SIAM J. ALG. DISC. METH.
Vol. 6, No. 4, October 1985

AN $O(|V|^2)$ ALGORITHM FOR THE PLANAR 3-CUT PROBLEM*

DORIT S. HOCHBAUM† AND DAVID B. SHMOYS‡

Abstract. A 3-cut for a connected graph G is a set of edges which, when deleted, separate G into 3 components. In this paper we present an $O(|V|^2)$ algorithm to find the minimum 3-cut for a *planar* graph G.

AMS(MOS) subject classifications. F.2.2., G.2.1., G2.2

Given a connected (simple) graph $G=(V, E)$, a *k-cut* is a subset of edges E_1 such that the graph $G_1=(V, E-E_1)$ contains exactly k components. The problem of finding the smallest 3-cut is interesting, not only as an extension of the ordinary minimum (2-)cut problem, but also because of applications in cutting plane methods for the traveling salesperson problem. In this paper we present a polynomial-time algorithm to find a minimum size 3-cut for *planar* graphs. It is easy to see that the 3-cut problem is polynomial for planar graphs; since the minimum degree of a planar graph is at most five, there is a trivial cut of size 10. To find the minimum cut we can simply check all subsets of edges of size less than 10. To simplify notation, for a graph $G=(V, E)$, $|G|$ will be used to denote $|E|$; note that for a planar graph $|G|=O(|V|)$. Thus, the simple enumerative algorithm requires $O(|G|^{10})$ time. In this paper we present a simple efficient algorithm that requires only $O(|G|^2)$ time.

One simple approach for finding an optimal 3-cut that might be considered is the greedy approach; that is, choose a triple of vertices (v_1, v_2, v_3). First find a minimum (2-)cut between v_1 and v_2. This cut leaves v_3 in a component with either v_1 or v_2; suppose, without loss of generality, that v_1 lies in the same component as v_3. Next find the minimum cut in this component between v_1 and v_3. This yields a 3-cut, but is it an optimal one? A further extension of this would be to try this greedy heuristic for all ordered triples of vertices. Unfortunately, even in the planar case, this procedure does not yield this optimum solution. Consider the graph given in Fig. 1. The minimum 3-cut is the set of six edges separating the three square-like sets of four vertices each. Attempting to find such a cut by enumerating all possible triples of vertices, we check the triple (v_1, v_2, v_3) given in Fig. 1. Suppose that we find a minimum 2-cut

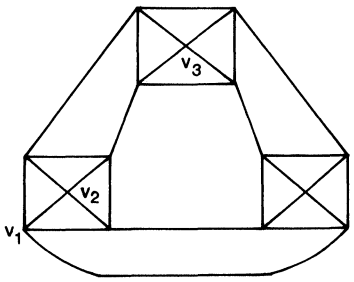

FIG. 1

* Received by the editors November 15, 1983, and in revised form June 28, 1984.

† School of Business Administration, University of California, Berkeley, California 94720. The research of this author was supported in part by the National Science Foundation under grant ECS-8204695.

‡ Department of Computer Science, Harvard University, Cambridge, Massachusetts 02138. The research of this author was supported in part by the National Science Foundation by a graduate fellowship and under grant MCS-8311422, and in part by DARPA order 4031, monitored by Naval Electronic System Command under contract N00039-C-0235.

between v_1 and v_2; one such cut consists of the four edges incident to v_1. Now find a minimum cut between v_2 and v_3; this cut consists of the three remaining edges incident to v_2. In total, we have found a 3-cut of size seven. By a careful examination of all cases, it is straightforward to verify that for any choice of 3 vertices, there is a cut that can be the result of this approach that is not optimal.

We will solve the minimum 3-cut problem in G by solving the corresponding problem in G^*, the dual graph of G. Note that the dual graph G^* might not be a simple graph; both self-loops and multiple edges can occur.

Recall that for a planar graph G, there is a 1-1 correspondence between 2-cuts in G and cycles in G^*; we want to generalize this idea. A cycle is a minimal graph with 2 faces. For each face of a subgraph of G^*, there are some vertices of G that are "contained" in that face that are separated from vertices "contained" in the other faces of the subgraph. This leads us to make the following observation.

Observation. G has a 3-cut of size k if and only if G^* contains a subgraph H with exactly three faces such that H has k edges.

Since G^* can be found in $O(|G|)$ time, we will focus attention on the problem of finding a minimum size subgraph with 3 faces. Thus, for the remainder of the paper, we will assume that we are given both G^*, and an embedding of it. We begin by examining the structure of possible optimal solutions. First of all, the optimal subgraph H may or may not be connected. Suppose that H is disconnected; since it contains exactly three faces, it must be precisely two disjoint cycles (see Fig. 2(a)). If H is connected then for a given embedding either the boundary of the exterior face includes all of the edges of H or it does not. By considering these cases separately, it is not hard to see that H must be one of the configurations depicted in Fig. 2(b) and 2(c).

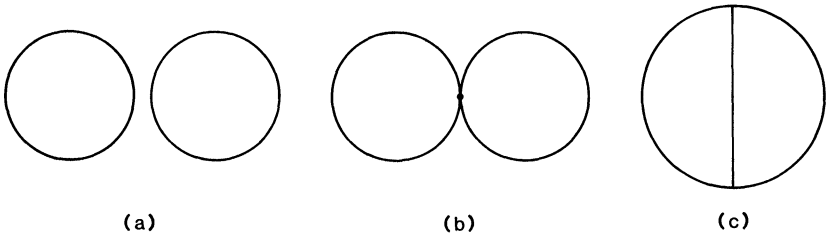

FIG. 2

We will show that a variant of the greedy approach works in either of the cases 2(a) and 2(b). Let $G \cdot e$ denote the graph formed from G by contracting the edge e, i.e., identifying the endpoints of e and deleting e. In terms of the dual graph, the variant of the greedy heuristic that we shall use is the following:

procedure *greedy* (G^*)
  begin
    find a shortest cycle C_1 in G^*
    for all edges e in the cycle C_1 **do** $G^* \leftarrow G^* \cdot e$
    find a shortest cycle C_2 in G^*
    output $C_1 \cup C_2$
  end

In this variant we avoid trying all triples of vertices, which of course improves the efficiency of the algorithm. Although this algorithm does not always give an optimal

solution, there are a number of cases in which it does work, including those cases when the optimal solution is of either of the configurations given in Fig. 2(a, b). In fact, our procedure to find an optimal 3-cut works by first performing *greedy* (G^*), and then performing a special procedure designed to handle the case depicted in Fig. 2(c). The following theorem justifies this approach.

THEOREM 1. *Let C_3 be any shortest cycle of the graph G^*. If there exists an optimal 3-face subgraph that is the union of 2 cycles with at most one vertex in common, then there exists some other cycle C such that C_3 and C share at most one vertex and $C_3 \cup C$ is an optimal 3-face subgraph.*

Proof. Suppose not. Let $C_1 \cup C_2$ be an optimal 3-face subgraph, where C_1 and C_2 have at most one common vertex. Note that by our assumptions, C_3 must have at least 2 vertices in common with C_i for either $i=1$ or 2. Recall that $|G|$ denotes the number of edges in G. Notice that a cycle C has $|C|$ vertices.

If C_i and C_3 are edge disjoint but have at least two vertices in common, then, by using Euler's formula $|F|=2+|E|-|V|$, we see that the number of faces of $C_i \cup C_3$ is at least

$$2+(|C_i|+|C_3|)-(|C_i|-(|C_3|-2))=4.$$

Therefore, $C_i \cup C_3$ contains a *proper* subgraph with 3 faces and fewer edges than $C_1 \cup C_2$, which is a contradiction.

If C_i and C_3 have an edge in common, then

$$|C_i \cup C_3| < |C_i|+|C_3| \leqq |C_1|+|C_2|.$$

Furthermore, once again using Euler's formula, it is not hard to see that $C_i \cup C_3$ must have at least 3 faces. Therefore, $C_i \cup C_3$ is a 3-face subgraph with fewer edges than $C_1 \cup C_2$, which is a contradiction. □

It is important to note that if the optimal 3-face subgraph consists of 2 cycles sharing at most one common vertex, Theorem 1 says that *any* shortest cycle is part of an optimal solution. Thus we may greedily contract it, and search for its partner.

Note that it is very simple to use breadth-first search (BFS) to find the shortest cycle of a graph that contains a specified vertex u. Recall that BFS partitions the vertex set of a graph into levels, and the edge set into cross edges and level edges. (For a comprehensive treatment of BFS, see [AHU].) Furthermore, for each vertex, each edge incident to it either goes to a vertex of the next level closer to the root, the same level or the next level further from the root; therefore, it is possible to partition the degree of any vertex v into the *up-degree*, *up* (v), the *cross-degree*, *cross* (v), and the *down-degree*, *down* (v), respectively. Note that it is possible to compute all three parameters as part of the BFS without increasing the order of the running time of the search algorithm. We detect the smallest cycle containing u by conducting a BFS from u until we find a vertex v that has *up* (v) + *cross* $(v) \geqq 2$. Since BFS requires only $O(|G|)$ time, the procedure outlined above requires only $O(|G|^2)$ time.

We next consider the case where the optimal solution is of the form depicted in Fig. 2(c). This graph may be viewed as a pair of points connected by three vertex-disjoint paths. The following result will be very useful in gaining a better perspective of the cases in which *greedy* (G^*) is successful.

THEOREM 2. *Suppose that there exists an optimal 3-face subgraph G^* which is the union of three paths P_1, P_2 and P_3, between vertices u and v. If one of the paths P_i is at least as long as the shortest cycle C in G^*, then greedy (G^*) delivers an optimal solution.*

Proof. Consider the solution produced by *greedy* (G^*), $C_1 \cup C_2$. Suppose without loss of generality that P_1 is at least as long as C_1, which is a shortest cycle of G^*.

There are two possible cases: either $C_2 = P_2 \cup P_3$, or $C_2 \neq P_2 \cup P_3$. In the first case the result is trivial since then it is clear that

$$|C_1 \cup C_2| = |C_1| + |C_2| \leqq |P_1| + (|P_2 \cup P_3|) = |P_1 \cup P_2 \cup P_3|.$$

If $C_2 \neq P_2 \cup P_3$ then after contracting C_1 the subgraph corresponding to $P_2 \cup P_3$ must still be a cycle (since not all of the edges of $P_2 \cup P_3$ have been contracted). Since C_2 is no longer than this cycle the total length of $C_1 \cup C_2$ is no more than that of $P_1 \cup P_2 \cup P_3$. □

We can use Theorem 2 to gain considerable insight into the structure of the optimal solution when *greedy* (G^*) does *not* deliver an optimal solution. Such optimal solutions, which must be of the type given in Fig. 2(c), can be characterized by the length of the three paths P_1, P_2 and P_3. More specifically, since *greedy* (G^*) always delivers a solution of size no more than 10, we know that the three path lengths sum to at most 9, where the longest is strictly shorter than the length of the shortest cycle. From these two conditions we know that the shortest path is of length at most three. This means that if we were conducting BFS from u and the optimal 3-face subgraph consisted of three disjoints paths from u to v, v must be, at most, in level three. (We will say from here on, that v is "labeled" 3.)

We will consider these cases, based on the length of the shortest cycle C_1 in G^*. We shall use the notation (x, y, z) to denote the configuration where P_1, P_2, P_3 have lengths x, y, z where $x \leqq y \leqq z$.

Case 1. $|C_1| = 1$. This case is vacuous since it is impossible for the longest path to have length 0.

Case 2. $|C_1| = 2$. In this case, the only possible configuration is $(1, 1, 1)$, or equivalently that there are three edges between some pair of vertices u and v. Notice that if we were conducting BFS from u we would find that $up\ (v) \geqq 3$.

Case 3. $|C_1| = 3$. Here there are two possible configurations, $(1, 2, 2)$ and $(2, 2, 2)$. These are depicted in Fig. 3.

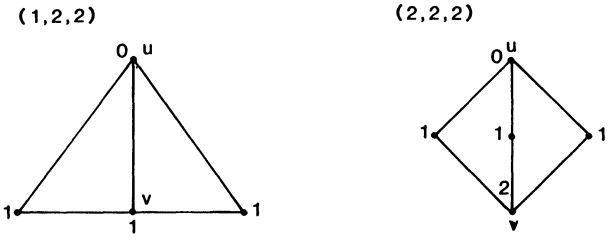

FIG. 3

(Notice that $(1, 1, 1)$ and $(1, 1, 2)$ cannot occur because then the shortest cycle would have at most 2 edges.) The labels assigned to the nodes in these figures, and the ones that follow, are the BFS labels of the nodes on the assumption that the given configuration is an optimal 3-face subgraph. For the first situation, in performing BFS from u, we find a vertex labeled 1 with *cross* $(v) \geqq 2$. In the second we find a vertex labeled 2 with $up\ (v) \geqq 3$. Notice that in both of these cases we have found vertices with $cross\ (v) + up\ (v) \geqq 3$.

Case 4. $|C_1| = 4$. In this case, there are the configurations $(2, 2, 2)$, $(2, 2, 3)$, $(1, 3, 3)$, $(2, 3, 3)$ (see Fig. 4).

(Note that $(3, 3, 3)$ need not be considered here, since *greedy* (G^*) is sure to find C_2 with at most 5 edges.) As above, $(2, 2, 2)$ can be detected by finding a vertex v

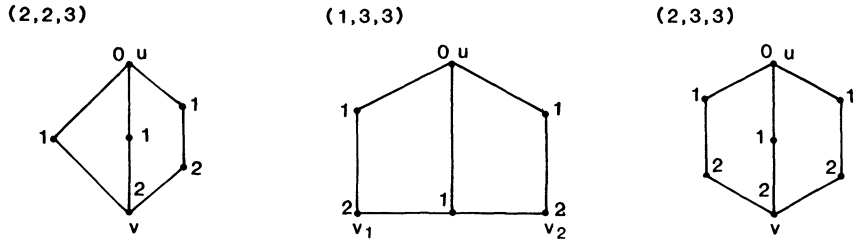

FIG. 4

labeled with a 2 with $up\ (v) \geqq 3$. For the case (2, 2, 3), we find a vertex v with $up\ (v) = 2$ and $cross\ (v) \geqq 1$. For (1, 3, 3), we find a vertex labeled 1 with two neighbors v_1 and v_2 with $up\ (v_i) = 2$. For (2, 3, 3), we find a vertex v with $cross\ (v) = 2$ (and $up\ (v) \geqq 1$).

Case 5. $|C_1| = 5$. For this final case, only the configurations (2, 3, 3), (3, 3, 3), (2, 3, 4) and (1, 4, 4) must be considered. The first configuration is handled exactly as in Case 4, and the remaining ones are depicted in Fig. 5.

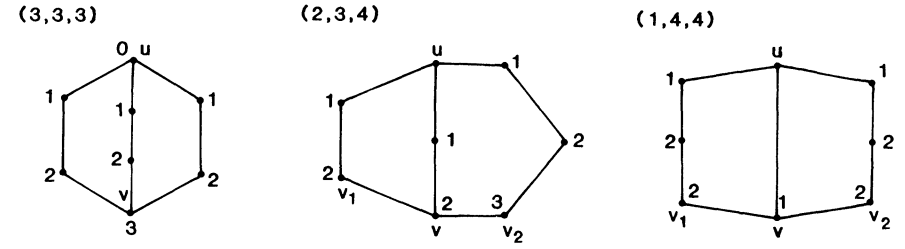

FIG. 5

For (3, 3, 3) we find a vertex labeled 3 with $up\ (v) = 3$. For (2, 3, 4), a vertex v labeled 2 with $cross\ (v) = 1$ has a neighbor v_2 with $up\ (v_2) \geqq 2$. Finally, for (1, 4, 4) a vertex v labeled with a 1 has two neighbors, v_1 and v_2 with $cross\ (v_i) \geqq 1$.

Summarizing what we have done, there are only 2 different types of configurations that we must search for in performing a BFS from u:

1. A vertex v with $cross\ (v) + up\ (v) \geqq 3$.
2. A vertex with two neighbors v_1 and v_2 with $cross\ (v_i) + up\ (v_i) \geqq 2$.

Therefore, *if greedy* (G^*) does not find an optimal solution, then by performing BFS from each vertex and searching for one of these two configurations, we are guaranteed to find an optimal 3-face subgraph. It is quite straightforward to modify BFS to detect these configurations. Therefore, we see that it can be implemented in $O(|V|^2)$ time, since BFS from each node requires $O(|V|)$ time for planar graphs. Furthermore, by performing *greedy* (G^*) and this procedure as well, and then choosing the best solution found, we find an optimal 3-face subgraph in $O(|V^2|)$ time.

It is significant to note that the planarity of G is used in two ways. Most importantly, it insures the existence of the dual graph; it is used secondarily in guaranteeing an "easy" cut of size 10, and thus limiting the search considerably.

Another interesting point is that the same argument as was used above, can be used to show that a greedy approach never delivers a k-cut of size more than $5k$. Thus, for fixed k the minimum k-cut problem is easily seen to be polynomial using an enumerative approach. Can a simpler approach, analogous to the one given here be used instead? Such an approach is complicated by the fact that the types of possible

configurations become more complicated (and much more numerous) than what was given in Fig. 2 for the 3-cut problem.

The complexity of the 3-cut problem for arbitrary graphs remains a challenging open problem. Recently, Hochbaum and Tsai [HT] have shown that a greedy heuristic for the 3-cut problem gives a solution, even for the weighted case, that is at most 4/3 the optimal solution, and that this bound is tight. Independently, Johnson, Papadimitriou, Seymour and Yannakakis [JPSY] proved this result in more general form, that for the weighted k-cut problem a greedy approach yields a solution no more than $(2-2/k)$ times the optimal solution. They also consider a more general problem: the problem of finding a minimum k-cut with the additional constraint that they specify k vertices that must be in different components. They showed that for planar weighted graphs the minimum 3-cut problem with specified vertices is polynomial. Their algorithm, although polynomial, is not efficient, since the polynomial is a polynomial in $k!$ and the degree of the polynomial depends linearly on k. Furthermore, they showed that the 3-cut problem with specified vertices for arbitrary graphs is NP-complete. This does not appear to imply anything about the complexity of the 3-cut problem for arbitrary graphs. If the general 3-cut problem were to be polynomial, it would be a remarkable extension of one of the oldest results in combinatorial optimization.

Acknowledgment. We are grateful to the anonymous referee whose suggestions greatly enhanced the clarity and simplicity of this paper.

REFERENCES

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

[JPSY] D. S. JOHNSON, C. H. PAPADIMITRIOU, P. D. SEYMOUR AND M. YANNAKAKIS, Private communication, 1983.

[HT] D. S. HOCHBAUM AND L.-H. TSAI, *A greedy heuristic for the minimum 3-cut problem*, manuscript, 1983.