The Steiner Problem in Graphs

S. E. Dreyfus
Department of Industrial Engineering and Operations Research
University of California
Berkeley, California

R. A. Wagner
Computer Science Department
Cornell University
Ithaca, New York

ABSTRACT

An algorithm for solving the Steiner problem on a finite undirected graph is presented. This algorithm computes the set of graph arcs of minimum total length needed to connect a specified set of k graph nodes. If the entire graph contains n nodes, the algorithm requires time proportional to

$$n^3/2 + n^2(2^{k-1} - k - 1) + n(3^{k-1} - 2^k + 3)/2.$$

The time requirement above includes the term $n^3/2$, which can be eliminated if the set of shortest paths connecting each pair of nodes in the graph is available.

1. THE STEINER PROBLEM IN A GRAPH

Let a *graph* G be given as a pair (N,A), where N is a set of nodes, and A is a set of undirected arcs connecting nodes of N. With each arc $a \in A$ is associated a positive number $|a|$, called the length of a. Suppose a subset Y of N is also given. The problem we consider is that of computing a subset S of A such that:

(1) all members of Y are connected by paths composed only of arcs in S;
and
(2) $|S| = \sum_{s \in S} |s|$ is a minimum.

We call this problem the *Steiner Problem on* Y *in graph* G. G is assumed to be connected. S is called the *Steiner path connecting* Y *in* G, or, where G is evident, just the *Steiner path connecting* Y. If Y consists of just one node, the Steiner

Networks, 1: 195-207

path consists of no arcs and has length zero.

The original Steiner problem (rather than the problem on a graph which we are studying) is an old problem in geometry. In its usual formulation, the Steiner problem requires finding the set of lines of minimum total length which connect a given set of points Y. For properties of its solution, see Reference [1]. Usually both lines and points are assumed to lie in the same Euclidean plane. Nothing in the formulation of the problem requires that the lines drawn intersect only on points in Y; on the contrary, "interior" points of intersection, called Steiner points, can frequently reduce the total length of line. For example, given the three points y_1, y_2 and y_3 to be connected, the lines shown in Figure 1

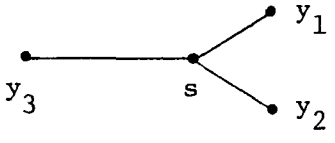


Fig. 1

with constructed point S used as a point of intersection, produces a total line-length smaller than that of say, Figure 2.

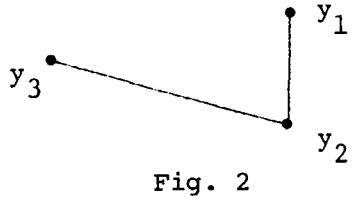


Fig. 2

The present investigation attempts to study this problem in a space where length-measurement is not Euclidean and where paths must consist of elements of a specified set A of arcs. Specifically, we allow lengths to be assigned to arcs (point-to-point distances) arbitrarily. Such lengths need not even satisfy the triangle inequality. For example, Figure 3 shows a perfectly valid assignment of lengths to arcs.

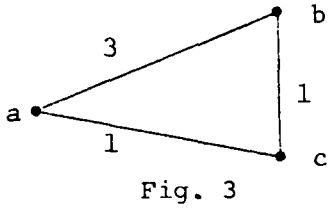


Fig. 3

In this graph, a shorter distance is traversed by going from a to c and thence to b, than by going directly from a to b.

One of the easily proven facts about a Steiner path S is that S must be a *tree*. That is, S must contain no cycles. For if a cycle exists, then there are two different paths available for connecting some node of Y to the rest. One can eliminate some arc of one of these paths without destroying the connecting property of S. But this reduces $|S|$, contradicting the assumption that $|S|$ is minimal.

Now, consider Figure 4, showing a typical solution S to a Steiner problem on $Y \subset N$ in $G = (N,A)$.

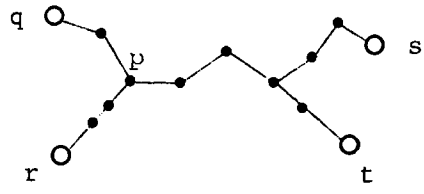


Fig. 4

Here, $Y = \{q,r,s,t\}$. Note that any node belonging to Y, say q, is connected, in this solution, by a branch of the solution tree to a junction node p (which need not be an element of Y). Clearly the path connecting q and p is the shortest path, else S would not be the minimal solution. Also note that each of the other branches of S touching p represents the solution of a Steiner problem connecting fewer nodes than the number of the set Y. $\{p,s,t\}$ cannot be connected by a shorter path, nor can $\{p,r\}$. If they could, again S would not be the solution.

We call the above "division" of the problem by p into three smaller parts the optimal decomposition property. It can be stated as follows.

Let S be any Steiner tree connecting Y, where $Y \subset N$ is a subset of the nodes of graph $G = (N,A)$, and let q be any node of Y. If Y contains at least 3 members then there exists a node $p \in N$ and a subset D of Y such that:

D is a proper subset of $Y - \{q\}$, and D is nonempty.

S consists of 3 disjoint subsets, S_1, S_2 and S_3.

S_1 connects $\{p,q\}$, S_2 connects $\{p\} \cup D$, while

S_3 connects $\{p\} \cup (Y - D - \{q\})$.

Furthermore, S_1, S_2 and S_3 are all Steiner paths connecting their respective sets.

Three "degenerate" situations are shown below (in all cases $Y = \{q,r,s,t\}$). Consider Figure 5.

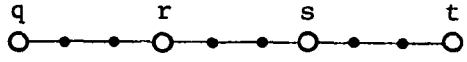


Fig. 5

Here p is node r. Then $D = \{s,t\}$, S_2 is the Steiner path connecting $\{r,s,t\}$ and S_3 connects $\{r\}$ and hence contains no arcs, or conversely. In Figure 6,

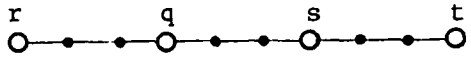


Fig. 6

p is node q, $D = \{r\}$, S_2 is the Steiner path connecting $\{q,r\}$ and S_3 is the Steiner path connecting $\{q,s,t\}$, or conversely. In Figure 7,

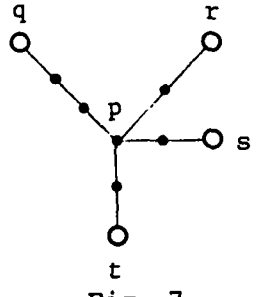


Fig. 7

D can be $\{r\}$, $\{s\}$, $\{t\}$, $\{r,s\}$, $\{r,t\}$ or $\{s,t\}$. Ignoring switching of D and $Y - D - \{q\}$, there are 3 different decompositions which satisfy the principle above.

(A general proof of the existence of an optimal decomposition of the type described above, covering all degenerate cases, appears in Appendix A.)

2. A SOLUTION ALGORITHM

Our algorithm exploits the optimal decomposition property. A straightforward application of the property would entail choosing $q \in Y$ (any q will do), then searching for the optimal choice of p. In turn, an optimal choice of p requires that an optimal choice of the subset $D \subset Y$ be made, and that the Steiner trees S_2 and S_3 connecting $D \cup \{p\}$ and $(Y - D - \{q\}) \cup \{p\}$ be

known. Thus, the original problem could be solved recursively. However, we could also build up the desired solution by means of the following $\|Y\| - 1$ steps (where $\|Y\|$ denotes the number of elements in the set Y).

1. Remove one node, q, from Y. Call the set $Y - \{q\}$ set C.
2. Solve Steiner problems connecting each set of 2 nodes of C and 1 node $n \in N$. (n can be a member of C, or it can even be node q itself).
3. Use this result to solve Steiner problems connecting each set of 3 nodes of C and 1 node $n \in N$.

.
.
.

$\|Y\| - 2$. Solve Steiner problems connecting each set of $\|Y\| - 2$ nodes of C and 1 node $n \in N$.

$\|Y\| - 1$. Solve the Steiner problem connecting q and set C.

Given a subset D of C, and $n \in N$, each step in the solution above involves two searches: Search 1 locates the intermediate node, $p \in N$; Search 2 finds the optimal proper subset E of D so that the lengths of the Steiner paths connecting $\{p\} \cup E$ and $\{p\} \cup (D - E)$, plus the distance from n to p add to the smallest possible such total length.

The efficiency of this procedure stems from the fact that only *optimal* solutions for the relevant subsets are ever considered. Nonoptimal solutions to smaller subproblems are disposed of at the time that subproblem is solved. The optimal solution is retained, for use in solving later subproblems, and the smaller subproblem is never solved again. Straightforward enumeration of all possible solutions to the entire problem would unnecessarily consider nonoptimal subsolutions many times. This building up of larger optimal solutions from optimal solutions of all possible smaller problems is the fundamental technique in the general methodology called dynamic programming.

Let us discuss in some detail the procedure whereby the Steiner solution for a given subset D consisting of a certain j (≥ 2) nodes of C and one node, $m \in N$, is found. Here again we avoid some unnecessary calculation by first solving all possible smaller problems of a certain form. First we associate with each node $k \in N$ a number $S_k(D)$ which is found by (1) breaking D into two proper subsets E and F, and adding the Steiner distance for the set consisting of the members of E and node k to the Steiner distance for the set consisting of the nodes in F and node k, and (2) minimizing this sum over all

distinct choices of sets E and F. (The number $S_k(D)$ is *not* necessarily the minimum Steiner distance for the set composed of the elements of D and node k since no saving due to coalescing the subsolutions at a node other than k is considered.) Having done this for given D and all k, to solve the Steiner problem for m and D, we let d_{mk} denote the length of the shortest path from m to k and we minimize $d_{mk} + S_k(D)$ over all nodes $k \in N$. Let S(m,D) denote the Steiner path for nodes $\{m \cup D\}$. Since $S_k(D)$ does not depend on the choice of node m, knowledge of $S_k(D)$ for all $k \in N$ allows easy computation of Steiner solutions for any $m \in N$, all, of course, for given D. The computation is repeated, then, for all choices of the set D.

Since shortest paths between pairs of general nodes are used repeatedly in the above calculations, we begin the whole procedure by first computing the length of the shortest paths between all pairs of nodes in the graph, Reference [2].

So far we have described a procedure for generating the *length* of the Steiner tree, but not the actual tree. To determine the tree, there are two "pure" strategies available (and several hybrids). For each choice of m and D one can record the value of k that minimized $d_{mk} + S_k(D)$ and the sets E and F that generated $S_k(D)$. Then k is the node which produces the optimal decomposition, with m connected to k by shortest path, while E and F, respectively, are joined to k by Steiner trees. Or, on the other hand, the values of $S_k(D)$ can be stored and the minimizing value of k and associated set E and F can be recomputed as needed in the reconstruction of the Steiner tree. In either case, as is typical in dynamic programming procedures, the Steiner tree is constructed (after the optimal length has been determined) by processing sets in the reverse order of that of the length-determination algorithm. The first method of tree-construction involves less computation while the second uses less computer storage. Since tree-construction by method two requires at most 1/nth the computation time of the length-generation, we recommend, and use, it.

3. NUMERICAL ILLUSTRATION OF THE PROCEDURE

Let $\|N\| = 7$, $\|Y\| = 4$, $Y = \{1,2,3,4\}$, and the matrix A of

direct distances ($a_{ij} = a_{ji}$ is the length of the arc (i,j) between nodes i and j) be

$A =$

i \ j	1	2	3	4	5	6	7
1	x	2	2	2	1	1	2
2	2	x	2	2	2	1	2
3	2	2	x	2	2	2	1
4	2	2	2	x	1	2	1
5	1	2	2	1	x	2	1
6	1	1	2	2	2	x	1
7	2	2	1	1	1	1	x

First we compute the matrix D of shortest lengths ($d_{ij} = d_{ji}$ = length of the shortest path between nodes i and j) by the method of Reference [2]. Clearly, by our choice of data, matrix D is identical with matrix A. We now remove one node, say node 1, from Y. Let $C = \{2,3,4\}$.

Let $D = \{2,3\}$. Then $S_1(D) = d_{12} + d_{13} = 4$, $S_2(D) = 2$, $S_3(D) = 2$, $S_4(D) = 4$, $S_5(D) = 4$, $S_6(D) = 3$, $S_7(D) = 3$. Letting $S(m,D)$ denote the Steiner solution for nodes $\{m \cup D\}$, we have $S(1,D) = \min_k (d_{1k} + S_k(D)) = 4$ (with several different trees yielding the result). $S(2,D) = 2$, $S(3,D) = 2$, $S(4,D) = 4$, $S(5,D) = 4$, $S(6,D) = 3$, $S(7,D) = 3$.

Now let $D = \{2,4\}$. Then $S_1(D) = 4$, $S_2(D) = 2$, $S_3(D) = 4$, $S_4(D) = 2$, $S_5(D) = 3$, $S_6(D) = 3$, $S_7(D) = 3$. Hence $S(1,D) = 4$, $S(2,D) = 2$, $S(3,D) = 4$, $S(4,D) = 2$, $S(5,D) = 3$, $S(6,D) = 3$, $S(7,D) = 3$.

Finally, let $D = \{3,4\}$. Then $S_1(D) = 4$, $S_2(D) = 4$, $S_3(D) = 2$, $S_4(D) = 2$, $S_5(D) = 3$, $S_6(D) = 4$, $S_7(D) = 2$. Hence $S(1,D) = 4$, $S(2,D) = 4$, $S(3,D) = 2$, $S(4,D) = 2$, $S(5,D) = 3$, $S(6,D) = 3$, $S(7,D) = 2$.

We are now ready for step $\|Y\| - 1$, since step 2 = step $\|Y\| - 2$. Let $D = \{2,3,4\}$.

Let $E = \{2\}$ and $F = \{3,4\}$. Then $S_1(D \mid E,F) = S(1,E) + S(1,F) = 2 + 4 = 6$. Now let $E = \{3\}$, $F = \{2,4\}$. Then $S_1(D \mid E,F) = 2 + 4 = 6$. Finally, let $E = \{4\}$, $F = \{2,3\}$. Then $S_1(D \mid E,F) = 2 + 4 = 6$. Hence $S_1(D)$ = minimum over all choices of E and F of $S_1(D \mid E,F) = 6$.

Letting E = {2} and F = {3,4}, $S_2(D \mid E,F) = 4$. Letting E = {3} and F = {2,4}, $S_2(D \mid E,F) = 4$. Letting E = {4} and F = {2,3}, $S_2(D \mid E,F) = 4$. Hence $S_2(D) = 4$.

Similarly, $S_3(D) = \min(4,4,4) = 4$. $S_4(D) = \min(4,4,4) = 4$. $S_5(D) = \min(5,5,5) = 5$. $S_6(D) = \min(4,5,5) = 4$. $S_7(D) = \min(4,4,4) = 4$.

Hence $S(1,D) = \min_k (d_{1k} + S_k(D)) = 5$, which is the minimal $|S|$. To reconstruct the solution, we note that since k = 6 yielded the minimum in the above minimization, node 1 is to be connected to node 6 by shortest path, which in this case is the arc connecting nodes 1 and 6. Now $S_6(D)$ resulted when E = {2} and F = {3,4}. Hence the shortest path from 6 to 2 is part of the solution tree. This is the arc between 2 and 6. Finally, node 6 must be connected to nodes 3 and 4 by Steiner path. Referring to S(6,D) above for D = {3,4} we see S(6,D) = 3 and the value 3 was obtained when k = 7 yielded $\min_k (d_{6k} + S_k(D))$. Hence node 6 must be connected to node 7 by shortest path (the arc connecting 6 and 7 in this case) and node 7 must be connected to nodes 3 and 4 by shortest paths (the arcs between 7 and 3 and 7 and 4 in this case). Hence the solution tree consists of the arcs (1,6), (2,6), (6,7), (7,3) and (7,4). The sum of these arc-lengths is indeed 5, checking with our computed value of S(1,D) for D = {2,3,4}.

With reference to the statement of the optimal decomposition property, if node 1 is taken as node q, then node 6 is node p and node 2 can constitute set D. Then set S_1 consists of the arc (1,6), set S_2 consists of (2,6) and S_3 consists of (6,7), (7,3) and (7,4).

4. THE ALGORITHM

The details of the algorithm described and illustrated above are best understood when expressed in precedural form. Below is the algorithm, expressed in a modified form of Algol.

Let D(i,j) = length of the shortest path in G from i to j, $i,j \in N$. Choose $q \in Y$, and define $C = Y - \{q\}$. Also, let A[1] denote the first element of the (ordered) set A and $A \subsetneq B$ mean that A is a subset of B but not B itself.

Algorithm A: (Computes the length of the Steiner tree connecting Y, and assigns this length to variable "v"):

```
   COUNT
                (1)  for each t ε C do
                (2)    for each J ε N do
                (3)      S[{t},J] ← D(t,J);
‖C‖ - 2         (4)  for m = 2 to ‖C‖ - 1 do
(‖C‖ choose m)  (5)    for each D such that D ⊂ C ∧ ‖D‖ = m do begin
                (6)      for each I ε N do
                (7)        S[D,I] ← ∞ ;
  ‖N‖           (8)      for each J ε N do begin
                (9)        u ← ∞ ;
2^(m-1) - 1     (10)       for each E such that D[1] ε E ∧ E ⊊ D do
                (11)         u ← min (u,S[E,J] + S[D - E,J]);
  ‖N‖           (12)       for each I ε N do
                (13)         S[D,I] ← min (S[D,I],D(I,J) + u);
                (14)         end; end;
                (15) v ← ∞ ;
  ‖N‖           (16) for each J ε N do begin
                (17)   u ← ∞ ;
2^(‖C‖-1) - 1   (18)   for each E such that C[1] ε E ∧ E ⊊ C do
                (19)     u ← min (u,S[E,J] + S[C - E,J]);
    1           (20)   v ← min (v,D(q,J) + u); end.
```

Here, we have solved all subproblems involving subsets of C of size j, before solving any subproblem involving a subset of C of size j + 1. But it is unnecessary to consider the subsets of C in precisely this order. For example, suppose C = {d,e,f,g,h,i}, and we wish to find the Steiner path connecting node m and {d,e,f,g}. We need solutions for all sets D ∪ {p}, where D ⊂ {d,e,f,g} and p ε N, but for no subsets of C involving h or i. The program that we have written to implement Algorithm A actually processes subsets in an order different from that given in Algorithm A, to use efficient methods of enumerating subsets. But the philosophy is the same, as is the number of elementary comparison — and — addition operations. No vast saving of time can be expected from this reordering of subset choice -- only a linear improvement factor due to reduced bookkeeping effort.

5. ENUMERATION OF THE NUMBER OF OPERATIONS REQUIRED BY ALGORITHM A

Most of the time required by Algorithm A is spent executing "elementary" statements of the form

$$a \leftarrow \min\ (a, b + c).$$

(Although the "bookkeeping" needed to compute successive sets, such as E in line 10, appears complicated, in actuality, its cost can be reduced to the execution of only 2 or 3 computer instructions per loop iteration. The time-cost of one such iteration is roughly the same as that needed for execution of the elementary statement above.)

The formulae presented in the column labelled "count" give the number of repetitions that control statements call for. Thus, for each value of m, statement 5 calls for $\binom{\|C\|}{m}$ iterations or repetitions of the statements (6 through 13) under its control.

Statement 11 is executed $\sum_{m=2}^{\|C\|-1} \binom{\|C\|}{m} \|N\|\ (2^{m-1} - 1) = S_{11}$ times;

Statement 13, $\sum_{m=2}^{\|C\|-1} \binom{\|C\|}{m} \|N\|^2 = S_{13}$ times;

Statement 19, $\|N\|\ (2^{\|C\|-1} - 1) = S_{19}$ times;

Statement 20, $\|N\| = S_{20}$ times.

Each is an elementary statement.

Since

$$\sum_{i=0}^{n} \binom{n}{i} 2^i = (1 + 2)^n = 3^n$$

we obtain $S_{11} + S_{13} + S_{19} + S_{20} = S$ as:

$$S = \|N\|^2 [2^{\|C\|} - \|C\| - 2] + \|N\| [1/2(3^{\|C\|} + 3) - 2^{\|C\|}].$$

Letting $k = \|C\| + 1 = \|Y\|$ = number of nodes to be connected, and $n = \|N\|$ = number of nodes in the graph,

$$S = n^2(2^{k-1} - k - 1) + n(3^{k-1} - 2^k + 3)/2.$$

To this must be added T, the number of elementary statements needed to compute D(i,j) = the length of the shortest path between each pair of nodes in N. By the method of Reference [2], exploiting symmetry, $T \simeq n^3/2$.

On a modern computer, each elementary statement should require about 10^{-5} seconds. For a problem of size n = 100, k = 10, we estimate that about 7×10^6 operations would be needed, requiring about 70 seconds. Our program was compiled by PL/I rather inefficiently and requires about 6 times this estimated time, tending to confirm the above estimate for an efficient machine language program. It is interesting to note that if 2^k is small relative to n (e.g., n = 200, k = 6), the solution of the all-pairs shortest path problem is much more time-consuming than the solution of the Steiner problem, given the all-pairs shortest path result.

APPENDIX A PROOF OF THE OPTIMAL DECOMPOSITION PROPERTY

The optimal decomposition property states, roughly, that *any* Steiner path connecting all nodes in the set Y can be decomposed into precisely 3 disjoint sets of arcs. All these pieces touch a particular node, p. One piece joins p with q, a previously chosen node of Y. The others join p with the remaining members of Y, grouped into two disjoint nonempty subsets, D and Y - D - {q}. Furthermore, all these pieces of the original Steiner path are themselves Steiner paths connecting, respectively: {p,q}, {p} ∪ D and {p} ∪ (Y - D - {q}).

We will establish, first that a decomposition of Steiner paths at any node along that path produces *Steiner* subpaths, rather than some longer connecting path. The remaining problem is the existence of an appropriate node to decompose the Steiner path into the three pieces we have characterized above. For this latter result, we add the additional hypothesis that Y contain at least 3 nodes. Note that the Steiner path connecting 2 nodes is easily computable. It is just the shortest direct graph distance (minimal tree) between the two given nodes.

First we insert some definitions useful in stating and proving both propositions:

> Let x be any node on a Steiner tree S connecting the nodes in Y.
> Let B(x) be the set of arcs of S which touch x.
> If $C \subset B(x)$, let $Y_C(x)$ be that subset of Y which is reachable from x along paths in S whose first arc belongs to C.

Theorem 1: Let S be any Steiner tree connecting Y, and let x be any node touching an arc of S. Let $C \subset B(x)$. *Then the arcs of S involved in connecting the nodes of* $Y_C \equiv Y_C(x) \cup \{x\}$ *form a Steiner tree connecting the nodes of* Y_C.

PROOF: Node x decomposes the path S into two parts, joined at x. One portion of S, S_1, connects the nodes in Y_C; let S_2 be the remaining arcs of S. The length of $S = |S| = |S_1| + |S_2|$. Only if there were a set of arcs $P \neq S_1$ which connected the nodes in Y_C, and $|P| < |S_1|$, would S_1 not be a Steiner tree connecting Y_C. But if such a P existed, then S_1 could be replaced in S by P, making $|S^1| = |S_2| + |P| < |S_2| + |S_1| = |S|$. But this would contradict the hypothesis that S was the shortest path connecting the nodes of Y. We have thus shown that Steiner trees can be viewed as collections of subtrees joined at their roots; we now want to show the existence of a particular kind of sub-Steiner-tree collection in any Steiner tree.

Optimal Decomposition Theorem: Let S be any Steiner tree connecting Y, where $Y \subset N$ *is a subset of the nodes of graph* $G = (N,A)$, *and let q be any node of Y. If Y contains at least three members then there exists a node* $p \in N$ *and a subset D of Y such that:*

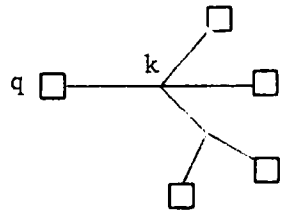
D is a proper subset of $Y - \{q\}$, *and D is nonempty.*
S consists of 3 disjoint subsets, S_1, S_2, *and* S_3.
S_1 *connects* $\{p,q\}$, S_2 *connects* $\{p\} \cup D$, *while*
S_3 *connects* $\{p\} \cup (Y - D - \{q\})$.
Furthermore, S_1, S_2, *and* S_3 *are all Steiner paths connecting their respective sets.*

Case 1: Suppose $\|B(q)\| \geq 2$. Then q itself is a suitable choice of p; for q is connected to itself by a shortest path (of length zero), and D can be chosen as $Y_a(q)$, for $a \in B(q)$.

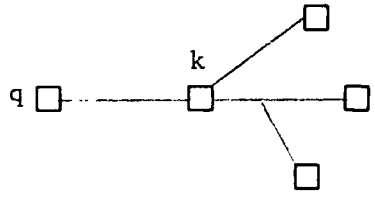
Case 2: Suppose $\|B(q)\| = 1$. Then two subcases arise; either (a): the branch of S touching q divides before reaching another $x \in Y$, or (b): it does not.

In Case a, there is a node k such that $\|B(k)\| > 2$, and k is connected to q by a path r in S which doesn't pass through any nodes of Y. Suppose r leaves k along arc b, $b \in B(k)$.

Since $\|B(k)\| > 2$, $\|B(k) - \{b\}\| > 1$, so $D = Y_c$, $c \in B(k) - \{b\}$ and $Y - D - \{q\}$ are nonempty and connected as required.



In Case b, there is some node $k \in Y$ such that k is connected to q by a path r passing through no other node of Y. Since $\|Y\| \geq 3$, $\|Y - \{k\} - \{q\}\| \geq 1$. Therefore, $\|B(k)\| \geq 2$, since r fails to touch at least one $y \in Y$. This node k is a suitable choice for p: Choose $D = \{k\}$.



REFERENCES

1. Gilbert, E. N. and H. O. Pollak, "Steiner Minimal Trees," *SIAM Journal for Applied Mathematics*, Volume 16, No. 1, pp. 1-29, (1968).

2. Floyd, R. W., "Algorithm 97, Shortest Path," *Communications of the ACM*, Volume 5, p. 345, (1962).

This research has been partially supported by the National Science Foundation under Grant GP-15473 with the University of California.

Also, the RAND Corporation, which through the U.S. Air Force under the Project RAND, supported the authors during their initial research on this problem. Reproduction in whole or in part is permitted for any purposes of the United States Government.

Paper received May 11, 1971.