DOCUMENT RESUME

ED 098 626 CS 500 840

AUTHOR Richards, William D., Jr.
TITLE Network Analysis in Large Complex Systems: Techniques and Methods--Tools.
PUB DATE Apr 74
NOTE 40p.; Paper presented at the Annual Meeting of the International Communication Association (New Orleans, Louisiana, April 17-20, 1974)

EDRS PRICE MF-$0.75 HC-$1.85 PLUS POSTAGE
DESCRIPTORS Analog Computers; *Computer Programs; Higher Education; Information Processing; *Networks; *Organizational Communication; *Program Descriptions; *Systems Analysis

ABSTRACT

Divided into five major sections, this paper describes a new algorithm which has been implemented in an extended FORTRAN program which runs on a CDC 6500 computer. The first section of the paper briefly outlines the goals of network analysis and presents the context in which these goals must be met. Section 2 describes the algorithm and the rationale behind it. In section 3 some especially important programing considerations are described, and section 4 covers some general characteristics of running the program. The final section of the paper briefly describes the historical development of this algorithm. (RB)

Network Analysis in Large Complex Systems:

Techniques and Methods--Tools

Paper presented to the 1974 meetings of the International Communication Association April 16-21, 1974

William D. Richards,Jr.

Institute for Communication Research

Stanford University

TECHNIQUES AND METHODS--TOOLS

In 1971 a formal algorithm for analyzing communication networks in large complex organizations was presented. (1) Since that time there have been many advances in the area--some conceptual (2) and others operational. This paper will describe a new algorithm which has been implemented in an Extended FORTRAN program which runs on a CDC 6500 computer. (3) This algorithm could be realized on any large general purpose machine, and it far surpasses any other similar analytic technique we are aware of, in terms of utility, capacity, and efficiency.

This paper will not discuss the theoretical basis of network analysis, nor will it report any empirical findings. For coverage of these areas, the reader is urged to see (2) and (4). Because the computer program mentioned above is highly complex and system-dependent, the actual code will not be presented. It should be possible, however, to write a similar program for any given machine with the information that will be presented here.

The paper will be divided into five major sections. The first will briefly outline the goals of network analysis and present the context in which these goals must be met. The second will describe the actual algorithm and the rationale behind it. In section three some especially important programming considerations are described; section four covers some general characteristics of the running program. Finally, the last section will describe briefly the historical development of this algorithm. Throughout the major portion of the first three sections, the approach taken in the description of techniqu will be to discuss goals and constraints of each facet of the analysis;

and then to show that the methods used are able to efficiently meet the goals, given the constraints.

Because much of the technique of network analysis depends on the theory of network analysis, it is difficult to discuss analytic methods without referring to the theoretical basis, which is covered extensively in (2). Rather than repeating discussions which are given better treatment in (2), the symbol "**" will be used to indicate that a point covered in this text is discussed more fully in (2).

Part One--Network Analysis: The Problem

The goals of network analysis are to a) detect and b) describe any structuring at each of three levels of the communication network. (The nature of structuring in complex systemr is covered at length in (5) and will not be discussed here.) These three levels are the individual, the group, and the whole-system. The detection of structure is a straightforward statistical problem which is easily done. If there is any structuring, it is then possible to describe it, and to analyze various characteristics of the system at different levels of analysis. The kinds of characteristics that can be examined will also not be covered in this paper; they are described in (6).

The basic problem we are thus faced with is this: Given some particular network which we know to be structured, we must determine what the units of analysis are at the various levels of analysis. At the individual and whole-system levels this doesn't seem to be much of a problem. At intermediate levels, however, the determination of the component boundaries is a complex problem. This is the problem area upon which our attention is focused in this paper.

What is meant by "the determination of component boundaries at intermediate levels"? Basically this: if the system as a whole is structured, it will show differentiation into parts. (5,7) These parts will take the form of groups of individuals which will meet certain specified criteria. We would like to know who is in these groups. This, then, is our main goal.

What are the constraints under which we must meet this goal? They center on the nature of the data we have available. The data ususlly take the form of lists of links between pairs of nodes.** There is no limit on how many links any individual may have--i.e. the number of links may vary from one individual to the next--there is no set number.[2] All we know about each link is who it connects and how strong it is. (We may also know if it is directed or not.)

A major consideration is size of network--we would like to be able to study very large systems--we want a maximum capability of at least a thousand nodes, and it might be nice to be able to look at networks with several thousand nodes.[3] In networks we have observed, the number of links per node has ranged from zero to a maximum average of about twenty. If we allow a limit of 1000 nodes, we should have room enough for at least twenty times that number of links. This is a lot of data!

Network data is not like most data we are used to seeing in the social sciences. The data elements do not describe properties of individuals. Rather, they describe properties of relationships between individuals. What we have, then, is a topological problem. We cannot approach this problem with the Euclidian distance paradigm used to structure other kinds of data. We cannot, therefore, hope to use the mathmatical tools which produce unique exact solutions based on a distance model. Instead, we use heuristic pattern-recognition

techniques which result in topological representations which may then be characterized along a number of dimensions.

The problem then is to arrange and represent the data in such a way that it becomes possible to see the groups. We do not use traditional scaling approaches because our data do not fit those models. Instead we use less elegant pattern-recognition techniques. These techniques are described in the next part of the paper.

Part Two--The Algorithm

In any searching procedure it is necessary that the investigator have a good idea of what it is that he is looking for. This common sense notion cannot lightly be dismissed, especially if we wish to write a computer program to do the searching for us. Computers, like chickens, are monumentally stupid. Unlike chickens, however, computers will do what we want them to do if we tell them exactly what to do and how to do it. This means that we must know exactly what we are looking for. This demand for precision lead to the somewhat complex definition of groups and other network roles which is presented here.**

I. Nodes may be of two types--participants and non-participants. Non-participants are either not connected to the rest of the network or are only minimally connected. They include:
   A. Isolate type one. These nodes have no links of any kind.
   B. Isolate type two. These nodes have one link.
   C. Isolated dyad. These nodes have a single link between themselves.
   D. Tree node. These nodes have a single link to a participant, and have some number of other isolates attached to them.

II. Participants are nodes that have two or more links to other participant nodes. They make up the bulk of the network in most cases, and allow for the development of structure. They include:

A. Group member. A node with more than some percentage of his linkage with other members of the same group. (this percent is called the alpha-percent or α-percent)

B. Liaison. These nodes fail to meet the α-percent criterion with members of any single group, but do meet it for members of groups in general.

C. Type other. These nodes fail to meet the α-percent criterion for any set of group members.

III. To be called a group, a set of nodes must satisfy these five criteria.

A. There must be at least three members.

B. Each must meet the α-criterion with the other members of this group.

C. There must be some path, lying entirely within the group, from each member to each other member. (This is called the connectiveness criterion.)

D. There may be no single node (or arbitratily small set of nodes) which, when removed from the group, cause the rest of the group to fail to meet any of the above criteria. (This is called the critical node criterion.)

E. There must be no single link (or subset of links) which, if cut, causes the group to fail to meet any of the above criteria. (This is called the critical link criterion.)

Obviously, if we think a certain set of nodes might be a group, we can be sure by applying tests to see if the set meets the five group criteria. If there are slight errors, we can adjust the group boundary by adding or removing nodes to or from the group. In other words, if we can get an approximate answer to the question

of group boundaries, we can "clean it up" and make it exact by the application of the criteria.

It turns out that we can take advantage of this fact and make significant savings because of it. This is because it is easier to make an educated guess out the group structure and then adjust this to an exact solution than it is to begin right away with a search for the exact solution. The algorithm to be presented in this paper follows this two-staged approach. In the first stage, the data are rearranged in such a way that an approximate solution is readily obtained. In the second, that tentative solution is tested and cleaned up so that it becomes exact enough to satisfy the criteria.

The Approximate Solution

There are three stages to this part of the analysis. In the first stage the non-participants are identified and removed for the rest of the analysis. This is done because the presence of non-participants only serves to complicate things by increasing the number and variety of nodes in the analysis. The non-participants are easily identified by their patterns of interaction with other nodes.

In the second and third stages of this part of the analysis the data are re-arranged and tentatively partitioned into parts which will correspond roughly to the final group structure.

Re-arranging the Data to Make the Groups Visible

This part of the analysis is essentially a refined version of the algorithm that was presented in 1971. (1) What is being done

can be understood easily with the following analogy. Imagine the nodes to be like billiards balls scattered about in space. Imagine there to be rubber bands connecting the balls corresponding to nodes with links between them. Imagine there to be springs between balls corresponding to nodes that do not have links between them.[4] The rubber bands will act to pull the balls connected to each other closer to each other, while the springs will push the balls not connected to each other apart from each other. If we hook up the rubber bands and springs and release the balls, they will re-arrange themselves so that the balls corresponding to nodes with links to each other will be close to each other, while the balls corresponding to nodes that are not linked to each other will be pushed away from each other. This example is shown in Figure 1.

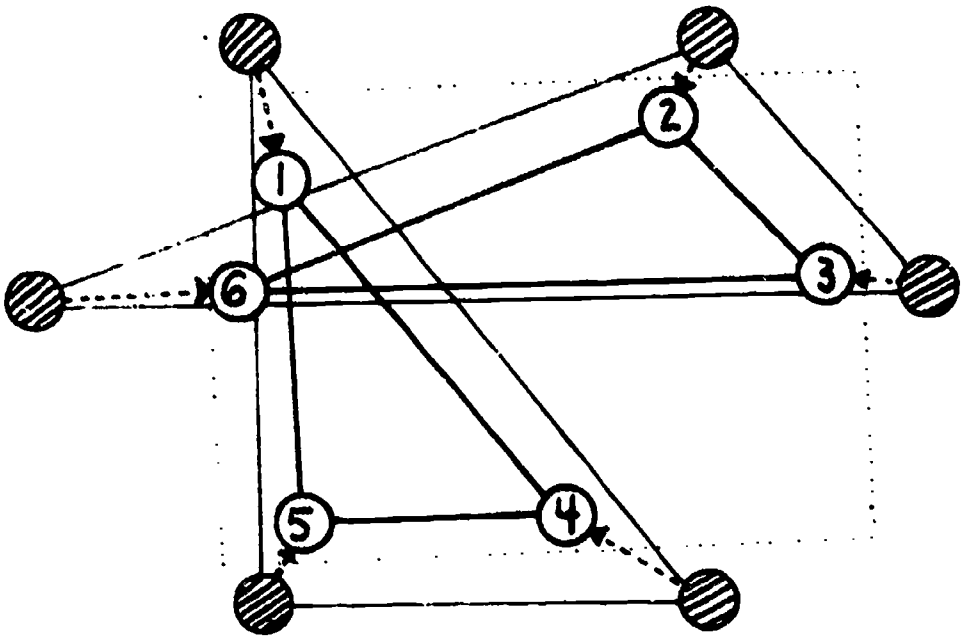
We could refine this technique by using heavier rubber bands to represent the links that occur more often or are more important. Since our objective here is to make it easier to identify groups, we could make the process work even better if we could make the rubber bands for within-group links heavier than the ones for other kinds of links. In order to do this, we will need some indicator that tells us which links look like within group links.

If two nodes are in the same group, they are likely to have many links to the same people. There is likely to be a high number of shared links, or two step links between this pair of nodes. If they are not in the same group, they are not likely to talk to the same people, and there are not likely to be many two-step links between the nodes. Thus, the number of two-step links is used as an indicator of the probability that the link is a within group link.

Figure 1

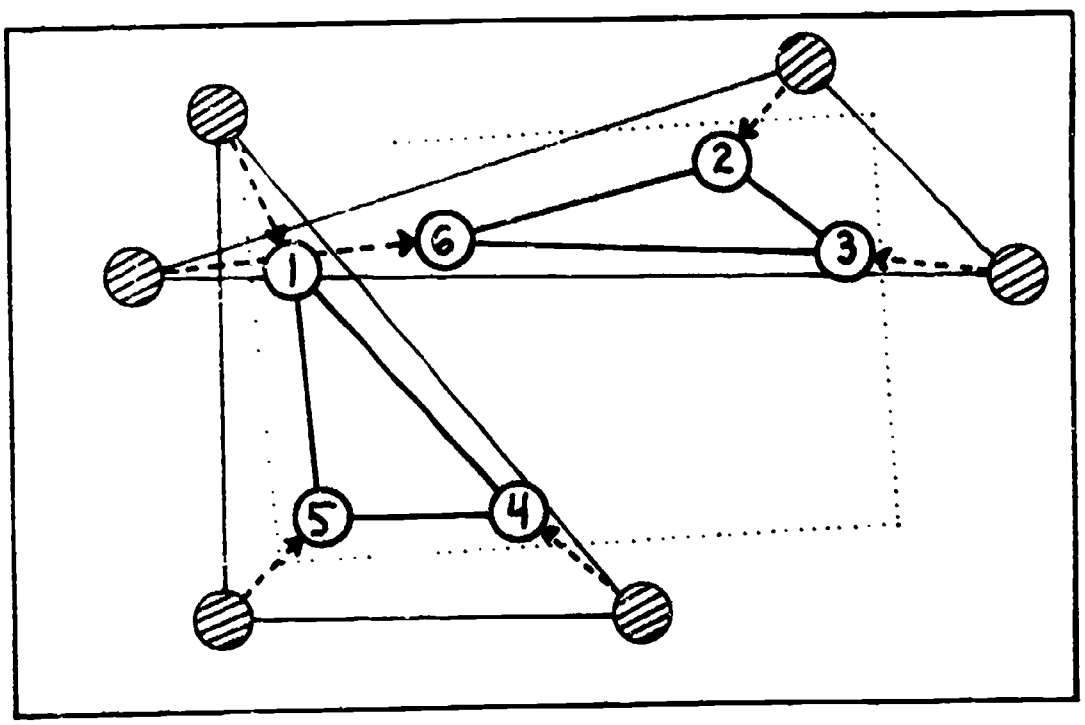
This figure illustrates the billiard ball and rubber band model described in the text. The network shown has two groups of three nodes each. The three drawings represent three successive increments of time, as the nodes move farther and farther in response to the forces exerted by the rubber bands.

The original position of the balls is shown by the shaded circles in the top drawing. Movement of balls during each time increment is shown by the dotted arrows in the three drawings. The scale was changed in going from the first to the second to the third drawing, in order to show smaller and smaller regions in space as occupying the same sized area in the drawings. The region of the top drawing shown in the middle one is indicated by the dotted box in the top. Similarly, the area of the bottom drawing is shown by the dotted box in the middle one.



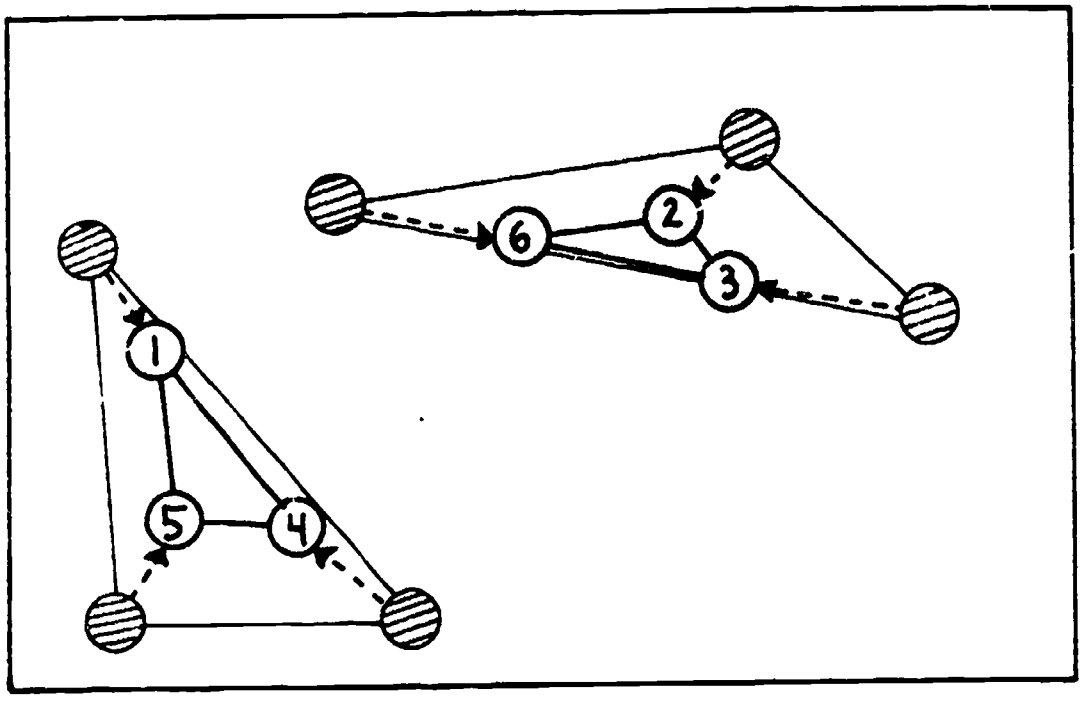
SHADED CIRCLES - NODES AT TIME 0

NUMBERED CIRCLES - NODE AT TIME



SHADED CIRCLES - TIME

NUMBERED CIRCLES - TIME 2



SHADED CIRCLES - TIME 2

NUMBERED CIRCLES - TIME 3

Figure 1

Now, it is hard to represent large numbers of points in multi-dimensional space. It takes a lot of information to do this, and it is fairly difficult to move objects in this kind of a space. Extensive experimentation with real data, however, showed that it was not necessary to use a multi-dimensional representation for this analysis; a single line segment was sufficient. This kind of reduction in complexity of representation greatly reduced the amount of information needed to perform the analysis at the same time it made the analysis itself easier to do.

The analysis is performed as follows: Nodes are scattered at unit points along a line segment N units long, where N is the number of nodes. We then treat each link from, say, node A to node B, as a vector, starting at A and pointing at B. We take all the vectors for each person and compute the average, weighting the individual vectors for strength of the link and probability that the link is a within-group link. We then get a single point for each individual, that point being the mean of that person's vectors. This is illustrated in Figure 2. After all the means have been computed, each node is moved to the point indicated by his mean.

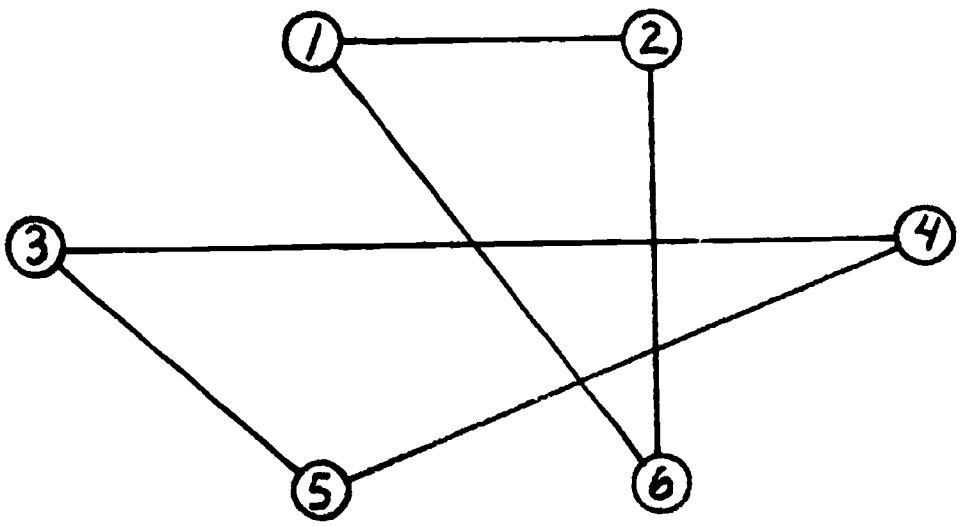
After this process has been completed, nodes with links to each other will be closer to each other than they were before. They will not, however, be as close as they could be. This fact is due to the way nodes are scattered initially, and also because of the statistical properties of the mean. For this reason, the entire process is repeated, using the new locations instead of the original positions used for the first set of calculations. A

Figure 2

At the top of this figure is shown a hypothetical network consisting of two groups, each of which has three members.

The diagram in the middle shows how the six nodes are initially placed along a line segment. The two solid arrows pointing to the right in the top of this figure are the vectors representing the links of Node #1 to Node #2 and Node #6. The dashed arrow between the solid ones is the average of the two. Below the line segment are shown the vectors for the links of Node #6.

The diagram on the bottom of Figure 2 shows how the iterative process of vector averaging works. The first line shows the initial positions of the six nodes. The second shows what the means could look like. Moving from the second to the third lines, the scale has been expanded so that the nodes range over the entire length of the continuum. The fourth and sixth lines show the second and third sets of means; while the expanded versions are shown on the fifth and seventh lines. (Note that the values shown are not the actual values that would be obtained for this particular network; they are intended merely to illustrate how the process might typically look.)


1
2
3
4
5
6

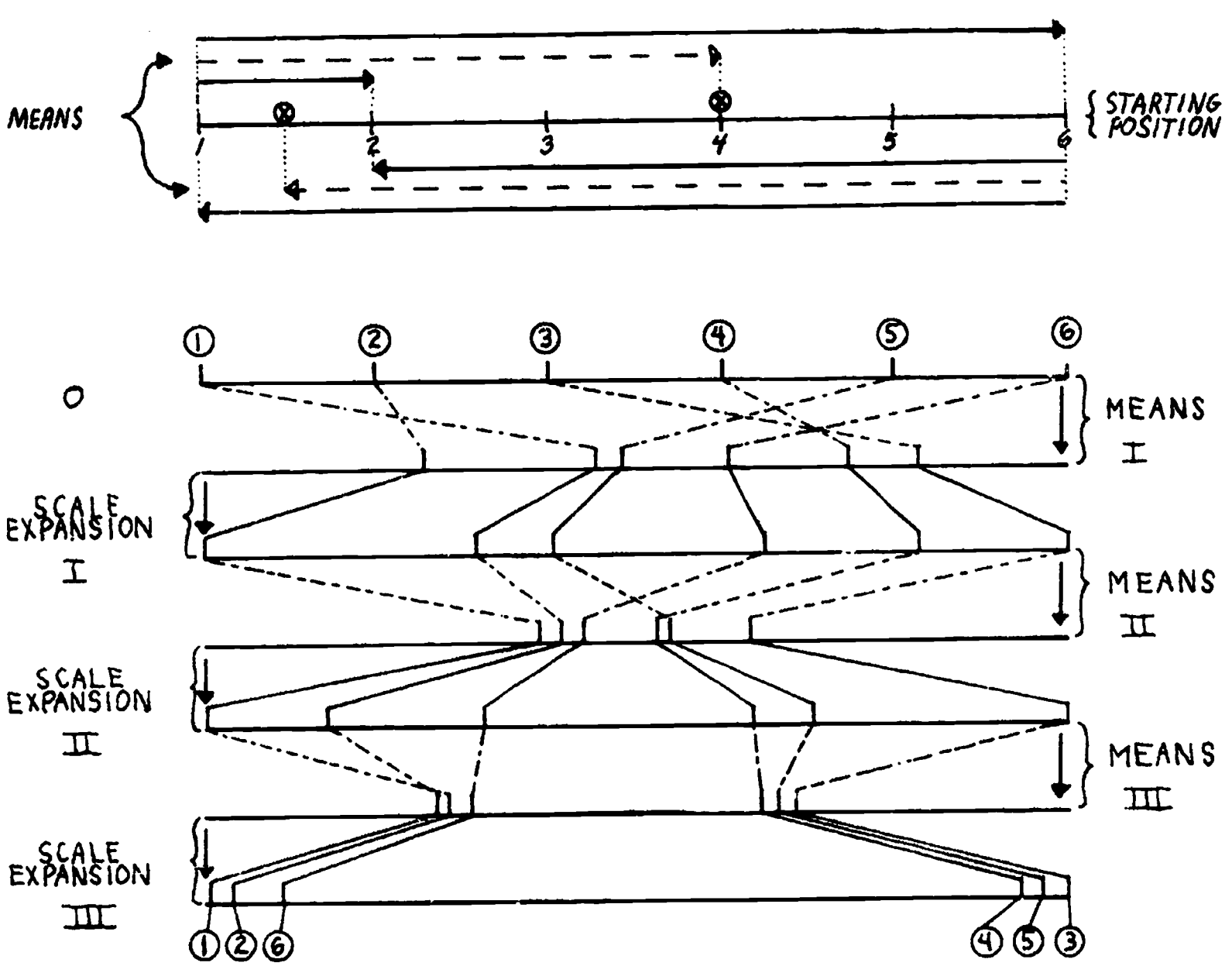

MEANS
1
2
3
4
5
6
STARTING POSITION
0
SCALE EXPANSION I
SCALE EXPANSION II
SCALE EXPANSION III
MEANS I
MEANS II
MEANS III
1 2 6
4 5 3

Figure 2

Plot showing how the nodes moved in successive iterations is shown in the bottom half of Figure 2. Between each set of calculations it is necessary to expand the scale of the continuum so that the spread or range which is occupied by the nodes remains N units long. If this is not done, the points will move closer and closer to each other, finally collapsing on a single spot. This is the "scale expansion" referred to in Figure 2.

The formula used for calculating a person's mean is shown here:

$$M' = \frac{\Sigma\,(wf_i \cdot S_i \cdot M_i)}{(wf_i \cdot S_i)}$$

where wf_i is the two-step weighting factor described above; S_i is a ratio-level indicator of the strength of the link; and M_i is the old mean of the person to whom the link goes. The summation is done as i goes from 1 to ℓ, where ℓ is the number of links that the individual whose mean we are calculating has.

In the development of this algorithm different numbers of iterations; different ways of varying relative contributions of wf_i's, S_i's, and M_i's; and different ways of assigning the original M_i's were tried. In general, four to six iterations seemed to be sufficient for any data set that was examined. If nodes are given subject numbers running from 1 to N, where N is the number of nodes, and these subject numbers are used as the first approximation for the M_i's, the process seems to work well for all types of data. In actual tests, when different subject numbers were assigned to individuals, the solution obtained was identical to the first soluti which indicates that the process is not terribly sensitive to the original positions. Usually, the wf_i's and S_i's are given equal

weight, although this has not been tested extensively.

The result of the application of this process is a continuum, N units long, with a scattering of nodes along its length. A sample network, together with the continuum that might result, is shown in Figure 3. This continuum is used as the input to the next stage of the analysis, in which tentative boundaries for groups are drawn.

Drawing the Tentative Boundaries

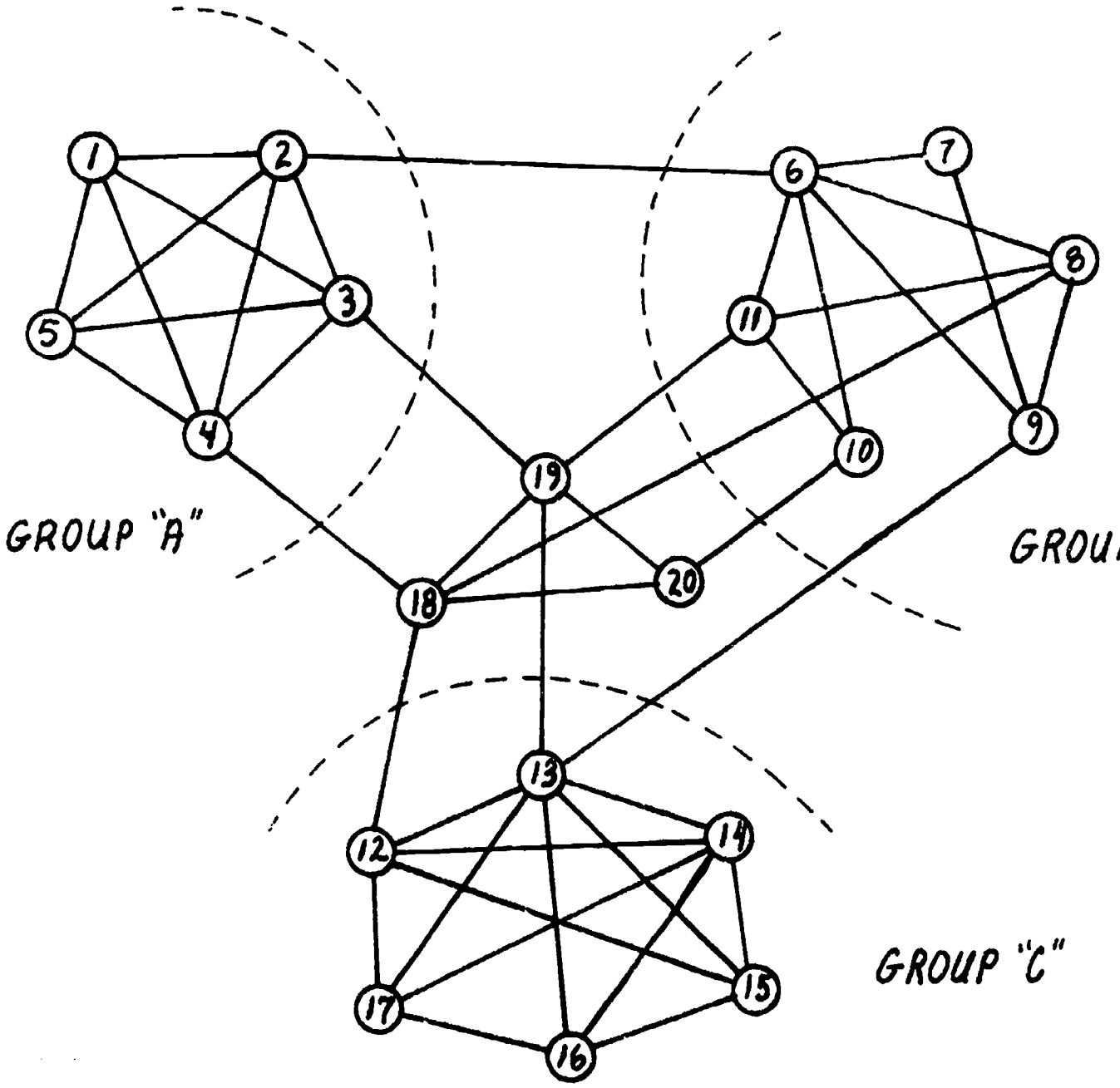
For any human observer, even a casual glance at Figure 3 will be enough to suggest that there are three clusters of nodes. The computer, however, must be told what a cluster looks like, and how to look for one. People probably identify a cluster as an area in which there are a lot of nodes, surrounded by areas in which there are fewer nodes. This is essentially what we have the machine look for.

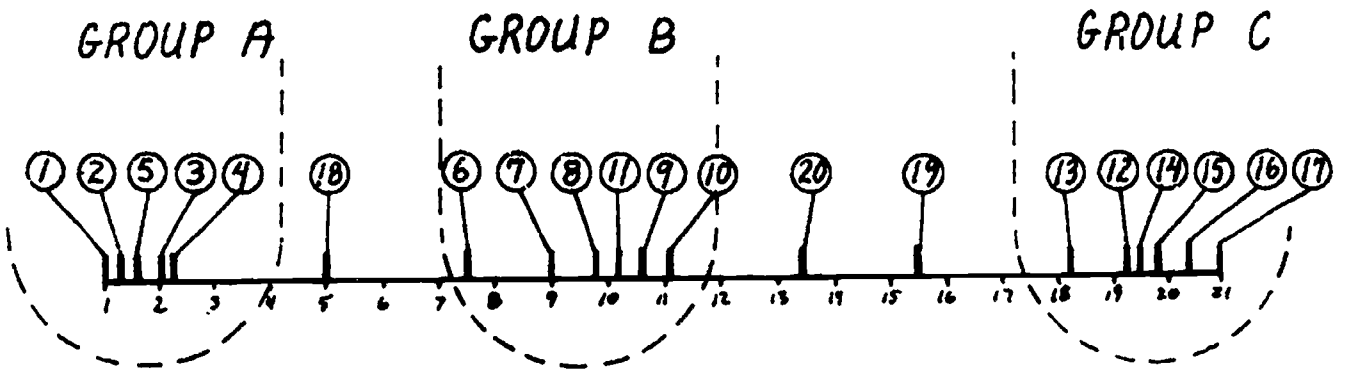
We will need a plot of the "density" of nodes along the continuum. In order to get such a plot, we construct a "window" and move it along the continuum, counting the number of nodes visible through the window at each point. This is shown at the top of Figure 4. Extensive testing has lead to the conclusion that the most efficient way to proceed is to center the window over each node, rather than to "slide" it gradually down the continuum. The optimum size of the window, also determined by experimentation, appears to be about two units on an N unit line. Windows smaller than this introduce spurious statistical information, while with windows larger than this, group boundaries tend to blur and merge into indistinction. This is shown in Figure 4, where density plots

Figure 3

The top of this figure shows a hypothetical network composed of twenty nodes. Group boundaries are indicated by the dashed lines.

The bottom shows what the final continuum might look like for the network shown in the top. Again, the group boundaries have been indicated by dashed lines.





THE CONTINUUM

Figure 3

Figure 4

This figure shows how the density plot is made. The example uses the continuum shown in Figure 3. In the top part, the window is shown, centered successively on the first eight nodes.

The three bar graphs in the middle show the effects of differently sized windows.

On the bottom is shown the refined version of the plot, with numbers of nodes visible to the right of the center of the window plotted above the horizontal and numbers visible on the left of the window plotted below the horizontal.

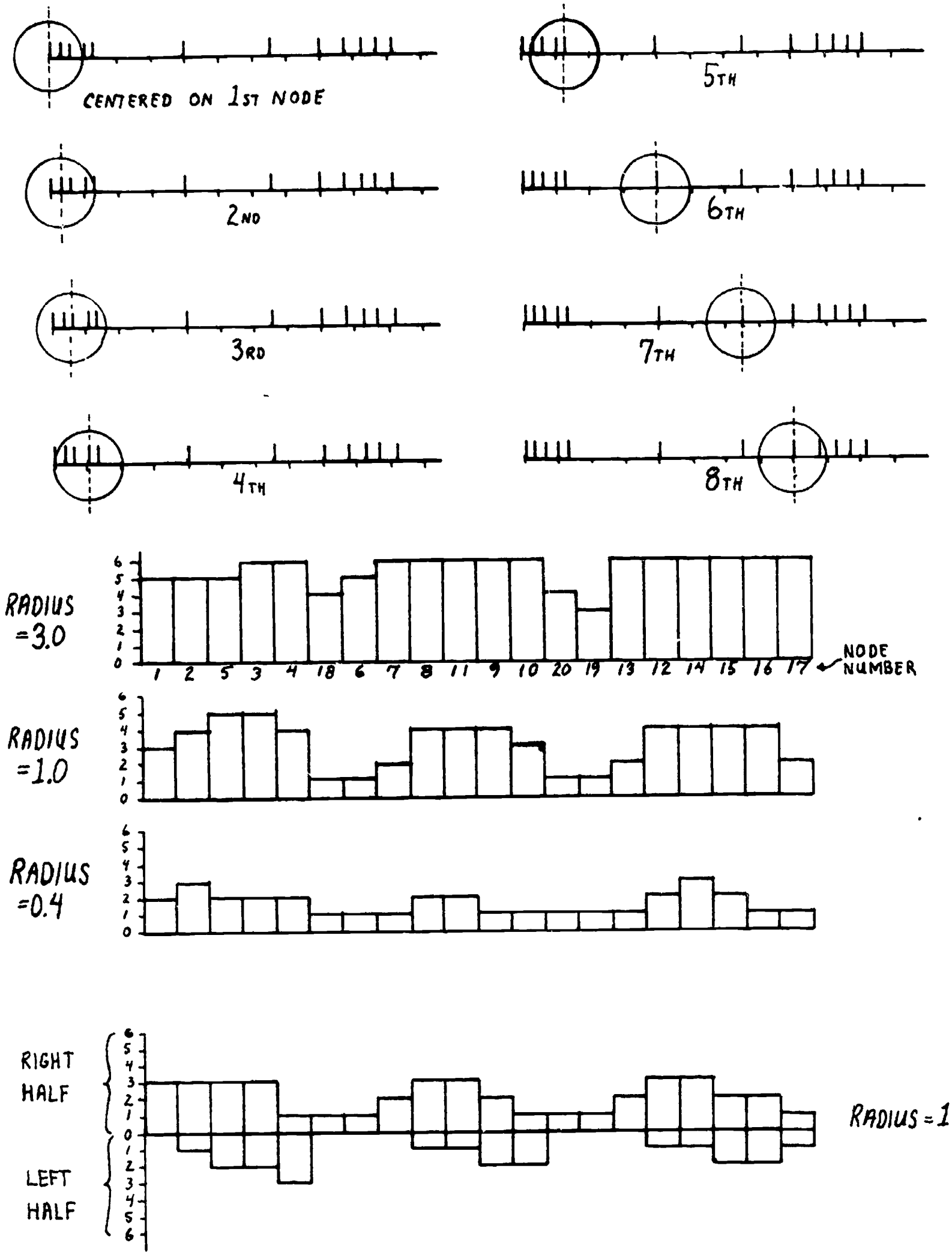

CENTERED ON 1ST NODE
2ND
3RD
4TH
5TH
6TH
7TH
8TH
RADIUS =3.0
RADIUS =1.0
RADIUS =0.4
NODE NUMBER
1 2 5 3 4 18 6 7 8 11 9 10 20 19 13 12 14 15 16 17
RIGHT HALF
LEFT HALF
RADIUS = 1

Figure 4

appear for windows of varying widths.[5] The result of moving the windown down the continuum will be a list of densities, with one value for each individual. Such a list could be represented as a bar plot like the one shown in Figure 4.

With this representation, groups will look like mounds, with boundaries between groups being indicated by low points. Although it seems as though this representation would be adequate, there arose problems which lead to an improvement over this simple plot. Although the problems will not be discussed here, the improvement will: Instead of just counting the number of nodes visible through the window, two numbers are counted--the number visible on the right half of the window, and the number visible on the left half. When constructing the bar graph, the number visible on the right half is plotted above the horizontal, while the number visible on the left half is plotted below the horizintal. The result is shown at the bottom of Figure 4.[6]

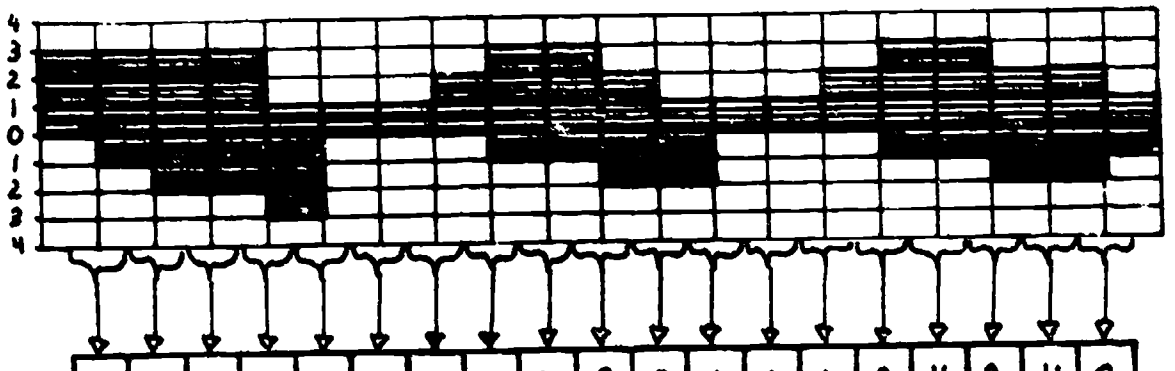
The final step in this stage is to have the computer draw lines around the groups. The way this is done is by locating spots at which there is a large change as we move from one point on the continuum to the next. If we count the number of non-overlapping points and divide by the number of overlapping points for each pair of adjacent nodes on the final bar plot, we will have a fairly sensitive indicator of group continuity. This is shown in Figure 5. High values for this ratio will indicate that there is a large change as we move from one node to the next. Low values, on the other hand, will indicate that there is only a small change. If we choose a cutting point, and instruct the computer to draw a line whenever the ratio goes above the cutting point, we will have

Figure 5

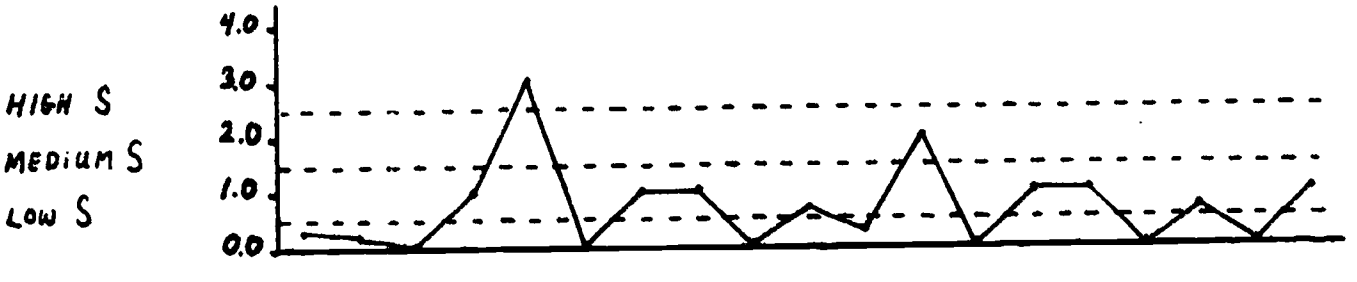
This figure illustrates the boundary-drawing process. The density plot on the bottom of Figure 4 is shown on the top of this figure. The table below the plot shows the number of overlapping points, the number of non-overlapping points, and the ratio of the two numbers; for each successive pair of bars on the bar plot.

The ratios are plotted in the graph in the middle of the page. The three dotted lines show the three different cutting points.

Below the ratio plot, the original continuum is shown three times. The first shows the effect of a high cutting point, while the second and third ones show the results for moderate and low values of the cutting point.



OVERLAP	3	4	5	3	1	1	1	2	4	3	3	1	1	1	2	4	3	4	2
NON-OVERLAP	1	1	0	3	3	0	1	2	0	2	1	2	0	1	2	0	2	0	2
RATIO	.3	.2	0	1.	3.	0	1.	1.	0	.7	.3	2.	0	1.	1.	0	.7	0	1.



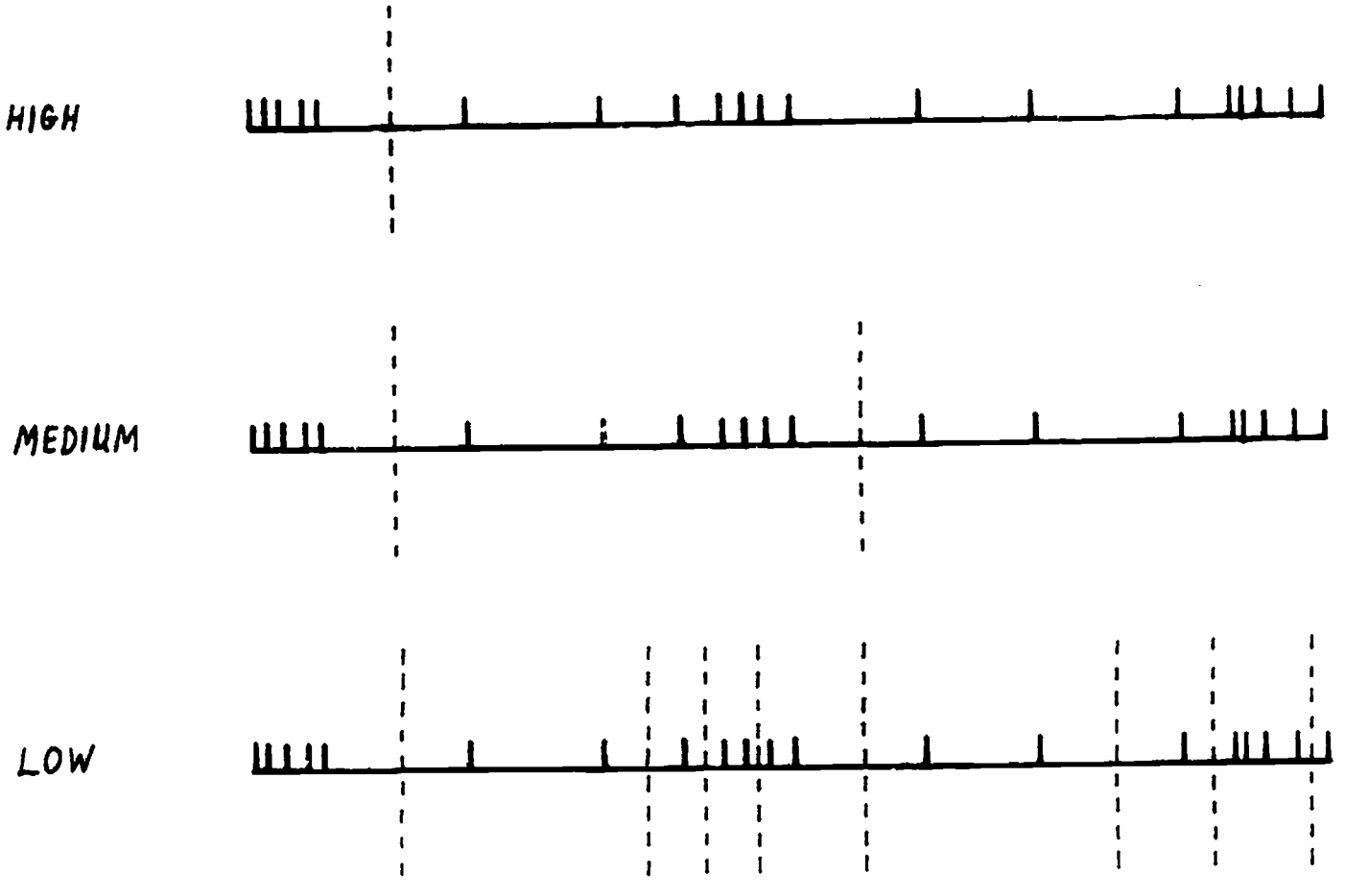


Figure 5

told the computer how to draw the boundaries around groups. If the value of the cutting point is variable, we can alter the sensitivity of the group spotting routine in either direction. With a window of two units, a cutting point of 1.0 appears to be optimum for most networks. Different values, along with the results, are shown in Figure 5.

This concludes the approximate phase of the analysis. The result of this stage is a list of tentative groups of nodes. The next part of the analysis involves the testing of this tentative solution, and any alteration that may have to be done to "clean it up."

Using the Criteria for an Exact Solution

This part of the analysis can be divided into two parts. In the first, individual nodes are tested to see if they meet the relevant criteria for their role in the network. If they do not, the appropriate changes are made. In the second, whole groups are tested for the criteria that are relevant at that level. Again, appropriate changes are made if necessary. We begin with the individual testing, which is very simple.

Individual Testing

First, people not in groups are tested to see if they meet the α-criterion for either liaison or group membership in any group. If any individual does meet the criterion, he is reclassified on that basis. If the individual fails both tests, he is labelled as "type other".

Second, members of groups are tested to see if they meet the α-criterion for group membership. Again, if the criterion is not met the appropriate changes are made.

Because changes made at any point in time can affect the roles of other people who were tested earlier, the tests are applied twice, to make sure that the final classification will be consistent with itself.

Group Testing

In this section we change our level of analysis to whole groups, rather than separate individuals. The criteria to be tested in this part are the connectiveness and critical link/node criteria. Since the information generated in the testing of the connectiveness criterion is necessary in the testing of the other two, it will be covered first.

The basic device used in the testing of these criteria is the distance matrix, which is constructed for each group. In this n by n matrix (n is the number of members in the group), the element in row i, column j gives the number of steps needed to get from individual i to individual j in the group. If there is some finite number in each element of the matrix, the group will be connected. This means that there will be some path from each individual in the group to every other individual in the group. The longest any path could ever be is n-1 steps. A sample network, together with its distance matrix is shown in Figure 6.

The way the distance matrix is constructed is as follows.

Figure 6

At the top of Figure 6 is shown a hypothetical eight-node network. The matrix directly below the network is a binary version of the network. In this matrix, each node has a row and a column. The i,j entry of the matrix is 1 if node i is linked to node j.

The second matrix is the distance matrix for the same network. The entry in the i,j element of the matrix is the number of links in the shortest path from node i to node j.

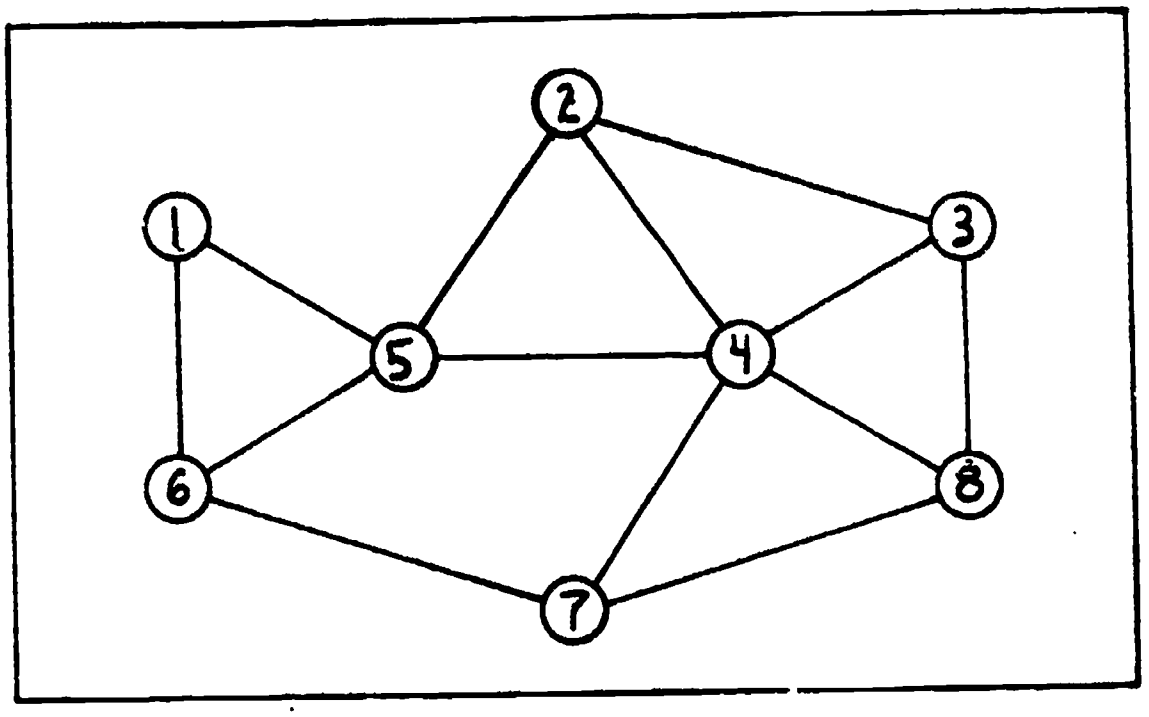


Figure 6

BINARY FORM

	1	2	3	4	5	6	7	8
1	0	0	0	0	1	1	0	0
2	0	0	1	1	1	0	0	0
3	0	1	0	1	0	0	0	1
4	0	1	1	0	1	0	1	1
5	1	1	0	1	0	1	0	0
6	1	0	0	0	1	0	1	0
7	0	0	0	1	0	1	0	1
8	0	0	1	1	0	0	1	0

DISTANCE FORM

	1	2	3	4	5	6	7	8
1	0	2	3	2	1	1	2	3
2	2	0	1	1	1	2	2	2
3	3	1	0	1	2	3	2	1
4	2	1	1	0	1	2	1	1
5	1	1	2	1	0	1	2	2
6	1	2	3	2	1	0	1	2
7	2	2	2	1	2	1	0	1
8	3	2	1	1	2	2	1	0
	1	2	3	4	5	6	7	8

A matrix is constructed in which there is a row and a column for each node in the group. All the elements are initialized to zero. Whenever there is a link from node i to node j we enter a 1 in row i, column j. If the link is reciprocated we also enter a 1 in row j, column i.**

We then repeatedly perform a boolean logic operation which is analogous to raising the matrix to successively higher and higher powers. Instead of entering the cross product of the ith row and the jth column as the i,j element in the product matrix, however, we enter the first power on which this value becomes non-zero. (This operation is performed with a series of nested DO-loops and IF statements in FORTRAN. With careful organization, the process can be optimized to take significantly less time to compute than a standard algebraic multiplication of matrices.)

We stop raising the matrix to higher powers when one of two conditions obtains: either a) all off-diagonal elements become non-zero, which implies the group is connected; or b) when going from any power k to the next power k+1 no entries change value, which implies the group is not connected at level k and will never be connected at any level.[7]

If the group is not connected, it is split into a connected part and all the rest. Each of the two parts is then treated as a separate group, and subjected to all the tests that any group must undergo.

At this point, there are only the critical links/nodes criteria remaining to be tested. These criteria serve as checks against situations like those shown in the bottom half of Figure 7, where two groups have been mistakenly identified as one. This situation

is generalized to include situations in which there are any number of multiple groups, connected in some relatively minimal way, which we wish to separate into distinct groups. The occurrance of these confusions is a result of the inelegance of the approximate techniques used in the first half of the analysis. For analytic purposes, it is practical to combine these two criteria into a single rule which says that no subset of some arbirtary size may be removed from a group and cause the group to become disconnecte If there is such a subset, the group will be seen to be "really" two or more groups.** As a result of this combination, whenever two groups are joined by a bridge link (a link between members of different groups), one of the nodes of this link will be identified as a liaison. That node will later be tested for the α-criterion of group membership, and if he passes, will be returned to his group.

The problem has thus been reduced to one of identifying any critical nodes which may exist in a group. If there is one, he will be the node with the lowest average distance from all other nodes. This is because all paths from nodes in either half of the group to the other half must go through the critical node. The average distance from any node to all the other nodes is given by the average of all the entries in that node's row in the distance matrix. This is illustrated in Figure 7. If there is a set of critical nodes, they will be the nodes with the smallest row means.

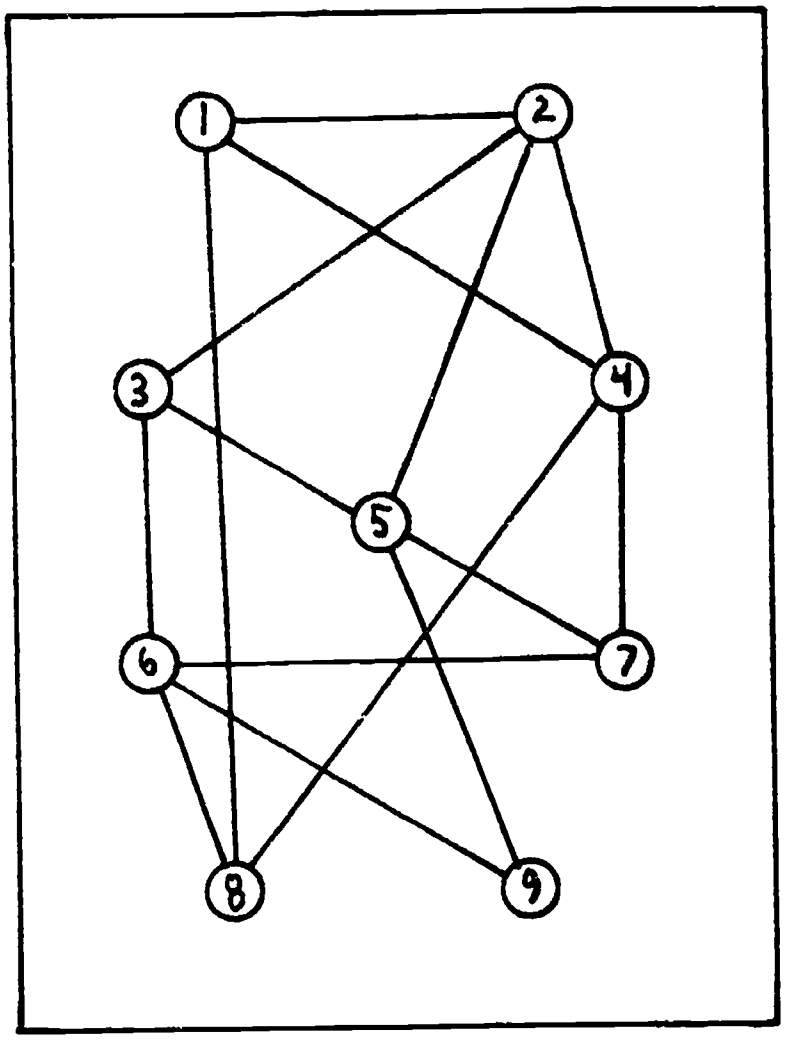
The fact that critical nodes have lower row means than the other members suggests that there must be some variation in the row means if there are any critical nodes. We can take advantage of

Figure 7

On the upper left-hand corner of this figure is shown a hypothetical nine-member network. To the right of this is the distance matrix for that network. The rightmost column of the matrix contains the means of the rows of the matrix. The values in this column are thus the mean number of steps it takes that node to reach all other nodes. The overall mean for the group, together with the standard deviation of the distribution of means, is shown below the matrix.

The network in the bottom left-hand corner is an example of the kind of situation that occurs when two or more groups are identified as a single group. Clearly, Node #5 is a liaison between the two groups. The middle matrix on the right half of the page is the distance matrix for this group. Note the relatively high standard deviation for this group, compared to the one above it.

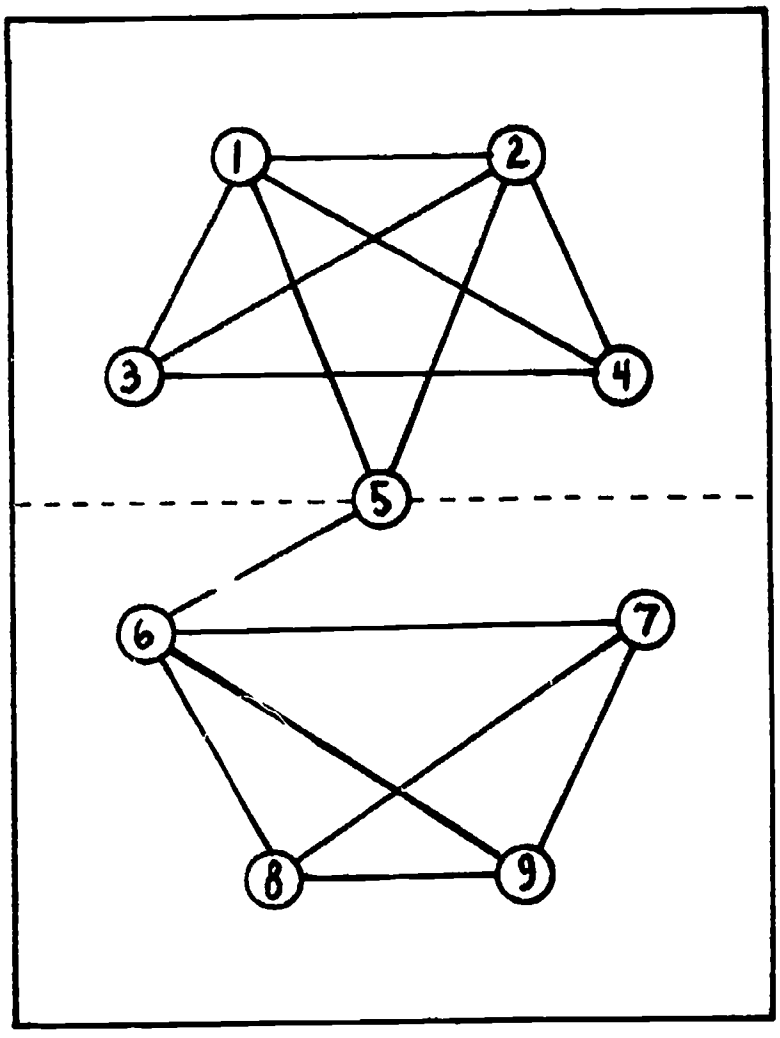
The third matrix was constructed after removing node #5. Note that there are no values for many of the elements, indicating that the group is no longer connected. The means shown for this bottom matrix are the values that would be obtained if the group were split in two, and the means for each group calculated separately.



	1	2	3	4	5	6	7	8	9	
1	0	1	2	1	2	2	2	1	3	1.75
2	1	0	1	1	1	2	2	3	2	1.62
3	2	1	0	2	1	1	2	2	2	1.62
4	1	1	2	0	2	2	1	1	3	1.62
5	2	1	1	2	0	2	1	3	1	1.62
6	2	2	1	2	2	0	1	1	1	1.5
7	2	2	2	1	1	1	0	2	2	1.62
8	1	3	2	1	3	1	2	0	2	1.87
9	3	2	2	3	1	1	2	2	0	2.0

MEAN=

S.D. =



	1	2	3	4	5	6	7	8	9	
1	0	1	1	1	1	2	3	3	3	1.87
2	1	0	1	1	1	2	3	3	3	1.87
3	1	1	0	1	2	3	4	4	4	2.5
4	1	1	1	0	2	3	4	4	4	2.5
5	1	1	2	2	0	1	2	2	2	1.62 ←
6	2	2	3	3	1	0	1	1	1	1.75
7	3	3	4	4	4	1	0	1	1	2.61
8	3	3	4	4	4	1	1	0	1	2.61
9	3	3	4	4	4	1	1	1	0	2.61

MEAN =

S.D.=

	1	2	3	4	5	6	7	8	9		
1	0	1	1	1	–	–	–	–	–	1.0	"A"
2	1	0	1	1	–	–	–	–	–	1.0	
3	1	1	0	1	–	–	–	–	–	1.0	
4	1	1	1	0	–	–	–	–	–	1.0	
5	–	–	–	–	–	–	–	–	–		
6	–	–	–	–	–	0	1	1	1	1.0	"B"
7	–	–	–	–	–	1	0	1	1	1.0	
8	–	–	–	–	–	1	1	0	1	1.0	
9	–	–	–	–	–	1	1	1	0	1.0	

Figure 7

of this fact if we only look for critical nodes when there is some variance. It turns out that this leads to a large saving, in terms of computation time. This is because of the way we test for critical nodes.

To check a node to see if it is critical, we remove it from the group and re-calculate the distance matrix. If, as a result of the removal, the group becomes disconnected, we have found a critical node. If the group is still connected, we try the next candidate--the node who, of all the remaining nodes, has the smallest row mean. We will usually stop this process after taking out some percentage of the original group (usually ten) if the group continues to remain connected. If this happens, we put all the removed nodes back into the group.

It is easy to see that there is a lot of work involved in the searching for critical nodes. This is why the heuristic device of checking the variance of the row means is so important. In every network that has been examined so far, this heuristic has worked correctly. That is, it did not prevent any critical nodes from being found. Similarly, the approach of looking at nodes with the lowest row means always seems to find the critical nodes. The optimum value to use as a cutting point for the variance test seems to be about 0.3. Whenever the standard deviation of the row means exceeds this value, there is likely to be a critical node. Whenever the standard deviation is less than this value, there is not.

After all groups have passed these tests, the obtained classification of nodes to groups and other roles will be exact. At this

point various indices may be calculated and the results tabled in any convenient manner. A flow chart of the algorithm is shown in Figure 8.

Part Three--Programming Considerations

This section will discuss several aspects of the analysis technique that are relevant to the actual coding of a program to perform the analysis. Some of these involve programming approaches which make the program both more powerful and easier to write, while others include programming "tricks" that greatly increase the efficiency of the program.

Programming Approaches--General Considerations

FORTRAN seems to be a good language to use for this type of program. The logical structure of FORTRAN is sufficiently powerful to handle the logic of the analysis, and the relative efficiency of FORTRAN makes the large amount of arithmetic affordable. In addition, FORTRAN is widely available, easily learned, and easily written. Finally, the only operating version of the program was written in FORTRAN, and it would seem to be easier to do another program in the same language, rather than a different one.

The other major general consideration involves internal data representation. The data should be stored in the form of variable length lists, rather than matrices. The use of the matrix format will greatly limit the capacity of the program, will make it prohibitively expensive to run, but will greatly simplify the programmer's task. To utilize a list processing

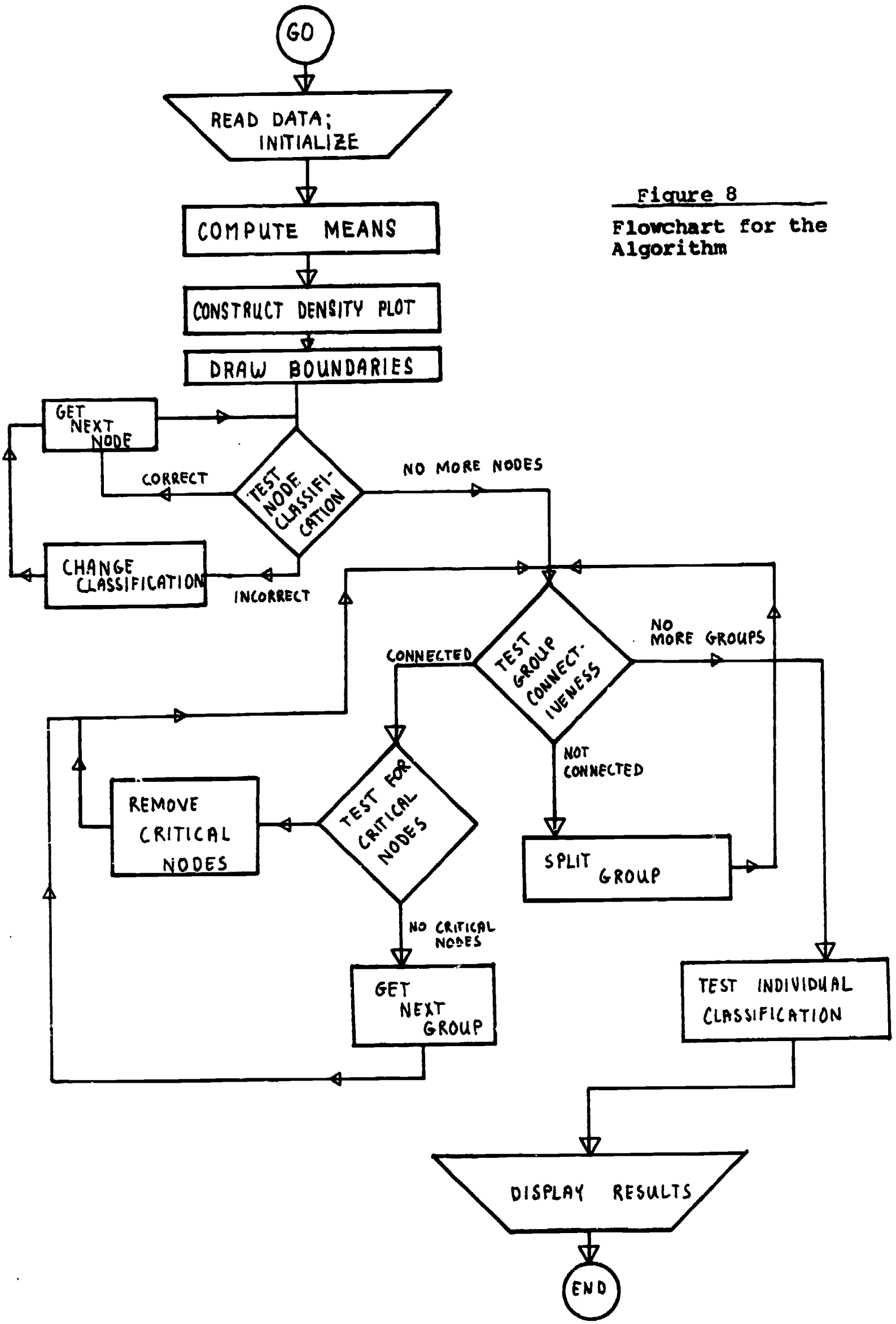


Figure 8

Flowchart for the Algorithm

approach in FORTRAN is not difficult. A set of standard statement functions will handle most routine list-processing procedures with ease.

Careful organization of logical steps is crucial if an efficient program is to be written. This is especially true in an algorithm as complex as this one, where huge numbers of decisions must be made about vast amounts of information. The programmer should expend considerable amounts of time deciding how to organize both the data he is working on and the operations he is performing on the data. In general, everything should be modularized, standardized, and clearly organized. This means:

a) The algorithm should be broken down into logical steps.

b) All notation--i.e. variable names, statement numbers, logical organization, etc.--must be consistent throughout the program.

c) Extreme care must be taken to clearly organize the code. That is, code should be "clean", concise, and "nice". If a section "feels clumsy" there is probably a better way to do it, and it is worth looking for a better way.

Documentation is essential. It is probably not possible to write a program for this algorithm without extensive documentation. This means flow charts, verbal descriptions of logic flows and objectives, memory use maps, and comment cards are all essential.

Programming "Tricks"--Optimization

There is a lot of room to optimize in this kind of program. Careful attention to organization at all levels of analysis will reveal parallel logic processes which could be handled by the

same section of code, for example. There are many points at which a single recursive loop of code, although more difficult to write than a sequential series of instructions, will perform an operation faster and more efficiently.

All list searching can be optimized if the lists areordered in such a way as to minimize mean search time. The use of indirect addressing in most of these kinds of situations can make many search operations almost automatic.

Finally, the use of a CDC machine (or one similar to it) with its more powerful version of Extended FORTRAN, rather than an IBM with standard FORTRAN G or H, will make significant differences in the ease with which the program is written, as well as the actual cost of running the final program.

Because different computer installations have different ways of doing certain things, it is not possible to be more specific with any of these comments.[8]

Part Four--General Characteristics of the Network Analysis Program

The only implementation of the algorithm, as of this date, is an Extended FORTRAN program which operates on the CDC 6500 at Michigan State University. The code for the entire program including data cleaning routines and complex table-producing routines, occupies approximately 2,300 computer cards. The program is stored in the form of a compiled version of the most recent revision, eliminating the compilation cost each time the program is run.

The code alone occupies $31{,}000_8$ 60-bit words of core,

leaving about $130,000_8$ words for data. This gives a capacity of 4,095 nodes and 32,767 links. Execution time increases linearly as a function of network size and complexity. Some times appear below.

Number of links	Number of Nodes	Execution Time
2000	725	57 seconds
1000	270	68 seconds

Execution time is a function of network complexity, as well as the absolute size, as can be seen above. The printout describes the network in great detail, and print charges usually exceed compute charges by a factor of about four. The printout for a network of 1,000 nodes is three to five inches thick.

The program is well protected against many user errors and odd data configurations. It has been tested on random data, and it performed as expected. (9) The largest network ever run was with an N of 960. Estimates suggest that to do this analysis by hand would take ten tireless errorless men over a century. It took the computer less than two minutes.

Part Five--Historical Development

Work was begun in this area at MSU in late 1970. The first working program, operationalizing parts of the algorithm described in (1) was completed in late 1971. The output of that program was a large matrix, with rows and columns arranged so that people who talked to each other were close to each other in the matrix. In June of 1972 the analytic methods used were refined and extended to include the group detection routines.

These were further refined and extensively tested in Summer and Fall of 1973.

In the time from 1972 to the present, the program was continuously being improved, as better and better methods of problem solving were discovered. In addition, errors of various types were tracked down and fixed. The theoretical basis on which the program stands was being refined at the same time improvements were being made on the program.

At this point in time, we know of no significant errors in the program itself, although users of the program sometimes make errors when running their data through it. Attempts are being made to include routines in the program which will identify even these kinds of error, thus protecting the user from himself.

NOTES

1. William D. Richards is a graduate student at Stanford's Institute for Communication Research.

2. In the past, some investigators have limited the number of links per node to some constant, like three or four. There is no reason to do this, as it severely distorts the data.

3. Many of the most interesting properties of networks don't seem to be found in small simple systems. We have seen several moderately large networks which show qualitatively different properties than smaller ones. Perhaps very large networks will be different in similar ways.

4. This analogy may have been suggested by James A. Danowski, a colleague of mine who has provided much invaluable assistance throughout the development of this methodology.

5. A major problem with the examples used to illustrate the different parts of the algorithm is that they are too simple to accurate mirror the kinds of things that happen with real data. This simplicity was felt to be necessary, if the examples were to be clear.
The size of the window is given a good information theoretical treatment in a paper by Gauthier. (10)

6. With real networks, the bar plot is much more complex and rich in detail. There are usually more nodes in each group, and the shapes of the groups in the bar plot is strikingly different from the ones seen in Figure 4. However, the ones shown there do illustrate the concept being put forth.

7. The author was not able to devise a proof for this theorem, so he tested it empirically with a large number of examples. It never received any disconfirming evidence, and if a T-test were done on the results, the significance level would be with p less than 0.00001. He is therefore confident in the truth of the theorem.

8. The author suggests that he is not primarily a computer programmer and regrets that his research interests prevent him from learning the peculiarities of other computer systems, which would allow him to be of more assistance to others working in the area. He is, however, willing to discuss any problems

that may be encountered in attempting to program the algorithm.

Acknowledgements

Professor R. Vincent Farace offered valuable moral assistance over the course of many months of difficult programming. James Danowski provided valuable assistance with the day to day aspects of the task. The Department of Communication at MSU funded the work, and my own department at Stanford allowed me the freedom to work in a strange area at another place for long periods of time. People who I didn't mention here but also deserve thanks know who they are, and I thank them. Finally, thanks to Control Data Corporation, for developing a fine computer.

References

1. An Improved Conceptually-Based Method for Analysis of Communication Network Structures of Large Complex Organizations. 1971, ICA convention in Phoenix. ERIC document # 064 876. William D. Richards, Jr.

2. Network Analysis in Large Complex Systems: Theoretical Basis. Paper presented to 1974 ICA Convention in New Orleans. William D. Richards, Jr. Institute for Communication Research, Stanford.

3. Network Analysis in Large Complex Systems: The Network Analysis Program--Version 3.2. Paper presented to ICA Convention in 1974 at New Orleans. William D. Richards, Jr. Inst. for Comm. Research, Stanford.

4. A Comparative Analysis of Communication Networks in Six Organizations. Paper presented at 1974 ICA Convention in New Orleans. R. Vincent Farace and J. David Johnson, Dept. of Communication, Michigan State University.

5. Network Analysis in Large Complex Systems: The Nature of Structure. Paper presented at 1974 ICA Convention in New Orleans. William D. Richards, Jr. and Mark Steinberg. Mich. State Univ. Dept of Communication.

6. Network Analysis in Large Complex Systems: Metrics. Paper presented at 1974 ICA Convention in New Orleans. William D. Richards, Jr. Inst. for Comm. Research, Stanford.

7. The Nature of Random Structures. Unpublished paper by George A. Barnett. 1973. Dept. of Communication, Mich. State Univ.

8. A Quantitative Measure for Structure, Control, and Learning. unpublished paper by Richard Gauthier. Dept. of Psychology, Stanford University. 1973.