Use of Finite Random Graphs to Model Packet Radio Networks

A Thesis Presented to

The Faculty of the College of Engineering and Technology

Ohio University

In Partial Fulfillment

Of the requirements for the Degree

Master of Science

By

Yang Wang

March, 1990

Acknowledgement

I would like to express my sincere thanks to my advisor, Dr. Jeffrey C. Dill for all his guidance and help during this research. I also would like to thanks my thesis committee members, Dr. Joseph E. Essman, Dr. Mehmet Celenk and Dr. John D. Gillam for their advice.

Finally I would like to extend special thanks to my wife, Yan Tang, for her encouragement and support during my study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Packet radio networks (PRnet) provide data communications by packet switching technology to mobile users where direct radio or wire connection between the source and destination users is not available. Like any packet communication system, to analyze a packet radio network, we first need to determine a model which can represent the network topology. For the N-node radio network, there are $2^{N(N-1)/2}$ range of possible topologies. Moreover, when PRnet's are mobile, the topologies change dynamically so that it is impossible to analyze the network performance over all possible topologies. Clearly, no single model can be formulated which incorporates all the necessary parameters and leads to an optimum solution. Therefore we must develop some representations of network topologies which allow us to describe mathematically a wide range of realistic network topologies and use those to analyze the network operations.

Kleinrock and Silvester [8] identify the following three prevalent assumptions used in modeling the topological structure of PRnets: (1) regular structure, in which nodes are considered to be located in some regular

pattern on the plane; (2) a continuum of nodes, in which nodes are considered to be continuously present throughout the plane and network traffic is generated at some rate per unit area; and (3) random locations, which nodes are considered to be randomly distributed in the space of interest. In this thesis we will focus on the random location topology and attempt to develop a useful approximation of the topological structure of realistic packet radio networks.

There are two possible random location models which can be used [3]. The Euclidean model assumes that the transmission radius of nodes is fixed in conjunction with an assumed Poisson distribution of locations, and therefore the link existence is strictly a function of internode distance. Thus, each node has a fixed transmission radius and the resulting connectivity matrix describes a Euclidean random graph as shown in figure 1-1. In the Euclidean model, an implicit assumption is made that the transmission range of a node is dominated by the free space propagation of the radio signal, and the transmission range is equal in all directions as represented in the "range circles" of figure 1-1.

Another possible model is the Bernoulli random graph model proposed by Dill [3]. This model assumes the existence of each link is the result of a Bernoulli random

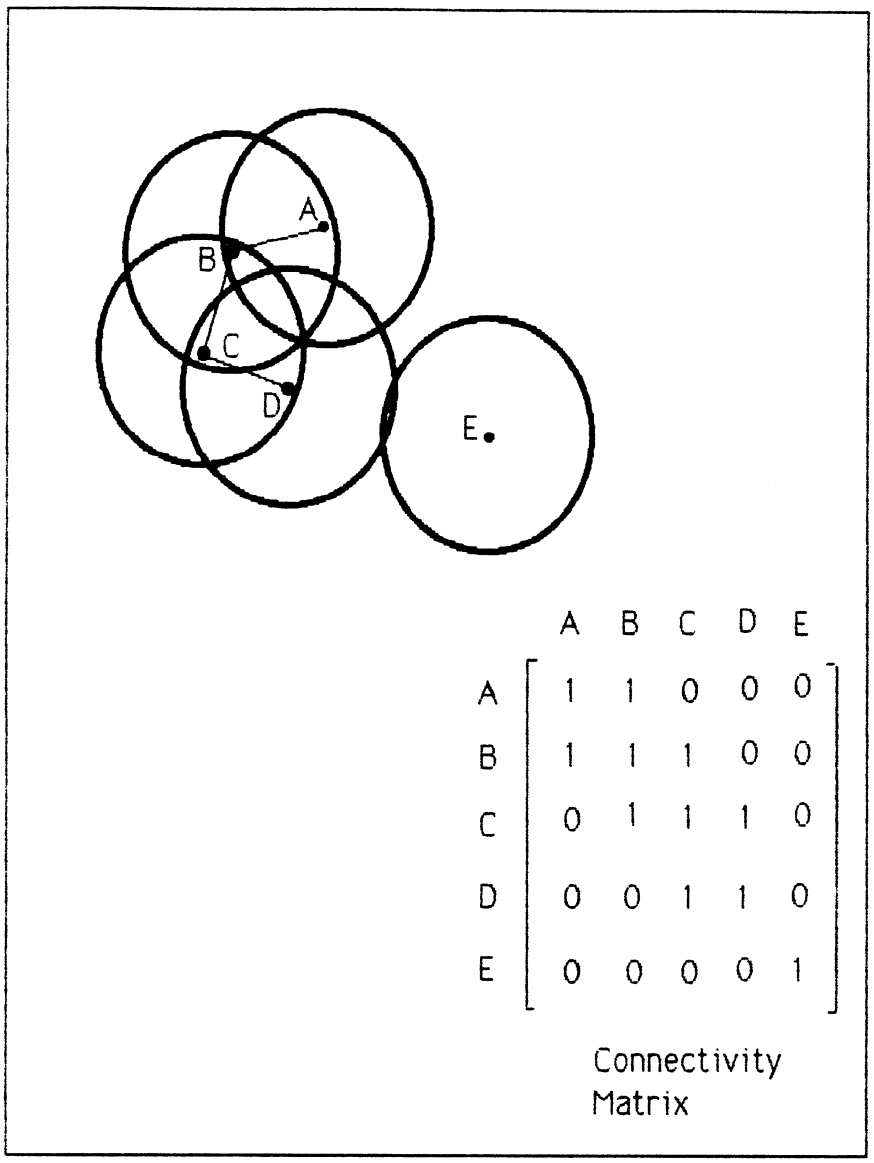

Figure 1-1. Euclidean Random Graph of 5 Nodes. (from [3])

trial with probability P, which is independent of all other links in the network. Thus, nodal positions are ignored, and the adjacency matrix is determined by a series of independent Bernoulli trials.

Th Bernoulli and Euclidean models represent the extremes of possible topologies for real radio networks [3]. For example, if the terrain is flat with minimal foliage or man-made structures, or if the radio nodes are carefully sited with their antennas placed on hilltops, then the Euclidean model is an excellent approximation. If, however, the terrain is rough and heavily foliated, or if the network is operating in an urban or suburban environment, then the Bernoulli model is preferable. The topology of real networks is best described by a combination of these two models.

In this thesis, we attempt to find some useful approximations to describe the topologies of finite random graphs for both the Euclidean and the Bernoulli models. We are particularly interested in the path length distributions, assuming a minimum hop routing algorithm is available.(Many such algorithms exist, e.g. Dijkstra's algorithm, the Bellman-Ford algorithm, etc. [12]).

The structure of *infinite* random Euclidean graphs has been studied extensively ([4], [6]) and results have been

found for the probability of connectedness for very large graph components. However, little theoretical work has been done on finite Euclidean graphs. The difficulty in analyzing the structure of finite Euclidean graphs results from the boundary effects at the edges of the finite plane. Rather than attempting to solve the more general analytical problem, in this thesis we develop an approximation technique which is based on a combination of analytical and simulation results, and enables us to predict the average minimum hop path length of a finite Euclidean random graph quite accurately.

For finite Bernoulli random graphs, on the other hand, there are no complicating edge effects, and an exact solution for the path length distribution has been found by Dill [3]. The algorithm, however, has a computation time which is proportional to N^3, and is computationally infeasible for large networks. In this thesis we develop an approximation to the exact solution which is quite accurate for large networks, and is computationally proportional to N.

Infinite Euclidean Graphs:

To construct a sample instance of an infinite Euclidean network, we first pick nodes from the infinite plane by a Poisson process with density λ points per unit area. Next

we connect each pair of nodes by a line if the pair is separated by distance less than R, i.e., if the pair can hear each other. Figure 1-2 shows two networks obtained from the same random node pattern but with different values of R. The parameter D in figure 1-2 represents the average number of neighbors posessed by each node (also called the average degree); $D=\pi R^2\lambda$,. Not all lines of the networkare drawn, just enough to show the connected components (called subnets). Naturally, subnets for larger D (D_L) are bigger than those for smaller D (D_S) but there is a qualitative difference as well. The small subnets of the D_S network are so merged together in the D_L network that most of the nodes belong to a single large subnet. This phenomenon suggests that there is an expected D in which the subnet will merge to a single connected network with infinitely many nodes in the infinite plane [6]. So the limit

$$\lim_{n \to \infty} P(n) = P(\infty) \qquad (1.1)$$

need not be zero. It represents the probability of belonging to an infinite connected network. In [6] it is shown that $P(\infty) = 0$ when $D < 1.75$ and $P(\infty) > 0$ when D is sufficiently large. Thus there exists a critical value D_c which is the largest value of D for which $P(\infty) = 0$.

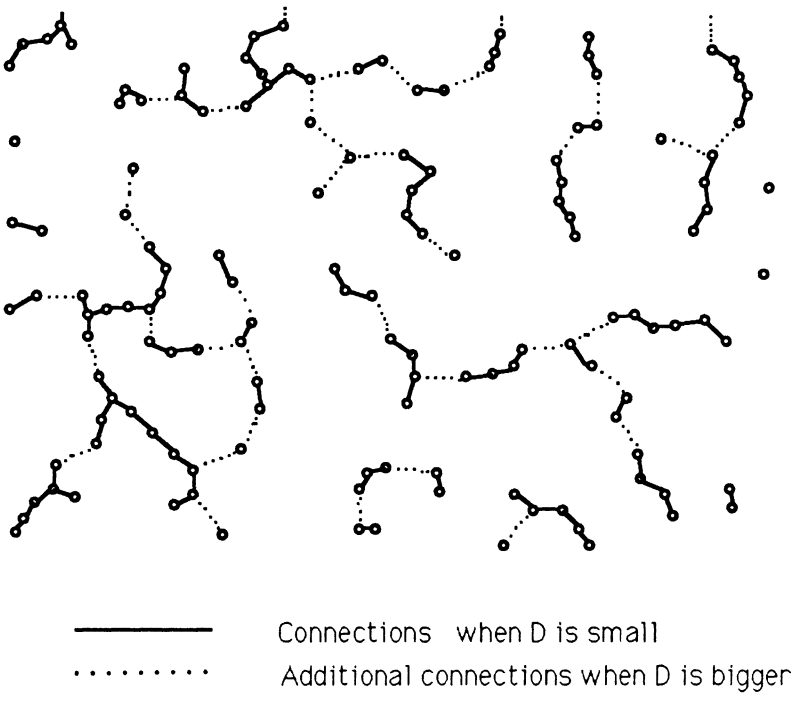

Figure 1-2. Different Average Degree Random Plane Network. (from [6])

Erdos and Renyi [4] have considered the issue of connectivity for large random graphs and found that if the average degree is log(N)+c then the probability of the graph being connected is $e^{-e^{-c}}$. Dewitt [2] has found a lower bound on the probability of connectedness if the average degree is 4log(N)+4loglog(N)+4c then Pr{connected} $\geq e^{-e^{-c}}$.

However, the graphs that we are primarily concerned with in this research are finite Euclidean graphs where the existence of edges is significant, and is not an independent process. Thus these results are asymptotically true for large graphs, but may differ significantly for finite graphs.

In the finite Euclidean plane with a finite number of nodes, the analysis of connectivity is much more complex because of the edge effects. First, the nodes near the edges of the plane tend to have fewer neighbors because their range circle extends beyond the edge of the finite plane, where no potential neighbors exist as shown on Fig 1-3. Second, the edge effects will increase as when the number of nodes increases. The questions are: does a criteria still exist for a minimum value of average degree D_C despite the edge effects, and is there some relationship between the total number of nodes and average path length? Practically, when we design a network, we would like to know the minimum average degree that the network should have, i.e.

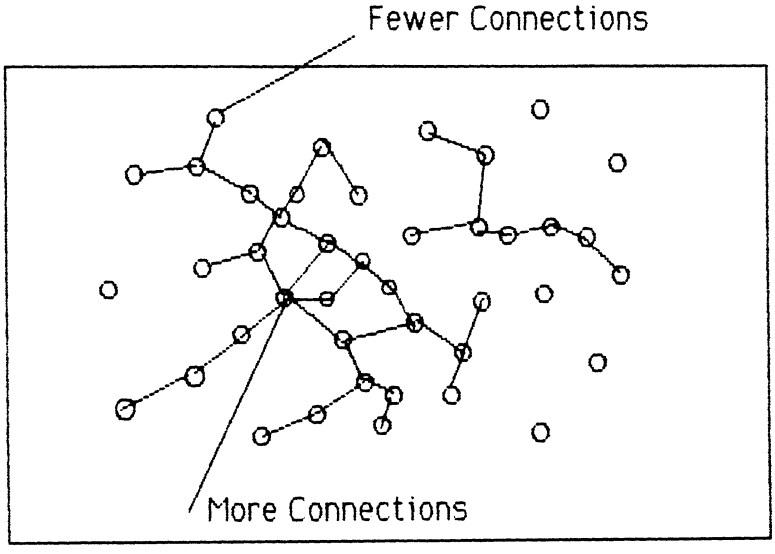

Figure 1-3. Edge Effect

the desired transmission radius, and thus the average path length of the network.

In this thesis we focus on finite plane networks. We introduce Gilbert's hex method to analyze the finite Euclidean model and simplify Dill's result for the Bernoulli model. Because analyzing the finite Euclidean model is analytically difficult, we will use experimental simulation data to obtain statistical results.

The following is the outline of this thesis. In chapter 2 we analyze the Bernoulli model and the Euclidean model. In chapter 3 we use a computer simulation to find statistical properties of the finite Euclidean model. Finally the conclusions of this research are presented in chapter 4.

Chapter 2

Analysis of Bernoulli and Euclidean Models

2.1 Overview of Random Graph Models:

A simple undirected graph G=(V,E), without multiple edges and self-loops, is defined by a finite node (vertices) set V and a finite unordered node pair set E. An element of E is called an "edge" and is represented by (u,v) (u, v $\in$ V, u$\neq$ v). From the definition, it holds that (u, v) = (v, u). For an edge e = (u, v) $\neq$ E, u and v are end points of the edge e. The degree of a node v($\neq$ V) is the number of edges adjacent to v, and so the average degree of the graph or network G is defined as the average number of neighbors which each node possesses.

2.1.1 Euclidean Random Graphs:

In a Euclidean random graph model of a radio network, each node represents a station which is capable of transmitting and receiving messages and each edge (line) represents a bidirectional channel. Each node is assumed to use a predetermined fixed radius for transmission so that the network structure (topology) is completely specified. In this thesis, we assume all stations have the same radius so that the performance of the network will then be

studied as the transmission radius is varied as shown in figure 2-1. Any nodes falling within a circle of radius R about a node will be able to hear that node and also will be able to transmit to it. Clearly by increasing the transmission radius, we can increase the average degree, each node being able to communicate with more nodes. In the limit, if the transmission radius spans the entire network, then the graph becomes fully connected.

As the nodes are randomly distributed in a two dimensional plane, the average number of nodes that will be in a circle of radius R is assumed, most commonly, to be described by a two dimensional Poisson distribution, and the probability of finding k nodes in a region of area A is

$$P = \frac{(\lambda A)^i \cdot e^{-\lambda A}}{i\ !} \qquad (2.1.1)$$

Since $A = \pi R^2$ is the area covered by the transmission, we can define D to be the average degree of the network (It is also an indication of network connectivity).

$$D = \lambda A = \lambda \pi R^2 \qquad (2.1.2)$$

In the unit square plane, we can assume λ equal to N nodes to estimate the average degree, i.e.

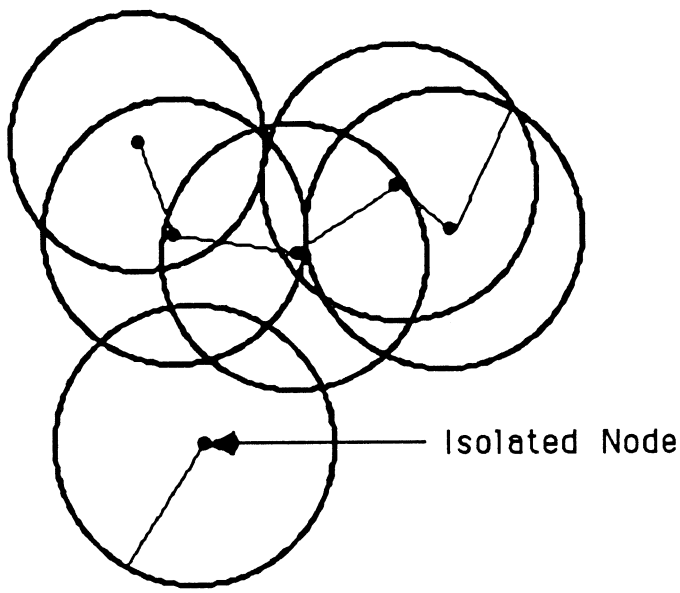

Figure 2-1 Simple Euclidean Network.

$$D = N \cdot A = N\pi R^2 \qquad (2.1.3)$$

The topology of the graph can be represented by a binary connectivity matrix of the form:

$$A=\{ a_{ij} \} \; ; \; a_{ij}=\begin{cases} 1 \text{ if } j \text{ and } i \text{ adjacent in } G; \\ 0 \text{ otherwise.} \end{cases} \qquad (2.1.4)$$

From this adjacency matrix, we can generate the set of minimum hop paths for the network by using any standard routing algorithm.

We next introduce the definition of the connectivity fraction and average path length. The connectivity fraction of the random network, X, is defined as the fraction of potential links which actually exist. The parameters, N, D, X are related by the equation

$$D = X\,(N-1) \qquad (2.1.5)$$

Thus, a connectivity fraction of X=1 corresponds to a fully connected network, and for the random network, the X is characterized statistically by the two parameters N and D.

The average path length is the expected number of hops required to reach an arbitrary destination using a minimum hop routing algorithm.

2.2 Bernoulli Model Analysis:

As stated in Chapter 1, nodal positions in the Bernoulli model are ignored, and the existence (or nonexistence) of any link is therefore not a function of internode distance. Thus, with a Bernoulli random trial probability P, the adjacency matrix is determined by a series of independent Bernoulli trials.

In order to solve the distribution of (minimum hop) path lengths in a Bernoulli random network, we begin by choosing a random node, v, and finding the distribution of path lengths from v to all other nodes. Then, since the links in the Bernoulli random graph are independent, it follows that this is the path length distribution for all paths in the entire graph.

In [3] an exact recurrence relation solution to the finite Bernoulli random graph has been found. It proceeds in successive levels, as shown graphically in figure 2-2 for an example network of 7 nodes, where the state variables are defined as:

(a) i = the current level, or distance in hops from node v.

(b) k_i = the number of nodes which are exactly i hops (minimum) from node v.

(c) l_i = the number of nodes "left over" or greater than i hops from node v.

The initial state (i=0) of the recurrence relation is (0,1,N-1), i.e., since node v is 0 hops from itself, and all other nodes are left over.

nodes at this level and l_{i-1} nodes remaining, which is simply a binomial term given by

$$P_r[k_i = k_i \mid k_{i-1} = k_{i-1} \text{ and } l_{i-1} = l_{i-1}] = \binom{l_{i-1}}{k_i} P_{i-1}^{k_i} (1 - P_{i-1})^{l_{i-1} - k_i} \qquad (2.2.1A)$$

where

$$l_i = l_{i-1} - k_i \qquad (2.2.1B)$$

$$P_i = 1 - (1 - X)^{k_i} \qquad (2.2.1C)$$

A terminal node is reached whenever $l_i = 0$ (no more nodes left, so all minimum hop paths have been found), or $k_i = 0$ (no nodes at this level, so the rest are disconnected).

The state probabilities are then computed by summing over all previous states as

$$\Pr\left[\mathbf{k}_i = k_i \text{ and } \mathbf{l}_i = l_i\right] = \sum_{j=1}^{J} \Pr\left[\mathbf{k}_i = j \text{ and } \mathbf{l}_{i-1} = l_{i-1}\right] \binom{l_{i-1}}{k_i} p_{i-1}^{k_i} (1-p_{i-1})^{l_{i-1}-k_i} \quad (2.2.2)$$

where

$$J = N - i - l_{i-1} + 1 \quad (2.2.3)$$

and

$$P_1 = 1 - (1 - X)^{k_1} \quad (2.2.4)$$

It was found that the transition probability that there are k_i nodes at the next level, given that there are k_{i-1} So the expected number of nodes at any level, i, is computed as

$$E\{\mathbf{k}_i\} = \sum_{j=1}^{N} j \sum_{m=1}^{N} \Pr\left[\mathbf{k}_i = j \text{ and } \mathbf{l}_i = m\right] \quad (2.2.5)$$

And finally, the path length distribution is found as

$$\mathbf{h}_i = \Pr\{\text{ path is exactly i hops }\} = \frac{E\{\mathbf{K}_i\}}{N-1} \quad (2.2.6)$$

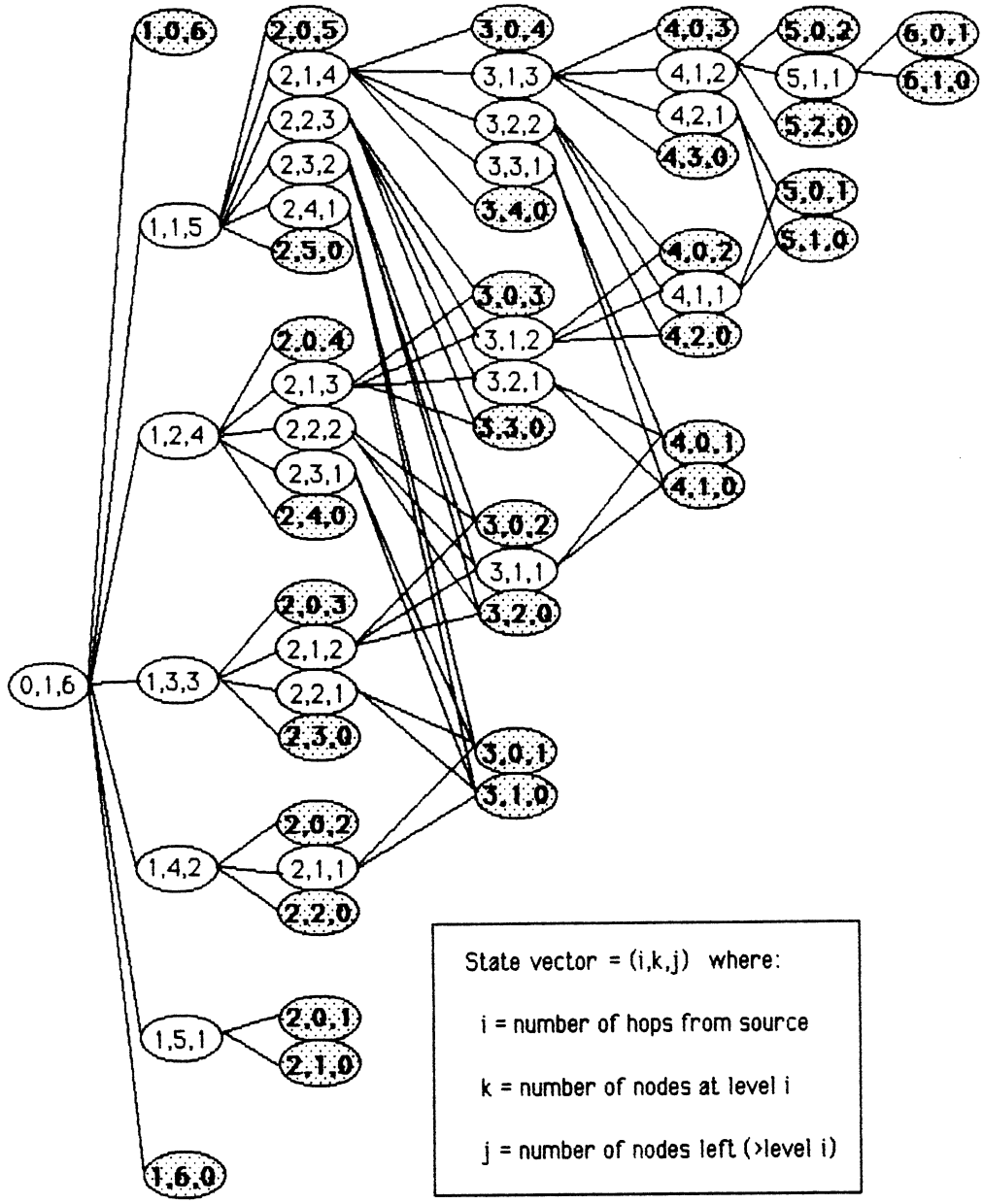

Figure 2-2. Example recurrence relation for determination of path length distribution for a 7 node network. (from [3])

From the (2.2.2) and (2.2.5) we see that the exact computation of path length distribution of nodes is of order N^3 and very computationally intensive for large N, since k_i and k_{i-1} are not independent random variables. Therefore, we wish to find some way to make an approximation for this distribution which is computationally efficient.

To do this, we first recall the initial values $k_0 = 1$, $l_0 = N-1$, $P_0 = X$, then compute the expected value of k_1.

$$E\{k_1\} = \overline{k}_1 = \sum_{k_1=0}^{l_0} k_1 \binom{l_0}{k_1} P_0^{k_1} (1-P_0)^{l_0-k_1} \qquad (2.2.7A)$$

Let $\overline{k}_i$ be the expected value of k_i. Then

$$\overline{l}_1 = l_0 - \overline{k}_1 \qquad (2.2.7B)$$

$$\overline{P}_1 = 1 - (1-X)^{\overline{k}_1} \qquad (2.2.7C)$$

Continuing,

$$\overline{k}_2 = \sum_{k_2=0}^{\overline{l}_1} k_2 \binom{\overline{l}_1}{k_2} \overline{P}_1^{k_2} (1-P_1)^{\overline{l}_1 - k_2} \qquad (2.2.8A)$$

$$\overline{l}_2 = \overline{l}_1 - \overline{k}_2 \qquad (2.2.8B)$$

$$\overline{P}_2 = 1 - (1-X)^{\overline{k}_2} \qquad (2.2.8C)$$

So on, in general,

$$\overline{k}_i = \sum_{k_i=0}^{\overline{l}_{i-1}} k_i \binom{\overline{l}_{i-1}}{k_i} \overline{P}_{i-1}^{k_i} (1-\overline{P}_{i-1})^{\overline{l}_{i-1} - k_i} \qquad (2.2.9A)$$

$$\overline{l}_i = \overline{l}_{i-1} - \overline{k}_i \qquad (2.2.9B)$$

$$\overline{P}_i = 1 - (1-X)^{\overline{k}_i} \qquad (2.2.9C)$$

For equation (2.2.9A), we make use of the formula

$$a = \sum_{j=0}^{l} j \binom{l}{j} P^j (1-P)^{l-j} = lP \qquad (2.2.10)$$

which is simply the mean value of a binomial distribution.

The set of recurrence relation equations can then be written as

$$\overline{k}_i = \overline{l}_{i-1}\overline{P}_{i-1} \qquad (2.2.11A)$$

$$\overline{l}_i = \overline{l}_{i-1} - \overline{k}_i \qquad (2.2.11B)$$

$$\overline{P}_i = 1 - (1 - X)^{\overline{k}_i} \qquad (2.2.11C)$$

Finally, the average path length is given by

$$h_i = \frac{\Sigma\ \overline{k_i}}{(N-1)} \qquad (2.2.12)$$

From the (2.2.11) equations, with initial condition $k_0 = 1$, $l_0 = N-1$, $P_0 = X$, successive expected values of k_i can be quickly computed recursively since each new value depends only on its immediate predecessor. The problem is that in general, $\overline{k_i}$ and $\overline{l_i}$ are not integers, and thus the summation and combinations are inexact.

In [3] the exact results for networks of 25, 50, 100 and 200 nodes are presented (see Table 2-1 and (Figure 2-3). Table 2-2 presents the results using the approximation method given here for 100 and 200 nodes. Comparing the two methods, the errors are less than 2% (see Table 2-3 and Figure 2-4).

From the figure 2-3, we find that as the average degree D increases, the average path length approaches a

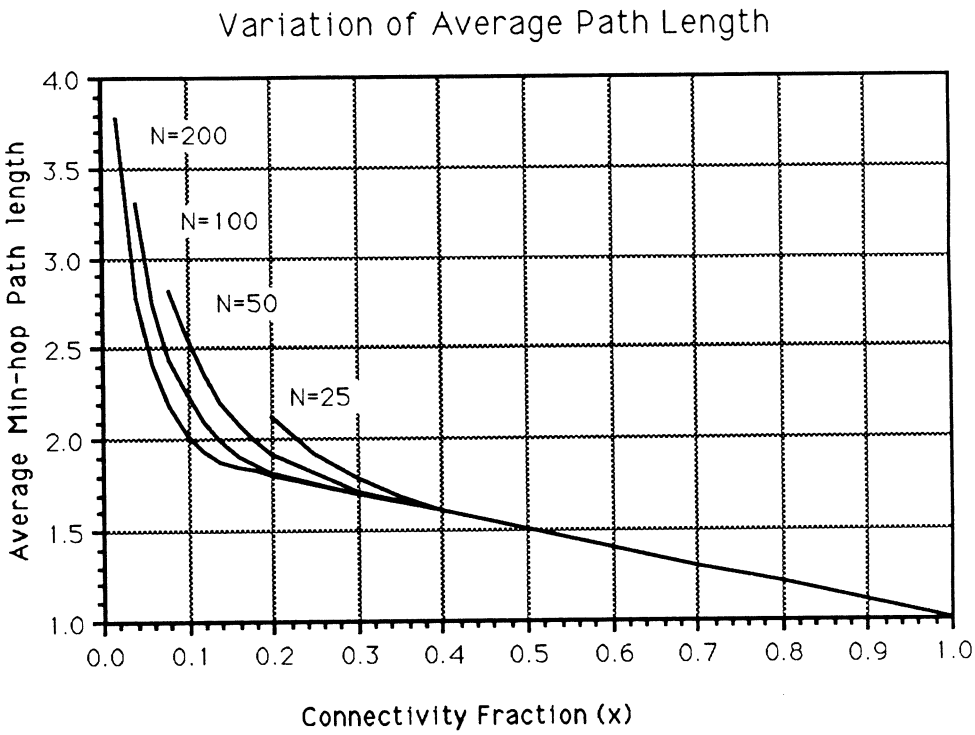

Figure 2-3. Average path length for Bernoulli random graphs. (from [3])

value of 2-X, since X of the paths are one hop, and all remaining paths will be 2 hops with high probability.

2.3 Euclidean Model Analysis:

There are many methods (such as branching processes, percolation processes) by which we can analyze infinite Euclidean graphs. Here we introduce Gilbert's hex method [6] to estimate D_c. It will be shown that this result is close to our result derived expermentally via statistical simulation in Chapter 3 for finite Euclidean graphs.

Hex is played on a honeycomb pattern of hexagonal cells. A typical cell $H(\beta,\partial)$ has its center at the point with Cartesian coordinates:

$$X = \left(\beta + \frac{\partial}{2}\right)b, \quad Y = \partial b\frac{\sqrt{3}}{2}, \qquad (2.3.1)$$

where b is the separation between cell centers and β and ∂

Table 2-1. Exact Minimum Hop Path Lenth Distribution for 100, 200 Nodes (from [3]).

N=100	1 hop	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	no path	mean*
2%	0.020	0.038	0.064	0.093	0.108	0.100	0.077	0.052	0.032	0.019	0.397	3.235
4%	0.040	0.139	0.321	0.318	0.118	0.023	0.003				0.038	3.303
6%	0.060	0.280	0.503	0.145	0.008						0.004	2.747
8%	0.080	0.430	0.463	0.027							0.001	2.436
10%	0.100	0.564	0.333	0.003								2.239
20%	0.200	0.785	0.015									1.815
30%	0.300	0.700										1.700
40%	0.400	0.600										1.600
50%	0.500	0.500										1.500
60%	0.600	0.400										1.400
70%	0.700	0.300										1.300
80%	0.800	0.200										1.200
90%	0.900	0.100										1.100
100%	1.000											1.000

N=200	1 hop	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	no path	mean*
2%	0.020	0.075	0.221	0.357	0.222	0.057	0.009	0.001			0.038	3.784
4%	0.040	0.261	0.587	0.110	0.001						0.001	2.770
6%	0.060	0.480	0.457	0.003								2.404
8%	0.080	0.662	0.258									2.178
10%	0.100	0.777	0.123									2.023
20%	0.200	0.800										1.800
30%	0.300	0.700										1.700
40%	0.400	0.600										1.600
50%	0.500	0.500										1.500
60%	0.600	0.400										1.400
70%	0.700	0.300										1.300
80%	0.800	0.200										1.200
90%	0.900	0.100										1.100
100%	1.000											1.000

***Mean is average path length of existing paths**

Table 2-2. Estimation of Minimum Hop Path Lenth Distribution for 100, 200 Nodes.

N=100	1 hop	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	no path	mean*
2%	0.020	0.038	0.007	0.113	0.154	0.160	0.122	0.070	0.033	0.014	0.206	4.339
4%	0.040	0.143	0.359	0.350	0.081	0.007	0.001				0.019	3.257
6%	0.060	0.289	0.540	0.107	0.002						0.002	2.696
8%	0.080	0.445	0.463	0.012								2.407
10%	0.100	0.583	0.316	0.001								2.218
20%	0.200	0.790	0.010									1.810
30%	0.300	0.700										1.700
40%	0.400	0.600										1.600
50%	0.500	0.500										1.500
60%	0.600	0.400										1.400
70%	0.700	0.300										1.300
80%	0.800	0.200										1.200
90%	0.900	0.100										1.100
100%	1.000											1.000
N=200	1 hop	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	no path	mean*
2%	0.020	0.076	0.237	0.410	0.208	0.028	0.002				0.019	3.746
4%	0.040	0.266	0.614	0.079								2.732
6%	0.060	0.491	0.448	0.001								2.390
8%	0.080	0.676	0.244									2.164
10%	0.100	0.789	0.111									2.011
20%	0.200	0.800										1.800
30%	0.300	0.700										1.700
40%	0.400	0.600										1.600
50%	0.500	0.500										1.500
60%	0.600	0.400										1.400
70%	0.700	0.300										1.300
80%	0.800	0.200										1.200
90%	0.900	0.100										1.100
100%	1.000											1.000

***Mean is average path length of existing paths**

Table 2-3. Estimation Error

Error %		Connection Fraction (X) 2%	4%	6%	8%	10%	20%
N	100	34%	1.4%	1.9%	1.2%	1%	0.2%
	200	1%	1.4%	0.6%	0.6%	0.6%	0

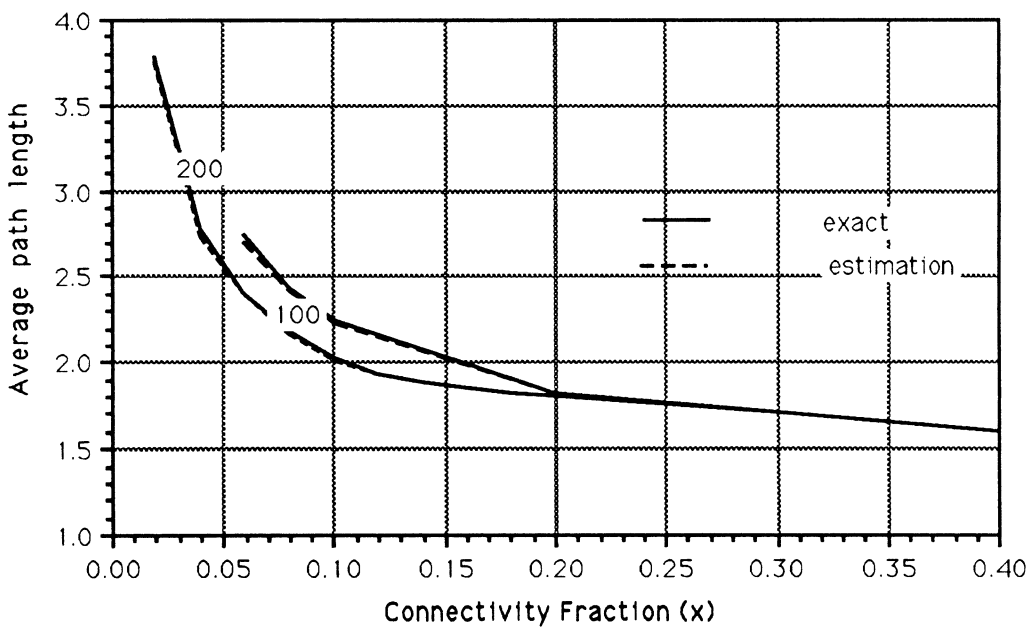

Figure 2-4. Comparison between exact solution and estimation.

assume integer values. A hexboard consists of all cells $H(\beta,\partial)$ such that $|\beta| \leq K$, $|\partial| \leq K$ for some integer K (see figure 2-4). One player, called White, tries to acquire enough cells to have a connected path from $\beta = -K$ to $\beta = K$. His opponent, Black, needs a connected path from $\partial = -K$ to $\partial = K$. Black and White select cells in turn, until a winner is determined. No draw is possible in hex; indeed every assignment of cells to White and Black finds exactly one of the players with a winning path.

Imagine a hexboard drawn on the plane of the random plane network. Give White the cell $H(\beta,\partial)$ if and only if it contains a node of the network. The probability that Black receives a particular cell is

$$P = e^{-\lambda b^2 \sqrt{3}/2} \qquad (2.3.2)$$

Suppose λb^2 is chosen so as to make P = 0.5, i.e.

$$0.5 = e^{-\lambda b^2 \sqrt{3}/2} \qquad (2.3.3)$$

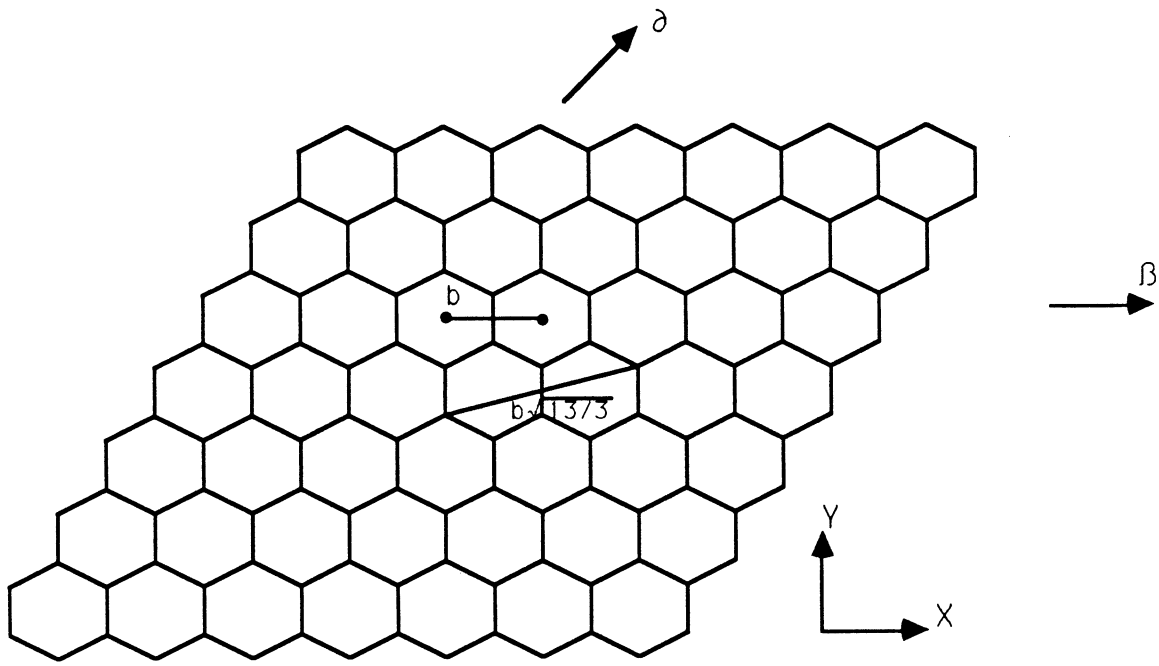

Figure 2-5. Hexboard (K = 3)
(from [6])

so

$$\lambda b^2 = (2\log_e 2)/\sqrt{3} \qquad (2.3.4)$$

Then the game is symmetrical and Black and White each have probability 1/2 of winning; there is probability 1/2 of a connected path of hexagons from $\beta = -K$ to $\beta = K$. If in addition one picks

$$b = R\sqrt{3/13} \qquad (2.3.5)$$

then points in neighboring hexagons are within distance R. A path which provides a component of the random network containing at least 2K+1 points. From [6] we get

$$\lambda R^2(3/13) = (2\log_e 2)/\sqrt{3} \qquad (2.3.6)$$

multiply both side by π, and from $D = \pi\lambda R^2$, then

$$D(3/13) = (2\pi\log_e 2)/\sqrt{3} \qquad (2.3.7)$$

$$\text{So:} \quad D = \frac{26\pi\log_e 2}{3/\sqrt{3}} = 10.9 \qquad (2.3.8)$$

Let K be an integer arbitrarily large, and let H_k be a hexboard of $(2k+1)^2$ cells with separation $b = R\sqrt{3/13}$

between cell centers. There is P probability at least 1/2 that a component of the random network joins a pair of opposite sides of the hexboard and contains at least 2k+1 points.

Since K can be arbitrarily large, the result shows that arbitrarily large components are easy to find when $D_c \geq 10.9$. This means that a random Euclidean graph with equal transmission radii will almost certainly be fully connected if the average degree is greater than 10.9.

In this chapter, we have analyzed the finite Bernoulli random graph model, and developed an approximation technique to estimate the path length distribution quickly and efficiently. We have also presented previous theoretical results for the infinite Euclidean model. In the next chapter we will use a simulation technique to produce some numerical results to analyze the finite Euclidean model, and develop an approximation method for the path length distribution of a finite Euclidean graph.

Chapter 3

Simulation and Results

3.1 Finite Euclidean Simulating Model:

From previous analysis we see that to find the connectivity of finite Euclidean graphs is much more complex because of the "edge effect". In this chapter, we use a computer simulation hosted on a VAXstation 3100 computer to model finite Euclidean graphs. First we create a set of random node locations, which satisfy the two dimensional Poisson distribution.

From previous analysis, we know that by choosing the transmission radius, we can control average degree. In the simulation we wish to find the average path length under a specified average degree, so we need to determine the radius which will result in a specified average degree for each random set of locations.

In our program we initialize R using equation (2.1.3). We then find the adjacency matrix and compute the exact average degree of the random network, given the selected value of R. If the average degree is bigger than the expected value, then we decrease the radius by half of R. If the average degree is smaller than expected value, we

increase the radius by half of R. We iterate in this fashion until we achieve the desired value of average degree to within a specified threshold.

Once we have the correct radius, we next find the adjacency matrix. We then use a shortest path routing algorithm to find the minimum path length. In our simulation we choose Dijkstra's Algorithm [12]. This algorithm is illustrated Fig. 3-1 and Table 3-1 for a small network of 7 nodes.. The objective is to find the least cost (i.e. minimum hop) path from node 1, as the source, to all other nodes in the network. The algorithm does this in a step-by-step process, building a shortest-path tree, rooted in the source node (node 1 in this example), until the most distant node in the network has been included. By kth step the shortest paths to the k nodes closest to the source have been calculated. These are defined to be within a set V.

Let D(v) be the number of hops from source 1 to nodes v. Let L(i,j) be the hops between node i and node j. There are then two parts to the algorithm: an initialization step, and a step to be repeated until the algorithm terminates.

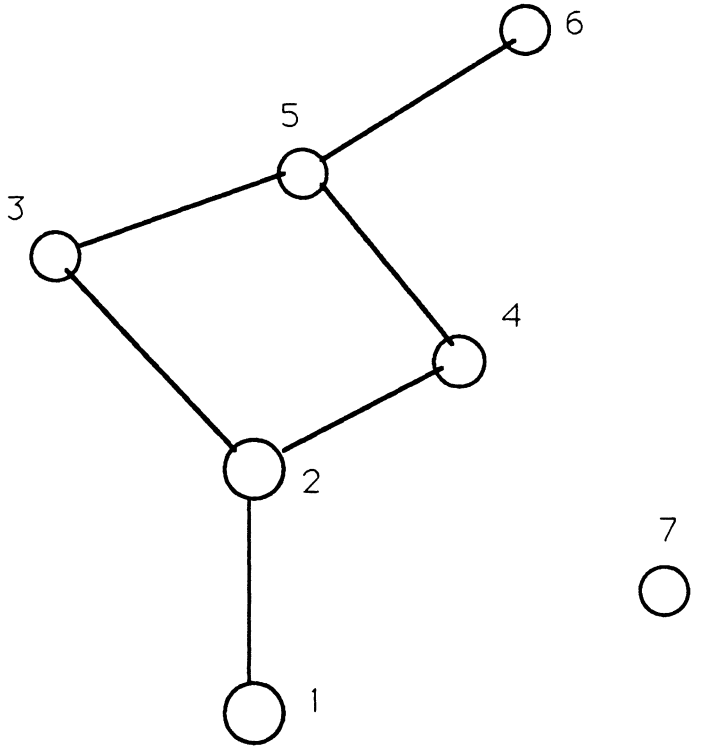

Figure 3-1. An Example Network

Table 3-1. The Shortest Path Tree From Node 1

steps	D(2)	D(3)	D(4)	D(5)	D(6)	D(7)
Initialize	1	∞	∞	∞	∞	∞
1		2	2	∞	∞	∞
2				3	∞	∞
3					4	∞
4						∞

The algorithm is stated below:

1. Initialization. Set V={1}. For each node v not in V, set D(v)=L(1,v). (We use ∞ for nodes not connected to 1; any number larger than the maximum cost or radius distance in the network would suffice.)

2. At each subsequent step. Find a node w not in V for which D(w) is a minimum, and add w to V. Then update D(v) for all nodes remaining that are not in V by computing

$$D(v) \leftarrow \text{Min}[D(v),\ D(w)+L(w,v)]$$

Step 2 is repeated until all nodes are in V.

Using this algorithm we can find the minimum path lengths for all nodes and then compute the average path length. The dynamic procedure is illustrated in figure 3-2 and the program source code is included in Appendix A.

3.2 Analysis of Numerical Results:

Using the previously stated algorithms, we simulate the construction of finite Euclidean graphs. We generate10 independent networks for each value of 100, 200, 300,400 and 500 nodes. From the data we plot the average path length versus average degree in Fig,3-3, 3-4, 3-5, 3-6 and 3-7.

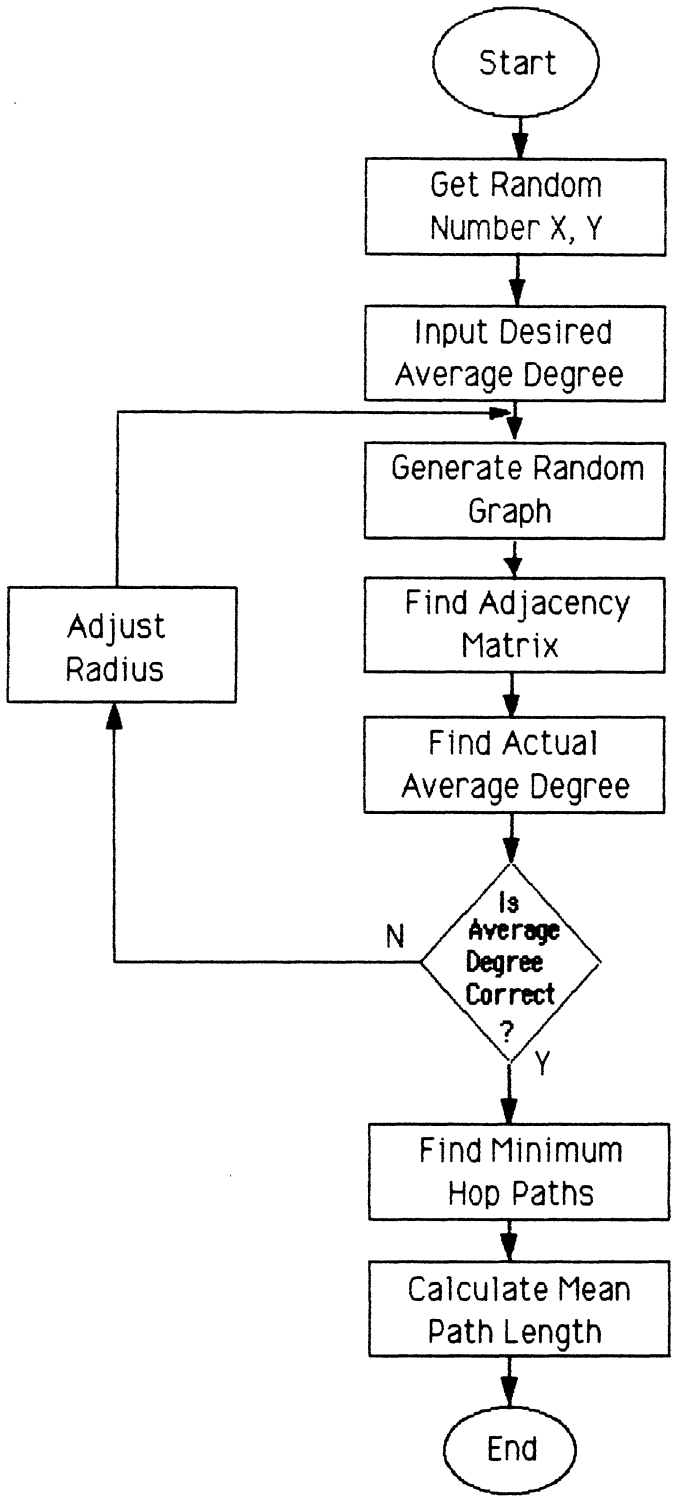

Figure 3-2. Dynamic Procedure For Simulating Finite Euclidean Model.

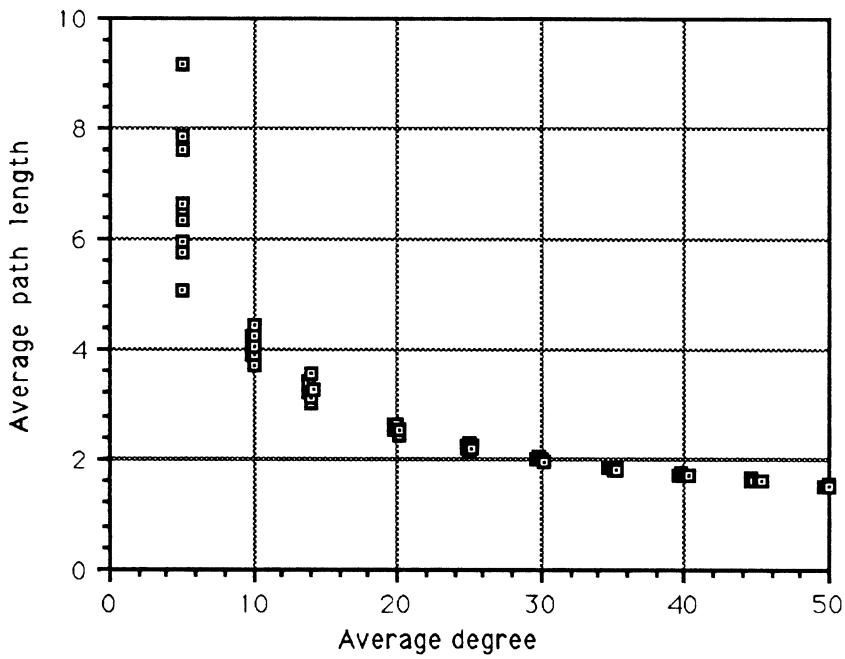

Figure 3-3. Distribution of 10 independent network of 100 nodes.

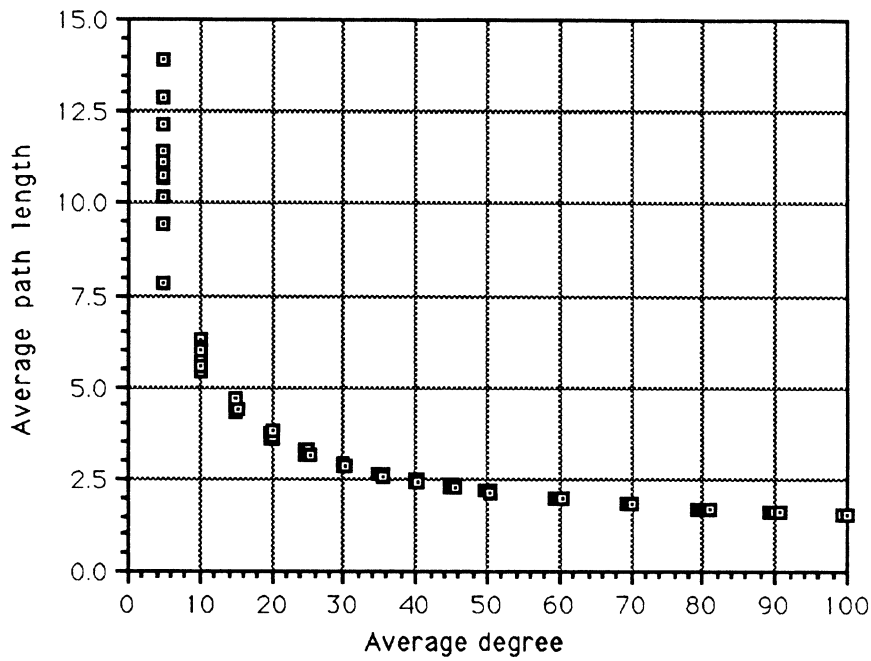

Figure 3-4. Distribution of 10 independent networks of 200 nodes.

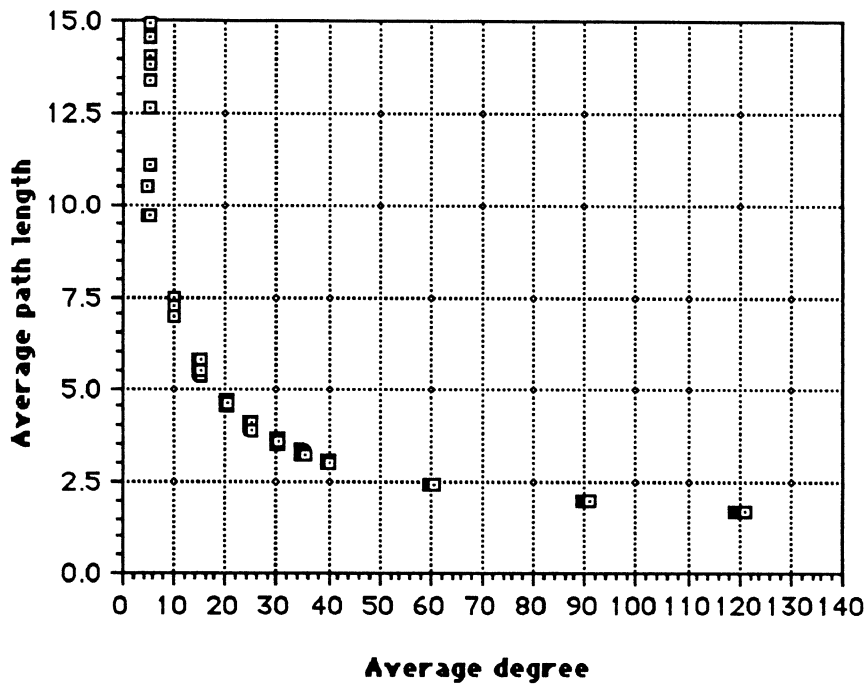

Figure 3-5. Distribution of 10 independent networks of 300 nodes.

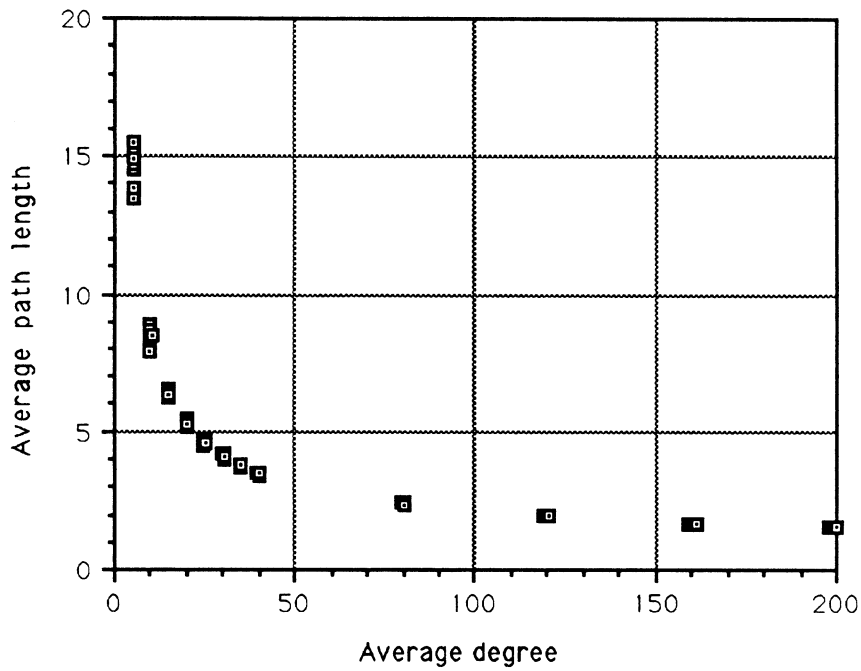

Figure 3-6. Distribution of 10 independent networks of 400 nodes .

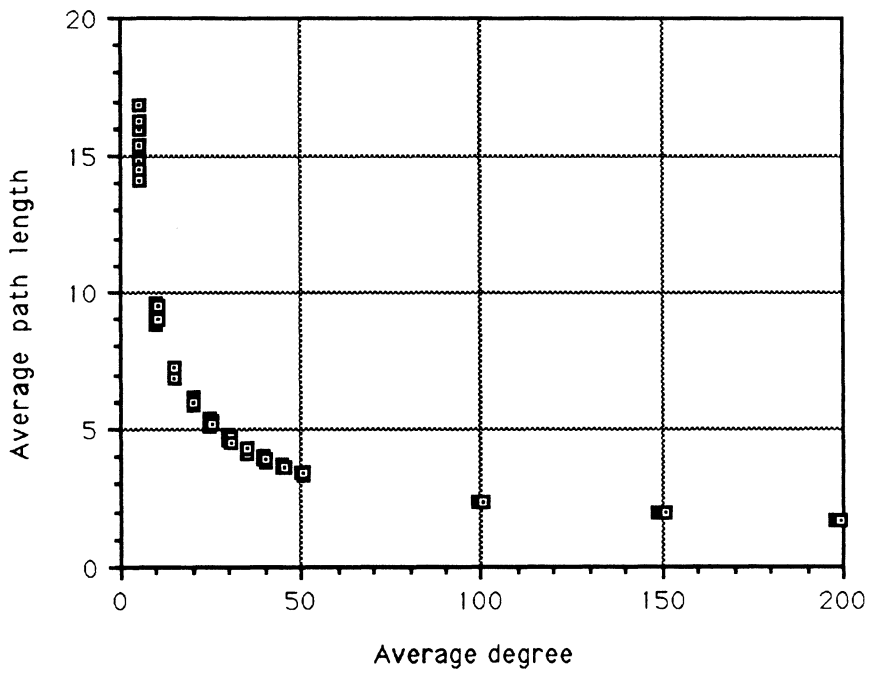

Figure 3-7. Distribution of 10 independent networks of 500 nodes.

3.2.1 Lower-bound on Average Degree:

From the above figures, we can see that when the average degree is small the sparse topology network will yield widely varying average path lengths. When the average degree is increased, the average path length of different networks result in smaller variance. This indicates that there is a certain value of average degree D_c, below which the network will likely be disconnected and the average path length will be difficult to predict.

To find D_c, we use the standard deviation of the average path length to measure the spread of the average path length for different random networks. This is plotted in Figure 3-8. From Fig. 3-8 we find that when the average degree $D>10$, the standard deviation is very small, since 10 individual networks have nearly the same value of the average path length. We choose to use the approximation $D_c=10$. When number of nodes increases, D_c will increase slightly. This result agrees closely with the result of Gilbert's's hex method discused in chapter 2. Thus, when the average degree of a Euclidean network is less than 10, the network has a significant probability of being disconnected, and also exhibits a dramatic increase in the

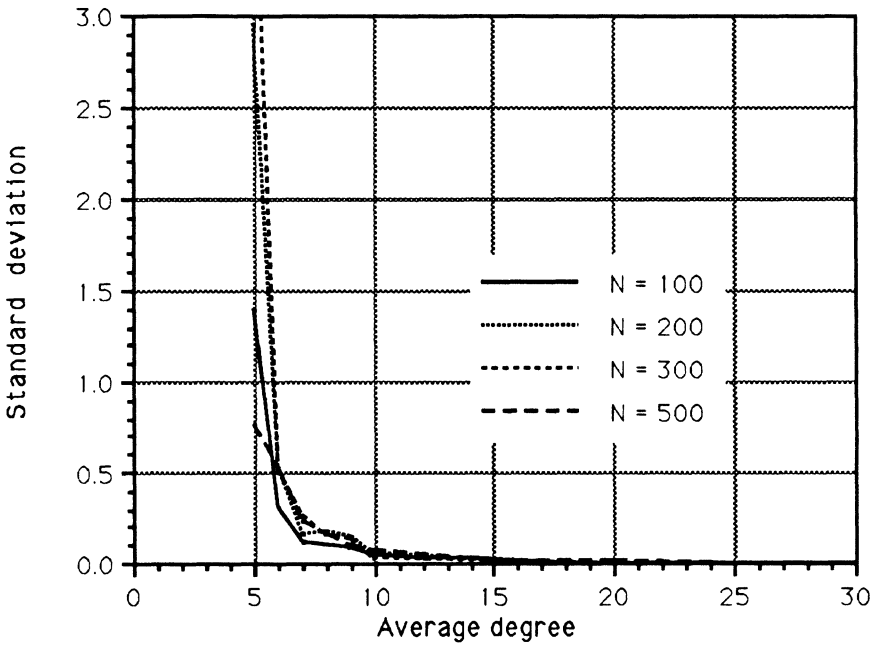

Figure 3-8. Standard deviation of 10 independent networks for average path length.

variance of path length. This represents a fundamental limitation on subsequent results in this paper. If the average degree of the network is below 10, then the approximation technique presented here is invalid.

3.2.2 Approximation Model for Average Path Length

After finding the lower limit on average degree , we now try to find an approximate relationship between the average path length, the number of nodes, and the connectivity fraction. We compute the mean path length for 10 random networks and plot this in figure 3-9. From the figure we find that when the connectivity fraction is bigger than 0.5, the relation with path length is $\overline{h} = 2 - X$. This is equivalent to the Bernoulli model, since x of the paths are one hop, and remaining paths will be 2 hops with very high probability.

Next we must find an empirical relationship for networks with connectivity fraction below 0.5.

From the plotted curves, we can see they are roughly of hyperbolic shape in the region of interest, i.e.

$$y = c\, x^{b}, \quad b < 0. \tag{3.2.1}$$

We next use a least squares curve fit approach to find a hyperbolic curve which fits the simulation data.

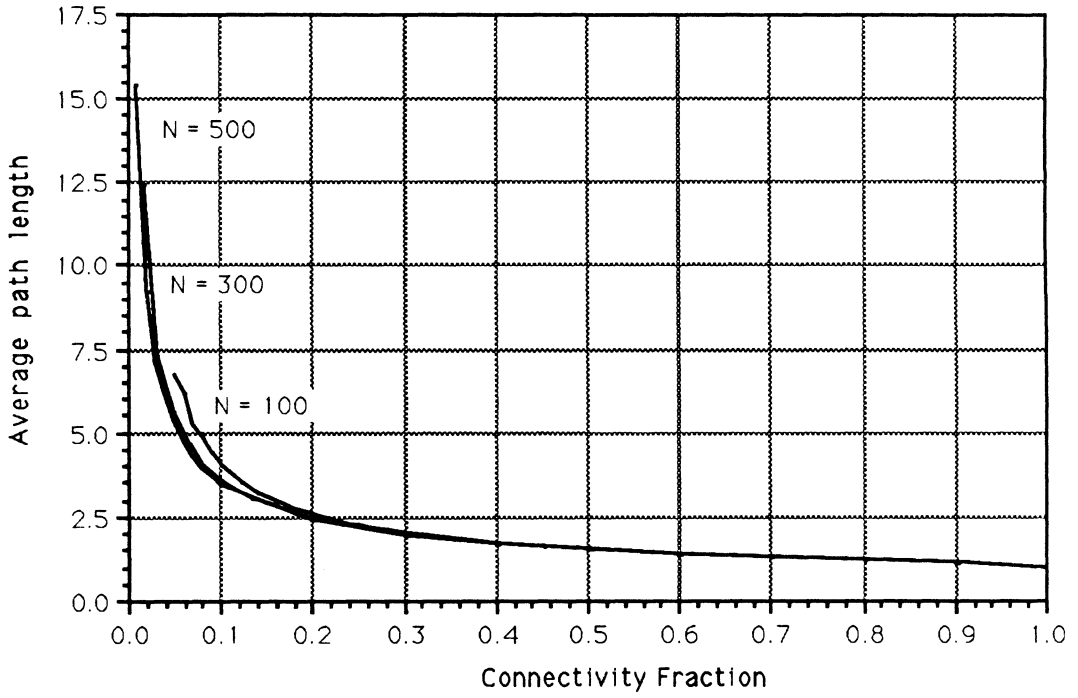

Figure 3-9. Simulation Results for Mean Path Length of 10 Independent Networks.

For the hyperbolic equation $y = c\ x^{b}$ ($b<0$), we take logarithms of both sides

$$\text{Log}\ (y) = \log c + b \log (x), \qquad (3.2.2)$$

which is a linear equation in log x and log y.

Let x_i = Log(x_i), y_i = log(y_i), A = log(c). We wish to find a line that is, of the type $\hat{y}_i = A + Bx_i$ - that will give values $\hat{y}_i$ that are as close as possible to the data values y_i. Suppose that trial values A and B have been chosen and the corresponding estimates $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ of the values $y_1, y_2, \ldots, y_n$ have been computed. To decide whether the values A and B are good choices, we determine the distance between the $\hat{y}$'s and the y's. The differences $y_i - \hat{y}_i$ are squared and summed, giving the criterion

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Since $\hat{y}_i = A + Bx_i$, this is

$$\sum_{i=1}^{n} [y_i - (A + Bx_i)]^2$$

The quantity $y_i = (A + Bx_i)$ is the vertical distance from the line y=A+Bx to the point (x_i, y_i). The criterion is

the sum of the squares of these vertical distances. The least-squares estimates, which will be denotes by a and b, are the values of A and B that minimize the criterion. these values are

$$b = \frac{S_{xy}}{S_{xx}^2} \tag{3.2.3}$$

$$a = \bar{y} - b\bar{x} \tag{3.2.4}$$

where

$$S_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n - 1} \tag{3.2.5}$$

$$S_{xx} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}} \tag{3.2.6}$$

S_{xy} is the covariance between x and y and S_x^2 is the variance of x. The statistics a and b are the least-squares estimates.

Applying this method to the simualtion data, we find that c is approximately equal to 0.99. For b we plot the results in figure 3-10 and we find the relationship between b and N to be

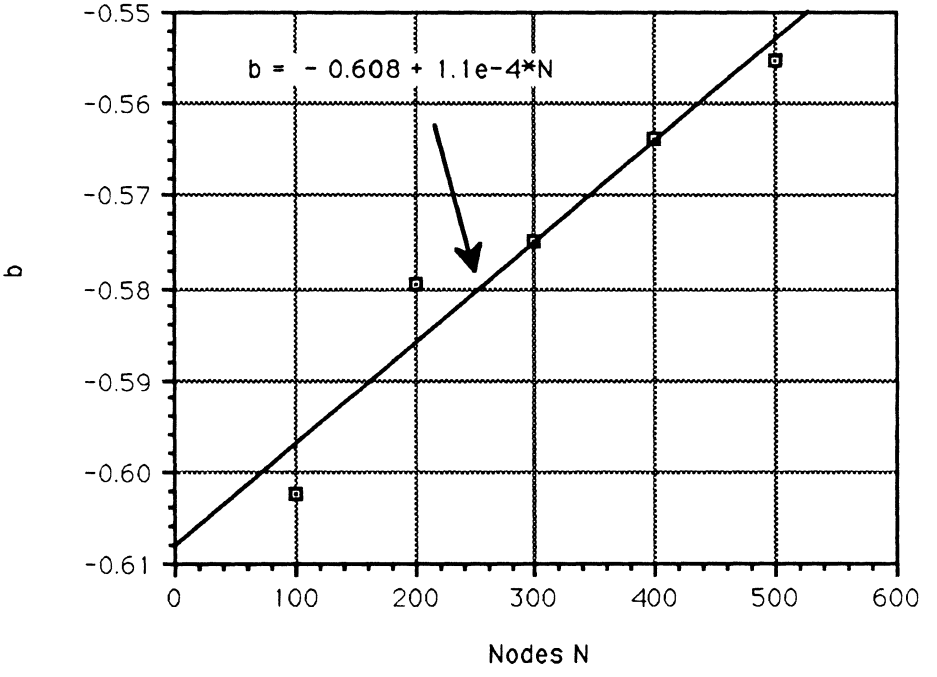

Figure 3-10. Statistical relationship between b and N.

$$b = -0.608 + 1.1\ e{-}4 * N \qquad (3.2.7)$$

arrive at the Euclidean model approximation:

$$\bar{h} = \begin{cases} 0.99 * X^{-0.608+1.1e-4*N} & X < 0.5 \\ 2 - X & X \geq 0.5 \end{cases} \qquad (3.2.8)$$

This is another primary result . It shows the average
length as a function of connectivity fraction x and
al nodes N. It should be noted here this result is only
itable for networks with average degree greater than 10,
ecause below that value the network is likely to be
disconnected. Given N, the average path length is a
hyperbolic function of average degree.

In Fig. 3-11, we plot the two approximate models,
Bernoulli and Euclidean by using the approximation
techniques. As noted before it this represents the lower
and upper bounds on the modeling of "real" topology.

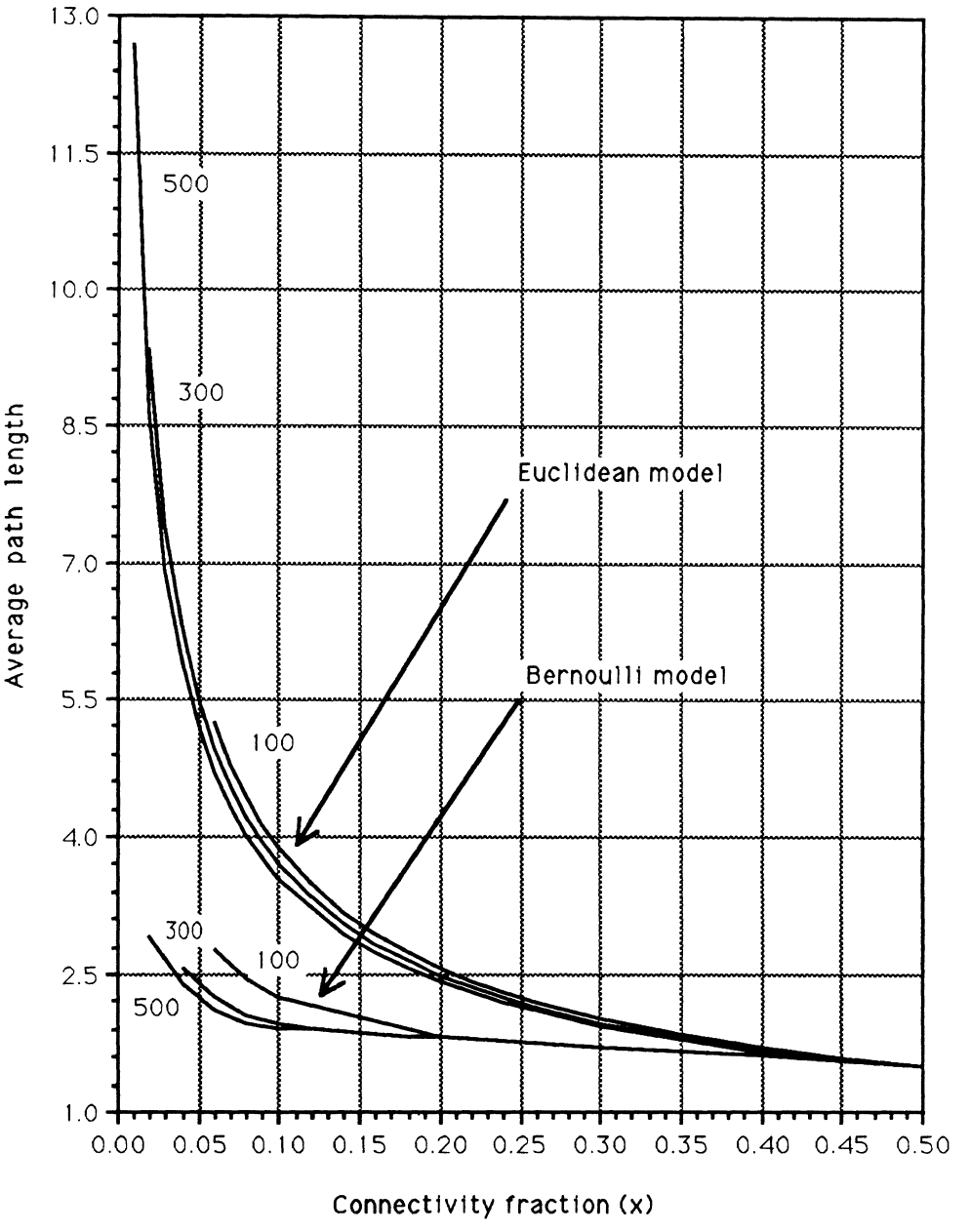

Figure 3-11. Approximations of Finite Euclidean and Bernoulli Models ($x \leq 0.5$).

Chapter 4

Summary and Conclusion

In this paper we have examined the structure of finite random graphs using both the Euclidean and the Bernoulli models, which are the limiting cases for the topology of real mobile radio networks. We have developed approximation methods which accurately predict the path length distribution and / or the average path length of arbitrary random networks. This path length distribution can then be subsequently used in conjunction with other information (i.e. the channel access protocol, message length distribution, routing algorithm and traffic distribution) to predict the overall throughput and delay of the network.

For the Bernoulli model, an approximation technique with computational complexity N was derived, which is very close to the exact solution (with computational complexity N^3). For the Euclidean model, an analytical solution is difficult due to the boundary effects of a finite Euclidean graph. Instead, a least square curve fit is used in conjunction with Monte Carlo simulation data to develop a piecewise continuous approximation of the average path length. For low connectivity fractions, the approximation

has a hyperbolic shape, while at high connectivities (X>0.5), the average path length approaches the value 2-X, where X is the connectivity fraction. Thus for highly connected networks, the Bernoulli and Euclidean models converge.

In analyzing the performance of real networks, these approximations will provide upper and lower bounds on the average path length and consequently on the achievable throughput of the network. Greater prediction accuracy could be achieved by a combination of these two models into a single network model, which incorporates both. This, however, would require much experimental data to determine the precise effects of terrain, foliage, etc., and would likely result in a model which is highly dependent upon the particular terrain and chosen node locations, thus losing much of its generality.

References

1. Anderson, T.W. and Sclove, Stanley L., "The Statistical Analysis of Data". The Scientific Press, 1986, pp.528-p530.

2. Dewitt, H., "The Theory of Random Graphs with Applications to the Probabilistic Analysis of Optimization Algorithms", Ph.D.Dissertation, 1977, Department of Computer Science, U.C.L.A., Los Angeles.

3. Dill, J. C., "On the Throughput of Multihop Receiver Directed CDMA Packet Radio Networks with Dynamic Random Topology," Ph.D. Dissertation, 1987, University of Southern California.

4. Erdos, P. and Renyi, A., "On Random Graphs I", Publications Mathematics, December 6, 1959, pp.290-297.

5. Feller, William An Introduction to Probability Theory and Its Applications, Volume 1, John Wiley & Sons, Inc, 1968.

6. Gilbert, E. N., "Random Plane Networks,"SIAM Journal, Vol.9, No.4, December, 1961, pp.533-543.

7. Kleinrock, L. and Silvester, J., "Optimum Transmission Radii for Packet Radio Networks or Why Six is a Magic Number",IEEE Conf. Rec. Nat. Telecommun. Conf., Dec, 1978, pp.4.3.1-4.3.5.

8. __,"Spatial Reuse in Multihop Packet Radio Networks", Proc. of IEEE, Vol. 75, No.1, Jan. 1987, pp.156-167.

9. Larson, H. J. and Shubert, B. O., Probabilistic Models in Engineering Sciences, Volume 1, John Wiley & Sons, Inc, 1979.

10. Palmer, Claude Irwin, Analytic Geometry, with introductory chapter on the calculus, McGraw-Hill, 1921

11. Press, W. H., Flannery, B. P., Teukolsky, S. A., Wetterling, W. T., Numerical Recipes, the Art of Scientific Computing, Cambridge University Press, 1986.

12. Schwartz, M., Telecommunication Networks. Addison-Wesley, 1987.

13. Tajibnapis, W. D. "A Correctness Proof of a Topology Information Maintenance Protocol for Distributed Computer Network", Communications of the ACM, July 1977, Vol. 20, Number 7, pp.478-485.

Appendix A

Simulation Program

```
c  ************************************
c *      Simulating finite Euclidean model     *
c *                    program                  *
c  ************************************
c
c ** Dimension **
c
          real  x(1000), y(1000), av_deg(20), radius(20),
         mean_path(20)
          real  ran, degr, radiu, xx, yy, rr, de, expect_value,
         total_hop
          integer    z(1000,1000), d(1000,1000), deg(1000),
         c(20,50),st
          integer  hop(30), i, j, num1, num, seed, net_num
          integer  degr_star, degr_end, degr_step
          character*16  datafile_name
c
c
          write(6,*) ' input the numbers of nodes'
         read (6, *) st
          write(6,*) ' input start, end and step average
```

```
                    degree ='
          read (6, *) degr_star, degr_end, degr_step
          write(6,*) ' input creating networks number ='
         read (6,*) net_num
          write(6,*) ' input the data file name: '
          read (6,5) datafile_name
   5     format (a16)
c
c ** to generate the random position of the random graphs
**
c
            open ( unit=82, file = datafile_name, status =
'new')
            do 250  degr = degr_star, degr_end, degr_step
             do 200 se = 1, net_num
             seed = -100*se
               do i = 1, st
                x(i) = ran(seed)
               end do
               do i = 1, st
                y(i) = ran(seed)
               end do
              radiu = sqrt(degr/(3.14*st))
     19      num = 0
             do  30 i = 1, st
                do 20j = 1, st
```

```
                if (i .eq. j) then
                 z(i,j) = 1
                 goto 20
                end if
                 xx = abs(x(i) - x(j))
                 yy = abs(y(i) - y(j))
           rr = sqrt(xx*xx + yy*yy)
                           ! find the distance between nodes
                 if (rr .le. radiu) then
                                ! find the adjancent matrix
                   z(i,j) = 1
                else
                   z(i,j) = 0
                   num = num + 1
                end if
   20          continue
   30        continue
c
c ** find the average degree **
c
             do 45 i = 1, st
                deg(i) = 0
                do 40 j = 1, st
                   if (i .eq. j) then
                   goto 40
                   end if
```

```
                deg(i) = z(i,j) +deg(i)
                      ! find degree in i column
40              continue
45            continue
              de = 0
                do 47 i = 1, st
                   de = deg(i) + de
                         ! find the total degree
47              continue
               de = de / st
                         ! find the average degree
                write (6, *)  ' the average degree = ', de
                if (de .gt. (degr + 0.01*degr))  then
                 radiu = radiu / 2
                 goto 19
                end if
                if (de .lt. (degr - 0.01*degr))  then
                  radiu = radiu + radiu / 2
                 goto 19
                end if
                av_deg(se) = de
                            ! find the average degree
                radius(se)  = radiu
                           ! find the radius
c
c ** find the minimum hops **
```

```
c
                do 60 i = 1, st
                  do 50 j = 1, st
                     d(i,j) = z(i,j)
     50            continue
     60           continue
                  num1 = num
     70           do 100 i = 1, st
                    do 90 j = 1, st
                       if (d(i,j) .eq. 0) then
                        call xmin(i,j,st,d,num1)
                       end if
     90              continue
    100           continue
                   if (num .eq. num1) then
                     num = 0
                  else
                     num = num1
                  end if
                  if (num .gt. 0) then
                    goto 70
                  end if
                  do 106 i = 1, st
                    do 105 j = 1, i
                       if ( i .eq. j ) then
                         d(j,i) = 0
```

```
                    else if (d(j,i) .eq. 0) then
                      d(j,i) = -1
                    end if
                    d(i,j) = d(j,i)
  105             continue
  106           continue
c
c ** find the means path links **
c
              do 120 i = 1, st
                do 110 j = i, st
                   do k = 1, 29
                       if ( d(i,j) .eq. k) then
                         hop(k) = hop(k) + 1
                         goto 110
                       end if
                        if (d(i,j) .gt. 29 .or. d(i,j) .eq. -1)
                          hop(30) = hop(30) + 1
                       goto 110
                       end if
                     end do
  110             continue
  120           continue
                do  i = 1, 30
                   c(se, i) = 2*hop(i)
                end do
```

```
                expect_value = 0
                total_hops = 0
                hop(30) = 0
                do i = 1, 29
                   expect_value = i * hop(i)
                    total_hops = hop(i) + total_hops
                  hop(i) = 0
                end do
                  mean_path(se) = total_hops / expect_value
   200          continue
c
c ** save and print out the result **
c
             do 300 k = 1, net_num
             write(82, 290) st, av_deg(k), radius(k),
mean_path(k),(c(k,i),j=1,30)
    290  format(3x, i4, 1x, f7.3, 1x, f6.4, 1x, f6.3, /, 3(2x,
10(i6, 1x),/),/,/)
    300   continue
             do 400 k = 1, net_num
             write(82, 290) st, av_deg(k), radius(k),
              mean_path(k),(c(k,i),j=1,30)
    400  continue
    500  continue
    999  end
c
```

```
c ** uniform random number generate **
c
            real function ran(idum)
           real r(97)
            parameter (m1=259200, ia1=7141, ic1=54773,
           rm1=1./m1)
            parameter (m2=134456, ia2=8121, ic2=28411,
           rm2=1./m2)
            parameter (m3=243000, ia3=4561, ic3=51349)
           data iff /0/
            if (idum .lt. 0 .or. iff .eq. 0) then
           iff = 1
            ix1= mod(ic1- idum, m1)
            ix1= mod(ia1*ix1 + ic1, m1)
            ix2= mod(ix1, m2)
            ix1= mod(ia1*ix1+ ic1,m1)
            ix3= mod(ix1, m3)
           do 11 j = 1, 97
               ix1=mod(ia1*ix1 + ic1, m1)
               ix2=mod(ia2*ix2 + ic2, m2)
                r(j) = (float(ix1) + float(ix2)*rm2)*rm1
    11     continue
           idum = 1
            ix1 = mod(ia1*ix1 +ic1, m1)
            ix2 = mod(ia2*ix2 + ic2, m2)
            ix3 = mod(ia3*ix3 + ic3, m3)
```

```
            j = 1 + (97*ix3)/m3
            if (j .gt. 97 .or. j .lt. 1)  pause
           ran = r(j)
            r(i) = (float(ix1) + float(ix2)*rm2) * rm1
           return
           end
c
c ** mini hops subroutine **
c
            subroutine xmin(i,j,st,d,num1)
           integer  d(1000,1000)
           do 10 v =1, st
              if (d(i,v) .ne. 0) then
                 if (d(v,j) .ne. 0) then
                    if (d(i,j) .eq. 0) then
                    d(i,j) = d(i,v) +d(v,j)
                   num1 = num1 - 1
                   else
                     d(i,j) = min0(d(i,j), (d(i,v) + d(v,j)))
                   end if
                end if
              end if
   10      continue
           return
           end
```