E^* be a minimum weight k-terminal cut for G, and let c_1 and c^* be the weights of E_1 and E^*, respectively. Label every vertex u of G with a pair (j,l) where $j = 1$ or 2 depending on whether u is in V_1 or not, and l is the index of the terminal to which u is left connected under E^*. (This index is defined, since u must be left connected to some terminal if E^* is to have minimum weight.)

The above labelling partitions the vertices into $2k$ groups. We first claim that, by the minimality of G, each group can consist of at most one vertex. Suppose u and v have the same label, and consider the graph G' in which we merge u and v. The cuts induced on G' by E_1 and E^* will have the same weights as did the original cuts, and clearly no better cuts of the corresponding types can exist. Thus G' would be a counterexample with fewer vertices, a contradiction of our minimality assumption.

Next we claim that if $\{u,v\}$ is an edge of G and the labels of u and v are (j_u, l_u) and (j_v, l_v) respectively, then it cannot be the case that both $j_u \neq j_v$ and $l_u \neq l_v$. Suppose such an edge e were in G, and note that e must be in both E_1 and E^*. Thus, consider the graph G' obtained by removing e from G. The sets $E_1 - \{e\}$ and $E^* - \{e\}$ will continue to be isolating and k-terminal cuts respectively for G' and the weight of each will drop by $w(e)$. Furthermore, there cannot be a k-terminal cut E' of weight less than $c^* - w(e)$ (isolating cut $E_1{}'$ of weight less than $c_1 - w(e)$) in G', as otherwise we could get a k-terminal cut in G of weight less than c^* (isolating cut of weight less than c_1) by simply adding e to E' (to $E_1{}'$). Thus $E_1 - \{e\}$ and $E^* - \{e\}$ continue to be optimal in G', so that G' would be a counterexample with fewer edges and the same number of vertices, again a contradiction of our minimality assumption.

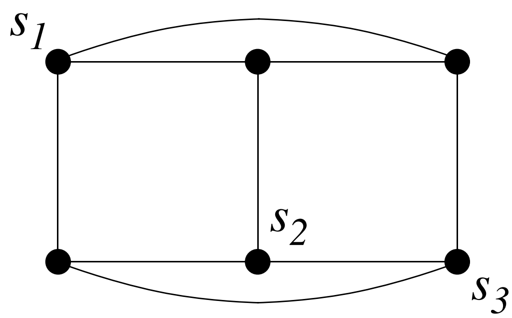


FIGURE 12. Schematic of the assumed minimal counterexample in Lemma 4.2 (if $k = 3$).

Thus for $k = 3$ our minimal counterexample is a subgraph of the 6-vertex graph depicted in Figure 12, where the vertices of G are placed on a 2×3 grid, with the vertex in row i and column j (v_{ij}) being the one with label (i,j). For $k > 3$, we would have a $2 \times k$ grid, with all k vertices in the top row connected in a clique, as are all k vertices in the bottom row. Note that for G to be a counterexample, at least one of the nonterminal vertices in the graph must be present in G. The set V_1 of vertices connected to s_1 under E_1 is simply the set of all vertices in the top row that are present in G.

Now observe that since E_1 consists of the vertical edges in this figure, these edges have total weight c_1. Similarly, E^* consists of the horizontal edges, and these edges consequently have total weight c^*. Thus the total weight of all the edges in G is $c_1 + c^*$. Consider the set $\hat{E}_1$ of edges incident on s_1 in G. This is clearly an isolating cut, and so $w(\hat{E}_1) \geq c_1$. But then consider the k-terminal cut $\hat{E}^*$ consisting of all edges incident on s_2 through s_k. This is disjoint from $\hat{E}_1$, and so can have weight at most $c^* + c_1 - w(\hat{E}_1) \leq c^*$. Hence it has weight precisely c^* and is itself an optimal k-terminal cut. But note that it leaves s_1 connected to all the vertices in V_1. Thus G actually satisfies the lemma statement, and there can be no counterexample. □

Using Lemma 4.2, we can reduce the number of vertices in our instance by $|V_i - 1|$: Simply construct a new graph in which all the vertices of V_i are merged into the terminal s_i. An optimal k-terminal cut for this shrunken graph will induce an optimal cut for the original one.

In order to get the maximum effect from applying Lemma 4.2 in this way, we should start with the minimum isolating cut for s_i such that V_i is as big as possible. Let us say that a set V_i is an *isolation set* for a terminal s_i if it is the set of vertices left connected to s_i by some minimum weight isolating cut. Let us call it an *optimum isolation set* for s_i if it has maximum cardinality over all isolation sets for s_i. From observations made in [7, pp.10-13], it can be seen that the optimum isolation set for a given terminal is unique and contains all other isolation sets for that terminal. It can be found by performing one maximum-flow computation followed by some linear-time post-processing. A corollary of the following Lemma is that k optimum isolating set computations suffice to shrink G as far as it can go.

Lemma 4.3. Let V_i be the optimum isolation set for s_i, $1 \le i \le k$, and let $\overline{G}$ be the graph obtained from G by merging all the vertices of V_1 into s_1. Then in $\overline{G}$ the optimum isolation set for s_1 is $\{s_1\}$, and the optimum isolation set for s_i is $V_i - V_1$, $2 \le i \le k$.

Proof. For $i = 1,2,...,k$, let w_i and $\overline{w}_i$ denote the weights of a minimum weight isolating cut for s_i in G and $\overline{G}$, respectively. Let $\overline{V}_i$ be the optimum isolation set for s_i in $\overline{G}$.

We begin with $\overline{V}_1$. Note that the isolating cut for s_1 that separates s_1 from all other vertices in $\overline{G}$ has weight w_1 by hypothesis, and we must have $\overline{w}_1 \le w_1$. On the other hand, $\overline{V}_1 \cup V_1$ induces an isolating cut for s_1 in G that has weight $\overline{w}_1$, so we must have $\overline{w}_1 \ge w_1$. Thus equality holds and $\overline{V}_1 \cup V_1$ is an isolation set for s_1 in G. But this means that $\overline{V}_1 \cup V_1$ is contained in V_1, by the properties of maximum isolation sets mentioned above, and hence $\overline{V}_1 \subseteq V_1$. Since the only member of V_1 that exists in $\overline{G}$ is s_1, this implies that $\overline{V}_1 = \{s_1\}$, as claimed.

Now consider $\overline{V}_2$. (The arguments for $\overline{V}_i$, $k > 2$ are analogous and hence omitted.) Let $\hat{V}_2 = V_2 - V_1$. We first show that $\hat{V}_2$ induces a minimum weight isolating cut for s_2 in G (although $\hat{V}_2$ will not be an optimum isolation set for s_2 in G unless $V_1 \cap V_2 = \phi$). Partition the vertices of V into the four sets $A = V_1 - V_2$, $B = V_2 - V_1$, $C = V_1 \cap V_2$ and $D = V - (V_1 \cup V_2)$. If X and Y are two disjoint subsets of the vertices of G, let $E(X;Y)$ denote the set of edges that link vertices in X with vertices in Y, and let $w(X;Y)$ denote their weight.

By hypothesis $w_1 = w(A;B) + w(A;D) + w(B;C) + w(C;D)$ and $w_2 = w(A;B) + w(B;D) + w(A;C) + w(C;D)$. Thus $w_1 + w_2 = W + w(A;B) + w(C;D)$, where W is the total weight of the edges that link vertices in different sets of our partition. Now consider the isolating cuts for s_1 and s_2 induced by $V_1 - V_2$ and $V_2 - V_1$, respectively, and let $\hat{w}_1$ and $\hat{w}_2$ be their weights. Then $\hat{w}_1 = w(A;B) + w(A;C) + w(A;D)$ and $\hat{w}_2 = w(A;B) + w(B;C) + w(B;D)$, so that $\hat{w}_1 + \hat{w}_2 = W + w(A;B) - w(C;D) \le w_1 + w_2$. Since by hypothesis we must have $w_i \le \hat{w}_i$, $i \in \{1,2\}$, this implies that equality must hold in both cases, and so $\hat{V}_2 = V_2 - V_1$ must induce a minimum weight isolating cut for s_2 in G. Since none of the vertices of $\hat{V}_2$ were merged with s_1 in the construction of $\overline{G}$, $\hat{V}_2$ must consequently also induce a minimum weight isolating cut for s_2 in $\overline{G}$, and so $\hat{w}_2 = \overline{w}_2 = w_2$. Thus $\hat{V}_2$ is an isolation set for s_2 in $\overline{G}$, and by the properties of optimal isolation sets, we must have $\hat{V}_2 = V_2 - V_1 \subseteq \overline{V}_2$.

On the other hand, since $\overline{V}_2$ induces an isolating cut of weight $\overline{w}_2 = w_2$ for s_2 in $\overline{G}$, it induces an isolating cut of the same weight in G, and so $\overline{V}_2$ is an isolation set for s_2 in G. Thus we must have $\overline{V}_2 \subseteq V_2$. Since the only vertices of V_2 that remain in $\overline{G}$ are those in $V_2 - V_1$, this implies that $\overline{V}_2 = V_2 - V_1$, as claimed.

This completes the proof of the lemma. □

Lemma 4.3 indicates both the efficiency with which we can apply Lemma 4.2 to reduce the instance size, and the bounds on how much shrinkage can be obtained. In particular, k optimum separation set computations suffice to yield all the shrinkage one can expect: Note that the proof of Lemma 4.3 would apply just as well if we renamed the terminals in any order. So let $G_0 = G$, and inductively obtain G_i from G_{i-1} by performing an optimum isolation set computation for s_i

and merging all the vertices in the set obtained into the terminal s_i. Lemma 4.2 says that an optimal k-terminal cut in G_i induces one in G_{i-1} (and, by induction, in G), and Lemma 4.3 says that the optimum isolation set for s_i in G_i is $\{s_i\}$ (and by induction, the optimum isolation set for s_h, $1 \le h \le i$, is $\{s_h\}$). Thus in G_k, the optimum isolation set for each terminal consists of the terminal itself, but a minimum weight k-terminal cut still induces one in the original graph G. Thus G_k is a maximally shrunken graph that can still induce an optimal k-terminal cut.

4.4. Near-Optimal Multiterminal Cuts

If one is willing to settle for cuts that are only *near*-optimal, one can exploit a bit further our ability to construct optimum isolating cuts. Consider the following straightforward heuristic.

Isolation Heuristic

1. For $1 \le i \le k$ construct a minimum weight isolating cut $\hat{E}_i$ for terminal s_i.
2. Determine h such that $\hat{E}_h$ has maximum weight among all the $\hat{E}_i$'s.
3. Let $\hat{E}$ be the union of all cuts $\hat{E}_i$ except $\hat{E}_h$.
4. Return $\hat{E}$.

Note that the Isolation Heuristic clearly outputs a k-terminal cut. Moreover, it can be implemented to run in $O(knm\log(n^2/m))$ time by using the max flow algorithm of [12] to compute each of the k required isolating cuts. This is the heuristic to which we referred in Theorem 4 of the Introduction. A more precise statement of that theorem can now be given.

Theorem 4. The Isolation Heuristic constructs a k-terminal cut whose weight is guaranteed to be no more than $2(k-1)/k$ times the optimal weight.

Proof. We first consider the upper bound. Let $\bar{E}$ be an optimal k-terminal, and let $\bar{w} = w(\bar{E})$. For $1 \le i \le k$, let $\bar{V}_i$ be the set of vertices left connected to s_i by $\bar{E}$, and let $\bar{E}_i$ be the set of edges in $\bar{E}$ with one endpoint in $\bar{V}_i$. Note first that for each i, the set $\bar{E}_i$ is an isolating cut for s_i. Hence $w(\bar{E}_i) \ge w(\hat{E}_i)$. On the other hand, each edge in $\bar{E}$ is in exactly two different sets $\bar{E}_i$, and so $\Sigma_{i=1}^{k} w(\bar{E}_i) = 2\bar{w}$. Thus

$$w(\hat{E}) \;\le\; \frac{k-1}{k}\sum_{i=1}^{k} w(\hat{E}_i) \;\le\; \frac{k-1}{k}\sum_{i=1}^{k} w(\bar{E}_i) \;=\; 2\frac{k-1}{k}\bar{w}$$

as claimed. □

The bound given by Theorem 4 is tight. More precisely, for each $k \ge 3$ and any $\varepsilon > 0$, there exist instances for which the cut constructed by the Isolation Heuristic has weight $(2-\varepsilon)(k-1)/k$ times the optimal. The generic construction contains $2k$ edges and vertices, with the nonterminal vertices $v_1, v_2, \ldots, v_k$ linked in a simple k-cycle of weight-1 edges, and terminal s_i linked by a weight-$(2-\varepsilon)$ edge to vertex v_i, $1 \le i \le k$. See Figure 13 for an illustration of the case for $k = 4$.

Note that for each terminal s_i, the minimum weight isolating cut is unique and consists of the edge of weight $2-\varepsilon$ connecting it to v_i. Thus the weight of the cut constructed by the heuristic is $(k-1)(2-\varepsilon)$. An optimal cut, on the other hand, would consist simply of all the k weight-1 edges in the cycle linking the nonterminals, for a total weight of k. The ratio thus has the claimed value. (Note that if we are willing to assume that our heuristic always breaks ties in the worst

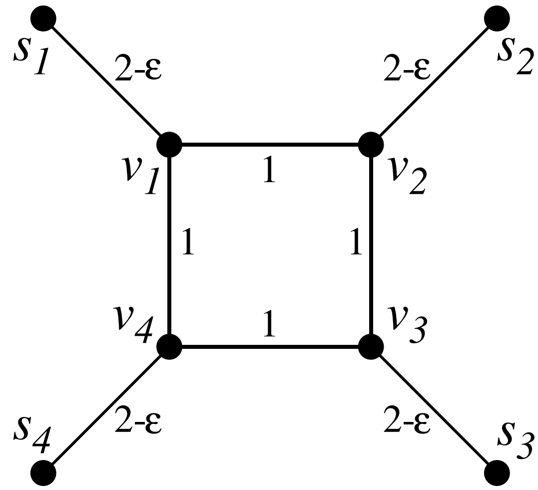


FIGURE 13. Worst-case graph for the Isolation Heuristic when $k = 4$.

possible way, we can take $\varepsilon = 0$ in the above example and obtain a ratio that precisely matches the upper bound of Theorem 4.)

For $k = 3$, the ratio guaranteed by Theorem 4 is 4/3 and fairly close to 1. An interesting question is whether there are any polynomial time heuristics that provide better guarantees. In particular, is it conceivable that we could guarantee ratios arbitrarily close to 1? Formally, is there a *polynomial time approximation scheme* for 3-Terminal Cut, i.e., a sequence of polynomial time algorithms A_t, where A_t is guaranteed to find a 3-terminal cut of weight at most $1 + 1/t$ times the optimal weight? (The algorithms need not be polynomial in t, only in the instance size.)

There is significant evidence that the answer to this question is no. In particular, our proof of Theorem 3 implies that the 3-Terminal Cut problem is what is known as *MAX SNP-hard* [21], and this implies that 3-Terminal Cut cannot have a polynomial time approximation scheme unless P = NP. Let us explain.

An optimization problem is in MAX SNP if the optimal value sought can be described as $\max_S |\{\bar{x} : \phi(\bar{x}, I, S)\}|$, where I is an instance, S is a relational structure such as a truth assignment in the case of 3-SAT, x is the object being counted, such as satisified clauses, and ϕ is a first order logical expression. A second relevant example of a problem in the class MAX SNP is *Max Cut*, the optimization version of the problem SIMPLE MAX CUT that we used to prove that 3-TERMINAL CUT is NP-complete in Section 4.2. In the Max Cut problem, the instance I is a graph, the structure S is a partition of the vertices into two sets, and the objects $\bar{x}$ are edges, where $\phi(\bar{x}, I, S)$ is true if and only if $\bar{x}$ does not have both its endpoints in the same set of the partition S.

MAX SNP was introduced in [21], where it was observed that although each problem in MAX SNP could be approximated to within *some* constant ratio in polynomial time, no problem in the class was known to have a polynomial time approximation scheme. The concept of MAX SNP-hardness was also introduced in [21], and defined in such a way that it could be applied both to problems, such as MAX 3-SAT and Max Cut, that are in MAX SNP and to problems, such as 3-Terminal Cut, that are minimization problems and hence by definition not in MAX SNP. The key observation about MAX SNP-hardness in [21] was that no MAX SNP-hard problem could have a polynomial time approximation scheme unless all problems in MAX SNP had them. Arora et al. [1] have now shown that the latter event cannot occur unless P = NP. Consequently no MAX-SNP-hard problem can have a polynomial time approximation scheme unless P = NP.

Proofs of MAX SNP-hardness involve *linear reductions*, a generalization of the familiar polynomial transformations used in NP-hardness proofs. Let A and B be two optimization problems (either maximization or minimization). We say that A linearly reduces to B if there are two polynomial time algorithms f and g and constants $\alpha, \beta > 0$ such that

1. Given an instance a of A, algorithm f produces an instance $b = f(a)$ of B such that the cost of an optimal solution for b, $opt(b)$, is at most $\alpha \cdot opt(a)$, and
2. Given a, $b = f(a)$, and any solution y of b, algorithm g produces a solution x of a such that $|cost(x) - opt(a)| \le \beta |cost(y) - opt(b)|$.

It can be shown [21] that linear reductions are transitive, i.e., are closed under composition, and that if A linearly reduces to B and B has a polynomial-time approximation algorithm guaranteed to produce solutions y with $|cost(y) - opt(b)| / opt(b) \le \varepsilon$, then A has a polynomial-time approximation algorithm which guarantees to keep the analogous ratio bounded by $\alpha\beta\varepsilon$. Thus if B had a polynomial time approximation scheme, so would A. A problem is MAX SNP-hard if every problem in MAX SNP linearly reduces to it, or equivalently, given the transitivity of linear transformations, if some single previously identified MAX SNP-hard problem linearly reduces to it.

Theorem 5. For any fixed $k \ge 3$, k-Terminal Cut is MAX SNP-hard.

Proof. We prove the result for $k = 3$. The extension to $k > 3$ follows immediately. Our proof is by a linear reduction from the Max Cut problem described above, previously proved MAX SNP-hard in [21]. For the reduction, we need only reinterpret the transformation from SIMPLE MAX CUT to 3-TERMINAL CUT used in the proof of Theorem 3. Note that for that construction we showed that if K is the size of the maximum cut in the original instance, then the size of the optimal 3-terminal cut is $28|E| - K$. Now note that the maximum cut must have size at least $|E|/2$, since a simple greedy heuristic will construct a cut that large. (Start with two adjacent vertices as the nuclei of V_1 and V_2, and then assign the rest of the vertices one at a time, choosing for each the set that maximizes the number of edges added to the cut.) Thus if we let $f(G)$ denote the instance of 3-Terminal Cut derived from an instance G of Max Cut, we have $OPT_{3-TerminalCut}(f(G)) \le 56 OPT_{MAXCUT}(G)$, and our transformation satisfies Property (a) of the definition of linear reduction with $\alpha = 56$.

For Property (b), note that our proof of Theorem 3 implies that any solution y for $f(G)$ of weight $28|E| - K$ can be easily converted to a solution $x = g(y)$ of size K for G. Thus for any solution y of $f(G)$ we have $|cost(x) - opt(a)| \le |cost(y) - opt(b)|$, and so property (b) holds with $\beta = 1$.

We conclude that the tranformation used in the proof of Theorem 3 was a linear reduction, and so 3-Terminal Cut is MAX SNP-hard. □

5. Related Results and Open Problems

Our NP-completeness results in Sections 3 and 4 can be adapted to several related problems of interest. In 1969, T. C. Hu [17] raised the question of the complexity of the following problem, which we might call the *Multipair Cut* problem. Suppose we are given a list of vertex pairs (u_i, v_i), $1 \le i \le k$, and are asked to find a minimum $C(u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k)$ cut, i.e., a minimum weight set of edges separating each pair of vertices u_i, v_i, $1 \le i \le k$. This is just 2-terminal Cut when $k = 1$. The problem is also polynomial time solvable when $k = 2$, by using two applications of a 2-terminal cut algorithm [24]. Our result for 3-Terminal Cut implies that it is NP-hard for arbitrary graphs when $k = 3$, even if all edge weights are equal: merely let the three pairs be (s_1, s_2), (s_2, s_3), and (s_3, s_1). (If one wants all the u_i and v_i to be distinct, the problem remains NP-hard, as can be proved via a simple modification to the input graph.)

Note that for any fixed k the Isolation Heuristic of Section 4.4 can be used in the design of a polynomial-time approximation algorithm for Multipair Cut. Consider partitions P of the vertices $u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k$ into sets $S_1, \ldots, S_{|P|}$ such that no pair (u_i, v_i) is in the same set, and such that for every pair of sets S_h and S_j there is some i such that u_i is in one set and v_i is in the other. Note that the latter constraint implies that $|P| \le \sqrt{2k} + 1$, and so there are at most $(\sqrt{2k} + 1)^{2k} / (\sqrt{2k} + 1)!$ such partitions. For any such partition P, let G_P be the graph obtained by merging all the vertices in the set S_j into a single terminal vertex s_j, $1 \le j \le |P|$. Run the Isolation Heuristic on each such graph G_P, at a cost of $O((\sqrt{2k} + 1)\, nm \log(n^2/m))$ per graph, and output the best cut found. Since the optimal $C(u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k)$ cut must induce one of the partitions P, and since no partition contains more than $\sqrt{2k} + 1$ sets, the weight of the cut we output is at most $2\sqrt{2k} / (\sqrt{2k} + 1) < 2$ times optimal by Theorem 4. The running time is

$$O\left[\frac{(\sqrt{2k})^{2k} (1 + 1/\sqrt{2k})^{2k}}{(\sqrt{2k} + 1)!} (\sqrt{2k} + 1)\, nm \log(n^2/m) \right] \quad = \quad O\left[(2k)^k nm \log(n^2/m) \right]$$

which is polynomial for fixed k. The question remains open as to whether there is a polynomial-time approximation algorithm that works for arbitrary k and provides a constant guarantee, although Garg et al. [11] have devised a polynomial-time algorithm that works for arbitrary k and has worst-case ratio $O(\log k)$.

More recently, Erdös and Székely in [5,6] proposed the following generalization of Multiterminal Cut. Suppose you are given a graph $G = (V, E)$ weighted edges, and a partial k-coloring of the vertices, i.e., a subset $V' \subseteq V$ and a function $f : V' \to \{1, 2, \ldots, k\}$. Can f be extended to a total function such that the total weight of edges that have different colored endpoints is minimized? The k-Terminal Cut problem is the special case where $|V'| = k$ and f is 1-1, i.e., each color is initially assigned to precisely one vertex. It is easy to see that for general graphs, this problem is in fact equivalent to Multiterminal cut: simply merge all the vertices with the same color, call the resulting merged vertices ''terminals,'' and find the minimum weight k-terminal cut for the resulting graph. For special classes of graphs, however, the ''Colored Multiterminal Cut'' problem can be more general. (The above merging trick need not for instance preserve planarity or acyclicity.) Nevertheless, in the case of trees the dynamic programming algorithm for Multiterminal Cut mentioned in the Introduction extends in a natural way to the Colored Multiterminal Cut problem, yielding an $O(nk)$ algorithm, as Erdös and Székely observe. This in turn implies that if G is such that deleting all the terminals renders it acyclic, then Multiterminal Cut can itself still be solved in $O(nk)$ time. (Simply split each terminal s_i into $degree(s_i)$ separate vertices, one for each edge incident on s_i, assign color i to all the derived vertices, and apply the abovementioned algorithm for Colored Multiterminal Cut on trees to the resulting graph [6].)

An obvious question is whether our algorithms for planar graphs also extend to Colored Multiterminal Cut problem. The answer is no. Colored Multiterminal Cut is clearly polynomial-time solvable if $k = 2$, even for general graphs. For any fixed $k \geq 3$, however, it remains NP-complete even for planar graphs and all weights equal to 1. The $k = 4$ case follows directly from our proof of Theorem 2. Simply use the four colors x, $\bar{x}$, c^+, and c^-, and assign x to each terminal x_i, $\bar{x}$ to each terminal $\bar{x}_i$, c to each terminal c_j^+, and c^- to each terminal c_j^-. It is straightforward to verify that the proof still goes through. For $k = 3$, the result can be proved by a transformation from PLANAR 3-COLORABILITY [9,10], using a local replacement argument in which each edge is replaced by a partially colored structure designed to make it expensive for the endpoints of the original edge to get the same color. We leave the details to the enterprising reader. (Note that this last result provides us with an alternate proof of the NP-completeness of 3-TERMINAL CUT for general graphs: once again simply merge all vertices with the same color. The fact that such an operation may destroy planarity is in this case irrelevant.)

Returning to the original Multiterminal Cut problem, in our opinion the most interesting open problem is whether one can improve upon the approximation results of Theorem 4. Although polynomial-time algorithms with worst-case ratios arbitrarily close to 1 are unlikely in light of Theorem 5, can we do better than the $2(k-1)/k$ guarantee we proved for the Isolation Heuristic? Noga Alon [private communication, 1991] has observed that for the special cases of $k = 4$ and $k = 8$ improvements can be obtained using a variant of our approach. For $k = 4$, the Isolation Heuristic provides a guarantee of $3/2$. An improved guarantee of $4/3$ can be obtained as follows: For each partition of the terminals into sets S_1, S_2 of size two, use max flow techniques to compute the minimum cut that separates the terminals in S_1 from those in S_2. Output the union of the two best such cuts. The reader can readily verify that this union is a 4-terminal cut whose weight is at most $4/3$ optimal. Note that this approach requires only three max flow computations versus the four needed by the Isolation Heuristic, so it is faster as well. (Cunningham reports in [3] that F. Zhang has independently obtained this $k = 4$ improvement.)

For $k = 8$, the guarantee of our theorem can be improved from $7/4$ to $12/7$. Here one computes minimum 2-terminal cuts based on partitions of the set of terminals into sets of size four. It can be shown that the average weight of these cuts is at most $4/7$ times the weight of an optimal 8-terminal cut, and that there exists a set of three of these cuts whose union is an 8-terminal cut and whose total weight is no more than average. This yields the claimed bound. Moreover, the running time is once again an improvement on the Isolation Heuristic, which in this case would require eight max flow computations. This is because we can show that it suffices to restrict attention to just seven of the 35 possible partitions of the eight terminals into sets of size four (the seven being derived from the rows of a Hadamard matrix).

Unfortunately, the above approach does not yield improvements over the Isolation Heuristic for any values of k other than 4 and 8. Is there some general technique that will improve on the Isolation Heuristic for arbitrarily large values of k? What about simply beating our bound for the case of $k = 3$?

Turning to our optimization algorithms for the planar case, the obvious question is whether the running times can be improved, although for the case of general k the improvement would have to be substantial to be interesting. For instance, although we would expect any algorithm to be exponential in k, the exponent containing k might not have to be attached to n. Could there be an algorithm whose running time was $c^k n^\alpha$, where α was independent of k?

Acknowledgement. The authors thank the anonymous referees for suggestions that improved the performance of our approximation algorithm for the Multipair Cut problem and more generally helped us clarify our presentation.

REFERENCES

1. S. ARORA, C. LUND, R. MOTWANI, M. SUDAN, AND M. SZEGEDY, ''Proof verification and hardness of approximation problems,'' *in* ''Proceedings 33rd Ann. Symp. on Foundations of Computer Science,'' IEEE Computer Society, Los Angeles, Calif., 1992, 14-23.
2. S. CHOPRA AND M. R. RAO, ''On the multiway cut polyhedron,'' *Networks* **21** (1991), 51-89.
3. W. H. CUNNINGHAM, ''The optimal multiterminal cut problem,'' *DIMACS Series in Disc. Math. and Theor. Comput. Sci.* **5** (1991), 105-120.
4. E. DAHLHAUS, D. S. JOHNSON, C. H. PAPADIMITRIOU, P. D. SEYMOUR, AND M. YANNAKAKIS, ''The complexity of multiway cuts,'' extended abstract (1983).
5. P. L. ERDÖS AND L. A. SZÉKELY, ''Evolutionary trees: An integer multicommodity max-flow min-cut theorem,'' *Adv. in Appl. Math.*, to appear.
6. P. L. ERDÖS AND L. A. SZÉKELY, ''Algorithms and min-max theorems for certain multiway cuts,'' Iin ''Integer Programming and Combinatorial Optimization,'' E. Balas, G. Cornuéjols, and R. Kannan (eds.), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1992, 334-345.
7. L. R. FORD, JR, AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.
8. G. N. FREDERICKSON, ''Fast algorithms for shortest paths in planar graphs, with applications,'' *SIAM J. Comput.* **16** (1987), 1004-1022.
9. M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979.
10. M. R. GAREY, D. S. JOHNSON, AND L. STOCKMEYER, ''Some simplified NP-complete graph problems,'' *Theor. Comput. Sci.* **2** (1976), 237-267.

11. N. GARG, V. V. VAZIRANI, AND M. YANNAKAKIS, ''Approximate max-flow min-(multi)cut theorems and their applications,'' unpublished manuscript (1993).

12. A. V. GOLDBERG AND R. E. TARJAN, ''A new approach to the maximum-flow problem,'' *J. Assoc. Comput. Mach.* **35** (1988), 921-940.

13. O. GOLDSCHMIDT AND D. S. HOCHBAUM, ''Polynomial algorithm for the k-cut problem,'' *in* ''Proceedings 29th Ann. Symp. on Foundations of Computer Science,'' IEEE Computer Society, Los Angeles, Calif., 1988, 444-451. Journal version to appear in *Math. of O.R.*.

14. M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, ''The ellipsoid method and its consequences in combinatorial optimization,'' *Combinatorica* **1** (1981), 169-198.

15. X. HE, ''On the planar 3-cut problem,'' *J. Algorithms* **12** (1991), 23-37.

16. D. S. HOCHBAUM AND D. B. SHMOYS, ''An $O(|V|^2)$ algorithm for the planar 3-cut problem,'' *SIAM J. Algebraic and Discrete Methods* **6** (1985), 707-712.

17. T. C. HU, *Integer Programming and Network Flows*, Addison-Wesley Publishing Co., Reading, MA, 1969.

18. E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

19. D. LICHTENSTEIN, ''Planar formulae and their uses,'' *SIAM J. Comput.* **11** (1982), 329-343.

20. C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

21. C. H. PAPADIMITRIOU AND M. YANNAKAKIS, ''Optimization, approximation, and complexity classes,'' *J. Comput. System Sci.* **43** (1991), 425-440.

22. H. SARAN AND V. V. VAZIRANI, ''Finding *k*-cuts within twice the optimal,'' *in* ''Proceedings 32nd Ann. Symp. on Foundations of Computer Science,'' IEEE Computer Society, Los Angeles, Calif., 1991, 743-751.

23. H. S. STONE, ''Multiprocessor scheduling with the aid of network flow algorithms,'' *IEEE Trans. Software Engineering* **SE-3** (1977), 85-93.

24. M. YANNAKAKIS, P. C. KANELLAKIS, S. C. COSMADAKIS, AND C. H. PAPADIMITRIOU, ''Cutting and partitioning a graph after a fixed pattern,'' *in* ''Automata, Languages, and Programming,'' Lecture Notes in Computer Science, Vol. 154, Springer, Berlin, 1983, 712-722.