Routing of Multipoint Connections

BERNARD M. WAXMAN

Abstract—**This paper addresses the problem of routing connections in a large scale packet switched network supporting multipoint communications. We give a formal definition of several versions of the multipoint problem including both static and dynamic versions. We look at the Steiner tree problem as an example of the static problem and consider the experimental performance of two approximation algorithms for this problem. Then we consider a weighted greedy algorithm for a version of the dynamic problem which allows for endpoints to come and go during the life of a connection. One of the static algorithms serves as a reference to measure the performance of our weighted greedy algorithm in a series of experiments. In our conclusion, we consider areas for continued research including analysis of the probabilistic performance of multipoint approximation algorithms.**

I. Introduction

In a packet switched network using virtual circuits, the primary goal in routing connections is to make efficient use of the network resources. For example, one favors an algorithm which can handle the largest number of connections for a given set of network resources. In a point-to-point network, routing is often treated as a shortest path problem in a graph. Here the network is modeled as a graph $G = (V, E)$ where the nodes of a graph represent switches and the edges represent links. In addition we have two functions cap: $E \rightarrow Z_0^+$ and cost: $E \rightarrow \Re^+$[1] which give us the bandwidth and the cost of each edge. At the time a connection is requested, a minimum cost (shortest) path connecting a pair of endpoints is selected. Of course only paths consisting of edges with sufficient unused bandwidth may be chosen.

Routing of multipoint connections may be modeled in a similar way. In the multipont problems, we wish to connect a set $D \subseteq V$. Instead of the shortest path, we are interested in the shortest subtree which contains the set D. Finding the shortest tree connecting a set of points is known as the Steiner tree problem in graphs. Of course the multipoint problem is complicated by the fact that the set D may change during the life of a connection.

II. Definition: The Multipoint Problem

Before proceeding, let us consider a definition for the multipoint problem. In the multipoint problem we can view the network as a graph $G = (V, E)$ with a function cap: $E \rightarrow Z_0^+$. For simplicity we will assume that this graph is undirected. We then define a connection request c to be a pair (D, b) where

- $D \subseteq V$ is the set of nodes to be interconnected
- $b \in Z^+$ is the bandwidth required for connection c.

Finally, we define a route $R_c = (\hat{D}, \hat{E})$, for connection request $c = (D, b)$, to be a connected subgraph of G such that $D \subseteq \hat{D}$ and the bandwidth constraint

$$(\forall e \in \hat{E})(b \leq \text{cap}(e)) \tag{1}$$

is satisfied. Of course, for a given connection request c there may not be any route R_c satisfying (1).

In what we shall refer to as the *basic* multipoint problem, we are given a network $N = (G, \text{cap})$ and a connection request c. We are then asked to find a route R_c for c that is optimum. The definitions that follow represent a compromise between simplicity and complete generality. These definitions capture the essential features of this problem without being overly complex. We consider three versions which we refer to as the *static* multipoint problem, the *unitary dynamic* multipoint problem, and the *dynamic* multipoint problem.

The static version of the multipoint problem is a formalization of the *basic* problem. In this version, we assume that a function cost: $R \rightarrow \Re^+$ is defined for a network N where R is the set of all possible routes on N. The static problem is stated as follows. Given a network N and a connection request c, find a route R_c such that cost (R_c) has a minimum value among all possible routes for c. The Steiner tree problem in graphs is a specific version of the static multipoint problem which we shall discuss shortly.

In the *unitary dynamic* version of the multipoint problem, we start with a network $N = (G, \text{cap})$ and consider a sequence of requests $S = [r_0, \cdots, r_n]$ where each r_i is a pair (v, d), $v \in V$, $d \in \{add, remove\}$. Each request includes a node of the network which is to be either added to, or removed from, a connection. Thus, in the *unitary dynamic* problem we are given a network N, a bandwidth b, and a sequence of requests S. We are asked to find a sequence of routes R_i, $i \in [0, \cdots, n]$ which yields a minimum cost sequence subject to the bandwidth constraint (1). We may wish to consider several measures for the cost of a sequence such as average cost or maximum cost. In this version, once a particular set of edges has been used in a route, no reconfiguration is allowed as the algorithm proceeds. Note that if we were to allow reconfiguration, this problem would be equivalent to a sequence

Manuscript received October 30, 1987; revised June 8, 1988. This work was supported by Bell Communications Research, Italtel SIT, NEC, and National Science Foundation under Grant DCI-8600947.

The author is with Computer and Communications Research Center, Washington University, St. Louis, MO 63130.

IEEE Log Number 8824403.

[1]We use Z_0^+ ($\Re_0^+$) to stand for the positive integers (reals) including 0, and Z^+ ($\Re^+$) to stand for the positive integers (reals) excluding 0.

0733-8716/88/1200-1617$01.00 © 1988 IEEE

of static problems. Clearly, an optimal algorithm which allows reconfiguration will always do at least as well as one for which reconfiguration is not allowed.

In the *dynamic* version of the multipoint problem we again are given a sequence of requests S which are triples of the form (c, v, d), $c \in C$ where C is a set of connection identifiers. Each $r_i \in S$ is a request to add or delete a node with respect to a particular connection. In addition we are given a function B: $C \rightarrow Z^+$ which associates a bandwidth with each connection. This version comes in two flavors, one which allows reconfiguration and a second for which reconfiguration is not allowed. Finally, in the *dynamic* multipoint problem, we impose the additional constraint that the sum of the bandwidths of all routes using a given edge e may not exceed the capacity of that edge. More formally we have that for all $i \in [0, \cdots, n]$

$$(\forall e \in E)\left(\text{cap}(e) \geq \sum_{R_{c_j} \in \Re_i, e \in R_{c_j}} b_j\right) \qquad (2)$$

where $\Re_i$ is the set of all routes created by an algorithm after request i and R_{c_j} is the route for connection c_j.

Here, we measure the performance of an algorithm by the number of requests of type *add* that it can service. If $a(N, S, C, B)$ is the number of *add* requests serviced by algorithm a, we define the number of requests serviced by an optimal algorithm for a given instance (N, S, C, B) of the *dynamic* multipoint problem as

$$O(N, S, C, B) = \max_{a \in A}\left(a(N, S, C, B)\right) \qquad (3)$$

where A is the collection of all *dynamic* routing algorithms which satisfy the bandwidth constraint (2). Note that the algorithms in A may or may not be allowed to use reconfiguration, depending on which version of the *dynamic* multipoint problem we consider.

III. Static Approximation Algorithms for the Steiner Tree Problem

The Steiner tree problem, was not originally conceived as a problem in communication but as a problem in geometry. The original version of the Steiner tree problem derives its name from Jacob Steiner [5] and was studied even earlier by Fermat [2]. The version most closely related to multipoint routing is the Steiner tree problem in graphs, which was shown to be NP-complete by Karp in 1972 [10]. Formally, the Steiner problem in graphs can be stated as follows. Given

- a graph $G = (V, E)$
- a cost function cost: $E \rightarrow \Re^+$, and
- a set of vertices $D \subseteq V$,

find a subtree $T = (V_T, E_T)$ of G which spans D, such that the cost $(T) = \Sigma_{e \in E_T} \text{cost}(e)$ is minimized.

Since the Steiner problem is NP-complete an algorithm that finds a minimum Steiner tree will not run in polynomial time unless $P = NP$. Therefore, we would like to find approximation algorithms which have worst case performance as well as average performance that is close to optimum. There are a number of heuristics for deriving solutions to the Steiner tree problem in graphs which run in polynomial time. At this time, none of these is known to give a solution that is better than twice optimal in the worst case [17], [16].

The approximation algorithms described here are not practical for very large networks, since they assume complete knowledge of the network and do not operate in a distributed fashion. Such algorithms are of theoretical interest and should be useful as tools to measure the performance of other routing algorithms. In addition, it is hoped that the study of such algorithms will contribute to a better understanding of the multipoint routing problem.

We have been looking at several heuristics for the Steiner tree problem in graphs. These include a heuristic which is based on minimum spanning tree algorithms, which we refer to as MST [14], [1], [7], [9], a contraction heuristic [11] and a heuristic due to V. J. Rayward-Smith [12], which we refer to as RS. Both MST and the contraction heuristic have worst case performance two times the cost of an optimal solution. In his paper, Rayward-Smith does not develop any results regarding the worst case performance of RS. We have recently proved [16] that the worst case bound for RS is also two, even though experimental studies discussed in this paper and in [13] indicate that RS does better than MST on average. In addition, analysis of probabilistic performance [15] also gives the edge to RS.

We have implemented both the MST and RS algorithms as well as an improved version of each. The improved versions result from a modification of the version of MST suggested by Kou, Markowsky, and Berman [7]. It should be noted that the improved versions do not have better worst case performance but do at least as well as the basic algorithms.

We briefly describe MST referring the reader to one of the references cited for more details.

Construct a complete graph $G' = (D, E')$ where $\text{cost}_{G'}(u, v)$ is the length of the shortest path from u to v in G.

Construct a minimum spanning tree T' for G'.

Convert the edges in T' to paths in G to form a solution T_1.

Fig. 1 illustrates the application of MST to a nine node graph, with a set $D = \{a, d, e, g\}$, and with the cost of each edge indicated. The complete graph $G[D]$, a minimum spanning tree for $G[D]$, and a final solution are shown. Note that in this example the solution given by MST is not optimum.

Kou, Markowsky, and Berman take the subgraph T_1 generated by basic MST and apply a minimum spanning tree algorithm to T_1. From the tree that results they prune branches which do not contain any vertexes in D, giving their solution to the Steiner problem. What we have done is to take the nodes D_1 in T_1, rerun the basic MST on this set of nodes to produce a subgraph T_2, and then prune those branches which do not contain any vertexes from D. This version of MST will yield a tree with cost that is

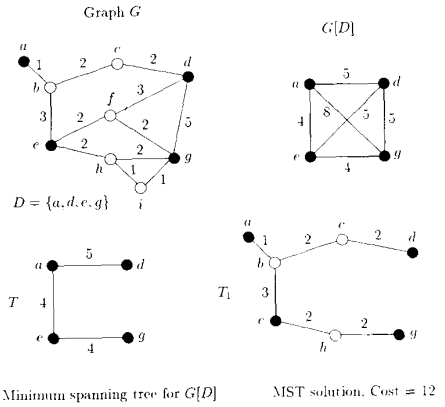

Fig. 1. An example of the application of MST to a graph.

no greater than the cost of the solution given by the basic algorithm, but in the worst case does not do any better.

We now consider the RS algorithm proposed by Rayward-Smith. The problem with the basic MST algorithm is that it only considers the distances between nodes in the set D and does not consider the value to a final solution of nodes not in D. The nodes outside of D that are important to a solution of the Steiner tree problem will be called Steiner points. More formally, we define a node s, in a Steiner tree $T = (V_T, E_T)$ for a set D, to be a Steiner point if $s \in V_T$, $s \notin D$ and $\deg_T(s) \geq 3$. These are the nodes that RS attempts to find. Rayward-Smith defines a function f: $V \rightarrow \Re^+$ such that those nodes for which the function has a minimum value are the ones that should be included in a final solution. We give a brief description of RS referring the reader to [12], [13] for a more details.

Construct a collection of single node trees $\mathfrak{I}$ consisting of the nodes in D.

Repeat the following steps until $|\mathfrak{I}| = 1$.

Choose $v \in V$ such that $f(v)$ is minimum where

$$f(v) = \min_{S \subseteq \mathfrak{I}, |S| > 1} \left\{ \frac{1}{|S| - 1} \sum_{T \in S} \text{dist}(v, T) \right\}.$$

Let T_1 and T_2 be two closest trees to v.

Join T_1 and T_2 by a shortest path through v.

Informally, f is the average cost of making r joins to $r + 1$ trees through a node v where $r + 1 = |S|$.

The application of RS to an eight node graph is illustrated in Fig. 2. In this example, the *repeat* statement is executed four times. The collection of trees $\mathfrak{I}$ and the values for f, at each iteration, are shown in Fig. 2 and 3.

RS uses the function f to determine if a node other than those in D will be useful in solving the Steiner problem. Whenever RS chooses such a node, it has found a potential Steiner point. As can be seen in Fig. 4, we have a set of instances where MST will yield a solution whose cost is almost twice that of an optimal solution while RS will actually find an optimal solution.

To gain some insight into the probable performance of MST and RS, we implemented both the basic and the improved versions. In our experimental tests of these algorithms, we also generated an optimal solution for each test case.

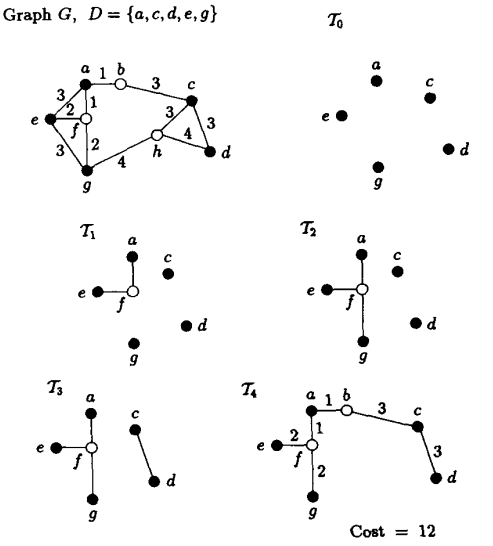

Fig. 2. The application of RS to an eight-node graph.

	f_0	f_1	f_2	f_3
a	3	3	4	4
b	4	4	4	4
c	3	3	3	4
d	3	3	3	7
e	3	3	7	7
f	2.5	2	5	5
g	3	2	7	7
h	5.5	5.5	5.5	7

Fig. 3. The values of $f_l(v)$ for $l \in [0, \cdots, 3]$.

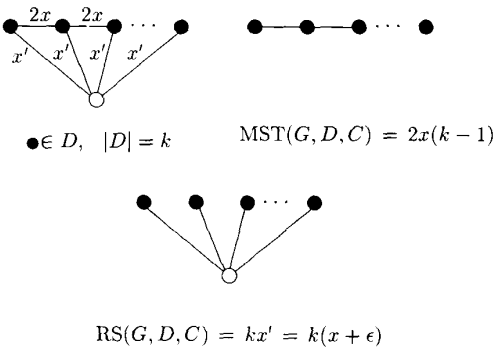

Fig. 4. MST approaches twice optimal, RS yields an optimal solution.

For the purpose of running these experiments, two simple random graph models RG1 and RG2, which have some of the characteristics of an actual network, were used. In RG1, n nodes are randomly distributed over a rectangular coordinate grid. Each node is placed at a location with integer coordinates. The Euclidean metric is then used to

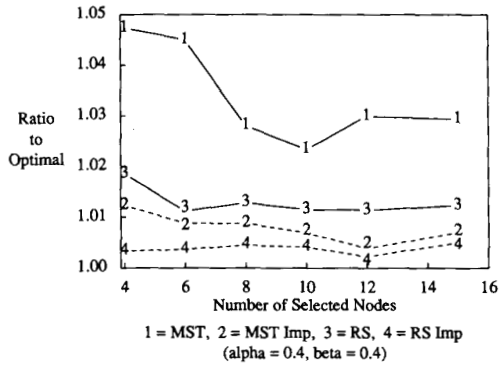

Fig. 5. Average performance of MST and RS.

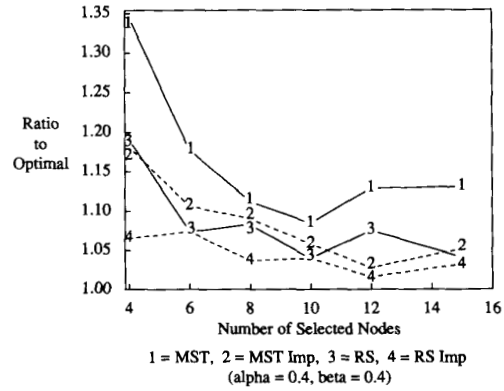

Fig. 6. Worst case performance of MST and RS.

determine the distance between each pair of nodes. In RG2, for each pair of nodes, a distance is chosen in (0, L] from a uniform random distribution. For both models, edges are introduced between pairs of nodes u, v with a probability that depends on the distance between them. The edge probability is given by

$$P(\{u, v\}) = \beta \exp \frac{-d(u, v)}{L\alpha} \qquad (4)$$

where $d(u, v)$ is the distance from node u to v, L is the maximum distance between two nodes, and α and β are parameters in the range (0 , 1]. Larger values of β result in graphs with higher edge densities, while small values of α increase the density of short edges relative to longer ones. Finally, we set the cost of an edge equal to the distance between its endpoints.

The experimental results of applying MST and RS to RG1 are illustrated in Figs. 5 and 6. For these experiments, five different 25 node random graphs were generated according to the RG1 model. Figs. 5 and 6 show the results of running 25 simulations, five for each graph, for subsets D of a given size. Each subset D contained nodes selected at random from the 25 node graphs. In each figure, the horizontal axis represents the size of the sets D, while the vertical axis indicates the ratio of the cost of a solution given by the algorithms to an optimal one. In Fig. 5, average values are plotted for the cost of the basic and improved solutions for both algorithms. In Fig. 6, worst case results for each set of simulations is plotted. As these figures indicate both MST and RS when applied to RG1 generally yield results which are near optimum with the improved version of RS[2] giving the best performance. Our experiments also indicate that the results are essentially the same for graph model RG2 and a wide range of values for α and β.

IV. A Dynamic Algorithm for the Steiner Tree Problem

Let us next turn our attention to the *unitary dynamic* multipoint problem. Here individual nodes are allowed to join or leave a connection at any point during the life of that connection. This version of the multipoint problem considers the dynamic nature that we expect from real applications, although it is still a simplification of the real problem since it considers a single connection in isolation. We have run experiments with a version of the *unitary dynamic* multipoint problem using both a greedy and a weighted greedy algorithm. In the greedy algorithm, to add a node u to an existing connection, we first find a closest node v already in the connection and then connect u to v by a shortest path. We delete a node from the connection by first marking it deleted. If that node is not an internal node in the connection, we prune the branch of which it is a part. In the weighted greedy algorithm, a distinguished node o is chosen to be the owner of the connection, with the restriction that node o must remain in the connection. We add a node u to a connection by choosing a node v, in the connection, which minimizes the function

$$W(v) = (1 - \omega)\, d(u, v) + \omega d(v, o) \qquad (5)$$

where $0 \le \omega \le 0.5$ and $d(x, y)$ is the distance from x to y. When $\omega = 0$ the algorithm is equivalent to the basic greedy algorithm. When $\omega = 0.5$, we add a node u to the connection by the shortest path to the owner o. Fig. 7 illustrates a sequence of five events handled by the greedy algorithm. Note that events three and four are examples of a situation where the greedy algorithm is not optimal.

In our experiments we used our improved version of MST to measure the performance of the greedy algorithm. To run these experiments we used a simple probability model to determine if an event was to be an addition or deletion of a node from the connection. The function

$$P_C(k) = \frac{\alpha(n - k)}{\alpha(n - k) + (1 - \alpha)k} \qquad (6)$$

was defined for this purpose. Here, P_C is the probability that an event is the addition of a node, k is the number of nodes in set D of the current connection, n is the number of nodes in the network, and α is a parameter in (0, 1). The value of α is the fraction of nodes in the connection

[2]Improved RS is analogous to the improved version of MST. We run RS a second time on the set of nodes generated by the initial RS solution.

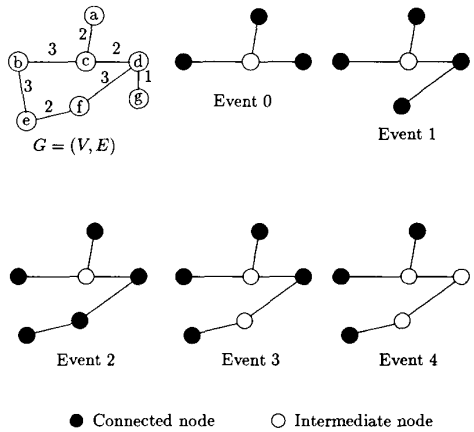

Fig. 7. Greedy algorithm with sequence: $(a, b, d), f, e, \bar{f}, \bar{d}$.

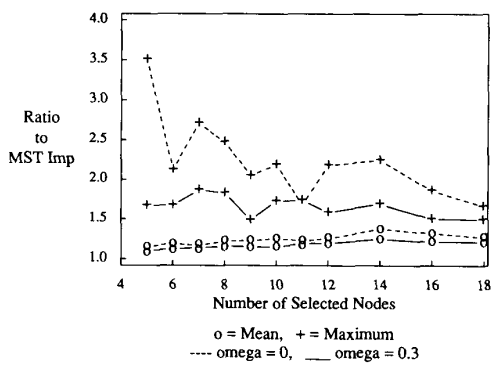

Fig. 8. Performance of the weighted greedy algorithm.

at equilibrium. For example, when $k/n = \alpha$, $P_C(k) = 1/2$.

We ran several sets of experiments on 195 node graphs. Fig. 8 indicates the results of one set of experiments. For each experiment we ran a sequence of 1000 events using a fixed value for α. The horizontal axis indicates the value of αn or, in other words, the number of nodes k in the connection for which $P_C(k) = 1/2$. The vertical axis represents the ratio of the cost of the solutions produced by the greedy algorithms to the cost of the solutions produced by improved MST. The results of the worst case ratio and the average ratio for each experiment are plotted for $\omega = 0$ and $\omega = 0.3$. We can see from Fig. 8 that the greedy algorithm[3] does reasonably well for most events in comparison to MST. Those cases where the greedy algorithm does much worse than MST occur after several nodes have left the connection. In those cases several internal nodes have been deleted from D but must remain in the connection since they are part of a branch that contains other nodes in D.

This phenomenon of poor performance after several nodes leave the connection would be expected with any algorithm that does not allow for reconfiguration after each event. It is under these circumstances that the weighted algorithm shows the greatest improvement. Note that for the set of experiments shown in Fig. 8 the worst case performance improves from 3.5 to 2.0 with the weighted algorithm using $\omega = 0.3$. From our experiments it appears that a value for ω in the neighborhood of 0.3 yields the best results. There are several other techniques that might be used to smooth out the performance of the greedy algorithm. For example, we might weight the choice of a path to a node v in the connection by the distance from v to the center of the connection, or perhaps by the number of connections routed through v.

[3] $\omega = 0$.

V. Conclusions

Our experiments with MST and RS indicate that they give near optimal results on our two graph models RG1 and RG2. For example, the improved version of RS gave performance within 8 percent of optimal for the entire set of experiments and within 2 percent of optimal on average. We are now looking at ways to obtain analytic results for the probable performance of Steiner tree approximation algorithms on several different random graph models [15]. Still open is the minimum value for the worst case performance of a polynomial time bounded Steiner tree algorithm, assuming $P \neq NP$.

Experimental results indicate that our weighted greedy algorithm gives reasonable results when compared to MST. Even though we got the best results with the weighting factor $\omega = 0.3$, a value for ω of 0.5 still does reasonably well. Since $\omega = 0.5$ is equivalent to shortest path routing these results indicate that extending a point-to-point routing scheme to the multipoint problem may be a reasonable thing to do.

Experimental studies of the extension of point-to-point routing as well as the extension of the weighted greedy algorithm to the general dynamic problem are worth additional consideration. These extension may then prove useful as a basis for the construction of a practical multipoint algorithm for a large network.

References

[1] K. Bharath-Kumar and J. M. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Trans. Commun.*, vol. COM-31, pp. 343-351, Mar. 1983.

[2] H. S. M. Coxeter, *Introduction to Geometry*. New York: Wiley, 1961.

[3] S. E. Dreyfus and R. A. Wagner, "The Steiner problem is graphs," *Networks*, vol. 1, no. 3, pp. 195-207, 1971.

[4] E. N. Gilbert, "Random graphs," *Ann. Math. Stat.*, vol. 30, pp. 1141-1144, 1959.

[5] E. N. Gilbert and H. O. Pollak, "Steiner minimal trees," *SIAM J. Appl. Math.*, vol. 16, no. 1, pp. 1-29, 1968.

[6] J. M. Jaffe, "Distributed multi-destination routing: The constraints of local information," *SIAM J. Comput.*, vol. 14, no. 4, pp. 875-888, Nov. 1985.

[7] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, pp. 141-145, 1981.

[8] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *IEEE Trans. Commun.*, vol. COM-28, pp. 711-719, May 1980.

[9] H. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs," *Inform. Process. Lett.*, vol. 27, no. 3, pp. 125-128, Mar. 1988.

[10] R. E. Miller and J. W. Thatcher, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. M. Karp, Ed. New York: Plenum, 1972, pp. 85–103.
[11] J. Plesnik, "A bound for the Steiner tree problem in graphs," *Math. Slovaca*, vol. 31, no. 2, pp. 155–163, 1981.
[12] V. J. Rayward-Smith, "The computation of nearly minimal Steiner trees in graphs," *Int. J. Math. Ed. Sci. Tech.*, vol. 14, no. 1, pp. 15–23, 1983.
[13] V. J. Rayward-Smith and A. Clare, "On finding Steiner vertices," *Networks*, vol. 16, pp. 283–294, 1986.
[14] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math. Japonic*, vol. 24, no. 6, pp. 573–577, 1980.
[15] B. M. Waxman, "Probable performance of Steiner tree algorithms," Washington Univ. Dep. Comput. Sci., St. Louis, MO, WUCS-88-4, 1988.
[16] B. M. Waxman and M. Imase, "Worst case performance of Rayward-Smith's Steiner tree heuristic," Washington Univ., Dep. Comput. Sci., St. Louis, MO, WUCS-88-13, 1988.
[17] P. Winter, "Steiner problem in networks: A survey," *Networks*, vol. 17, pp. 129–167, 1987.

Bernard M. Waxman was born in St. Louis, MO, on February 25, 1944. He received the B.S. degree in mathematics from Washington University, St. Louis, MO, in 1966, and the M.A. degree in mathematics from the University of Illinois, Urbana, IL, in 1968. More recently, he received the M.S. degree in computer science from Washington University, St. Louis, in 1985. He is currently working on a Doctorate degree in computer science at Washington University in the Advanced Communications Systems (ACS) group.

He has been investigating the multipoint routing problem as it relates to the large scale broadcast packet network being developed by ACS. Two areas of particular interest to him are the study of the probabilistic performance of Steiner tree approximation algorithms and the development of new Steiner tree algorithms with improved worst case performance. Previous to his work with ACS, he was the part owner of a professional photoprocessing laboratory.