Report Number: WUCS-87-10

1987-05-01

Thesis Proposal: Routing of Multipoint Connections

Bernard M. Waxman

This proposal addresses the problem of routing connections in a large scale packet switched communications system supporting multipoint communication. In this proposal a number of topics are considered including: the requirements imposed by routing in a large system, the Steiner tree problem of distributed algorithms which have access to limited information. In addition, this proposal outlines the work done to date and work that remains to be done.

THESIS PROPOSAL: ROUTING OF MULTIPOINT CONNECTIONS

Bernard M. Waxman

WUCS-87-10

May 1987

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Abstract

This proposal addresses the problem of routing connections in a large scale packet switched communications system supporting multipoint communication. In this proposal a number of topics are considered including: the requirements imposed by routing in a large system, the Steiner tree problem as an abstract version of multipoint routing, random graph models for real networks, and the problem of distributed algorithms which have access to limited information. In addition, this proposal outlines the work done to date and work that remains to be done.

This work supported by Bell Communications Research, Italtel SIT and National Science Foundation grant DCI-8600947.

THESIS PROPOSAL: ROUTING OF MULTIPOINT CONNECTIONS

Bernard M. Waxman

1. Introduction

This proposal addresses the problem of routing connections in a large scale packet switched communications system. The problem of routing in a network can be broken into three sub-problems which include: point to point, all-point, and multipoint routing. The first two have been studied extensively and should be less difficult to deal with. Multipoint routing is somewhat more complex and has not been investigated as extensively as point to point and all-point routing. Consequently this research will focus on the multipoint problem.

One aspect of this research will involve the development of efficient routing techniques for a large scale general purpose packeted switched network, with special consideration given to the multipoint problem. The algorithms which we develop must be able to handle an arbitrary network topology and yield connections which are reasonably close to optimal. At the same time the algorithms should require minimum network resources to determine the route for each connection. For example, it will be desirable to keep to a minimum the quantity of routing information that must be stored at each node, the computation time required to route a connection and the resources needed to update the routing data stored at each node.

I also plan to investigate the Steiner tree problem in graphs, which is an abstraction of the multipoint problem. Since this problem is NP-hard it will be necessary to investigate heuristics for solving the Steiner tree problem. This work will include investigation of the worst case and probable (average case) performance of these heuristics.

For the past several months I have been compiling a bibliography of the literature dealing with routing in communications networks. A considerable body of information dealing with point to point routing is available. In addition numerous packet switched networks have implemented point to point routing [2]. On the other hand there is a much smaller body of research dealing with the multipoint problem. The simpler problem of all-point routing in which a packet is sent to all nodes has a reasonably simple solution, and has been implemented in some existing networks [14]. It should be noted that many of the existing packet switched networks route each packet individually, a model sometimes referred to as datagrams. In the system proposed by the Advanced Communications Systems group a virtual circuit routing scheme will be implemented. Therefore, some of the existing point to point algorithms are not directly applicable to this network.

With regard to the multipoint problem I have looked at several heuristics for solving the Steiner tree problem in graphs. I have run computer simulations with one heuristic, referred to as the MST heuristic by Brath-Kumar and Jaffe [1]. The purpose of these experiments was to evaluate the performance of MST, with the intention of using it as a standard against which the performance of potential routing algorithms can be measured. In addition a number of simulations have been run

using a greedy algorithm that handles dynamic connections and disconnections, whose performance has been measured with reference to the MST heuristic. This algorithm is closer to one that might be used in an actual network, though it still requires complete knowledge of the network topology, which is not reasonable for a large scale communication network.

One of earliest packet switched networks was developed for the Defense Advanced Research Projects Agency and was appropriately named the ARPANET. The ARPANET was implemented in 1969 and is the oldest packet switched network. McQuillan et al. give an overview of the algorithm used for routing in "A Review of the Development and Performance of the ARPANET Routing Algorithm " [13]. This network implemented a point to point routing algorithm to route individual packets, with the object of achieving minimum transmission delay. The original algorithm maintained, at each node, a routing table of link numbers indexed by the set of network nodes. When a packet passed through a node in the network, the destination address in the header of that packet was used to index the routing table to find the outgoing link for that packet. These routing tables were updated periodically to minimize a cost function, based on delay and the number of links along a path. The techniques used for routing by the ARPANET worked fairly well for moderate sized networks. As the ARPANET grew, problems were encountered especially with regard to updating the routing tables. This growth lead to the development and implementation of a new routing algorithm [14]. The modifications dealt primarily with the way in which routing information is updated. In both cases each node has information about all destinations in the network. Hence, the techniques used for routing in the ARPANET are probably not practical for a very large communication network. It is interesting to note that this new routing algorithm incorporates the ability to send all-point packets, used to update routing information at each node, by a technique known as flooding.

During the same period of time work was underway at the National Physical Laboratory in Britain. This experimental network began operation in 1973 and probably influenced the proposal by the British post office to develop an experimental packet switched service in 1972 [2]. Tymnet, which was established in 1971, uses the concept of virtual circuits to do its routing and, therefore, is of special interest to us. That is the algorithm establishes a fixed path for each connection, and the path is maintained during the life of the connection. At the present time there are a sizable number of large scale packet switched networks throughout the world including those in North America, Western Europe and Southeast Asia. Most of the current packet networks support point to point communication for data interchange and are not set up as general purpose communication networks.

2. Definitions and the Steiner Tree 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 $\text{cap}: E \rightarrow \Re^+$. For simplicity we will assume that this graph is undirected, even though in the more general case it may be appropriate to use a directed graph as the model of a network. Next we define a connection request c to be a pair (D, b) where

- $D \subset V$ is the set of nodes to be interconnected
- $b \in \Re^+$ 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 $D \subset \hat{D}$ and such that the bandwidth constraint

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

is satisfied.

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 in some sense is optimum. Informally a route is optimum if it makes efficient use of the network resources. That is if we are given a set of connections request C on a network N and two different routing algorithms, the algorithm which is able to handle the largest subset of C without exceeding the capacity of the network is the better algorithm, at least with respect to the set C. We have considered a number of formal definition for multipoint routing and what it means for a routing algorithm to be optimal. The definitions that follow represent a compromise between simplicity and complete generality. The following definitions capture the essential features of this problem without being overly complex. We consider three versions of the problem 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 described previously. In this version we assume that a function $\text{cost} : \mathcal{R} \rightarrow \Re^+$ is defined for a network N, where $\mathcal{R}$ 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 $\text{cost}(R_c)$ has a minimum value among all possible routes for c. The Steiner tree problem in graphs is an example of a specific version of the static multipoint problem which will be discussed shortly.

In the *unitary dynamic* version of the multipoint problem, we start with a network $N = (G, \text{cap})$ where $G = (V, E)$ and consider a sequence of requests $S = [r_0, \ldots r_n]$, 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 and are asked to find a sequence of routes $R_i, i \in [0..n]$ which satisfy the largest number of requests of type add. The *unitary dynamic* problem can be further subdivided into two cases. In the first case once a particular set of edges has been used in a route no reconfiguration is allowed as the algorithm proceeds. In the second case reconfiguration is allowed for a connection as the algorithm progresses through the sequence of requests. An optimal algorithm is one that for a given input will always satisfy the maximum number of requests possible subject to the bandwidth constraint (1) for a route. 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, in this case, are triples of the form (c, v, d) where v and d are as above and $c \in C$, where C is a collection of connections. Hence each request $r_i \in S$ is a request to add or delete a node with respect to a particular connection. In addition we are given a function $B : C \rightarrow \Re^+$ which associates a bandwidth with each connection. Note that this version also 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 $i \in [0..n]$

$$(\forall e \in E)(\text{cap}(e) \geq \sum_{R_{c_j} \in R_i,\, e \in R_{c_j}} (b_j)) \tag{2}$$

where R_i is the set of all routes created by an algorithm at time i.

We again 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 requests serviced by algorithm A, we define the number of request serviced by an optimal algorithm for a given instance (N, S, C, B) of the *dynamic* multipoint problem as

$$|O(N, S, C, B)| = \max_{A \in \mathcal{A}}(|A(N, S, C, B)|) \tag{3}$$

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

The Steiner tree problem, an example of a *static* multipoint problem, has a long history. This 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 [10] and was studied even earlier by Fermat [3]. Steiner was interested in the problem of connecting three points in a plane in such a way so that the sum of the lengths of the edges would be a minimum. This problem naturally extends to the Steiner tree problem in a plane, in which the object is to connect a set of points with straight edges to produce a tree that is of minimum cost, where cost is the sum of the edge lengths.

It is often the case that a minimum tree includes nodes, called Steiner points, which are not among the set of original points. This makes the problem more difficult then it might at first appear to be. In fact the Steiner tree problem (in the Euclidean plane) has been shown to be NP-hard by Garey, Graham and Johnson [7] in 1977. Other versions of the Steiner problem include finding a minimum tree for a set of points in the plane with other than the Euclidean metric. An example is the rectilinear Steiner problem, which uses the Manhattan metric and was shown to be NP-complete, when stated as a decision problem, by Garey and Johnson [8]. The rectilinear version has direct application to the design of circuit layouts. The problem can be extended to sets of points in higher dimension spaces. The version most closely related to multipoint routing is the Steiner tree problem in graphs. This version of the problem is also NP-complete as shown by Karp in 1972 [15]. In this version a graph $G = (V, E)$ is given with a cost function $C: E \rightarrow \Re^+$. The problem is to connect a set of points $D \subset V$ with a minimum cost subgraph. This subgraph is called the minimum Steiner tree. The Steiner tree problem in graphs is a specific example of the static multipoint problem, and corresponds to the problem of connecting multiple points in a network where graph edge costs correspond to fixed link cost in a network.

An algorithm that finds an optimal solution, a minimum Steiner tree, to the Steiner problem will not run in polynomial time unless P = NP. Thus, it becomes necessary to find a heuristic to solve any practical version of this problem. In fact it has been shown by Jaffe [11] that with certain limited knowledge, i.e. where no node has knowledge of the entire graph topology, that in some cases it is not even possible to derive an optimal solution. Therefore, the need arises to find heuristics which will preferably have worst case performance close to optimal and which must have average performance which is close to optimal. At present there are several heuristics [12,17,18] for deriving solutions to the Steiner tree problem in graphs that run in polynomial time. But at this time none of these is known to give a solution that is better than twice the optimal solution in the worst case. There are also algorithms that will give an optimal solution [4] and although they run in exponential time they are more efficient than exhaustive search and may be useful for comparison purposes.

Even these approximation algorithms 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 have theoretical interest and should be useful as tools to measure the performance of other routing algorithms. In addition it is hoped the the study of such algorithms will contributed to our understanding of the multipoint routing problem and help us to develop algorithms for an actual network.

The all-point problem can be considered a special case of the multipoint problem, which corresponds to the Steiner tree problem where $D = V$. This special case of the Steiner problem corresponds to finding a minimum spanning tree for G, a problem for which there are polynomial time algorithms. In fact the two Steiner tree heuristics discussed in the sections that follow will always produce an optimal solution when $D = V$. All-point routing has been used for such tasks as updating routing tables and distributing information about changes in the network to the various

nodes. Since this type of information is not normally transmitted in high volumes the efficiency of the algorithms used in regard to network resources used has not been critical.

3. Static Approximation Algorithms for the Steiner Tree Problem

Formally the minimum Steiner tree problem can be stated as follows. Given

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

find a subtree $T = (V_T, E_T)$ of G which spans D, such that the cost$(T) = \sum_{e \in E_T} C(e)$ is minimized. The Steiner tree problem can be stated as a decision problem, where we are given G, C and D as above. We then ask, for a fixed $\beta \in \Re^+$, if there is a subtree T of G spanning D such that the cost$(T) \leq \beta$. This decision problem was first proved to be NP-complete by Karp.

I 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 I have referred to as MST [1,12], a contraction heuristic [17] and a heuristic due to V.J. Rayward-Smith [18], which I shall refer to as Rayward-Smith's heuristic (RS). Both MST and the contraction heuristic have a worst case performance which yields a solution that has cost within a factor of two of the cost of an optimal solution. In addition it can be demonstrated that there are cases where the performance of each of these heuristics can yield a solution which has cost nearly twice that of an optimal solution. In his paper Rayward-Smith does not develop any results regarding the worst case performance of his algorithm. I have found a class of graphs on which the performance of RS can have a cost nearly $3/2$ that of the optimal solution and have proved that its worst case performance is within a factor of two of optimal. Thus, it appears that RS may have a worst case performance that is better than either of the other heuristics.

During the past summer we implemented the basic MST heuristic and an improved version of MST which is a modification of the version suggested by Kou, Markowsky and Berman [12]. It should be noted that this improved version does not have better worst case performance but does do at least as well as the basic algorithm. In the description of the basic MST algorithm

- let $n = |V|$
- let $m = |E|$
- let $k = |D|$
- let $T_1 = (D_1, E_1)$ (basic solution)

The basic MST heuristic can then be stated as follows.

1. Construct the complete graph $G[D] = (D, E')$ with
   edge weights given by $C_1: E_1 \rightarrow \Re^+$, where
   $C_1(\{u, v\}) = \text{dist}_G(u, v)$ is the length of a shortest path from u to v in G.
   C_1 can be calculated in $O(km\frac{\log n}{\log(m/n+2)}) = O(kn^2)$ using Dijkstra's algorithm [20].
2. Construct a minimum spanning tree T for $G[D]$
   Use a standard algorithm e.g. Prim's algorithm which has running time $O(k^2)$ [20].

3. Convert edges in T to paths in G to form $T_1 = (D_1, E_1)$
   Take advantage of path overlap in G. This can be done in time $O(n^2)$.

The total running time for basic MST is $O(kn^2)$

Figure 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 optimal.

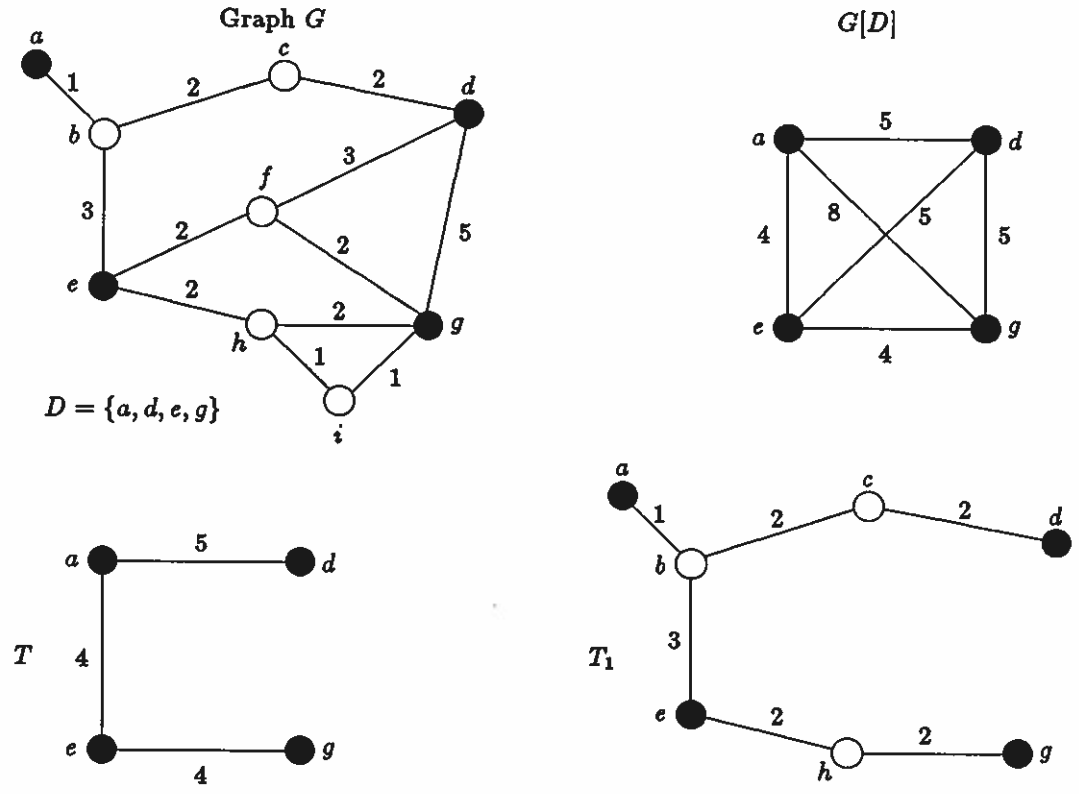

Figure 1: An Example of the Application of MST to a Graph

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 vertices in D, thus giving their solution to the Steiner problem. What I 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 that do not contain any vertices from D. Because the size of D_1 may be as large as n the improved heuristic has running time $O(nm\frac{\log n}{\log(m/n+2)}) = O(n^3)$ This version of MST will yield a tree with cost that is no greater than the cost of the solution given by the basic heuristic, but in the worst case does not do any better.

To gain some insight into the probable performance of MST we implemented both the basic version and the improved version of MST using the C++ language. In our experimental tests of these heuristics we also generated an optimal solution for each test case.

For the purpose of running these experiments a simple random graph model (RG1), which has some of the characteristics of an actual network, was used. In RG1 n nodes are randomly distributed over a rectangular coordinate grid. Each node is placed at a location with integer coordinates. Edges are then introduced between pairs of nodes u, v with a probability that depends on the Euclidean distance between them. The probability is given by

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

where $d(u,v)$ is the Euclidean distance from node u to v, L is the distance along the diagonal of the rectangular coordinate grid, 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.

The experimental results of applying MST to RG1 are illustrated in figures 2 and 3. For these experiments five different 25 node random graphs were generated according to the RG1 model. Figures 2 and 3 show the results of running 50 simulations, 10 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 the size of the sets D, while the vertical axis indicates the ratio of the cost of a solution given by MST to an optimal one. Thus, the result of an algorithm which produces optimal routings would appear as a horizontal line at a height of 1. In figure 2 the average values are plotted for the cost of the tree in the graph $G[D]$, the cost of the basic solution, and the cost of the improved solution. In figure 3 the worst case results for each of the 50 simulations are plotted. As these figures indicate the basic MST heuristic applied to RG1 generally yields results which are near optimal. This set of experiments also indicate that our improved version of MST, on the average, yields improved performance.

At this point there are several problems which I would like to investigate regarding RG1 and the MST heuristic.

- How do the characteristics of a graph G generated by RG1 depend on the parameters α and β? That is for what values of these parameters will the graphs almost always be connected, for what values will they almost always be disconnected, etc.
- For a given network model, what is the expected performance of the MST heuristic in relation to an optimal solution? I would like to answer this question for our random network model as well as some other possibly simpler models. Some other models that might be considered include the random graphs of E. N. Gilbert [9] with unit cost links, complete graphs with random assignment of link costs and complete graphs with unit link costs.
- More generally I would like to develop tools for analyzing the probable performance of any Steiner tree heuristic.

We now consider a second heuristic proposed by Rayward-Smith, which we call RS. The problem with the basic MST heuristic 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 $\deg_T(s) \geq 3$. It is these nodes that RS attempts to find. Rayward-Smith defines a collection of

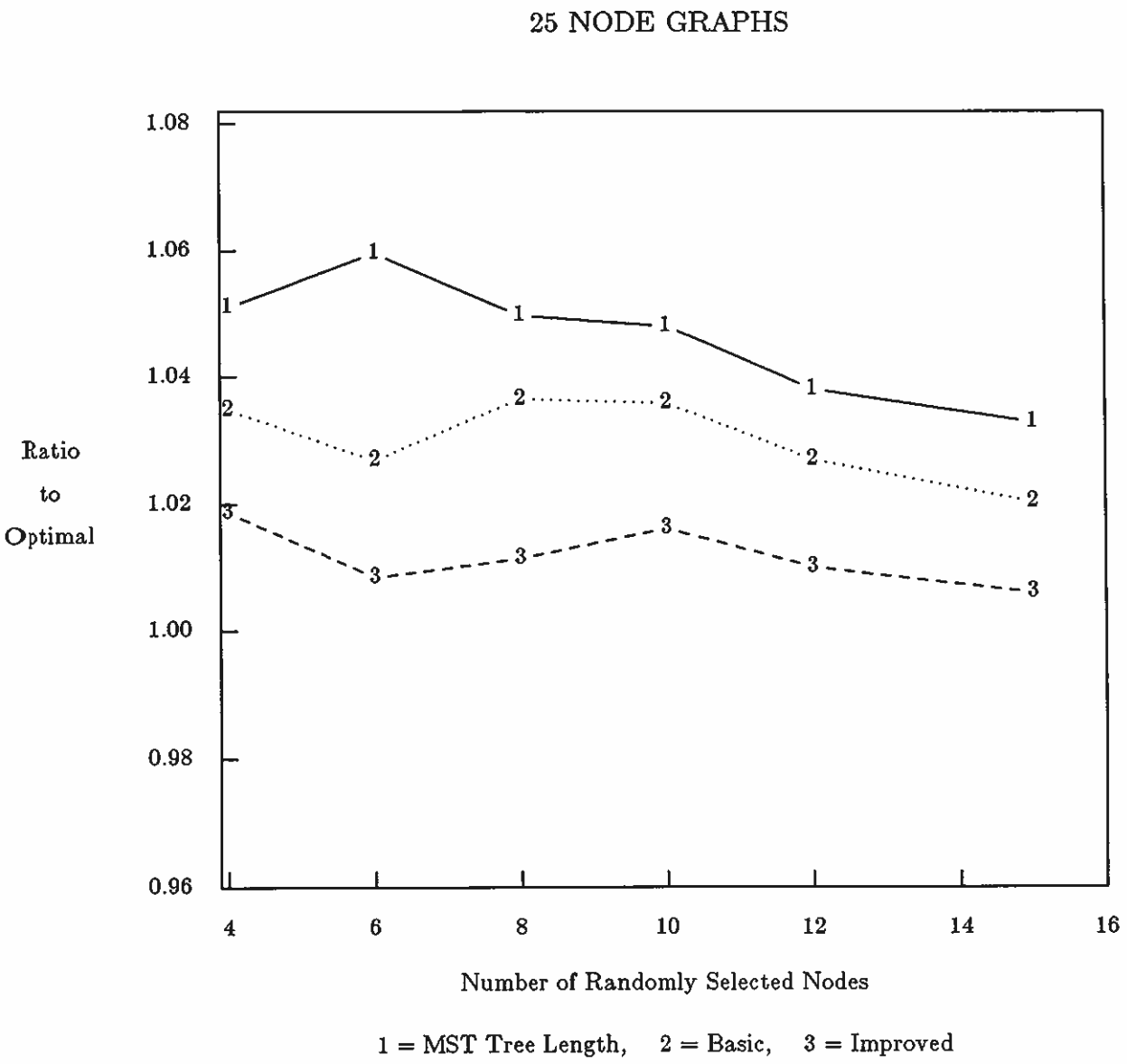

Figure 2: AVERAGE PERFORMANCE OF MST

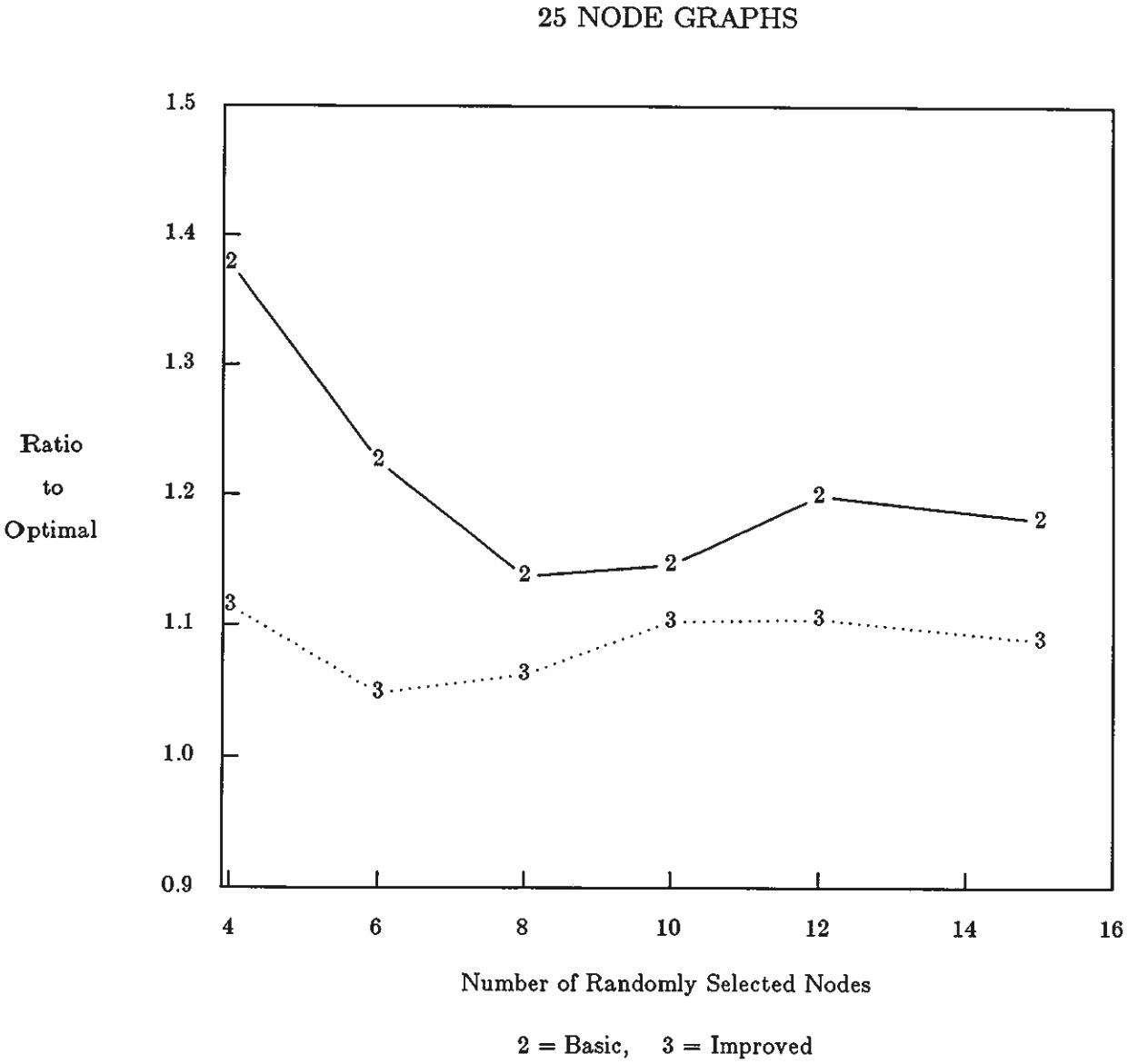

Figure 3: WORST CASE PERFORMANCE OF MST

functions $f_\ell : V \rightarrow \Re^+$ such that those nodes for which each function has a minimum value are the ones that should be included in a final solution. The subscript on f_ℓ indicates the ℓ-th iteration of the 'do' statement below. When it causes no confusion we will drop the subscript. The RS heuristic can be stated as follows.

Derive the function dist: $V \times V \rightarrow \Re$
  Use a standard algorithm
  e.g. Dijkstra's algorithm which has running time $O(nm\frac{\log n}{\log(m/n+2)}) = O(n^3)$.
Initialize $\mathcal{T} = \{(\{v\}, \emptyset) | v \in D\}$
  $\mathcal{T}$ is initially a set of single node trees.
$\forall v \in V$ create a list L_v of trees $T = (V_T, E_T) \in \mathcal{T}$ in ascending order by dist(v, T)
  Here dist$(v, T) = \min_{u \in V_T}\{\text{dist}(v, u)\}$
do $|\mathcal{T}| > 1 \rightarrow$
  $\forall v \in V \rightarrow f[v] = \text{val}(v)$
  Choose a vertex v that gives a minimum value for $f(v)$
  Join T_i and T_j with a shortest path through v to form T'
    where T_i and T_j are the first and second items in L_v
  Set $\mathcal{T} := (\mathcal{T} - \{T_i, T_j\}) \cup \{T'\}$
  $\forall v \in V$ calculate dist(v, T')
  $\forall v \in V$ update L_v
od
  The running time for the **do** statement is $O(k^2 n)$

The total running time for RS is $O(n^3)$.

Of course the crucial part of this heuristic is the calculation of f. We first define f_ℓ

$$f_\ell(v) = \min_{0<r<|\mathcal{T}|} \left\{ 1/r \sum_{i=0}^{r} \text{dist}(v, T_i) \right\} \tag{5}$$

where $T_i \in L_v$, after ℓ executions of the **do** statement. To calculate $f(v)$ we can evaluate the sums beginning with $r = 1$ and continuing until $\text{dist}(v, T_{r+1}) \geq 1/r \sum_{i=0}^{r} \text{dist}(v, T_i)$. Note that to reorder each L_v after the set T has been modified we need only remove two trees from the list and insert the new tree in the appropriate spot. More formally we describe the function val.

function val(node u)
  $f := g := \text{dist}(v, L_u[0]) + \text{dist}(v, L_u[1]); \quad r := 2$
  do $f > \text{dist}(u, L_u[r])$ and $r < |\mathcal{T}| \rightarrow$
    $g := g + \text{dist}(u, L_u[r])$
    $f := g/r$
    $r := r + 1$
  od
  return f

The total running time for val is $O(k)$

The application of RS to a eight node graph is illustrated in figure 4. In this example the **do** statement of the algorithm is iterated four times. The collection of trees $\mathcal{T}$ is shown after each iteration and the values for f, before each iteration, are shown in figure 5.

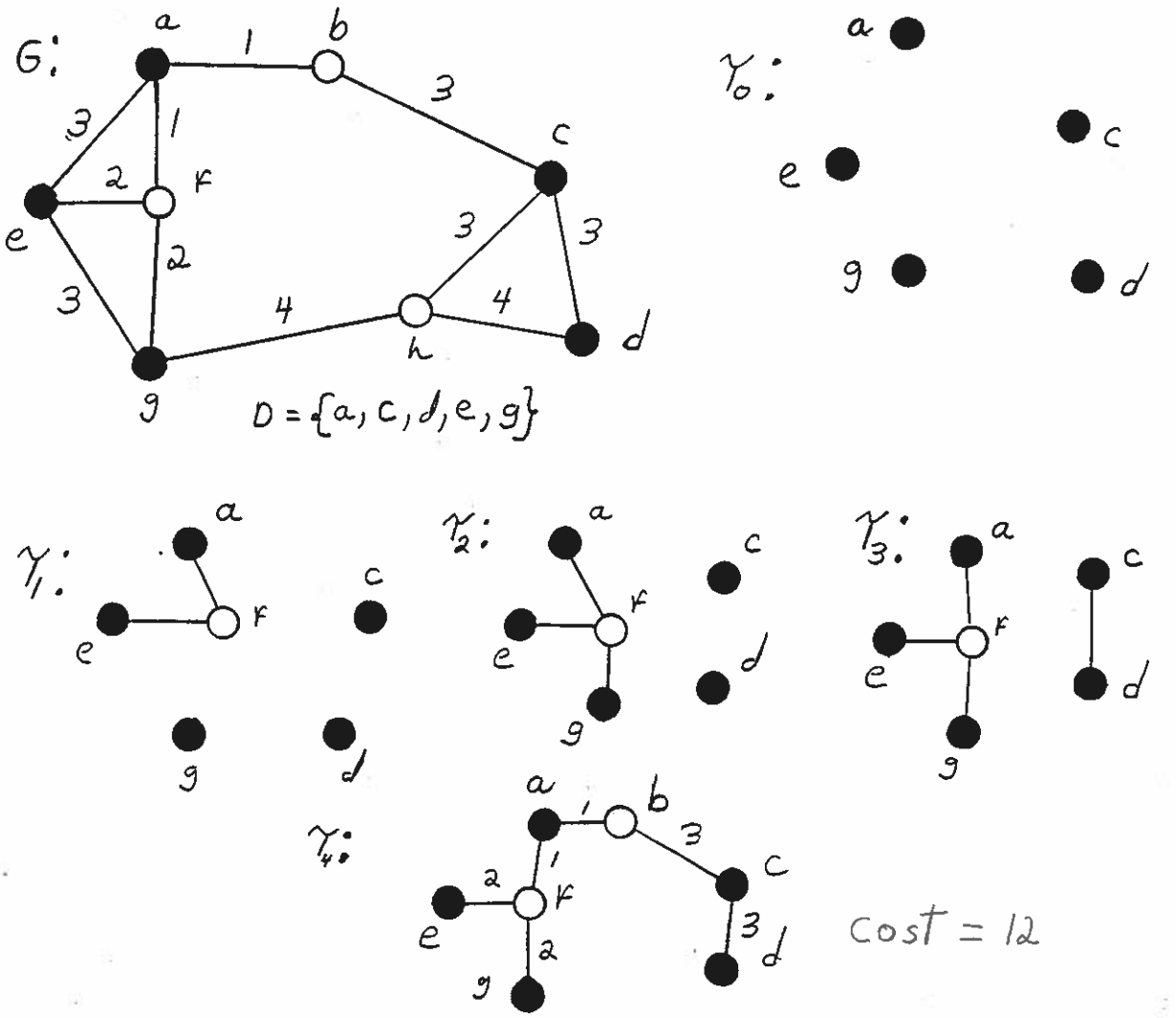

Figure 4: The application of RS to an eight node graph

	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

Figure 5: The value of $f_\ell(v)$ for $l \in [0..3]$

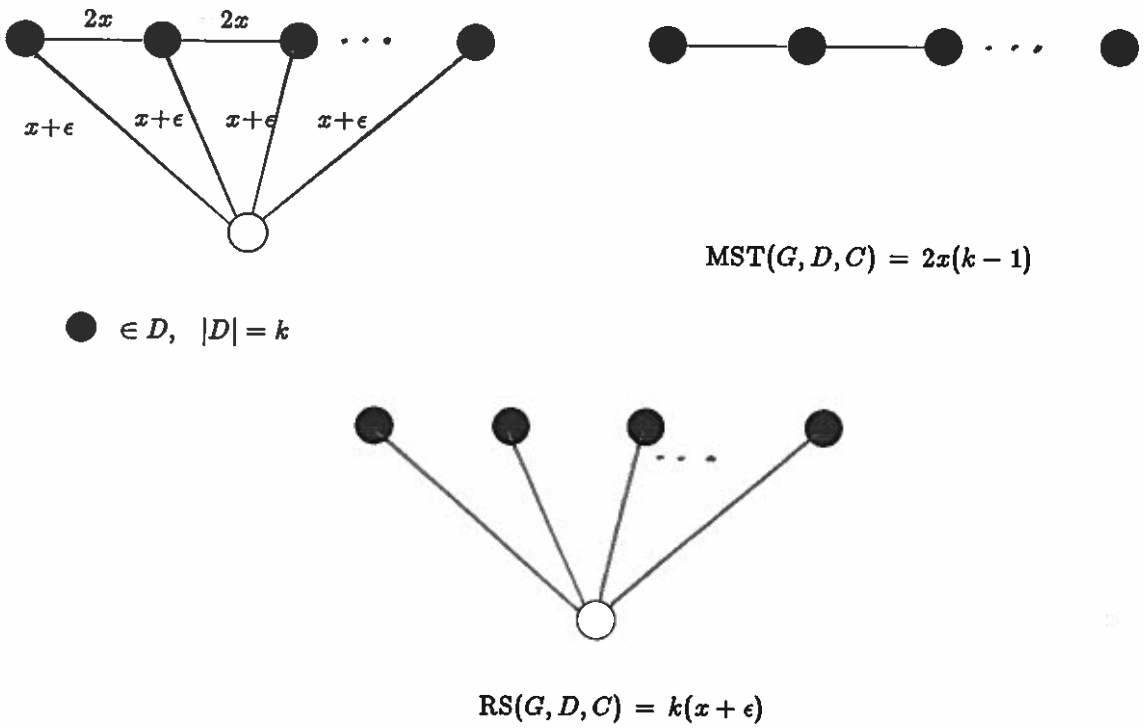

Figure 6: MST approaches twice optimal, RS yields optimal solution

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 figure 6 we have a class of examples where MST will yield a solution whose cost is almost twice that of an optimal solution while RS will actually find an optimal solution.

4. A Dynamic Algorithm for the Steiner Tree Problem

This past summer I ran another set of experiments dealing with a multipoint connection in the form of the *unitary dynamic* multipoint problem. Thus individual nodes were allowed to join or leave a connection at any point during the lifetime of that connection. These experiments were run using a greedy algorithm, in which a node was added to an existing connection using the shortest path among those paths to any node already in the connection. A node was deleted from the connection by pruning that branch of which it was a part, as long as that node was not an internal node in the connection. At each stage the performance of this algorithm was compared to that obtained by using our improved version of MST on the nodes in the connection. Figure 7 illustrates a sequence of five events handled by the greedy algorithm. Note that event four gives an example of a situation

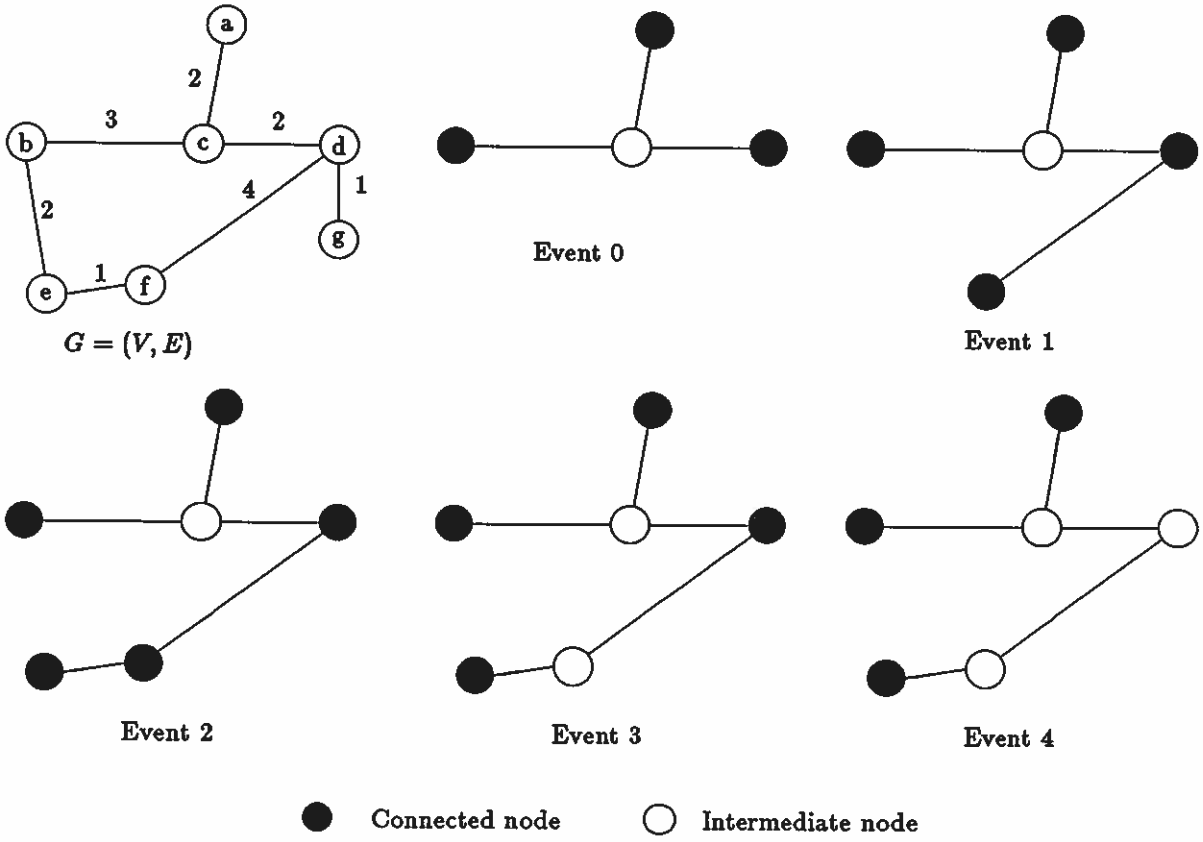

Figure 7: Dynamic Greedy Algorithm with sequence: a,b,d,f,e,$\bar{\text{f}}$,$\bar{\text{d}}$

where the connection resulting from this greedy algorithm is far from optimal. In this experiment we used the MST heuristic as a tool to measure the performance an algorithm that is a little closer to one that might be used in a real network. It should be noted that this greedy algorithm is still not a realistic one for a large network since it requires complete knowledge of the state of the network.

To run the experiments I used a simple probability model to determine if an event was to be an addition or deletion of a node from the connection. A function

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

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 at equilibrium or roughly αn is the average number of nodes, over time, in the connection. For example, when $k/n = \alpha$, $P_C(k) = 1/2$.

I ran several sets of experiments on 195 node graphs. Figures 8 and 9 indicate the results of these experiments. For each experiment a sequence of 1000 events was run using a fixed value of α. Figure 8 shows the results of the simulation for $\alpha = 6/195$. The horizontal axis indicates the

event number, with events 550 through 1000 being plotted. The vertical axis on the left indicates the ratio of the cost of the solution produced by the greedy algorithm to the cost produced by the improved version of MST and applies to the upper curve. The vertical axis on the right indicates the actual number of nodes in set D and is applied to the the lower curve. In figure 9 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 algorithm to the cost of the solutions produced by improved MST. The results of the worst case ratio, the average ratio, and the ratio of the $90th$ percentile for each experiment are plotted. We can see from figure 9 that the greedy algorithm 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 the cases where this result is observed the nodes which have been deleted from D are internal nodes and must remain since they are part of a branch that contains other nodes in set D. Such an occurrence can be see in figure 8 in the neighborhood of event 950.

This phenomenon of poor performance after several nodes leave the connection would be expected with any algorithm that does not allow for reconfiguration of the connection after each event. There are several techniques that might be used to smooth out the performance of the greedy algorithm. For example, we might weight the choice of a connection path for a node joining the connection by the distance to some fixed source node, or perhaps the distance to the center of the current connection.

5. Distributed Algorithms and Limited Information

A large scale communications network will necessitate the use of a distributed routing algorithm and will require that the quantity of data stored at each node be limited. There are a number of networks which use point to point routing algorithms where at least part of the computation is distributed. For example, the ARPANET uses an algorithm in which each node along a path from u to v takes responsibility for choosing the next link on the path [13,14]. Each node, including u, is responsible for determining just one link in the connection. This algorithm requires that each node $x \in V$, where V represents the set of nodes in the network, maintain a table which has entries for each node $y \in V - \{x\}$. For each node x the table indicates which outgoing link to use for a connection to each $y \in V - \{x\}$, plus the delay along that path. Even though each node in the ARPANET does not have complete knowledge of the network, these tables may still present a problem with frequent updates in a large network with thousands of nodes as opposed to the few hundred nodes of the ARPANET.

The problem of finding and maintaining shortest paths in a network is of interest in developing any routing algorithm. In the original version of the ARPANET routing algorithm the tables used to route along shortest paths were calculated in a distributed fashion. Each node measured the delays on each of its outgoing links. Then each node using this information along with the shortest route and delay tables stored by its neighbors would periodically update its own routing tables. With this scheme it was difficult to maintain consistent routing information at each node. Therefore, in a new version of the routing algorithm a procedure was instituted for the calculations of shortest path spanning trees rooted at each node. The new algorithm for determining these trees is based on Dijkstra's shortest path algorithm and is distributed only in the sense that each node calculates the tree for which it is the root. Thus, routing in the ARPANET has moved away from a distributed algorithm in the maintenance of its routing tables.

D. S. Parker, Jr. and B. Samadi [16] discuss several versions of a distributed algorithm (PS) for updating a minimum spanning tree for a graph. Parker and Samadi are interested in the application of a minimum spanning tree to the all-point routing problem. The PS algorithm assumes that:

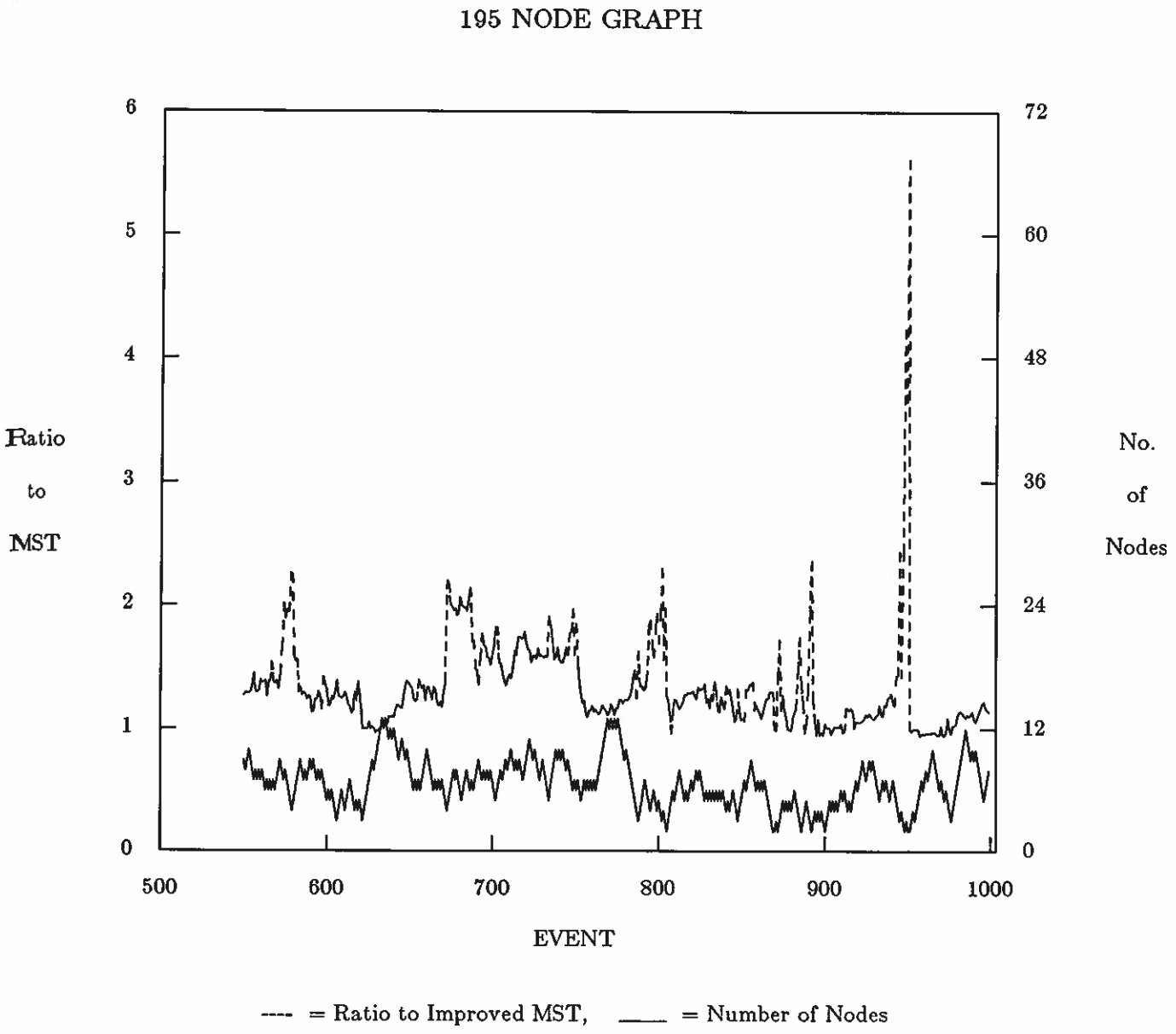

Figure 8: TIME SERIES FOR GREEDY ALGORITHM
(alpha = 6/195)

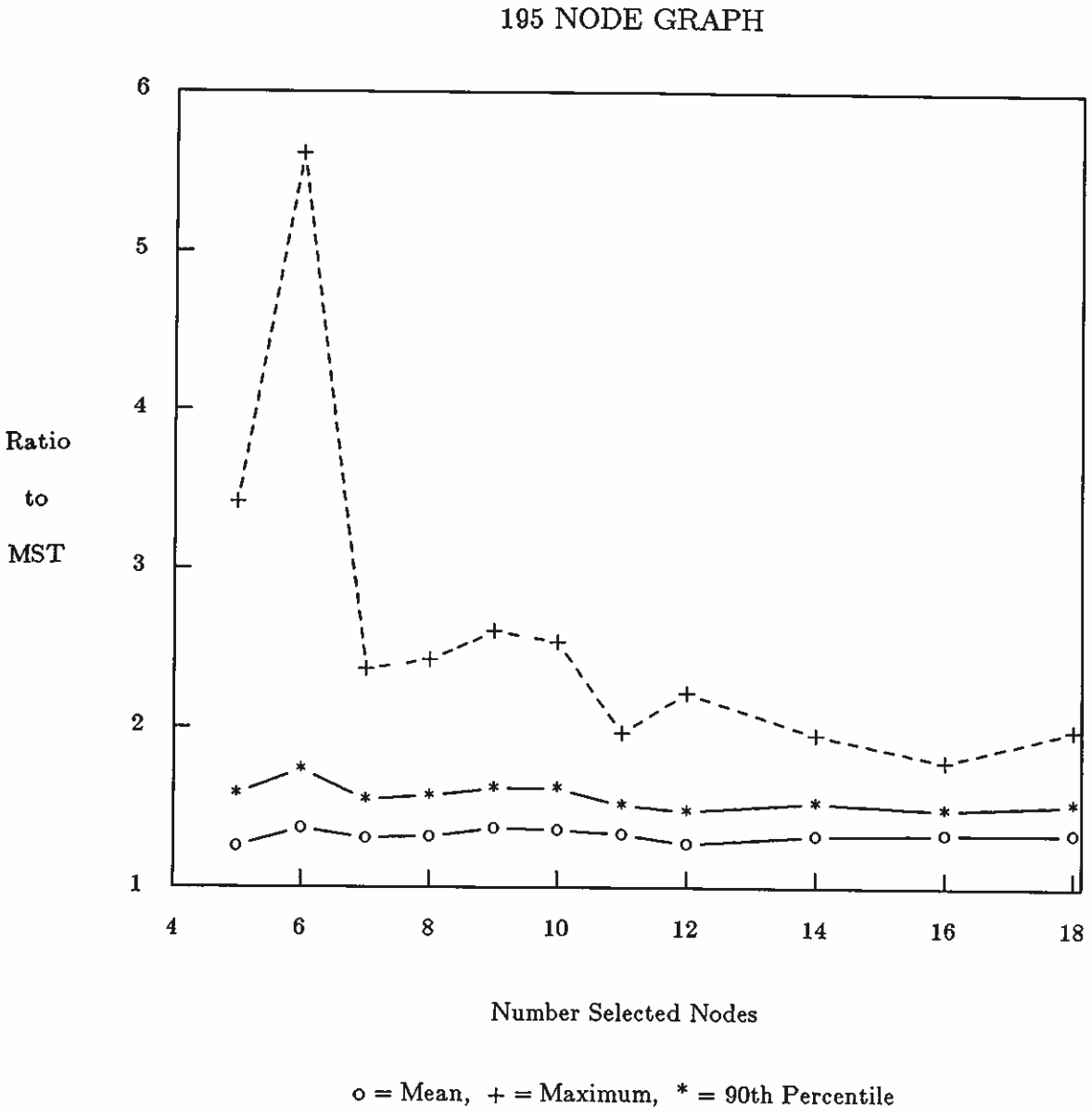

Figure 9: PERFORMANCE OF GREEDY ALGORITHM
(for 1000 event simulations)

- A network is represented by a graph $G = (V, E)$
- Each node knows the names of each of its neighbors.
- Every link has a unique cost.
- Three tables are maintained at each node u
  1. A table O_u of outgoing edges, indexed by the nodes $v \in V$, which contains the first edge on the path from u to v in the spanning tree
  2. A table E_u of the edge with maximum cost along the path from u to each node $v \in V$, indexed by v
  3. A table C_u, indexed by $v \in V$, which stores the cost of the edges stored in E_u

The algorithm uses an iterative technique which allows individual nodes to update the current spanning tree. The basic idea of this algorithm is:

- Each node u compares the cost of each incident edge $\{u, v\}$ to the value $C_u[v]$
- If $\text{cost}(\{u, v\}) < C_u[v]$ then $E_u[v]$ is replaced by $\{u, v\}$ in the spanning tree.

Parker and Samadi address such details as how to implement their algorithm in a concurrent fashion and how to handle changes in the network including changes in the topology of the network, but do not have complete solutions. This particular algorithm is not directly applicable to very large networks because of the need for large tables at each node. But PS does contain some interesting ideas and might be used as a starting point for an algorithm to construct a suboptimal minimum spanning tree, if we relax the requirement for complete tables at each node.

Even though distributed routing algorithms are used in some networks, such as the ARPANET, most of the algorithms do not deal with the multipoint problem. In the paper "Distributed multi-destination routing: the constraints of local information" by J. M. Jaffe [11], some of the problems of storing limited information with respect to the multipoint routing problem are considered. For example, Jaffe considers a class of algorithms for solving the Steiner tree problem which are restricted to the use of *local information*. Given a graph $G = (V, E)$, each node u has tables for the first link on the shortest paths to all other nodes in a network. For a given set of nodes $D \subset V$ one node s is chosen to be a distinguished node, called the source. The remaining set $D - \{s\}$ is called the set of destination nodes. A *local information* algorithm is then defined to be an algorithm for solving the multipoint problem in which the data stored at the source node s is all that can be used by the algorithm to make its initial routing decisions. As the algorithm builds a solution by choosing links to create the paths, each intermediate node makes choices based on only the data stored at that node. Jaffe proves that any *local information* algorithm will have worst case performance approaching $2|D|/3$ times that of an optimal solution.

Even if the source node is allowed to use the information stored at its neighbors, called a *nearby information* algorithm by Jaffe, the worst case performance does not change. It should be noted that this result is made under the assumption that only the source node can use *nearby information*. Jaffe's results do not tell us anything about those algorithms in which intermediate nodes along the route are allowed to use *nearby information*.

In this paper Jaffe looks at another set of algorithms, which he refers to as *destination information* algorithms. Such algorithms are ones in which a source node $s \in D$ knows the distance between all pairs of nodes in D. A *destination information* algorithm uses this information along with a set of paths corresponding to the shortest distances to make its routing decisions. MST is an example of

a *destination information* algorithm. Just as the worst case performance of MST can be almost as bad as twice the optimal solution, Jaffe proves that any *destination information* algorithm will also have worst case performance which is no better than twice optimal.

The results presented by Jaffe indicate, that with limited data stored at each node, we are not likely to find an algorithm which yields optimal routing. In fact Jaffe's results lead us to believe that any distributed algorithm, based on the quantity of data which can be reasonably stored in a very large network, is not likely to have worst case performance even within a constant factor of optimal, though Jaffe's results do not rule out this possibility. We should note the Jaffe's results do not give us any information about the average case performance of multipoint algorithms restricted to limited data. Since the average case performance of such algorithms is of great interest the analysis of average case performance with limited information is an area that warrants further study.

Even the assumption that each node in the network maintains tables of shortest path information for all other nodes, as required for the *local information* algorithms of Jaffe, may not be feasible for a very large network. Frederickson and Janardan [5] as well as Santoro and Khatib [19] consider techniques for routing in networks where the topology allows for compact storage of routing data. For example, Santoro and Khatib [19] present a method for shortest path routing in trees which only requires that each node be labelled in a specific way so as to eliminate the need for tables at each node. Frederickson and Janardan present methods for optimal shortest path routings with a technique they call *interval labelling*. They show, for example, that an optimum routing can be obtained for outerplanar graphs using a single interval label for each edge. Here an interval refers to a consecutive sequence of integers, where each integer stands for a unique node in the network.

Frederickson and Janardan [6] present another technique, which they call *separator based strategies*, for suboptimal routing in somewhat more general networks. The idea behind this technique is to subdivide a network into a hierarchy of subgraphs. Within each subgraph each node maintains information on the shortest paths to all other nodes in that subgraph. For routing to nodes outside of a given subgraph a route is chosen, based on the destination node, to a node in another subgraph in the hierarchy and control for routing is then passed to that node. Frederickson and Janardan claim that for a planar graph their technique yields paths that are at most 7 times that of an optimal path. They also claim that only $O(n^{\epsilon})$ items need be stored at each node for an n node network with $0 < \epsilon < 1/3$, as opposed to $O(n)$ items for complete shortest path routing tables. It is likely that for very large networks some technique, such as that proposed by Frederickson and Janardan, will be required to reduce the quantity of data stored at each node.

6. Research Goals

The primary motivation behind this work is to develop a workable routing mechanism for a large scale general purpose packet switched network. The general nature and features of the proposed network require that a routing algorithm for such a network meet a number of constraints. Among other things most of the data transmission will be made using the virtual circuit model, though some small percentage of the communications may use datagrams. The network will not be constrained to any particular topology. Thus, the algorithm must be general purpose and will not be able to assume any special topology for its operation. The algorithm will be required to handle variable bandwidth connections as well as the three categories of connection types. The algorithm will have to be adaptive so that it can handle varying loads in different parts of the network as well as being able to handle changes in network topology. The changes in network topology include both temporary changes due to equipment failure and permanent changes due to increases in the size of the network. Due to the large size of the network each node will be able to store only limited routing information. Hence, a distributed routing algorithm will be required.

At this point it is assumed that most of my effort will be in work related to the multipoint problem. As far as point to point communication and the all-point problem are concerned these may be treated as special cases of the multipoint problem or techniques used in existing algorithms may be incorporated into a general algorithm. Thus, nearly all of my effort up to this point has been directed toward the multipoint problem.

Work on the multipoint problem will fall into two categories. One segment of my research will deal with the theoretical aspects of multipoint routing which includes work with the Steiner tree problem, looking at random graph models for communication networks, and the study of probable performance of algorithms with respect to these specific models. This work will include looking at the existing heuristics for the Steiner problem in graphs, the investigation of their worst case performance, and the study of their probable performance. The other area will be the development of a practical algorithm that can handle multipoint connections in a large network.

With regard to the Steiner tree problem, I plan to develop mathematical tools to analyze the probable performance of algorithms on specific network models, to develop improved algorithms with respect to both probable performance and worst case performance, and finally to consider algorithms that lend themselves to distributed implementation and can operate in a situation where only limited knowledge of the network topology is available to each routing processor. All of these results should have bearing on the problem of developing a usable algorithm for the proposed network.

Finally, I expect to use any results that I obtain from the theoretical problems and incorporate them into the design and testing of a practical multipoint routing scheme for an actual packet switched network. For example, a good heuristic for the Steiner tree problem can be used as a reference tool against which we can measure the results of any practical algorithm even though the actual heuristic may not be incorporated into the algorithm. Any tools that I develop for analyzing the probable performance of an algorithm could potentially be used to analyze any proposed network algorithm. Hopefully the consideration of heuristics for the simpler Steiner tree problem will give us insight that will aid in the derivation of practical algorithms for an actual network.

6.1. Network models

In order to study the behavior of any routing algorithm we need a network model to which we can apply the algorithm. One appropriate technique is to use a random graph to represent a network model. A particular random graph should satisfy two conditions. First it should have as many of the properties of a real network as possible and second it should be simple with respect to our ability to analyze its properties and to analyze the probable performance of a routing algorithm running on a network corresponding it.

In order to test the MST heuristic I have used the random graph model RG1, described in section 2. The random graph model RG1 creates links between distant nodes with a lower probability than it creates links between nodes which are close together. In addition the cost of a link increases as the distance between the endpoints of that link increases. Both of these properties correspond to what we would expect in a real network, at least qualitatively. Clearly RG1 does not have all of the properties of a real network. For example, we would not expect the nodes of a network to be distributed uniformly over a plane but would, for example, expect nodes to cluster in areas of high population density. This brings us to the second point that our model needs to be simple. Hence, any particular model must be a compromise between simplicity and retaining as many properties of a real network as possible. In fact for the purpose of analyzing the probable performance of a routing algorithm we may need to look at some models which are very simple but do not correspond very well to a real network.

Graph model RG1 may not be as easy to analyze as we might like. Therefore, at least for the initial work in analyzing the probable performance of different algorithms I plan to use simpler

models. Some of the models include: complete graphs with unit cost links, complete graphs with uniform random assignment of link costs over a given range, and a random graph in which all edges are equally likely with unit costs. Finally, analysis of a simple model may be valuable in terms of comparing the relative performance of different algorithms even if it is not a precise predictor of the performance in a real network.

In addition to the probable performance of different algorithms on specific network models, it is of interest to consider some of the properties of the random graph models that are used. For example, Gilbert's random graphs are almost always connected when the edge probability has a fixed value $p > 0$ [9]. Where "almost always" means that the probability that a given property holds, for a particular random graph model, approaches one as the number of nodes goes to infinity. I plan to investigate the properties of RG1 and any other random graph models that I use.

6.2. Probable performance of Steiner tree heuristics

There are a number of heuristics that I plan to study in terms of their probable performance on a given network model. Two heuristics which are of particular interest are the MST heuristic and Rayward-Smith's heuristic (RS), both of which are discussed later in this proposal. I have already made some initial studies of the experimental performance of the MST heuristic. What I plan to do next is to analyze the probable performance of MST and RS on the models described in the previous section. For example, I plan to consider the problem of determining

$$P\left(\frac{MST(G,D,C)}{OPT(G,D,C)} > A\right)$$

for a fixed value of A in $[1,2)$, where P stand for probability, $MST(G,D,C)$ represents the cost of a solution given by MST and $OPT(G,D,C)$ represents the cost of an optimum solution. Note that the ratio of MST to OPT is always in $[1,2)$. The same problem will then be considered substituting RS for MST.

After these initial studies I will hopefully be able to analyze the performance of these and other algorithms on more complicated graph models which more closely resemble a real network.

We can also obtain some idea of the probable performance of an algorithm by running simulations of the operation of that algorithm on a specific network model. To this end, as mentioned above, I have run experiments with the MST heuristic and may run experiments with others. Such experiments can certainly give us a general indication of the average performance of a heuristic and the relative quality of different heuristics.

6.3. Worst case performance of Steiner tree heuristics

Of the heuristics already mentioned we know that the worst case performance of MST and the contraction heuristic are within twice that of optimal and that the worst case performance of each is worse than any value less than two. I have proven that the worst case performance of Rayward-Smith's heuristic is also within twice that of optimal, and have shown that its performance can be nearly as bad as 3/2 the cost of the optimal solution. I suspect, but have not yet been able to prove, that the performance of RS is within 3/2 of optimal. Thus, I would like to find the smallest value for the worst case performance of RS and prove that the value is correct.

In addition to the investigation of existing heuristics I would like to find heuristics which have better worst case performance. I would like to also determine if there is any limit to the best worst case performance that can be achieved by a polynomial time heuristic for the Steiner tree problem if $P \neq NP$.

6.4. A network algorithm

I also plan to develop algorithms that can be used in a real network. I have looked at the experimental performance of a greedy algorithm. The experiments with this greedy algorithm represent an initial step toward the development of a practical algorithm. The next step with regard to a practical routing algorithm will be the development of an algorithm that can be used in our prototype network. Since the prototype will be small we may use an algorithm in which each node has information about the topology of the entire network. In this case something similar to the greedy algorithm may actually be usable in the prototype. For the large scale network a more complicated distributed algorithm will be needed.

7. Summary

In the research undertaken for my thesis I will be investigating the problem of routing in a packet switched network with specific emphasis on the problem of routing multipoint connections. The goals of this project include:

- Development of a practical routing algorithm for a large scale packet switched network.
- A proof or disproof of the claim that Rayward-Smith's heuristic has worst case performance which is within 3/2 of optimum.
- Analysis of the probabilistic performance of Steiner tree heuristics including the MST heuristic and the RS heuristic; and comparison of this analysis with experimental results.
- Development of new heuristics for the Steiner problem whose worst case and/or average performance is better than that of existing heuristics.
- Determination of the probable characteristics of the random graph model RG1, which has been used to represent a communication network, and others which may be developed.
- Consider ways in which any of the heuristics or theoretical tools, which are developed, can be used to estimate the performance of a routing algorithm.

References

[1] K. Bharath-Kumar and Jeffrey M. Jaffe. "Routing to multiple destinations in computer networks." *IEEE Transactions on Communications*, COM-31(3):343–351, March 1983.

[2] M. Casey. "Packet switched data networks: an international review." *Inf. Technol. Res. & Dev. (GB)*, 1(3):217–44, July 1982.

[3] H.S.M. Coxeter. *Introduction to Geometry.* John Wiley & Sons, Inc., 1961.

[4] S.E. Dreyfus and R.A. Wagner. "The Steiner problem in graphs." *Networks*, 1(3):195–207, 1971.

[5] G.N. Frederickson and R. Janardan. "Optimal message routing without complete routing tables." *5th ACM SIGACT-SIGOPS, Symposium on Principles of Distributed computing*, August 1986.

[6] G.N. Frederickson and R. Janardan. "Separator based strategies for efficient message routing." *27th Annual Symposium on Foundations of Computer Science*, 428–37, October 1986.

[7] M.R. Garey, R.L. Graham, and D.S. Johnson. "The complexity of computing Steiner minimal trees." *SIAM J. Appl. Math.*, 32(4):835–59, June 1977.

[8] M.R. Garey and D.S. Johnson. "The rectilinear Steiner tree problem is NP-complete." *SIAM J. Appl. Math.*, 32(4):326–34, June 1977.

[9] E.N. Gilbert. "Random graphs." *Annals of Mathematical Statistics*, 30:1141–44, 1959.

[10] E.N. Gilbert and H.O. Pollak. "Steiner minimal trees." *SIAM J. Appl. Math.*, 16(1):1–29, 1968.

[11] J.M. Jaffe. "Distributed multi-destination routing: the constraints of local information." *SIAM J. Comput*, 14(4):875–88, November 1985.

[12] L. Kou, G. Markowsky, and L. Berman. "A fast algorithm for Steiner trees." *Acta Informatica*, 15:141–5, 81.

[13] J.M. McQuillan, G. Falk, and I. Richer. "A review of the development and performance of the ARPANET routing algorithm." *IEEE Trans. Commun.*, COM-26(12):1802–10, December 1978.

[14] J.M. McQuillan, I. Richer, and E.C. Rosen. "The new routing algorithm for the ARPANET." *IEEE Transactions on Communications*, COM-28(5):711–19, May 1980.

[15] R.E. Miller and J.W. Thatcher. *Complexity of Computer Computations*, chapter Reducibility among combinatorial problems - by R.M. Karp, pages 85–103. Plenum Press, 1972.

[16] D.S. Parker, Jr and B. Samadi. "Adaptive distributed minimal spanning tree algorithms." *Proceedings of a Symposium on Reliability in Distributed Software and Database Systems*, 138–44, July 1981.

[17] J. Plesnik. "A bound for the Steiner tree problem in graphs." *Math. Slovaca*, 31(2):155–63, 1981.

[18] V.J. Rayward-Smith. "The computation of nearly minimal Steiner trees in graphs." *International Journal of Math. Ed. Sci. Tech.*, 14(1):15–23, 1983.

[19] N. Santoro and Ramez Khatib. "Labelling and implicit routing in networks." *The Computer Journal*, 28(1):5–8, 1985.

[20] R.E. Tarjan. *Data Structures and Network Algorithms.* Society of Industrial and Applied Mathematics, 1983.