Pattern Recognition: AMOEBA Extension

Honors Program

1

4/6/91

Charles Phipps
University Undergraduate Fellow

ABSTRACTS

Descriptive: The general field of pattern recognition is discussed with emphasis on the development of AMOEBA, a two dimensional digital clustering program, into a three dimensional program. Moreover, a program analyzing phenyl rings from X-Ray Crystallographic analysis of organic molecules is discussed. Finally, the report analyzes the steps involved in writing a pattern recognition program.

Informative: Pattern recognition is the general scientific field dealing with computer aided analysis of information. Aspects of a computer, such as pipeline, data storage, processor, and mainframe, are described in this report. In addition, the general relationship between the researcher and the computer is explored using interactive techniques.

AMOEBA, a two dimensional digital digital clustering program written by Dr. Jack Bryant in 1972, is explained based on its unique five-point gradient technique. Moreover, the use of the Kth nearest neighbor method is discussed.

Phenyl ring identification is mentioned as an additional project used to train myself in writing a pattern recognition program.

Although the complete conversion of AMOEBA to a three dimensional digital clustering program was not accomplished, I managed to learn the necessary steps involved in developing such a program. These steps include, the choice of data units, the choice of variables, what to cluster, clustering criterion,homogenization of variables, similarity measures, algorithm and computer implementation, number of clusters, and interpretation of results.

The development of AMOEBA, as well as the development of the phenyl ring identification program, is delineated as per each step in the designing process. Starr is the current version of AMOEBA in three dimensions and is briefly discussed. Moreover, the phenyl ring program, Findp, is also discussed. Thus, two pattern recognition programs are studied within this report.

TABLE OF CONTENTS

APPENDICES

FOREWORD

The purpose of this report is to outline my progress in understanding the development of the three dimensional pattern recognition computer program AMOEBA. The scope of the report includes my understanding of the general scientific research field of pattern recognition, of the two dimensional digital clustering program AMOEBA, of Starr, the first attempt at extending AMOEBA,of Findp, the three dimensional program analyzing phenyl rings, and of the nine step process involved in writing a pattern recognition program.

I began working on this project last summer at Cornell University. The National Science Foundation provided me with a research grant in the form of the Supercomputing Program for Undergraduate Research(SPUR). SPUR was a one month program involving nineteen students chosen nationally. SPUR provided me with the basic supercomputing skills I need to do research in the field of pattern recognition. Cornell also gave me free access to their IBM 3090 Supercomputer. This has been a valuable resource with which to do research.

Moreover, Dr. Jack Bryant of the Texas A&M University Mathematics Department approached me last year with an opportunity to work for him. Under his guidance, I became involved in the Texas A&M University Undergraduate Fellows Research Program. In addition, I acquired an additional advisor in the Department of Chemistry, Dr. John Fackler, Dean of the College of Science.

Thus, my project will be sent to the National Science Foundation, Cornell University, the Texas A&M Honors Department, Dr. John Fackler, and Dr. Jack Bryant.

SUMMARY

This report deals with the progression of my understanding of the general scientific field of pattern recognition. In the report, I explain the basic concepts necessary for understanding a pattern recognition program. In addition, the nine major steps

involved in writing a pattern recognition program are discussed and applied to two programs. First, AMOEBA, a two dimensional digital program, is discussed as being converted to three dimensions in the form of Starr. Second, Findp, a program finding phenyl rings in an organic molecule is discussed.

My main problem in this research endeavor has been a shortage of time. AMOEBA is a very complex and long program. The conversion and development of AMOEBA could take years. However, I only have one year in which to complete the Senior Honors Thesis, which is the report requested by the Honors Department and the final report requested by Cornell. Thus, I must live with the realization that I was unable to accomplish that which I set out to do.

However, I did help Dr. Bryant develop a program which uses pattern recognition techniques to develop a program which finds phenyl rings. The data used for this program was collected from X-Ray Crystallographic analysis of organic molecules.

The planned procedure of conducting my research was simply to help Dr. Bryant add another dimension to his program. However, the accessibility of Cornell's 3090 was hindered by A&M's computer failures. Thus, unexpected technical failures delayed the completion of the AMOEBA conversion.

Thus, a knowledge of pattern recognition, of supercomputing, of programming, of carbon structures, and of research in general has been gained. In addition to writing Findp, Dr. Bryant and I developed several display techniques which will be used upon AMOEBA's complete conversion.

The costs of my research were largely paid by the National Science Foundation in the sum of $2,000. Moreover, the Honors department gave me a $300 budget. Thus, little or no expense has come out of my own pocket.

I recommend that research be continued on both Starr and Findp. I hope that once these programs are complete the general scientific community will have been improved.

DISCUSSION

Pattern Recognition: Although the complete conversion of the computer program AMOEBA to three dimensions was the original task, only an intermediate step was accomplished. I will elaborate on the larger scope of the project and then specify exactly what has been accomplished.

Pattern recognition is a general scientific field which involves computer aided data analysis. Data used for pattern recognition programs can range from the stellar to the molecular. Moreover, data can be either digital, composed of any real number, or binary, composed of ones and zeros. Each group of data used is termed a data set. Thus, different pattern recognition programs must be developed for the wide range of existing data sets.

The development of pattern recognition programs requires the knowledge of a few basics. For example, the way in which a computer uses data must be thoroughly understood if this process is to be manipulated. In essence, computers read commands and data in a linear string of arguments called the pipeline. Thus, a three dimensional data set will be read as a line first down one row, then another, and finally the logic will skip to another plane and repeat the process.

Another basic idea used in pattern recognition is the Kth nearest neighbor method. Imagine a point in space. Now imagine the points surrounding it. If calculations are performed to compare this point with its neighbor it is necessary to know two things: the value of the points, and the distance between them. Thus, if the analyst wishes to compare the point with its six closest neighbors, then the process would be termed the sixth nearest neighbor method.

In addition to understanding the logic sequence of a computer and the Kth nearest neighbor concept, my project requires an understanding of the way in which supercomputers operate. Supercomputers are computers that are capable of processing data at a rate many times faster than that of a normal or personal computer.

A supercomputer operates by dealing through a central processing unit or a mainframe. This central processing unit is like the human brain is to the body. Moreover, a supercomputer is connected to many other units which are able to perform tasks independent of one another. Thus, these individual units or processors are like human hands are to the brain. However, a supercomputer can have many more than just two processors.

Since my research problem could possibly deal with a staggering amount of data, the use of a supercomputer becomes necessary. Using a normal computer would take too long and require too much money. However, the use of a supercomputer entails dealing with special problems. For example, since many calculations will be occurring at the same time on different processors. How will the calculations be organized? How will the processors output be organized? How will the data be accessed.? These problems and more face the supercomputer user.

Actually, the identified supercomputer problems are easily solved with a little basic training. I received this training at Cornell University last summer under the National Science Foundation's Supercomputing Program for Undergraduate Research(SPUR). The SPUR program provided me with the essential skills necessary to access and use the Cornell IBM 3090 supercomputer. Moreover, they provided me with a cash stipend of over $1500 and with forty hours credit on the 3090.

Thus, with an understanding of a few simple concepts and techniques, pattern recognition can become an exiting field for practically any researcher.

AMOEBA: AMOEBA is a two dimensional digital computer program. The purpose of AMOEBA is to analyze any raw data set. AMOEBA should be able to produce a list of clusters found in the data. These clusters can then be analyzed by the programmer or user to understand more about the intended research. Thus, AMOEBA is a versatile research tool because it can be used by any researcher dealing with any type of two

dimensional data set. AMOEBA's history is summarized in Appendix 1.

What makes AMOEBA different from other two dimensional digital programs? AMOEBA uses a special five-point comparison technique. Once AMOEBA has gone through a data set and produced a cluster map, it takes five points from each cluster to represent the diversity within that cluster. Thus, AMOEBA reduces the problem of analyzing clusters to comparing sets of five points.

AMOEBA then takes five points and labels them so as to identify them with the cluster from which they were taken. Then the points are aligned based on increasing numerical value. AMOEBA then subtracts the fifth points value from the first and arrives at a difference or gradient measure for the cluster.

AMOEBA now has two values with which to categorize the identified clusters; the gradient, and the five point set. The program first separates the clusters into separate categories. Then, within each category, the clusters are classified further using the gradient. thus, each cluster is placed in a map. This map is called a cluster map.

Now that the data has been analyzed and categorized, the researcher can use his previous understanding of the data to help the computer label the clusters in the cluster map. For example, if AMOEBA had identified different geometric shapes, the researcher would then tell the computer what to call each shape. Thus, when the program runs again it will produce a labeled cluster map.

The labeled cluster map can only be developed when the programmer has some understanding of the data and is able to produce a cluster library within the program. Thus, in order for the computer to evaluate data the way a researcher would, it must be trained in recognizing clusters.

The value of AMOEBA comes from the new arrangements of data clusters it provides. This allows the researcher to identify new relationships and principles previously unnoticed. Thus, the substantive results are not the output of the computer, but the new

ideas prompted in the analyst's mind. The test version Starr will be discussed in Section V. In addition, Starr and its output is located in Appendix 2.

Carbon Rings: Organic chemistry is the study of groups of atoms containing the element carbon. Many techniques exist for the analysis of these carbon groups or organic molecules. One of which is X-Ray Crystallography.

X-Ray Crystallography is the technique involving the bombardment of organic molecules with X-Rays which are high energy electromagnetic radiation. The X-Ray Crystallographic machine collects deflected X-rays. Then, a computer evaluates the data and produces a digital map describing the electron density structure of the organic molecule. Electron density is simply a term used to define that space in which an electron is observed over a period of time.

Since the conversion of AMOEBA requires several trial runs Dr. Bryant and I decided to use X-Ray Crystallographic data as one trial run. What we did was design a program that would identify phenyl rings within an organic molecule. A phenol ring is simply a group of six carbon atoms connected to each other in a circle. Thus, a three dimensional cluster map may be produced to view the phenyl rings of an organic molecule.

Findp and its output is located in Appendix 3. Findp will be discussed more in Section VI.

Program Generation Process: According to Anderberg, the general rule in pattern recognition is that the development of a program involves nine basic steps. These are the choice of data units, the choice of variables, what to cluster, clustering criteria, homogenizing variables, similarity measures, algorithm and computer implementation, number of clusters, and interpretation of results. I will briefly explain each step and how it relates to the extension of AMOEBA and to the analysis of carbon rings.

A. Choice of Data Units

Data units are those numerical values which indicate something about the objects which the data describe. The proper choice of the data units which the analyst uses in the program is essential. For example, if an observer wished to determine the elevation of a mountain, then data pertaining to the height of the mountain relative to the sea level must be collected. Moreover, the same observer would not want or need to collect data concerning the relative locations of the mountain and of the sea. Hence, only the essential data should be used in analysis.

In AMOEBA, the program user will be allowed to specify the data units. Thus allowing AMOEBA to be used for a variety of data sets and purposes. Moreover, in the carbon ring experiment, the data units are the three dimensional coordinates of the carbon atoms.

B. Choice of Variables

Variables within the data indicate something specific about the existing interrelationships. For example, a data set describing an assortment of different colored geometric shapes would present the problem of choosing a variable to classify the objects. Should the objects be classified by color or shape? It all depends on what the analyst is intending to study about the data set.

Thus, AMOEBA should also be flexible in the analyst's choice of variables to allow a variety of things to be studied about a particular data set. Moreover, for the carbon experiment, the distance between carbon atoms is the choice of variable.

C. What to Cluster

Now that the data is collected and the aspect of the data that is intended to be studied is identified, the data can be analyzed . However, a question arises, what should be clustered that will enable the researcher to help understand what the data can tell him regarding his chosen variable?

For AMOEBA, this division will again be up to the analyst, but will be limited by both the choice of data units and data variables. Moreover, for the carbon experiment, a group of closely spaced carbon atoms is clustered. This is done because a phenol ring is composed of six closely spaced carbon atoms. If a group of atoms is found, then it is classified as a group. However, simply because a group of atoms is close to each other does not mean that they form a phenol ring. This leads to the next point.

D. Clustering Criteria

Clustering criteria is that information which tells the computer to discard or to store for later use a certain cluster. For example, a data set containing a number of different sizes of spheres was clustered for spheres. This cluster of spheres was then limited to a certain size of sphere, then those clusters not mentioning the size criteria would be discarded or stored elsewhere.

AMOEBA will allow the user to establish clustering criteria only after it has performed its unique operations. Then, the program becomes interactive with the user to develop the computer observations into a usable form. Moreover, the carbon atoms experiment clusters atoms based on their proximity to one another. However, only those groups

involving six atoms are stored. Thus the clustering criteria for the carbon atom experiment is the number of carbon atoms in a group.

E. Homogenizing Variables

Since a program can analyze data based on several variables, a relationship between the variables used must be described. For example, if the size and shape of geometric figures within a data set is studied, then what relationship do these variables have? Is there any correlation between a small cube and a small sphere or is the correlation between a small cube and a large cube? Thus , a comparative intuition must be developed in the program concerning the relationship between variables.

AMOEBA will be interactive in this area as well since it depends on the interest of the researcher as to what relative information is important. Moreover, in the carbon experiment, only groups of six carbons were found, so only one variable, distance, is studied.

F. Similarity Measures

As the number of variables increases, the number of possible comparisons also increases. Thus, not only do these variables need to be computable or homogeneous, but they need to be comparable. For example, assume a data set composed of three dimensional colored geometric shapes of different sizes. Now study the data and decide if the shapes are related to the size or the color , or is the color related to the shape and size? Thus, many possible relationships between a multi-variable study exist.

AMOEBA will certainly have to be user interactive in this respect since the number of possible relationships could be infinite, even with a small number of variables. Moreover, for the carbon experiment, this problem does not exist because one variable

not a set of variables is studied.

G. Algorithms and Computer Implementation

Algorithms are simply those concepts used to solve a specified problem. Computer Implementation of these algorithms means that the algorithm is written in a form that the computer can use. For example, if the relative size of geometric shapes is to be considered, then an algorithm comparing the area or volume of the different objects should be implemented into the computer logic.

The major algorithms which AMOEBA uses have previously been explained. The implementation of these algorithms has yet to take place. Moreover, the algorithms behind the carbon experiment have been completely implemented.

H. Number of Clusters

The number of clusters indicates the number of aspects found by the program based upon the decision made by the programmer. For example, the number of clusters found in a data set containing information on a deck of playing cards can vary greatly. If color is used to classify, then two clusters will be found. If suit is used, then four clusters will be found. This process could be continued on certain data sets indefinitely. Thus, the number of clusters found is a very important question which the programmer or user must answer.

AMOEBA will allow the user to decide on the number of clusters found. Moreover, the number of clusters found in the carbon experiment is not an aspect written into the program. This is because the original question posed by the researcher, myself, was how many phenol rings are there?

J. Interpretation of Results

The interpretation of results phase determines what the researcher is able to learn from the data. For example, if the program is designed to label a cluster by specific name then most of the interpretation has been done by the computer. However, if only the raw clusters are received as output, then most of the interpretation work must be done by the researcher. Thus, the ability to interpret the results depends on both the quality of the program and the skill of the user.

AMOEBA will not be trained to interpret the results as a stand alone program. However, additional supplemental programs or subprograms will hopefully be developed that will focus AMOEBA in a certain area. Moreover, the interpretation of the results of the carbon atom experiment is quite easier. I can simply read the output and determine the number of phenol rings present.

Starr: The history of AMOEBA is contained in the comment lines of the code. These comment lines have been edited out and placed in Appendix 1. AMOEBA itself was not included because it has approximately 13,000 lines of code.

In short, the ideas of AMOEBA, specifically the five point method, have been developed into a new program called Starr. Starr attempts to do in three dimensions what AMOEBA does in two. Starr is the first attempt at AMOEBA's conversion.

Starr begins by making artificial data. Starr then calculates the gradient of each point. The gradient can then be used to determine the amount of change occurring in the data. Thus, where the gradient change, or change in value, is large then that area is considered to be a boundary region. The points that comprise a connected group in which the gradient does not change a great deal are considered to be non-boundary points.

Once the gradients have been calculated Starr uses the constant data to form seed groups. These seed groups of constant data are collected until they are surrounded by

boundary points. Thus, groups are found by Starr. Starr then uses AMOEBA's five point

comparison technique.

However, Starr currently can not generate enough test sets because its data set is too small. Starr would need a minimum of about 100 test sets whereas the current data generate only seventeen.

Findp: Findp is a pattern recognition program that clusters the phenyl rings in a compound. The program has been successful on seven out of the nine data sets acquired from Dr. Fackler's research group. Thus, Findp needs a little work.

The program begins by calculating each points three closest neighbors. This data is then printed out. The point being considered is now called i and its closest neighbor j. These two points are used in a similarity measure. The program then produces a diagram of points that show a directional relationship to its most similar neighbor. If two points both point to each other, then these two points are called an attractor set.

The attractor sets are then used to cluster the program. For each set of six points in a group a centroid is calculated. Then a point not of the six is calculated for its distance away from the centroid. Theses calculations are completed and then the point with the largest distance associated with it is thrown out of the group. This is continued until a group of only six carbons remain.

Thus, phenyl groups are recognized. The points not clustered are thrown into a general pool which could be recalculated for phenyls if the researcher thought that the program missed some the first time around.

For this report, we print out only the first wave of calculations because we know it to be accurate. In addition, Findp prints out each group and the other points distance from the centroid of the group. Then the groups are identified as well as those points not in a group of six.

Conclusion: Thus, the general field of pattern recognition can be seen to have many applications. Although I was unable to complete the AMOEBA conversion, I was able to complete the phenyl ring clustering program which works 78% of the time, and the program Starr. In addition, I gained valuable incite into the requirements for a good pattern recognition program.

In essence, this research could produce valuable fruits for many disciplines. I hope that I will continue working with this research topic in the future. I would like to thank Dr. Bryant for everything he has taught me. I would like to thank Dr. Fackler for allowing me to work in his group. In addition, I would like to thank Cornell University and the National Science Foundation for their technical support. Finally, I would like to thank the Honors Department at Texas A&M for the fantastic experience.

Works Cited:

Anderberg, Michael R. Cluster Analysis For Applications. New York: Academic Press, 1973. pp. 1-16.

Bryant, Jack. AMOEBA. Pattern Recognition. January, 1972. pp. 1041.

APPENDIX A

AMOEBA History

Section 1. General comments.

LANGUAGE VAX-11 FORTRAN-77 (ANSI X3.9-1978)

PURPOSE Driver program for the AMOEBA clustering-classification-dimensionality reduction program.

---------- CALLED MODULES ----------

AMCPARM	Gets parameters.
AMOEBA	AMOEBA
MAKEOUT	Writes output file.
LIB$GET_VM	Get virtual memory in program region. A VMS Run-time Library function.
LIB$FREE_VM	Free virtual memory from program region. A VMS Run-time Library function.

PROJECT AMOEBA CLUSTERING/CLASSIFICATION/DISPLAY PROGRAM

---------- REVISION HISTORY ----------

VERSION	DATE	AUTHOR	AFFILIATION	REMARKS
10	12/1/83	Jack Bryant	Texas A&M	Initial VAX Version AMOBMAIN
10.1	12/20/83	Jack Bryant		Allow testing any input size & mask.
10.2	1/17/84	Jack Bryant		Add subroutine GETPAR for RSC.
10.3	3/5/84	Jack Bryant		Replace GETPAR with AMCPARM to allow many different data types and arrangements of headers. Also, replace READBYTE with READ_DATA and add module to rewind and skip header records on input data file(s). Include file added to pass parameters.

10.4	12/25/85	Jack Bryant	Corrected bug to allow self-generation of data. Changes in START, COUNTERR, and NUMCLU.
10.5	1/1/86	Jack Bryant	Sharpened estimates on memory required (trying to reduce same).
11	1/20/87	Jack Bryant	Extensive changes in classifier...over twice as fast now!
12	4/14/88	Jack Bryant	More classifier and NUMCLU changes. Also enough changes in START to mention.
13	3/1/89	Jack Bryant	Added Robert's-like gradient for boundaries. Added allowing old cluster map to be a mask for new run.
13.1	5/10/90	Jack Bryant	Removed Robert's stuff pending more study. Changes sprinkled throughout, mostly for minor increase in speed.
13.2	6/20/90	Jack Bryant	Big changes in NUMCLU and PERPIXEL make the program 30% faster! Minor changes in AMCPARM. Longstanding Bug fixed in THINMEAN.
14	8/31/90	Jack Bryant	Classification is faster yet. (Yet another simple idea.) Started putting in changes to do spatial filtering and logic on reclassification of apparent mixtures.
14.1	9/10/90	Jack Bryant	Implemented the pair mixture distance test in the AMOEBA classification option. Also put in logic to exit-when-classified in NUMCLU.
14.2	10/21/90	Jack Bryant	Put in 250 lines of help in the parameter read module. The statistic file now has file name <outfile>.TXT. Factored out mask tests everywhere. Made two read_data modules depending on whether the classification mask is set.
14.3	10/23/90	Jack Bryant	Added option to allow previous cluster map (not necessarily by AMOEBA) to be used to start. Included help statements for this.

Section 2. An outline of the model.

1. Axiom: Real classes exist.

2. Def: A pixel is "pure" if it and all four nearest neighbors are in the same real class.

3. Def: A "path" is a sequence $P_1,\ldots,P_n$ of pixels such that P_i is

   one of the four nearest neighbors of P_{i+1}, $i = 1,\ldots,n-1$.

4. Theorem: If P and Q are pure pixels and are neighbors then they are in the same real class.

5. Def: A set is "connected" if each pair of points in the set can be joined by a path lying in the set.

6. Def: If S is a set, then C is a "component" of S if C is a maximal connected subset of S.

7. Theorem: If S is a set and P is an element of S then P is contained in a unique component C of S. Moreover, C is exactly the set $C = \{ Q : P$ can be joined to Q by a path lying in S $\}$.

8. Def: A "patch" is a component of the set of pure pixels.

The set of patches is thus a partition of the set of pure pixels.

9. Def: The "classifier" is the mapping from the set of all pixels to an unknown set of labels.

Let us note that the classifier exists but is unknown. It exists by virtue of Axiom 1.

10. Theorem: If F is a patch, then each point in F has the same label; that is, the partition induced by the classifier in the set of pure pixels is refined by the component partition.

11. Def: A "NNC" is a nearest neighbor classifier determined by a distance function and a set of cluster attractors.

12. Def: A measurement vector R is a "convex combination" of vectors P and Q if $R = a P + (1-a) Q$ for some a between 0 and 1.

13. Axiom: (But with supporting arguments...) If P and Q are measurement vectors in the same real class and R is a convex combination of P and Q, then a NNC should classify R in the same class as P and Q.

14. Def: A distance function is "natural" if it is induced by a norm on the vector space in which the measurement vectors are imbedded.

Note that the maximum likelihood classifier is a NNC induced by a natural distance function. Also natural are the Minkowski distances

$$d(P,Q) = (\text{sum}_i \mid p_i - q_i \mid^{**} p)^{**} 1/p.$$

Two popular choices for p are 1 ("city block" distance) and 2 (Euclidean distance). Note also that the different p_i may be weighted with unequal (positive) weights w_i. A "parallelepiped" classifier can be viewed as a weighted

p=infinity classifier.

15. Theorem: If a natural distance function induces a NNC consistent with Axiom 13, then it is weighted Euclidean distance.

Let C be a classifier. Let P and Q be two pure pixels. There are four possibilities:

(1) P and Q are in the same real class and C(P) = C(Q);

(2) P and Q are in different real classes and C(P) # C(Q);

(3) P and Q are in the same real class and C(P) # C(Q); or,

(4) P and Q are in different real classes and C(P) = C(Q).

16. Def: The "pair probability of misclassification" is the probability PPMC of case (3) or (4) above.

Note a NNC is completely determined by the class attractors and the distance function. In clustering unknown data (to search for

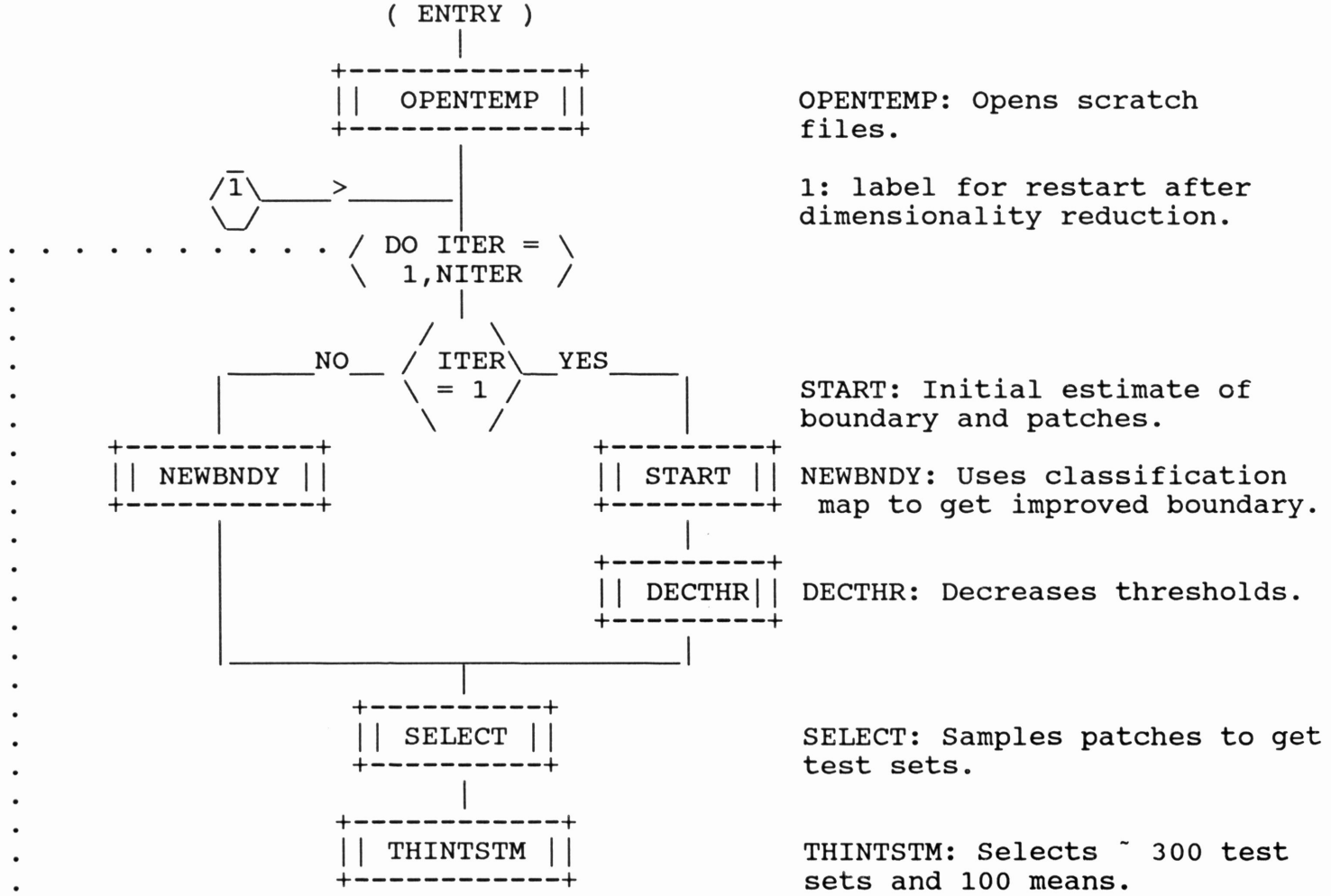
structure with little a priori knowledge), we will not know the "best" weights to apply to the only acceptable distance function (Euclidean distance). We therefore let all weights be equal. This fixes the distance function, and it remains to find the attractors.

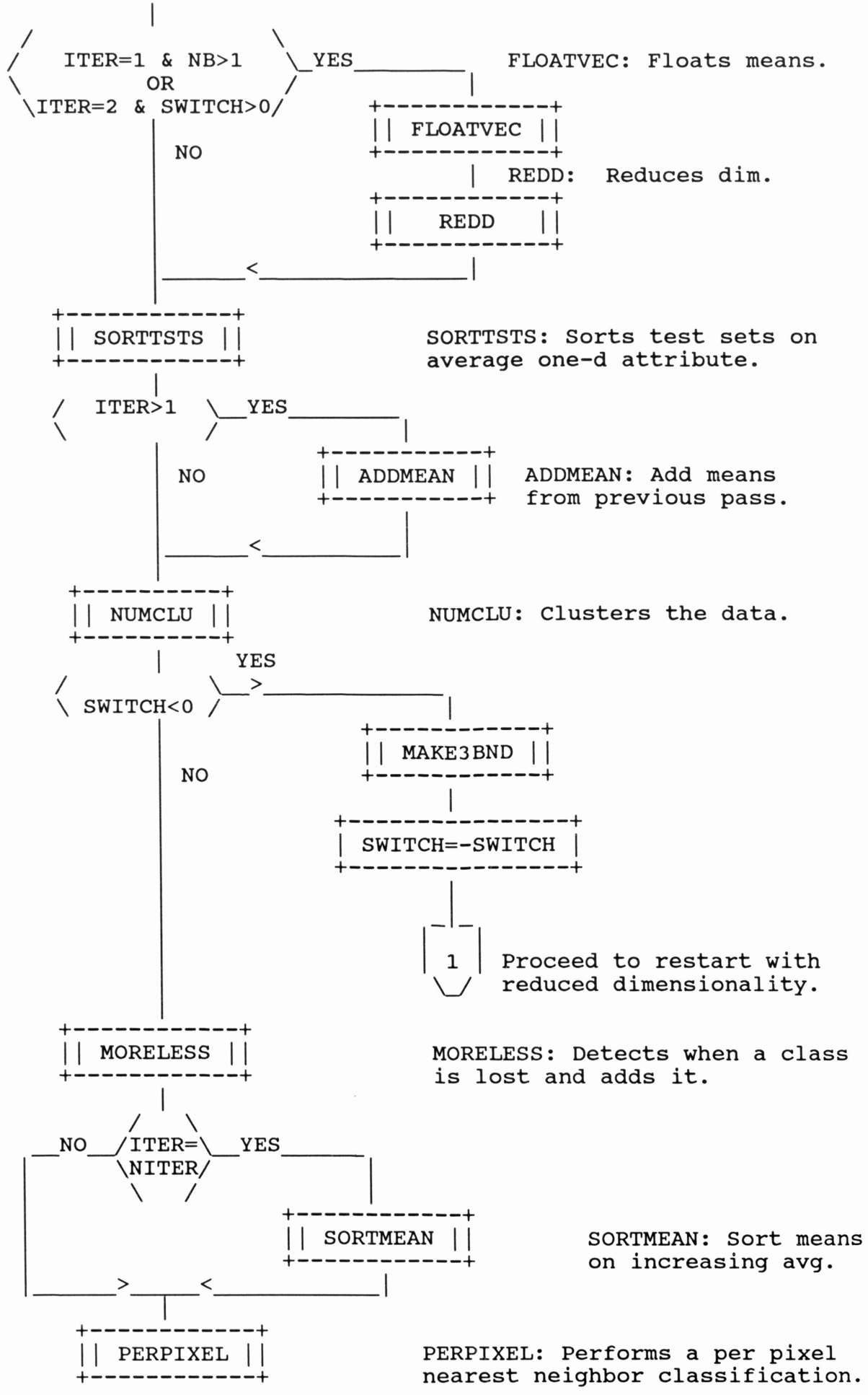
17. Objective: It is desired to minimize the PPMC.

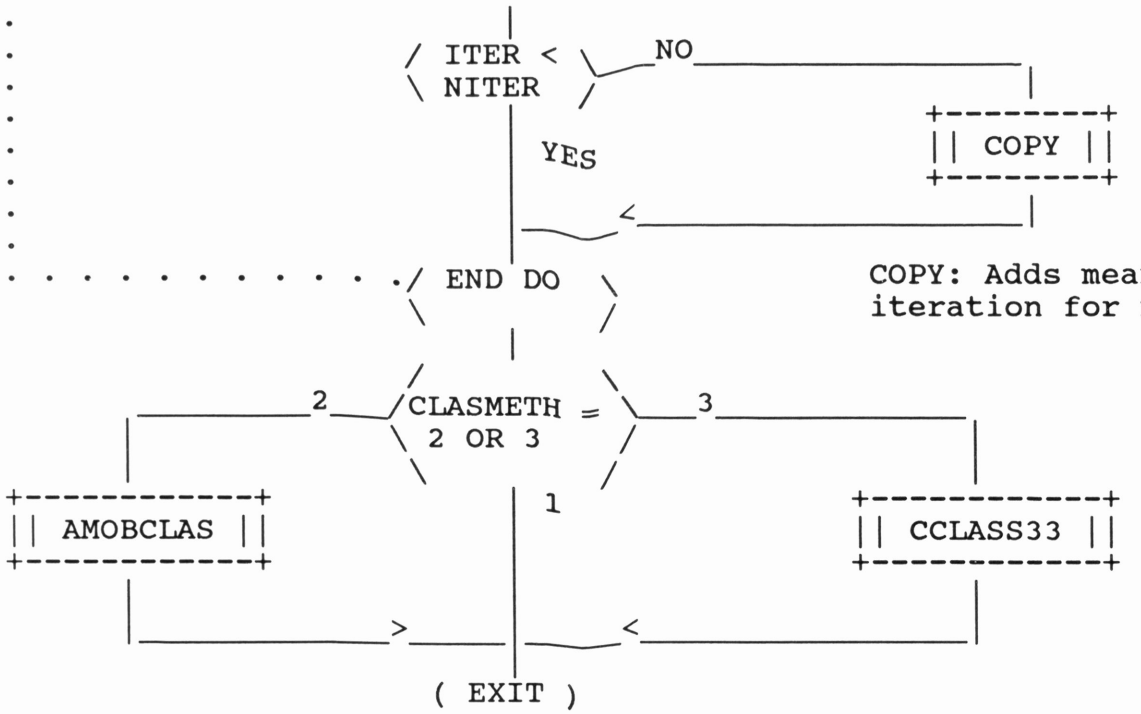
With this objective, we need a means of estimating the PPMC. Examine again Theorem 10. If we can estimate the set of mixed (i.e. not pure) pixels, then we can get an estimate of the patches. Using Theorem 10, we can extract samples from the case "same real class". It has been observed that the average brightness of the measurements is the single most significant attribute in most multi-image analysis problems. Therefore, sort the samples on this feature, and select pairs close, but not too close, in the order. This gives a way to estimate the "different real class" case. By counting errors (disagreements between NNC and the case pairs), we obtain both an estimate of the PPMC and an indication of which attractors are the cause of the (relatively) most errors.

This is the general outline of the idea; refinements, such as using a prior classification to refine the boundary estimate, and on selecting starting cluster attractors from the very large number of candidates, are described in the subroutines which do the work.

Crude flow chart and brief description of the modules.




ITER=1 & NB>1
OR
ITER=2 & SWITCH>0
YES
NO
FLOATVEC
FLOATVEC: Floats means.
REDD
REDD: Reduces dim.
SORTTSTS
SORTTSTS: Sorts test sets on average one-d attribute.
ITER>1
YES
NO
ADDMEAN
ADDMEAN: Add means from previous pass.
NUMCLU
NUMCLU: Clusters the data.
SWITCH<0
YES
NO
MAKE3BND
SWITCH=-SWITCH
1
Proceed to restart with reduced dimensionality.
MORELESS
MORELESS: Detects when a class is lost and adds it.
ITER=NITER
NO
YES
SORTMEAN
SORTMEAN: Sort means on increasing avg.
PERPIXEL
PERPIXEL: Performs a per pixel nearest neighbor classification.



AMOBCLAS Performs standard AMOEBA spatial classification using the boundary model.

CCLASS33 Context classification based on the context in a 3X3 neighborhood.

Section 4. Revision history of AMOEBA.

VERSION	DATE	AUTHOR	AFFILIATION	REMARKS
1	5/1/76	Jack Bryant	Texas A&M	Initial Version AMOEBA

The first version of AMOEBA was a starting procedure for extracting samples to test other clustering procedures. These, which were among the fastest methods known, included ISODATA [1], the K-Means Algorithm [2], and a Single Linkage Tree Algorithm [3]. The purpose of the testing was to evaluate various clustering methods and their potential in the Large Area Crop Inventory Experiment (LACIE). In the LACIE, a standard image size was the "segment", a 117 by 196 multispectral scanner Landsat image, often three or more rather poorly registered temporal acquisitions. The data were generally free of clouds but not haze, and were of areas of intense large field agricultural activity. The images were thus small and not spectrally complex. Still, we lacked the computational resources to adequately test the methods. Moreover, all the methods failed to solve the clustering problem when given samples which were measurement-space mixtures--mixtures which exist owing to the resolution and precision of the sensor, atmospheric effects, registration problems, and calibration errors. To avoid testing using mixtures, as well as to sample the data, we used the Fisher Algorithm [4] down scan lines to break the data into intervals.

The Fisher method divides a sequential data set into segments so that the sum of the within-segment "diameters" is minimal for that number of intervals. It is not a true clustering method, for no information on whether two distinct segments are the same "class" is obtained. Moreover, although for a fixed number of intervals there is an objective

function, there is no way to decide how many intervals to use. We used 12 intervals for scan lines of 117 pixels, and selected the center of each interval which contained at least 5 pixels. This brought the number of points to cluster from 23,000 to 2,000, a big saving since the methods seemed to need O(#samples**2) time.

2 7/1/76 Second Version AMOEBA

One drawback was that boundaries in the direction normal to the scan line direction were missed. Another application of the Fisher method in that direction picked up those boundaries (but also the every-six-line Landsat problem). By then we were making line printer maps, and the name AMOEBA had caught on because of the appearance of the blobs. Our sampling strategy now had us picking points with no neighbors on any of the thickened boundaries, much the same as the present subroutine SELECT does. We now had between 250 and 350 samples to cluster, another big improvement. We were also pleased to be using Dynamic Programming [5]. Pleased, that is, because of our background as mathematicians. Not so pleased, however, with the time the Fisher method was taking, and not at all happy with the results of the clustering methods. At that time, the idea was to use clustering to develop training sets for a Maximum Likelihood Classifier. But they were impossible to compare; none did well, nor did the classifier seem all that likely to be maximum given the poor training.

3 8/1/76 Third Version AMOEBA

Version 3 added a module to label and sample the different components of the complement of the boundary. David Egle and I wrote the fast and elegant component labelling program; unlike relaxation methods, only one pass was required to grow all the labeled components at one time. The program was optimal in the sense that each pixel was examined once. Also, the actual computations involved in the Fisher method were studied (we were trying to speed them up). I found that simply thresholding the spectral distance of a pixel to its nearby neighbors did as well. I determined the threshold by experiment, and made it adapt to data variability. Nothing "dynamic", and no "maximum likelihood" either. But, if results get to decide, the map of boundaries was better (less sensitive to 6-line noise), and the new version could do it in less than one percent of the time we had been using. The maps and lists of blob statistics from these simple, fast methods seemed better than classical methods, for example from [6]. To some extent, I had put aside my mathematical point of view. It was to return.

4 4/1/77 Fourth Version AMOEBA

None of the clustering methods being tested was doing well. This task was yielding inconclusive results, as might be expected given the variety of methods. Yet the data were as good as could be hoped for (given the hardware). Moreover, the problem seemed to be easy, with preselected data free of clouds, with large fields, flat terrain, and multiple acquisitions. Cluster maps were confetti, not wheat fields. Everyone blamed it on the data, and wanted to correct for bad clustering (and therefore poor classifier performance) by fancy neo-statistical methods. Version 4 of AMOEBA was a clustering program which used the boundary map from Version 3. Components of the complement of the boundary (now called patches) were sampled, forming what I called test sets. The averages of the test sets were used as starting cluster centers for a nearest neighbor classifier. Unpopular clusters were eliminated, reducing the clusters to a user-supplied number. Fast and simple, and the

resulting cluster maps were, by comparison, almost great.

5 11/1/77 Fifth Version AMOEBA

Three changes marked Version 5. One was motivated by discussion with people who interpret images manually: that boundary-between-field decisions are made based on one of the multi-images, not on some multi-dimensional distance function. This led to the present vector decision thresholds. Another was the introduction of reject classifications. This was really a natural result of a simple model for the spectral appearance of a spatial boundary with registration errors possible. The third was the initial version of the heart of the clustering program NUMCLU. For the first time we had a clustering program which determined the number of classes automatically. The idea then was to count the errors made when the classifier split (i.e., classified differently) test pixels from the same patch. An estimate of the number of errors expected (depending on the number of clusters) was compared to the number of errors observed, and the number of clusters determined following this comparison to to minimize the deviation.

6 2/1/78 Sixth Version AMOEBA

Version 6 was a result of a formal model for Landsat data. In particular, the objective in NUMCLU (the present version) was introduced. That objective, the pair probability of misclassification, allows one to compare the hypothetical (and desired) real clustering with the classification of samples
taken from the same patch and from different patches. We also introduced spatial classification aids. Points which appeared to be misclassified were reclassified if possible within the rejection threshold model. Version 6, a finished program, can cluster and classify a four pass LACIE segment in 12 seconds (on the AMDAHL), using `only' 320K memory in the process. Although subversions of Version 6 continued to appear until 10/31/79, the main ideas were complete [7].

Although the path that led us here has often strayed from conventional mathematics, the approach is in spirit mathematical. (The model outlined above is a mathematical model; the depth or lack of it, compared to "real" mathematics, is beside the point.)

The development of AMOEBA through Version 6 was supported in part by NASA contract NAS-9-14689, "Development and Selection of Clustering Procedures," L.F. Guseman, Jr., P.I. It is a pleasure to acknowledge this support. Significant contributions to the program came from Gary Breaux and David Egle.

7 11/1/80 Seventh Version AMOEBA

In the summer of 1980, I went to EROS Data Center, a U.S. Geological Survey center near Sioux Falls, S.D. The purpose of the visit was to transfer some of the techniques developed at Texas A&M during the LACIE to the Data Analysis Laboratory at EROS, as well as to develop new techniques as time permitted. One new technique was the use of hashing to histogram multidimensional data--e.g. Landsat data. Using this method, we were able to compress a Landsat one pass frame so it could be processed on the AMDAHL (i.e., the data were all in memory.) Version 7 uses data in this form. Unfortunately, a two pass Landsat image is too complex to be compressed in this manner. (Of course, the AMDAHL is a virtual machine: little memory limitation, in theory, is placed on a task. But the cost of memory resources at A&M was high.)

8 9/10/81 Jack Bryant and S.K. Jenson Eighth Version AMOEBA

Version 8 was written at EROS the following summer. The program uses several tricks to segment essentially infinite images into strips, to find boundary and samples for NUMCLU, and to classify data, all on a tiny (by AMDAHL standards) minicomputer (the Hewlett-Packard 3000 Series III). The program runs as an IDIMS function. Comparisons with ISOCLS [8] show it performs about as well while saving time (AMOEBA is faster) [9]. Major enhancements (in addition to the ability to handle very large images) include a provision for a mask and the generation of a "statistics" file.

9 6/1/82 Ninth Version AMOEBA

Version 9 is an iterative enhancement of Version 6. I noticed that boundaries were found where the classifier (i.e. the per pixel classifier) got lost. Classification induced boundaries were used to generate patches for the next pass of the clustering procedure. Although this uses more time, it seemed to result in better clustering, at least in the LACIE data I tested the new program on. Several related techniques on finding boundaries and on classification based distance functions and gradients were presented during a special session of a meeting of the American Mathematical Society [10].

10 12/1/83 Tenth Version AMOEBA

Version 10 is a complete rewrite of Version 8 (now in FORTRAN 77). Mainly owing to the better compiler, the program is written in well structured code. The advantages of the structured approach to coding are widely recognized [11]. Additions to Version 8 include:

@ The user may supply weights for counting error cases; these were fixed in all previous versions.

@ A choice of classifiers is made available. They are:

>> a per pixel classifier (not an option of versions past number 5)

>> the Version 8 spatially supervised classifier

>> a new context classifier [12], based on 3X3 blocks (new to all versions)

@ Each pixel is given a label; if necessary, the number of clusters is increased during the classification step. The difference between this and Version 8 lies in the more efficient classifier which only looks at the exceptional case when regular classification is rejected.

@ The number of iterations may be specified by the user. Version 8 did not iterate at all. Other versions handle only small images.

@ Using VAX-11 Run-time Library calls, all memory management is transparent to the user; the program runs in as little virtual memory as possible. No previous version has had this feature.

One minor change in Version 10 is the way in which the mask is labelled with 255 rather than 99 or 0 (in earlier versions). This allows 254 context classes, and reserves 0 for "unclassified".

10.1 5/1/84

The VMS version has been modified to merge clusters for the first time. The merging is carried out in subroutine MORELESS (formerly MOREQUES). The thresholds are one-fifth of the vector boundary decision thresholds determined in START.

10.2 1/10/86

A number of minor changes have been made. The test data generation portion was incorrect due to the removal of READBYTE. NUMCLU, COUNTERR, and COLAPS have been modified. Many fossil comments have been removed. The initial start has been given additional protection to guard against a lost boundary finder. In AMOEBA, code has been included to add the final clusters from the previous pass to the spatially determined starting clusters. The two print routines PRTSTATS and PRINT33 have been modified to make ASCII files. The AMOEBA classification method has been modified to use 8-neighborhoods instead of only 4-neighborhoods. A similar change has been made in the preliminary steps before context classification. All logic including BIGDATA has been commented out everywhere (awaiting true 12 bit data).

11 6/19/86 Eleventh Version AMOEBA

A major rewrite of the logic in the main clustering module NUMCLU is the principal change leading to Version 11. Although the objective function remains unchanged from Version 6, the minimization search has been considerably widened for the final iteration. Roughly speaking, Phase II of NUMCLU takes "subclusterings" of the results of the earlier Version (Phase I) and evaluates, in a 64 node tree search, each subclustering, looking for arrangements which make fewer errors in the way the clustering fits the model. In this way, essentially every arrangement which has any chance of improving the partition is examined. A report on this work is in preparation.

Another significant change (with a surprising improvement resulting) is the incorporation of the dimensionality reduction technique of Bryant and Guseman in the initial phase of NUMCLU; this module takes the starting means for NUMCLU and finds the best linear mapping to one dimension. Earlier versions simply used the sum of all the measurements as the one dimensional attribute. That was not bad for Landsat data, but the new method is better when far IR or UV bands are present.

Yet another change allows the user to reduce the dimensionality of the data to three to make a color display of the image, and, at the same time, save time in the program. The time saved depends on the number of bands in the input image: for six band input and two iterations, it amounts to at least 45%. The system is designed for MSS data. The reduced dimensionality image is designed for currently available systems. We hope for reproducible colors: that is, the display will be stable if the sensor and scene do not change materially.

Another change is to restart the process following the dimensionality reduction step. This is needed because the parameters which are used to detect probably identical classes are defined in START. If dimensionality reduction has been carried out, these parameters get lost.

A major change has been made in the context classifier: no one likes the many classes you get when you use it. So an option has been provided which will merge classes when they can be merged. The entire context classifier situation is incomplete, although the program does work as advertised. Probably, a 3x3 neighborhood is too small.

A number of less significant changes have also been incorporated. One is the way in which the "mask" is handled. Since experience has shown that the "unclassified" event never (well, almost never) happens on real data, and since the value 255 is hard to deal with on the IIS image display hardware, the mask label is made 0 on the final pass if the classification method is 1 or 2.

Also, the weights /default have changed to fit better with the new version of NUMCLU and the new option in the context classifier; it seems to work much better.

12 12/19/86 Twelfth Version AMOEBA

Another major change in NUMCLU uses the exhaustive search on all iterations now. Changes in the way classification is carried about, plus saving recently encountered trial arrangements, makes this possible. A seemingly minor change in the way distances are computed when performing nearest neighborhood classification has made the program at least twice as fast on 4-band data. Throughout the program, long to short integer conversions have been eliminated; this gains still more speed.

The AMOEBA classifier has been generalized to allow the user to select from one to eight neighbors (in the 8-neighborhood) which must match to accept the classification. The resulting cluster maps when (say) 4 alike is selected are very smooth and easy to interpret (if slightly less accurate). The 1-neighbor alike preserves fine detail such as urban features better and is more accurate than pure per pixel classification.

The starting boundary estimation procedure has been improved. When a very large patch is detected, a serious effort is made to detect additional boundary points without changing the thresholds. The time spent here is probably repaid later by not having to collect and analyze the extra points. In addition, a gradually changing natural feature, such as observed in rangeland or wetlands, is much less likely to be taken to be a patch.

Throughout the program, changes in the way classification is carried out have been made. This work, reported in [13], makes the program nearly three times as fast on 11 dimensional data, yet the idea is very simple and natural.

The module that makes the 3-band image has been improved. Safeguards have been incorporated to prevent the 'folding' operation to lose more that 1/2 percent of the counts (not pixels clipped, but actual counts lost). The original objectives are still met, although the resulting color displays are not as spectacular. (The spectacular appearance was probably a result of substantial 'folding.') The dimensionality reduction method is now the simple principal components method, rather than the method of Bryant and Guseman, pending the location of an efficient nonlinear optimization method. This work is presented in [14].

A number of fossil comments have been removed, shortening the source by a thousand lines.

13 1/20/90 Thirteenth Version AMOEBA

A number of changes have made this version faster. Most ammount to replacing subroutines and array indexing-loops with straight line code written out. Of course, this makes the program difficult to understand. These changes are documented in the modules affected.

14 1/10/91* Fourteenth Version AMOEBA

* Projected release date.
"The future lies ahead of us" -- R.M. Nixon

The changes and additions planned for Version 14 are listed here in no particular order; a few are complete indicated by a date e.g.:
8/10/90

Classification is faster still because of the test for "when classified" (the nearest attractor has been found when the distance to attractor tested is half the distance to the other nearest attractor). This change in PERPIXEL. A similar change in NUMCLU. 8/20/90

The statistics file should have a name: make it the same as the cluster map with extension .TXT 8/21/90

Some speed-up seems likely if the mask and classmask tests are recoded.
8/21/90

Help files should be added--or built-in help. 8/20/90

An option will be added to allow one to produce a large number of clusters for the purpose of using these to make a color display in the setting of a display device with very limited palette--e.g., 256 colors. The output will be the cluster map and table of cluster attractors. The map is inteded to be used to produce color display products.

An option will be added to allow dimensionality reduction from n to m dimensions (instead of just n to 3). Also the dimensionality reduction will not scale like the 3 band product does (except to make sure the full use of 8 bit unsigned integers is made).

In connection with this, the Moore-Penrose generalized inverse idea will be implemented.

Texture needs to be experimented with. Two uses: use texture measurements as additional bands to aid (?) classification, and use texture to switch how many alike in the 8-point neighborhood are required.

In connection with this, the 8-point neighborhood logic is slightly different now. The 4 nearest are weighted twice that of the corners. At the same time, the requested number is multiplied by 3/2, keeping the intention of the user the same. 7/7/90

An option to use another cluster map as the entry to the program will be added. This would be useful if the system had a hardware clustering program built in which was much faster than one pass through AMOEBA.

AMOEBA has been known to lose a bright high variance class on the first iteration. I believe this can be corrected. The main result is that the dimensionality reduction routine receives poor training.

I propose to make a pyramid structure of high resolution data and do a multi-resolution attempt to understand the image.

A 9x9 context classifier will be implemented. I think the information in an area this big can still be kept in a 32 bit word, and I believe fewer than 2^16 actual contexts are present even in a complex image.

The Moore-Penrose inverse can also be used to restore missing data or to recreate at apparently higher resolution low resolution data which

is registered to high resolution data. I see no reason why this could not be done automatically.

An option of making the cluster map a color map should be easy to provide, although this should clearly be a separate program which depends on the system.

Filtering: high frequency enhancement filters of the form

$$F_{a,b}(s) = (1+a||s||^2)\exp(-||s||^2/b^2)$$

are easily approximated (the Fourier transform can be computed by hand:

$$f_{a,b}(t) := \int_{R^2} F_{a,b}(s)\, e^{-i\, s.t}\, ds$$

$$= \text{pi } b^2/4\, [4+4ab^2-ab^4]\, e^{-b^2||t||^2/4}$$

Within reason, these filters provide modest sharpening and at the same some smoothing. They are easily implemented as small (say 7x7) real domain convolution filters.

Information about the platform which acquired the data could potentially be used to automatically produce color display products, pyramid structure, and spatial filtering. Much data is available, but it is only now being assembled in one place.

---------- REFERENCES ----------

[1] Ball, G.H. and Hall, J.D., A clustering technique for summarizing multivariate data, Behav. Sci. 12(1967),153-155.

[2] Fisher, L. and Van Ness, J.W., Admissible clustering procedures, Biometrika 58(1971),91-104.

[3] Grower, J.C. and Ross, G.J.S., Minimum spanning trees and single linkage cluster analysis, Appl. Stat. 18(1969),54-64.

[4] Fisher, W.D., On grouping for maximum homogeneity, J. Am. Stat. Assoc. 53(1958),789-798.

[5] Bellman, R.E. and Dreyfus, S.E., Applied Dynamic Programming, Princeton Univ. Press, Princeton N.J., 1962.

[6] Hartigan, J.A., Clustering Algorithms, John Wiley and Sons, New York, 1975.

[7] Bryant, J., On the clustering of multidimensional pictorial data, Pattern Recognition 11(1979),115-126.

[8] Kan, E.P.F. and Holly, W.A., More on clustering techniques with final recommendations on ISODATA, Tech. Rep. LEC 64-TR-112, Lockheed Electronics Co., Inc., HASD, Houston, Texas, 1972.

[9] Jenson, S.K., Loveland, T.R., and Bryant, J., Evaluation of AMOEBA: A spectral-spatial classification method, J. Appl. Photographic Engineering 8(1982),159-162.

[10] Bryant, J., Clustering and boundary detection in image data, Abstracts AMS 2(1981),515. Presented to the AMS at 789th Meeting, Amherst, October 17, 1981.

[11] Tanimoto, S.L., Advances in software engineering and their relations to pattern recognition and image processing, Pattern Recognition 15(1982),113-120.

[12] Wharton, S.W., A contextual classification method for recognizing land use patterns in remotely sensed data, Pattern Recognition 15(1982),317-324.

[13] Bryant, J., A fast classifier for image data, Pattern Recognition 22 (1989),45-48.

[14] Bryant, J., On displaying multispectral data, P.E. & R.S. 54 (1988), 1739-1743.

[15] Bryant, J., AMOEBA clustering revisited, P.E. & R.S. 56 (1990), 41-47.

ONICA>

APPENDIX B

Starr: Program and Output

```
      program starr
c
c  One step test of first cut.
c
      implicit none ! remove for old version 2.1.1 for parallel
      integer nc,nr,np,nb
      character*64 outfile,infile
      parameter(nb=1,nc=32,nr=32,np=32)
      integer i,j,k,l,m,n
      integer testsets(5,356),means(256)
      integer nmeans,ntstsets,nfc
      real*4 max
      integer nbits
      parameter (nbits=32)
      integer bndy(0:(nc-1)/nbits,nr,np)
      real*4 a(nc,nr,np),grad(nc,nr,np),thresh
      common/data/ a,grad,bndy
      write(6,*) ' Percent in homogeneous blobs?'
      read(6,*) thresh
      write(6,*) ' Input file name?'
      read(6,1) infile
      write(6,*) ' Output file name?'
      read(6,1) outfile
 1    format(a)
      write(6,*) ' Here we go'
      call formdata(infile)
      write(6,*) ' Oh boy, we have data'
      call formgrad(max)
      write(6,*) ' Gradient formed'
      call findthrs(thresh,max)
      write(6,*) ' Found a threshold:',thresh
      call findbndy(thresh)
      write(6,*) ' Boundaries marked'
      call extract(testsets,means,ntstsets,nmeans)
      write(6,*) ' Extracted test sets and initial means'
      write(6,*) ' There are ',ntstsets,' testsets.'
      call numclu(testsets,means,nb,ntstsets,nmeans,nfc)
      write(6,*) ' Clusters found: there are:',nfc
      call perpixel(means,nfc)
      write(6,*) ' Classification step finished'
      call cleanup(nfc)
      write(6,*) ' Spatial fixups finished'
      call storeit(outfile)
      stop
      end
      subroutine formdata(infile)
      implicit none ! remove for old version 2.1.1 for parallel
c
c  The idea is to place charges at points read in and to set the stren
c  equal to the charge/square of distance except not less than 2.0 on
c  distance.
c
      character*64 infile
      integer nc,nr,np,nb
      integer nbits
      parameter (nbits=32)
      parameter(nb=1,nc=32,nr=32,np=32)
      integer bndy(0:(nc-1)/nbits,nr,np)
      real*4 a(nc,nr,np),grad(nc,nr,np)
      common/data/ a,grad,bndy
```

```
      real*4 center1,center2,center3,charge,temp
      integer k1,k2,i1,i2,j1,j2
      integer nchar,i,j,k,l,is
      character*1 yesno
      open(99,file=infile,status='old')
      nchar = 0
      is = 12321
      do 10 l = 1,100000
          read(99,*,end=20) center1,center2,center3,charge
          nchar = nchar+1
          charge = 1.3*sqrt(charge)
          k1 = center1-charge
          if (k1.lt.1) k1 = 1
          k2 = center1+charge
          if (k2.gt.np) k2 = np
          j1 = center2-charge
          if (j1.lt.1) j1 = 1
          j2 = center2+charge
          if (j2.gt.nr) j2 = nr
          i1 = center3-charge
          if (i1.lt.1) i1 = 1
          i2 = center3+charge
          if (i2.gt.nc) i2 = nc
          do 50 k = k1,k2
          do 50 j = j1,j2
          do 50 i = i1,i2
            a(i,j,k) = charge+(ran(is)-ran(is))*.4
50        continue
10    continue
20    do 40 k = 1,np
      do 40 j = 1,nr
      do 40 i = 1,nc
        if (a(i,j,k).eq.0.) then
          a(i,j,k) = (ran(is)+ran(is)+ran(is)+ran(is)+ran(is)+
     +  ran(is)+ran(is)-ran(is)+ran(is)+ran(is)+ran(is)-ran(is))*9.
        end if
40    continue
      write(6,*)' Dump data?'
      read(6,1) yesno
1     format(a)
      if (yesno.eq.'y'.or.yesno.eq.'Y') then
        write(6,2) a
2       format(1x,16f7.2)
      end if

      write(6,*)' I read',nchar
      return
      end

      subroutine formgrad(max)
      implicit none ! remove for old version 2.1.1 for parallel
c
c  Form the square of the magnitude of the gradient of a.  Only inside
c
      integer nc,nr,np,nb
      integer nbits
      parameter (nbits=32)
      parameter(nb=1,nc=32,nr=32,np=32)
      integer bndy(0:(nc-1)/nbits,nr,np)
      real*4 a(nc,nr,np),grad(nc,nr,np),temp,max
```

```
      common/data/ a,grad,bndy
      integer k,j,i
      max = 0.
c
c  This logic defines grad inside.  Easy enough.
c
      do 10 k = 2,np-1
      do 20 j = 2,nr-1
      do 40 i = 2,nc-1
          temp = (a(i+1,j,k)-a(i-1,j,k))**2+
     +     (a(i,j+1,k)-a(i,j-1,k))**2+(a(i,j,k+1)-a(i,j,k-1))**2
          if (temp.gt.max) then
            max = temp
            write(6,*) max,i,j,k,a(i,j,k)
          end if
          grad(i,j,k) = temp
 40   continue
 20   continue
 10   continue
c
c  Now define grad on the boundary of the parallelepiped.
c
      do 110 k = 1,np,np-1
      do 120 j = 2,nr-1
      do 140 i = 2,nc-1
          temp = ((a(i+1,j,k)-a(i-1,j,k))**2+
     +     (a(i,j+1,k)-a(i,j-1,k))**2)*1.5
          if (temp.gt.max) then
            max = temp
            write(6,*) max,i,j,k,a(i,j,k)
          end if
          grad(i,j,k) = temp
 140  continue
 120  continue
 110  continue
      do 210 j = 1,nr,nr-1
      do 220 k = 2,np-1
      do 240 i = 2,nc-1
          temp = ((a(i+1,j,k)-a(i-1,j,k))**2+
     +     +(a(i,j,k+1)-a(i,j,k-1))**2)*1.5
          if (temp.gt.max) then
            max = temp
            write(6,*) max,i,j,k,a(i,j,k)
          end if
          grad(i,j,k) = temp
 240  continue
 220  continue
 210  continue
      do 310 i = 1,nc,nc-1
      do 320 k = 2,np-1
      do 340 j = 2,nr-1
          temp = ((a(i,j+1,k)-a(i,j-1,k))**2+
     +     (a(i,j,k+1)-a(i,j,k-1))**2)*1.5
          if (temp.gt.max) then
            max = temp
            write(6,*) max,i,j,k,a(i,j,k)
          end if
          grad(i,j,k) = temp
 340  continue
 320  continue
```

```
 310  continue
c
c  The corners.
c
      do 410 k = 1,np,np-1
      do 420 j = 1,nr,nr-1
      do 440 i = 1,nc,nc-1
          grad(i,j,k) = max
 440  continue
 420  continue
 410  continue
      write(6,*)' Down the center: data'
      write(6,*)(a(i,nr/2,np/2),i=1,nc)
      write(6,*)' Down the center: gradient:'
      write(6,*)(grad(i,nr/2,np/2),i=1,nc)
      write(6,*)' Maximum gradient:',max
      return
      end
      subroutine findthrs(th,max)
      implicit none ! remove for old version 2.1.1 for parallel
      integer nc,nr,np,nb
      parameter(nb=1,nc=32,nr=32,np=32)
      real*4 max
      integer nbits
      parameter (nbits=32)
      real*4 hist(100)
      integer bndy(0:(nc-1)/nbits,nr,np)
      real*4 a(nc,nr,np),grad(nc,nr,np),th
      integer cs,i,j,k,count
      common/data/ a,grad,bndy
      th = (100.-th)*(nc-2)*(nr-2)*(np-2)/8.
      do 5 i = 1,100
   5  hist(i) = 0.
      do 10 i = 2,nc-1,2
      do 10 j = 2,nr-1,2
      do 10 k = 2,np-1,2
          cs = 1+99*grad(i,j,k)/max
          hist(cs) = hist(cs)+100.
  10  continue
      count = 0
      write(6,*)' Histogram of counts'
      write(6,*)(ifix(hist(i)/100.),i=1,100)

      do 20 i = 100,0,-1
          count = count+hist(i)
          if (count.ge.th) go to 30
  20  continue
  30  th = (max*i-1.)/99.
      return
      end
      subroutine findbndy(th)
      implicit none ! remove for old version 2.1.1 for parallel
      integer nc,nr,np,nb,ix,iy,iz,i,j,k,nbits,ipset,ibset
      parameter (nbits=32)
      parameter(nb=1,nc=32,nr=32,np=32)
      real*4 a(nc,nr,np),grad(nc,nr,np),th
      integer bndy(0:(nc-1)/nbits,nr,np)
      common/data/ a,grad,bndy
      ipset(ix,iy,iz) = ibset(bndy((ix-1)/nbits,iy,iz),
     +                        nbits-1-mod(ix,nbits))
```

```
      do 10 k = 1,np
      do 10 j = 1,nr
      do 10 i = 1,nc
          if (grad(i,j,k).gt.th) then
              bndy((i-1)/nbits,j,k) = ipset(i,j,k)
          end if
 10   continue
      return
      end
      subroutine extract(testsets,means,ntstsets,nmeans)
      implicit none ! remove for old version 2.1.1 for parallel
      integer nc,nr,np,nb,ix,iy,iz,x,y,z,nsa,map
      parameter(nb=1,nc=32,nr=32,np=32,nsa=nc*nr/10,map=222)
      integer knt(nsa),lab(nsa)
      integer testsets(5,356),means(256)
      integer nbits
      parameter (nbits=32)
      integer nmeans,ntstsets,c,s,llbl,nsamax,i25,i1,i2,it,l,m,nts
      logical lfound
      integer lold,lsave,maxslot
      integer bndy(0:(nc-1)/nbits,nr,np),plane(nc,nr,2)
      real*4 a(nc,nr,np),grad(nc,nr,np),dat(map,nsa)
      logical point
      common/data/ a,grad,bndy
      point(ix,iy,iz) = btest(bndy((ix-1)/nbits,iy,iz),
     +                        nbits-1-mod(ix,nbits))
c
c  First, go through the array and set the gradient to -1 at points wh
c  it had been thresholded.  Do this for all six neighbors too.
c
      do z = 1,np
        do y = 1,nr
          do x = 1,nc
            if (.not.point(x,y,z)) go to 10
            grad(x,y,z) = -1.
            if (x.gt.1) grad(x-1,y,z) = -1.
            if (x.lt.nc) grad(x+1,y,z) = -1.
            if (y.gt.1) grad(x,y-1,z) = -1.
            if (y.lt.nr) grad(x,y+1,z) = -1.
            if (z.gt.1) grad(x,y,z-1) = -1.
            if (z.lt.np) grad(x,y,z+1) = -1.
 10       end do
        end do
      end do
c
c  Initialization:
      s = 0     ! summer of distances
      c = 0     ! count of test sets encountered
      llbl = 1  ! label
      nsamax = 0        ! number of slots in use for labels
      ntstsets = 0      ! count of number of test sets collected
      i25 = 0   ! pointer used by extract
      call opentemp(10,5)       ! a file to store the samples
      i1 = 1    ! circular buffer            i1 oldest
      i2 = 2    ! pointers for two planes    i2 newest
c
c  Build the first plane: it is special.
c
c       call zap(plane(1,1,i1),nc*nr)   ! not necessary in the present
        do y = 1,nr     ! Process this plane by rows
```

```
          do x = 1,nc   ! and columns.
            if (grad(x,y,i1).ge.0.) then        ! Find a critter.
              if (x.eq.1) then  ! Got one.  Start of a row?
                if (y.eq.1) then        ! Yep.  First row?
                  plane(x,y,i1) = llbl  ! Yep.  Assign the label.
                else if (plane(x,y-1,i1).gt.0) then     ! Label above?
                  plane(x,y,i1) = plane(x,y-1,i1)       ! Yep. Transfe
                  go to 20      ! and break out of logic.
                else
                  llbl = llbl+1 ! Build a new label
                  plane(x,y,i1) = llbl  ! and assign.
                  go to 20
                end if
              else
                if (plane(x-1,y,i1).gt.0) then  ! Look to left.
                  plane(x,y,i1) = plane(x-1,y,i1)       ! Got one
                  go to 20      ! Done...next
                end if
                if (y.eq.1) then        ! First row?
                  llbl = llbl+1 ! Yep.  Start a new label
                  plane(x,y,i1) = llbl  ! and assign.
                else
                  if (plane(x,y-1,i1).gt.0) then        ! Nope.  Look
                    plane(x,y,i1) = plane(x-1,y,i1)     ! above and as
                  else  ! else
                    llbl = llbl+1       ! make a new label
                    plane(x,y,i1) = llbl        ! and assign.
                  end if
                end if
              end if    ! First column logic.
            end if      ! Blob logic.
 20       end do        ! Column loop.
        end do  ! Row loop.
      do z = 2,np       ! Plane loop starts.
c
c now build plane i2
c
        call zap(plane(1,1,i2),nc*nr)
        do y = 1,nr     ! Again by rows
          do x = 1,nc   ! and columns.
            if (grad(x,y,z).ge.0.) then ! If a blob point is found
              if (plane(x,y,i1).gt.0) then      ! then look to the pre
                plane(x,y,i2) = plane(x,y,i1)   ! plane, else the logi
                go to 30        ! exactly the same as above.
              end if
              if (x.eq.1) then
                if (y.eq.1) then
                  plane(x,y,i2) = llbl
                else
                  if (plane(x,y-1,i2).gt.0) then
                    plane(x,y,i2) = plane(x,y-1,i2)
                    go to 30
                  else
                    llbl = llbl+1
                    plane(x,y,i2) = llbl
                    go to 30
                  end if
                end if
              else
                if (plane(x-1,y,i2).gt.0) then
```

```
                  plane(x,y,i2) = plane(x-1,y,i2)
                  go to 30
                end if
                if (y.eq.1) then
                  llbl = llbl+1
                  plane(x,y,i2) = llbl
                else
                  if (plane(x,y-1,i2).gt.0) then
                    plane(x,y,i2) = plane(x-1,y,i2)
                  else
                    llbl = llbl+1
                    plane(x,y,i2) = llbl
                  end if
                end if
              end if
            end if
 30       end do
        end do
      end do
c
c  doing plane z-1 in the sense of the data.
c
        do y = 1,nr     ! process one row at a time

c  throughout, l always is the pointer into the collected stuff

          do l = 1,nsamax       ! this initially falls through (no lab
            if (knt(l).gt.0) then       ! test if a label has been fou
              lfound = .false.  ! lfound is a flag that says that labe
c                                 was found.  This logic starts fresh
c                                 for each scan line.
              do x = 1,nc       ! Process this line searching for
                if (plane(x,y,i1).eq.lab(l)) then       ! find it?
                  plane(x,y,i1) = 0     ! Yes.
                  lfound = .true.       ! Fly the flag.
                  if (knt(l).lt.map) then       ! Is there room?
                    m = knt(l)+1        ! Yes. Incr knt
                    knt(l) = m  !
                    dat(m,l) = a(x,y,z-1)       ! and save the data.
                  else  ! No.  Close that label.
                    call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
                  end if
                end if
              end do
              if (.not.lfound)  ! If this label was not found then clo
     +          call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
            end if
          end do        ! Label loop.
          lold = 0      ! This is to the most recently encountered lab
          do x = 1,nc
            if (plane(x,y,i1).ne.0) then        ! Test for valid label
              if (plane(x,y,i1).ne.lold) then   ! Yes. Test if old one
                if (nsamax.eq.0) nsamax = 1     ! Now we are started.
                do l = 1,nsamax ! Not the old one.
                  if (knt(l).eq.0) then ! Look for room for it.
                    knt(l) = 1  ! start a new critter here
                    lsave = l   ! lsave is where lold goes
                    lold = plane(x,y,i1)
                    lab(l) = lold
                    dat(1,l) = a(x,y,z-1)
```

```
                  go to 1     ! Next point.
                end if
              end do  ! If this falls through, then no new slots wer
              nsamax = nsamax+1
              if (nsamax.gt.nsa) then ! test if we are absolutely ou
                l = maxslot(knt,nsa,map)      ! Yes.  Find fattest a
                call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
                knt(l) = 1    ! Start a new critter here.
                lsave = l
                lold = lab(l)
                dat(1,l) = a(x,y,z-1)
                nsamax = nsa  ! Reset pointer
              else    ! There is room for a new one.
                knt(nsamax) = 1       ! Start a new critter here.
                lsave = nsamax
                lold = plane(x,y,i1)
                lab(nsamax) = lold
                dat(1,nsamax) = a(x,y,z-1)
              end if
            else      ! Yes, we found the old one.
              m = knt(lsave)+1        ! Prepare to add it.
              if (m.le.map) then      ! is there room?
                knt(lsave) = m        ! Yes.  Add.
                dat(m,lsave) = a(x,y,z-1)
              else    ! No room.  Close the full label.
                call gettst(lsave,knt,dat,nts,i25,map,nsa,c,s,nsamax
                lold = 0      ! Point to nil label.
              end if
            end if    ! Label search logic.
          end if      ! Valid label in map logic.
 1      end do        ! Next point loop.
c swap planes
        it = i1
        i1 = i2
        i2 = it
      end do
c
c  Now the last plane of labels should be gathered.
c
        do y = 1,nr     ! process one row at a time
          do l = 1,nsamax       ! this initially falls through (no lab
            if (knt(l).gt.0) then       ! test if a label has been fou
              lfound = .false.  ! lfound is a flag that says that labe
c                                 was found.  This logic starts fresh
c                                 for each scan line.
              do x = 1,nc       ! Process this line searching for
                if (plane(x,y,i1).eq.lab(l)) then       ! find it?
                  plane(x,y,i1) = 0     ! Yes.
                  lfound = .true.       ! Fly the flag.
                  if (knt(l).lt.map) then       ! Is there room?
                    m = knt(l)+1        ! Yes. Incr knt
                    knt(l) = m  !
                    dat(m,l) = a(x,y,np)        ! and save the data.
                  else  ! No.  Close that label.
                    call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
                  end if
                end if
              end do
              if (.not.lfound)  ! If this label was not found then clo
     +          call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
```

```
            end if
          end do        ! Label loop.
          lold = 0      ! This is to the most recently encountered lab
          do x = 1,nc
            if (plane(x,y,i1).ne.0) then        ! Test for valid label
              if (plane(x,y,i1).ne.lold) then   ! Yes. Test if old one
                if (nsamax.eq.0) nsamax = 1     ! Now we are started.
                do l = 1,nsamax ! Not the old one.
                  if (knt(l).eq.0) then ! Look for room for it.
                    knt(l) = 1  ! start a new critter here
                    lsave = l   ! lsave is where lold goes
                    lold = plane(x,y,i1)
                    lab(l) = lold
                    dat(1,l) = a(x,y,np)
                    go to 11    ! Next point.
                  end if
                end do  ! If this falls through, then no new slots wer
                nsamax = nsamax+1
                if (nsamax.gt.nsa) then ! test if we are absolutely ou
                  l = maxslot(knt,nsa,map)      ! Yes.  Find fattest a
                  call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
                  knt(l) = 1    ! Start a new critter here.
                  lsave = l
                  lold = plane(x,y,i1)
                  lab(l) = lold
                  dat(1,l) = a(x,y,np)
                  nsamax = nsa  ! Reset pointer
                else    ! There is room for a new one.
                  knt(nsamax) = 1       ! Start a new critter here.
                  lsave = nsamax
                  lold = plane(x,y,i1)
                  lab(nsamax) = lold
                  dat(1,nsamax) = a(x,y,np)
                end if
              else      ! Yes, we found the old one.
                m = knt(lsave)+1        ! Prepare to add it.
                if (m.le.map) then      ! is there room?
                  knt(lsave) = m        ! Yes.  Add.
                  dat(m,lsave) = a(x,y,np)
                else    ! No room.  Close the full label.
                  call gettst(lsave,knt,dat,nts,i25,map,nsa,c,s,nsamax
                  lold = 0      ! Point to nil label.
                end if  ! If room logic.
              end if    ! Label search logic.
            end if      ! Valid label in map logic.
 11       end do       ! Next point loop.
        end do ! Next row loop.
c
c  Now all open slots should be closed.
c
      do l = 1,nsamax
        if (knt(l).ge.5)
     +    call gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
      end do
      ntstsets = nts
      return
      end
      subroutine gettst(l,knt,dat,nts,i25,map,nsa,c,s,nsamax)
      implicit none
      integer l ! the label about to be closed
```

```
      integer nts       ! count of the number of test sets found
      integer i25       ! circular buffer pointer
      integer map       ! max in a blob
      integer nsa       ! max open at once
      integer knt(nsa)  ! the count of the number with label l
      real dat(map,nsa) ! the data in each blob
      integer c ! counter
      real s    ! summer
      integer nsamax    ! current number of potentially open critters
      integer kn
      real tdist,tsp(5),last25(25),a
      integer m,is,ip
      kn = knt(l)
      if (kn.gt.4) then
        tsp(1) = dat(1,l)
        tsp(5) = dat(kn,l)
        tdist = abs(tsp(1)-tsp(5))      ! distance first to last
        if (tdist.eq.0.) go to 1
        s = s+tdist
        c = c+1
        if (c.gt.4096) then
          c = c/2
          s = s/2.
        end if
        a = s/c ! average so far
        if (nts.gt.25) then             ! Is it initially
          if (tdist*2..lt.a) go to 1    ! interesting?
          if (tdist.gt.4.*a) go to 1    ! believable?
          do m = 1,25
            if (abs(tsp(1)-last25(m)).lt.tdist) go to 1 ! new?
          end do
        end if
        is = (kn-1)/4
        m = 1+is
        do ip = 2,4
          tsp(ip) = dat(m,l)
          m = m+is
        end do
        i25 = i25+1
        if (i25.gt.25) i25 = 25
        last25(i25) = tsp(5)
        write(10,*) (tsp(m),m=1,5)
        nts = nts+1
        type*,tsp
      end if
1     knt(l) = 0
      return
      end
      subroutine numclu(testsets,means,nb,ntstsets,nmeans,nfc)
      implicit none ! remove for old version 2.1.1 for parallel
      integer nb,nmeans,nfc,ntstsets
      integer testsets(5,356),means(256)
      return
      end
      subroutine perpixel(means,nfc)
      implicit none ! remove for old version 2.1.1 for parallel
      integer nc,nr,np,nb
      parameter(nb=1,nc=32,nr=32,np=32)
      integer nbits
      parameter (nbits=32)
```

```
integer testsets(5,356),means(256),nfc
integer bndy(0:(nc-1)/nbits,nr,np)
real*4 a(nc,nr,np),grad(nc,nr,np)
common/data/ a,grad,bndy
return
end
subroutine cleanup(nfc)
implicit none ! remove for old version 2.1.1 for parallel
integer nc,nr,np,nb
parameter(nb=1,nc=32,nr=32,np=32)
integer nbits
parameter (nbits=32)
integer testsets(5,356),means(256),nfc
integer bndy(0:(nc-1)/nbits,nr,np)
real*4 a(nc,nr,np),grad(nc,nr,np)
common/data/ a,grad,bndy
return
end
subroutine storeit(outfile)
implicit none ! remove for old version 2.1.1 for parallel
character*64 outfile
integer nc,nr,np,nb
integer nbits
parameter (nbits=32)
parameter(nb=1,nc=32,nr=32,np=32)
integer bndy(0:(nc-1)/nbits,nr,np)
real*4 a(nc,nr,np),grad(nc,nr,np)
common/data/ a,grad,bndy
open(98,file=outfile,status='new')
return
end
subroutine opentemp(iu,nr)
implicit none
integer iu,nr
open(iu,status='new')!,disp='delete',form='unformatted'
return
end
subroutine zap(a,n)
integer a(n),n,i
do i = 1,n
  a(i) = 0
end do
return
end
function maxslot(k,n,m)
integer k(n),n,m,max
integer maxslot
maxslot = 1
max = k(1)
if (max.ge.m) return
do i = 2,n
  if (max.lt.k(i)) then
    max = k(i)
    maxslot = i
    if (max.ge.m) return
  end if
end do
return
end
```

Starr Output

output of
run of Starr

```
Trying 128.194.7.20...
Connected to 128.194.7.20.
Escape character is '^]'.


        Welcome to VAX/VMS V5.3

Username: AMOEBA
Password:
        Welcome to VAX/VMS version V5.3 on node MONICA
    Last interactive login on Thursday, 18-APR-1991 14:40
    Last non-interactive login on Tuesday,  9-APR-1991 08:16

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DUA0:                   Mounted              0  VAXVMSRL5         2379   139   1
%TYPE-W-SEARCHFAIL, error searching for SYS$SYSDEVICE:[AMOEBA]WELCOME.TXT;
-RMS-E-FNF, file not found
MONICA> type balls.dat
```

```
  3.000000       3.000000       3.000000       17.62415
  3.000000       3.000000       14.00000       19.16363
  3.000000       3.000000       25.00000       10.19850
  3.000000       14.00000       3.000000       19.16363
  3.000000       14.00000       14.00000       10.19850
  3.000000       14.00000       25.00000       4.367349
  3.000000       25.00000       3.000000       10.19850
  3.000000       25.00000       14.00000       4.367349
  3.000000       25.00000       25.00000       4.554211
  14.00000       3.000000       3.000000       19.16363
  14.00000       3.000000       14.00000       10.19850
  14.00000       3.000000       25.00000       4.367349
  14.00000       14.00000       3.000000       10.19850
  14.00000       14.00000       14.00000       4.367349
  14.00000       14.00000       25.00000       4.554211
  14.00000       25.00000       3.000000       4.367349
  14.00000       25.00000       14.00000       4.554211
  14.00000       25.00000       25.00000       10.82963
  25.00000       3.000000       3.000000       10.19850
  25.00000       3.000000       14.00000       4.367349
  25.00000       3.000000       25.00000       4.554211
  25.00000       14.00000       3.000000       4.367349
  25.00000       14.00000       25.00000       10.82963
  25.00000       25.00000       3.000000       4.554211
  25.00000       25.00000       14.00000       10.82963
  25.00000       25.00000       25.00000       19.52651
MONICA> run starr
 Percent in homogeneous blobs?
45
 Input file name?
balls
 Output file name?
temp
 Here we go
 Dump data?
n
 I read          27
 Oh boy, we have data
 4.6458825E-02          2          2          2   5.388172
 0.1607398              4          2          2   5.440030
 0.7804615              7          2          2   5.509752
  2.574856             19          2          2   6.008701
  4.938468             20          2          2   4.069189
  910.2798             29          2          2   4.070025
  1160.500             30          3          2   26.69777
  1649.536             29          4          2   3.992256
  2480.657             29          5          2   4.393729
  2509.603             30          6          2   32.90219
  3225.109             27          8          2   39.30929
```

```
 3368.889              27          11           2   2.851958
 3696.088              21          14           2   54.78817
 3830.919              11          27           3   2.799880
 4401.765              16          27           3   2.718645
 4599.657              11          22           5   2.713610
 4916.878              27          22           5   2.775974
 5389.722              16          25           5   2.508463
 6040.157              22           5          11   2.787701
 6915.316               5          27          22   2.593717
Down the center: data
 4.192015       4.101766       4.086305       4.477773       4.242034
 3.987885       4.058863       26.63864       46.00188       33.09476
 2.840770       2.688167       2.778892       2.902167       2.696835
 2.701120       41.67275       37.79688       22.37953       27.30846
 36.89902       2.596532       2.835341       2.611711       2.707472
 2.887982       2.695838       48.42439       53.47391       54.88974
 36.03139       51.11856
Down the center: gradient:
0.2917202      1.4465959E-02  0.1863152      6.7582548E-02  0.2725425
4.2965472E-02   513.1008       2216.435       400.3434       1888.532
 2731.161       2784.995       3468.301       2346.421       1901.320
 3477.114       1545.977       693.3981       238.9257       260.4082
 875.5566       3007.539       2302.260       2190.771       1688.093
 3481.299       5705.418       2749.697       262.5401       462.3196
 202.5814       66.00420
Maximum gradient:   6915.316
Gradient formed
Histogram of counts
      1111         120         134         137         120         108
        89          88         101          67          85          87
        78          75          74          73          72          62
        56          62          58          43          37          45
        34          34          32          34          27          21
        21          20          12          18          14          15
        13          14           7          12           5           5
         2           6           8           2           2           3
         5           4           2           1           1           1
         4           2           3           1           1           0
         1           2           0           0           2           0
         0           1           0           1           0           0
         0           0           0           0           0           0
         0           0           0           0           0           0
         0           0           0           0           0           0
         0           0           0           0           0           0
         0           0           0           0
Found a threshold:   349.2483
Boundaries marked
 35.39187       52.49762       34.37973       57.91971       50.60735
 44.08176       28.26985       46.18728       39.66749       49.94583
 27.17790       49.57285       31.75939       16.22164       36.11312
 30.28798       15.71326       40.85241       48.10483       54.05400
 50.10608       50.84894       38.71812       31.19532       34.46811
 40.86628       30.90541       41.06978       23.77833       37.57311
 47.80839       31.42234       32.52766       38.54172       35.83867
 35.45852       30.60879       38.59006       48.96417       48.17749
```

```
 48.81665       39.61263       31.82920       44.17275       37.47644
 33.88279       31.43385       30.24509       33.16746       38.61612
 44.65694       24.07245       51.83775       42.94830       33.40131
 36.13909       35.52843       33.60717       46.51528       30.83201
 35.39187       52.49762       34.37973       57.91971       50.60735
 33.76551       33.91352       42.39333       29.63771       34.29617
 44.08176       28.26985       46.18728       40.25033       24.77884
 18.95086       59.50185       32.19398       38.76727       46.82760
 39.72323       39.69683       26.37075       25.32755       28.58317
 Extracted test sets and initial means
 There are           17 testsets.
 Clusters found: there are:           0
 Classification step finished
 Spatial fixups finished
FORTRAN STOP
MONICA>
```

APPENDIX C

Findp: Program and Output

Findp

```
      integer numberd,ndx(20),ktr(26)
       parameter (numberd=52)
       real mind,mind1,mind2,dis(numberd,numberd),
     +      sim(numberd,numberd),smax,s
       logical changed,found(numberd),readf
       integer counter(numberd,numberd),similar(numberd,numberd)
       integer seed(26,52),nf(26)
       integer nbr(numberd,3)
       integer weirdd(numberd,numberd)
       integer rings(20,10)
       integer rn      ! ring number
       integer pndx(numberd)
       real c(3)
       character*64 filename
       real carbon(3,numberd)
       real pool(3,numberd)
       data carbon/.46141 ,.29111 ,.32291
     + ,.16608 ,.25634 ,.43206
     + ,.16040 ,.25822 ,.10400
     + ,.30720 ,-.05160 ,.17651
     + ,.69619 ,.16068 ,.16331
     + ,.79505 ,.17274 ,.13024
     + ,.86146 ,.21187 ,.16901
     + ,.82901 ,.23895 ,.24084
     + ,.73015 ,.22689 ,.27391
     + ,.66374 ,.18776 ,.23515
     + ,.60226 ,-.02971 ,.32649
     + ,.64252 ,-.10888 ,.37558
     + ,.65167 ,-.08196 ,.44516
     + ,.62055 ,.02414 ,.46564
     + ,.58029 ,.10331 ,.41655
     + ,.57114 ,.07639 ,.34697
     + ,-.16218 ,.36745 ,.44139
     + ,-.25599 ,.34612 ,.47637
     + ,-.25765 ,.24197 ,.49470
     + ,-.16550 ,.15913 ,.47805
     + ,-.07169 ,.18046 ,.44307
     + ,-.07003 ,.28462 ,.42475
     + ,-.03087 ,.48707 ,.30682
     + ,-.05452 ,.59600 ,.29306
     + ,-.01462 ,.66104 ,.33342
     +  ,.04893 ,.61715 ,.38754
     +  ,.07258 ,.50823 ,.40130
     +  ,.03268 ,.44318 ,.36094
     +  ,.14488 ,.56935 ,.12561
     +  ,.09509 ,.66912 ,.09901
     +  ,.08098 ,.68239 ,.02646
     +  ,.11666 ,.59590 ,-.01948
     +  ,.16646 ,.49613 ,.00712
     +  ,.18057 ,.48286 ,.07967
     +  ,.40680 ,.20952 ,.02345
     +  ,.51079 ,.17938 ,-.01289
     +  ,.57960 ,.24570 ,-.01334
     +  ,.54440 ,.34217 ,.02255
     +  ,.44041 ,.37232 ,.05890
     +  ,.37161 ,.30599 ,.05935
     +  ,.27985 ,-.28881 ,.28871
     +  ,.32932 ,-.39133 ,.31096
     +  ,.42587 ,-.41117 ,.34145
```

```
     +  ,.47295 ,-.32849 ,.34969
     +  ,.42348 ,-.22597 ,.32744
     +  ,.32693 ,-.20613 ,.29694
     +  ,.07155 ,-.07371 ,.34792
     + ,-.03848 ,-.06758 ,.36135
     + ,-.10047 ,-.05715 ,.30442
     + ,-.05243 ,-.05285 ,.23407
     +  ,.05760 ,-.05898 ,.22064
     +  ,.11959 ,-.06941 ,.27757 /
        type*,' Data from external file? If so, enter file'
        type*,' name, else enter a semicolan and hit return.'
        read33,filename
        if (filename.ne.';') then
          readf = .true.
          open(1,file=filename,status='old')
          read(1,*) number
          do i = 1,number
            read(1,*,end=11)(carbon(k,i),k=1,3)
          end do
        else
          number = numberd
          readf = .false.
        end if
 11     type*,number
        inpool = 0
        do i = 1,number-1
          do j = i+1,number
c           type*,i,j,dist(carbon(1,i),carbon(1,j))
            dis(i,j) = dist(carbon(1,i),carbon(1,j))
            dis(j,i) = dist(carbon(1,i),carbon(1,j))
          end do
          dis(i,i) = 0
        end do
        if (readf) go to 44
        do k = 5,number,6
          type 88,k,k+1,k+2,k+3,k+4,k+5
          smd = 10.**30
          big = 0.
          do i = k,k+5
            do j = k,k+5
              if (i.ne.j) then
                if (dis(i,j).gt.big) big = dis(i,j)
                if (dis(i,j).lt.smd) smd = dis(i,j)
              end if
            end do
            type 89,i,(dis(i,j),j=k,k+5)
          end do
          type*,' ratio', big/smd
          type*
          call zap(c,3)
          do i = k,k+5
            c(1) = c(1)+carbon(1,i)
            c(2) = c(2)+carbon(2,i)
            c(3) = c(3)+carbon(3,i)
          end do
          c(1) = c(1)/6.
          c(2) = c(2)/6.
          c(3) = c(3)/6.
          do l = 1,number
            if (l.lt.k.or.l.gt.k+5) then
```

```
              type*,dist(carbon(1,l),c(1))
            else
              type*,' in the group ',dist(carbon(1,l),c(1))
            end if
          end do
        end do
c  calculate new exp. similarity measure
 44     smax = 0
        type*,' weight'
        read*,w
c       w = .65 ! found by experiment
        smin = 10.**30
        do i = 1,number-1
          do j = i+1,number
            s = w/dis(i,j)
            do k = 1,number
              if (k.ne.i.and.k.ne.j)
     +          s = s + 1./(dis(i,k)+dis(j,k))
c             do l = 1,number
c               if (l.ne.k.and.l.ne.i.and.l.ne.j)
c     +           s = s + 1./(dis(i,l)*dis(j,l)*dis(k,l))**(1/3)
c             end do
            end do
            sim(i,j) = s
            sim(j,i) = s
            if (smax.lt.s) smax = s
            if (smin.gt.s) smin = s
          end do
        end do
c       do i = 1,number-1
c         do j = i+1,number
c           counter(i,j) = ifix(16.99*(sim(i,j)-smin)/(smax-smin))
c           counter(j,i) = counter(i,j)
c         end do
c       end do
c       type8
c       do i = 1,number
c          type7,i,(counter(i,j),j=1,number)
c       end do
c       type*
        do i = 1,number
c  find the three most similar neighbors.
          near2 = 0.
          mind2 = 0.
          do j = 1,number
            counter(i,j) = 0
            if (i.ne.j) then
              if (sim(i,j).gt.mind2) then
                near = near1
                near1 = near2
                mind2 = sim(i,j)
                near2 = j
              end if
            end if
          end do
            nbr(i,1) = near2
          if (near1.ne.0) then
            nbr(i,2) = near1
          else
            mind2 = 0.
```

```
            do j = 1,number
              if (i.ne.j.and.j.ne.near2) then
                if (sim(i,j).gt.mind2) then
                  near = near1
                  mind2 = sim(i,j)
                  near1 = j
                end if
              end if
            end do
            nbr(i,2) = near1
          end if
          if (near.ne.0) then
            nbr(i,3) = near
          else
            mind2 = 0.
            do j = 1,number
              if (i.ne.j.and.j.ne.near2.and.j.ne.near1) then
                if (sim(i,j).gt.mind2) then
                  mind2 = sim(i,j)
                  near = j
                end if
              end if
            end do
            nbr(i,3) = near
          end if
          write(*,1)i,(nbr(i,k),k=1,3)
c         if (i.lt.nbr(i,1))then
c           counter(i,nbr(i,1))=counter(i,nbr(i,1))+3
c         else
c           counter(nbr(i,1),i)=counter(nbr(i,1),i)+3
c         end if
c         if (i.lt.nbr(i,2))then
c           counter(i,nbr(i,2))=counter(i,nbr(i,2))+2
c         else
c           counter(nbr(i,2),i)=counter(nbr(i,2),i)+2
c         end if
c         if (i.lt.nbr(i,3))then
c           counter(i,nbr(i,3))=counter(i,nbr(i,3))+1
c         else
c           counter(nbr(i,3),i)=counter(nbr(i,3),i)+1
c         end if
        end do
        nseed = 0
        do i = 1,number
          found(i) = .false.
        end do
        do i = 1,number
          if (nbr(nbr(i,1),1).eq.i) then
            found(i) = .true.
            found(nbr(i,1)) = .true.
            nseed = nseed+1
            seed(nseed,1) = i
            seed(nseed,2) = nbr(i,1)
            nbr(i,1) = 0
            nf(nseed) = 2
          end if
        end do
333     changed = .false.
        if (nseed.gt.0) then
          do i = 1,number
```

```
            if (found(i)) go to 444
            it = nbr(i,1)       ! want to see if i points to something
here.
            do j = 1,nseed
 222          do k = 1,nf(j)
                if (i.eq.seed(j,k)) go to 111   ! however, skip it if i
dy in the list.
              end do
              do k = 1,nf(j)
                if (it.eq.seed(j,k)) then
                  found(i) = .true.
                  nf(j) = nf(j)+1
                  seed(j,nf(j)) = i
                  changed = .true.
                  go to 222     ! restart the search??
                end if
              end do
 111        end do      !  if this loop falls through we are through wi
 444      end do
        end if
        if (changed) go to 333
        do i = 1,number
          if (.not.found(i)) then
                type*,i,'not found'! do something...but didn't have to
test.
          end if
        end do
        do j = 1,nseed
          type2,j,(seed(j,k),k=1,nf(j))
        end do
c  now the problem is to get rid of the odd guy (if any)
        do j = 1,nseed
          if (nf(j).gt.14) then
            do k = 1,nf(j)
              nin = 0
              do m = 1,nf(j)
                if (m.ne.k) then
                  if (nbr(seed(j,k),2).eq.seed(j,m)) nin = nin+3
                  if (nbr(seed(j,k),3).eq.seed(j,m)) nin = nin+2
                end if
              end do
              ktr(k) = nin
            end do
            min = 1000
            do k = 1,nf(j)
              if (ktr(k).lt.min) then
                nout = k
                min = ktr(k)
              end if
            end do
            is = 0
            do k = 1,nf(j)
              if (ktr(k).eq.min) is = is+1
            end do
            if (is.eq.1) then
c             nseed = nseed+1
c             nf(nseed) = 1
c             seed(nseed,1) = seed(j,nout)
              inpool = inpool+1
              do k2 = 1,3
```

```
                pool(k2,inpool) = carbon(k2,seed(j,nout))
              end do
              pndx(inpool) = seed(j,nout)
              do k = nout+1,nf(j)
                seed(j,k-1) = seed(j,k)
              end do
              nf(j) = nf(j)-1
            end if
          end if
        end do
        do j = 1,nseed
          if (nf(j).eq.14) then
c  for each set of six...
            amax = 0
            do k = 1,nf(j)-7     ! leave eight out
            do k2 = k+1,nf(j)-6
            do k3 = k2+1,nf(j)-5
            do k4 = k3+1,nf(j)-4
            do k5 = k4+1,nf(j)-3
            do k6 = k5+1,nf(j)-2
            do k7 = k6+1,nf(j)-1
            do k8 = k7+1,nf(j)
              call zap(c,3)      ! compute the centroid of the rest
              do l = 1,nf(j)
                if (l.ne.k.and.l.ne.k2.and.l.ne.k3.and.k4.ne.l
     +          .and.k5.ne.l.and.k6.ne.l.and.k7.ne.l.and
     +          .k8.ne.l) then
                  do m = 1,3
                    c(m) = c(m)+carbon(m,seed(j,l))
                  end do
                end if
              end do
              do m = 1,3
                c(m) = c(m)/6.
              end do
              dt = dist(carbon(1,seed(j,k)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k
              end if
              dt = dist(carbon(1,seed(j,k2)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k2
              end if
              dt = dist(carbon(1,seed(j,k3)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k3
              end if
              dt = dist(carbon(1,seed(j,k4)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k4
              end if
              dt = dist(carbon(1,seed(j,k5)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k5
              end if
```

```
                dt = dist(carbon(1,seed(j,k6)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k6
                end if
                dt = dist(carbon(1,seed(j,k7)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k7
                end if
                dt = dist(carbon(1,seed(j,k8)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k8
                end if
              end do
              end do
              end do
              end do
              end do
              end do
              end do
              end do
c             nseed = nseed+1
c             nf(nseed) = 1
c             seed(nseed,1) = seed(j,nout)
                inpool = inpool+1
                pndx(inpool) = seed(j,nout)
                do k2 = 1,3
                  pool(k2,inpool) = carbon(k2,seed(j,nout))
                end do
              do l = nout+1,nf(j)
                  seed(j,l-1) = seed(j,l)
              end do
              nf(j) = nf(j)-1
            end if
            if (nf(j).eq.13) then
c  for each set of six...
              amax = 0
              do k = 1,nf(j)-6    ! leave seven out
              do k2 = k+1,nf(j)-5
              do k3 = k2+1,nf(j)-4
              do k4 = k3+1,nf(j)-3
              do k5 = k4+1,nf(j)-2
              do k6 = k5+1,nf(j)-1
              do k7 = k6+1,nf(j)
                call zap(c,3)     ! compute the centroid of the rest
                do l = 1,nf(j)
                  if (l.ne.k.and.l.ne.k2.and.l.ne.k3.and.k4.ne.l
     +                    .and.k5.ne.l.and.k6.ne.l) then
                    do m = 1,3
                      c(m) = c(m)+carbon(m,seed(j,l))
                    end do
                  end if
                end do
                do m = 1,3
                  c(m) = c(m)/6.
                end do
                dt = dist(carbon(1,seed(j,k)),c(1))
                if (dt.gt.amax) then
```

```
                    amax = dt
                    nout = k
                end if
                dt = dist(carbon(1,seed(j,k2)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k2
                end if
                dt = dist(carbon(1,seed(j,k3)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k3
                end if
                dt = dist(carbon(1,seed(j,k4)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k4
                end if
                dt = dist(carbon(1,seed(j,k5)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k5
                end if
                dt = dist(carbon(1,seed(j,k6)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k6
                end if
                dt = dist(carbon(1,seed(j,k7)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k7
                end if
              end do
              end do
              end do
              end do
              end do
              end do
              end do
c             nseed = nseed+1
c             nf(nseed) = 1
c             seed(nseed,1) = seed(j,nout)
                inpool = inpool+1
                pndx(inpool) = seed(j,nout)
                do k2 = 1,3
                  pool(k2,inpool) = carbon(k2,seed(j,nout))
                end do
              do l = nout+1,nf(j)
                  seed(j,l-1) = seed(j,l)
              end do
              nf(j) = nf(j)-1
            end if
            if (nf(j).eq.12) then
c  for each set of six...
              amax = 0
              do k = 1,nf(j)-5     ! leave six out
              do k2 = k+1,nf(j)-4
              do k3 = k2+1,nf(j)-3
              do k4 = k3+1,nf(j)-2
```

```
                do k5 = k4+1,nf(j)-1
                do k6 = k5+1,nf(j)
                  call zap(c,3)     ! compute the centroid of the rest
                  do l = 1,nf(j)
                    if (l.ne.k.and.l.ne.k2.and.l.ne.k3.and.k4.ne.l
     +                    .and.k5.ne.l.and.k6.ne.l) then
                      do m = 1,3
                        c(m) = c(m)+carbon(m,seed(j,l))
                      end do
                    end if
                  end do
                  do m = 1,3
                    c(m) = c(m)/6.
                  end do
                  dt = dist(carbon(1,seed(j,k)),c(1))
                  if (dt.gt.amax) then
                      amax = dt
                      nout = k
                  end if
                  dt = dist(carbon(1,seed(j,k2)),c(1))
                  if (dt.gt.amax) then
                      amax = dt
                      nout = k2
                  end if
                  dt = dist(carbon(1,seed(j,k3)),c(1))
                  if (dt.gt.amax) then
                      amax = dt
                      nout = k3
                  end if
                  dt = dist(carbon(1,seed(j,k4)),c(1))
                  if (dt.gt.amax) then
                      amax = dt
                      nout = k4
                  end if
                  dt = dist(carbon(1,seed(j,k5)),c(1))
                  if (dt.gt.amax) then
                      amax = dt
                      nout = k5
                  end if
                  dt = dist(carbon(1,seed(j,k6)),c(1))
                  if (dt.gt.amax) then
                      amax = dt
                      nout = k6
                  end if
                end do
                end do
                end do
                end do
                end do
                end do
c               nseed = nseed+1
c               nf(nseed) = 1
c               seed(nseed,1) = seed(j,nseed)
                  inpool = inpool+1
                  pndx(inpool) = seed(j,nout)
                  do k2 = 1,3
                    pool(k2,inpool) = carbon(k2,seed(j,nout))
                  end do
                do l = nout+1,nf(j)
                    seed(j,l-1) = seed(j,l)
```

```
            end do
            nf(j) = nf(j)-1
          end if
          if (nf(j).eq.11) then
c  for each set of six...
            amax = 0
            do k = 1,nf(j)-4     ! leave five out
            do k2 = k+1,nf(j)-3
            do k3 = k2+1,nf(j)-2
            do k4 = k3+1,nf(j)-1
            do k5 = k4+1,nf(j)
              call zap(c,3)      ! compute the centroid of the rest
              do l = 1,nf(j)
                if (l.ne.k.and.l.ne.k2.and.l.ne.k3.and.k4.ne.l
     +                    .and.k5.ne.l) then
                  do m = 1,3
                    c(m) = c(m)+carbon(m,seed(j,l))
                  end do
                end if
              end do
              do m = 1,3
                c(m) = c(m)/6.
              end do
              dt = dist(carbon(1,seed(j,k)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k
              end if
              dt = dist(carbon(1,seed(j,k2)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k2
              end if
              dt = dist(carbon(1,seed(j,k3)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k3
              end if
              dt = dist(carbon(1,seed(j,k4)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k4
              end if
              dt = dist(carbon(1,seed(j,k5)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k5
              end if
            end do
            end do
            end do
            end do
            end do
c           nseed = nseed+1
c           nf(nseed) = 1
c           seed(nseed,1) = seed(j,nout)
              inpool = inpool+1
              pndx(inpool) = seed(j,nout)
              do k2 = 1,3
                pool(k2,inpool) = carbon(k2,seed(j,nout))
```

```
              end do
            do l = nout+1,nf(j)
                seed(j,l-1) = seed(j,l)
            end do
            nf(j) = nf(j)-1
          end if
          if (nf(j).eq.10) then
c   for each set of six...
            amax = 0
            do k = 1,nf(j)-3     ! leave four out
            do k2 = k+1,nf(j)-2
            do k3 = k2+1,nf(j)-1
            do k4 = k3+1,nf(j)
              call zap(c,3)      ! compute the centroid of the rest
              do l = 1,nf(j)
                if (l.ne.k.and.l.ne.k2.and.l.ne.k3.and.k4.ne.l) then
                  do m = 1,3
                    c(m) = c(m)+carbon(m,seed(j,l))
                  end do
                end if
              end do
              do m = 1,3
                c(m) = c(m)/6.
              end do
              dt = dist(carbon(1,seed(j,k)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k
              end if
              dt = dist(carbon(1,seed(j,k2)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k2
              end if
              dt = dist(carbon(1,seed(j,k3)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k3
              end if
              dt = dist(carbon(1,seed(j,k4)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k4
              end if
            end do
            end do
            end do
            end do
c           nseed = nseed+1
c           nf(nseed) = 1
c           seed(nseed,1) = seed(j,nout)
              inpool = inpool+1
              pndx(inpool) = seed(j,nout)
              do k2 = 1,3
                pool(k2,inpool) = carbon(k2,seed(j,nout))
              end do
            do l = nout+1,nf(j)
                seed(j,l-1) = seed(j,l)
            end do
            nf(j) = nf(j)-1
```

```
          end if
          if (nf(j).eq.9) then
c  for each set of six...
            amax = 0.
            do k = 1,nf(j)-2     ! leave three out
            do k2 = k+1,nf(j)-1
            do k3 = k2+1,nf(j)
              call zap(c,3)      ! compute the centroid of the rest
              do l = 1,nf(j)
                if (l.ne.k.and.l.ne.k2.and.l.ne.k3) then
                  do m = 1,3
                    c(m) = c(m)+carbon(m,seed(j,l))
                  end do
                end if
              end do
              do m = 1,3
                c(m) = c(m)/6.
              end do
              dt = dist(carbon(1,seed(j,k)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k
              end if
              dt = dist(carbon(1,seed(j,k2)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k2
              end if
              dt = dist(carbon(1,seed(j,k3)),c(1))
              if (dt.gt.amax) then
                  amax = dt
                  nout = k3
              end if
            end do
            end do
            end do
c           nseed = nseed+1
c           nf(nseed) = 1
c           seed(nseed,1) = seed(j,nout)
              inpool = inpool+1
              pndx(inpool) = seed(j,nout)
              do k2 = 1,3
                pool(k2,inpool) = carbon(k2,seed(j,nout))
              end do
            do l = nout+1,nf(j)
                seed(j,l-1) = seed(j,l)
            end do
            nf(j) = nf(j)-1
          end if
          if (nf(j).eq.8) then
c  for each set of six...
            amax = 0.
            do k = 1,nf(j)-1     ! leave two out
             do k2 = k,nf(j)
              call zap(c,3)      ! compute the centroid of the rest
              do l = 1,nf(j)
                if (l.ne.k.and.l.ne.k2) then
                  do m = 1,3
                    c(m) = c(m)+carbon(m,seed(j,l))
                  end do
```

```
                  end if
                end do
                do m = 1,3
                  c(m) = c(m)/6.
                end do
                dt = dist(carbon(1,seed(j,k)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k
                end if
                dt = dist(carbon(1,seed(j,k2)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k2
                end if
               end do
              end do
c             nseed = nseed+1
c             nf(nseed) = 1
c             seed(nseed,1) = seed(j,nout)
                inpool = inpool+1
                pndx(inpool) = seed(j,nout)
                do k2 = 1,3
                  pool(k2,inpool) = carbon(k2,seed(j,nout))
                end do
              do l = nout+1,nf(j)
                  seed(j,l-1) = seed(j,l)
              end do
              nf(j) = nf(j)-1
            end if
             if (nf(j).eq.7) then
c  for each set of six...
              amax = 0.
              do k = 1,nf(j)       ! leave one out
                call zap(c,3)      ! compute the centroid of the rest
                do l = 1,nf(j)
                  if (l.ne.k) then
                    do m = 1,3
                      c(m) = c(m)+carbon(m,seed(j,l))
                    end do
                  end if
                end do
                do m = 1,3
                  c(m) = c(m)/6.
                end do
                dt = dist(carbon(1,seed(j,k)),c(1))
                if (dt.gt.amax) then
                    amax = dt
                    nout = k
                end if
              end do
c             nseed = nseed+1
c             nf(nseed) = 1
c             seed(nseed,1) = seed(j,nout)
                inpool = inpool+1
                pndx(inpool) = seed(j,nout)
                do k2 = 1,3
                  pool(k2,inpool) = carbon(k2,seed(j,nout))
                end do
              do l = nout+1,nf(j)
```

```
                seed(j,l-1) = seed(j,l)
              end do
              nf(j) = nf(j)-1
            end if
          end do
          if (inpool.ge.6) then
            number = inpool
            do j = 1,3
              do i = 1,inpool
                carbon(j,i) = pool(j,i)
              end do
            end do
c restart with these after printing.
            type*
            type*,' Groups found so far ...'
            do j = 1,nseed
              type2,j,(seed(j,k),k=1,nf(j))
            end do
            go to 11
          else if (inpool.gt.0) then
            type*
            type*,' Groups found so far ...'
            do j = 1,nseed
              type2,j,(seed(j,k),k=1,nf(j))
            end do
            type*
            if (inpool.gt.0)type*,' These are not in a group'
            do j = 1,inpool
              type2,j,pndx(j)
            end do
          end if

 33       format(a)
 2        format(i3,2x,40i3)
 88       format(2x,6i7)
 89       format(1x,i3,1x,6f7.4)
 8        format(5x,'                    1 1 1 1 1 1 1 1 1 1 ',
     +   '2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 ',
     +   '4 4 4 4 4 4 5 5 5'/
     +   4x,' 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 ',
     +   '3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 ',
     +   '7 8 9 0 1 2')
 7          format(1x,i2,1x,52z2)
 1          format(1x,13I4)
          end
          real function dist(a,b)
          real a(3),b(3)
          dist = 0.
          do i = 1,3
            dist = dist+(a(i)-b(i))**2
          end do
          dist = sqrt(dist)
          return
          end

          subroutine zap(a,n)
          real a(n)
          do i = 1,n
            a(i) = 0
          end do
```

```
return
end
```

Findp Output

```
bright2> telnet 128.194.7.20
Trying 128.194.7.20...
Connected to 128.194.7.20.
Escape character is '^]'.


        Welcome to VAX/VMS V5.3

Username: AMOEBA
Password:
        Welcome to VAX/VMS version V5.3 on node MONICA
    Last interactive login on Thursday, 18-APR-1991 14:45
    Last non-interactive login on Tuesday,  9-APR-1991 08:16

Device                  Device          Error    Volume         Free  Trans Mnt
 Name                   Status          Count     Label        Blocks Count Cnt
DUA0:                   Mounted             0  VAXVMSRL5         2376   139   1
%TYPE-W-SEARCHFAIL, error searching for SYS$SYSDEVICE:[AMOEBA]WELCOME.TXT;
-RMS-E-FNF, file not found
MONICA> sd
MONICA> run findp
 Data from external file? If so, enter file
 name, else enter a semicolan and hit return.
;
          52
        5       6       7       8       9      10
  5   0.0000  0.1049  0.1731  0.1726  0.1333  0.0834
  6   0.1049  0.0000  0.0863  0.1333  0.1667  0.1687
  7   0.1731  0.0863  0.0000  0.0833  0.1687  0.2099
  8   0.1726  0.1333  0.0833  0.0000  0.1049  0.1731
  9   0.1333  0.1667  0.1687  0.1049  0.0000  0.0863
 10   0.0834  0.1687  0.2099  0.1731  0.0863  0.0000
 ratio   2.518283

 0.3371215
 0.6418127
 0.6129234
 0.5208189
 in the group   8.6282626E-02
 in the group   8.3346076E-02
 in the group   0.1049382
 in the group   8.6281128E-02
 in the group   8.3343178E-02
 in the group   0.1049403
 0.3063810
 0.3739187
 0.3883196
 0.3471391
 0.2975718
 0.2699718
 0.9698404
  1.064973
  1.062222
 0.9691162
 0.8686152
```

0.8660532
0.8503419
0.9126476
0.9132637
0.8472840
0.7816249
0.7856509
0.7238655
0.8224595
0.8534200
0.7894360
0.6936812
0.6586781
0.3982403
0.3317176
0.2863543
0.3163954
0.3925098
0.4295547
0.6923202
0.7409720
0.7114174
0.6203169
0.5585796
0.6029915
0.7573878
0.8594168
0.9063084
0.8538952
0.7512286
0.7011728

	11	12	13	14	15	16
11	0.0000	0.1015	0.1388	0.1503	0.1621	0.1125
12	0.1015	0.0000	0.0752	0.1621	0.2249	0.2006
13	0.1388	0.0752	0.0000	0.1125	0.2006	0.2030
14	0.1503	0.1621	0.1125	0.0000	0.1015	0.1388
15	0.1621	0.2249	0.2006	0.1015	0.0000	0.0752
16	0.1125	0.2006	0.2030	0.1388	0.0752	0.0000

ratio 2.991955

0.3379711
0.5164840
0.5973534
0.3783226
0.2967898
0.3676921
0.4001977
0.3603922
0.2859595
0.2548333
in the group 7.5161621E-02
in the group 0.1124452
in the group 0.1014902
in the group 7.5161584E-02
in the group 0.1124452

```
in the group   0.1014903
0.8588144
0.9383804
0.9082348
0.7978216
0.7088068
0.7401204
0.8126745
0.9014487
0.9146012
0.8371202
0.7426279
0.7314651
0.7862124
0.8979306
0.9420315
0.8808371
0.7734148
0.7222019
0.4751616
0.4588591
0.4799675
0.5128325
0.5325630
0.5159658
0.4508483
0.4876290
0.4518676
0.3569372
0.2997278
0.3634570
0.5466185
0.6540291
0.7198058
0.6851465
0.5836367
0.5102568
     17     18     19     20     21     22
17  0.0000 0.1024 0.1664 0.2115 0.2077 0.1250
18  0.1024 0.0000 0.1058 0.2077 0.2500 0.2026
19  0.1664 0.1058 0.0000 0.1250 0.2026 0.2047
20  0.2115 0.2077 0.1250 0.0000 0.1024 0.1664
21  0.2077 0.2500 0.2026 0.1024 0.0000 0.1058
22  0.1250 0.2026 0.2047 0.1664 0.1058 0.0000
ratio   2.442571

0.6406472
0.3311506
0.4813473
0.6334385
0.9154456
 1.017953
 1.066957
 1.016982
0.9138212
```

```
0.8608288
0.8309692
0.8920808
0.8857016
0.8200586
0.7623565
0.7667078
in the group   0.1057722
in the group   0.1250173
in the group   0.1023641
in the group   0.1057749
in the group   0.1250200
in the group   0.1023632
0.3018885
0.3878425
0.4431957
0.4191622
0.3454016
0.2841439
0.5482795
0.6015433
0.6506119
0.6472579
0.6067619
0.5579138
0.7203155
0.8279666
0.8813624
0.8360291
0.7332553
0.6699458
0.7286441
0.8329865
0.9036840
0.8762500
0.7757707
0.6983622
0.4260026
0.3672441
0.3616867
0.4040748
0.4583173
0.4735000
     23     24     25     26     27     28
23  0.0000 0.1123 0.1767 0.1726 0.1417 0.0943
24  0.1123 0.0000 0.0863 0.1417 0.1886 0.1886
25  0.1767 0.0863 0.0000 0.0943 0.1886 0.2246
26  0.1726 0.1417 0.0943 0.0000 0.1123 0.1767
27  0.1417 0.1886 0.1886 0.1123 0.0000 0.0863
28  0.0943 0.1886 0.2246 0.1767 0.0863 0.0000
ratio   2.602268

0.5228370
0.3454710
0.4103922
```

```
0.6946232
0.8119211
0.8993405
0.9349577
0.8841639
0.7944505
0.7575936
0.8311837
0.9159837
0.9080932
0.8165437
0.7297761
0.7363958
0.2688649
0.3596641
0.4348198
0.4494689
0.3922188
0.2895158
in the group   8.6321287E-02
in the group   9.4306767E-02
in the group   0.1123122
in the group   8.6318716E-02
in the group   9.4303660E-02
in the group   0.1123155
0.2604720
0.2875511
0.3535683
0.3846312
0.3788918
0.3252436
0.6167586
0.7213477
0.7412240
0.6603644
0.5491076
0.5242960
0.8853878
0.9969854
 1.049619
0.9953328
0.8817991
0.8237202
0.6289373
0.6216717
0.6204986
0.6185061
0.6259428
0.6351050
     29      30      31      32      33      34
29  0.0000 0.1146 0.1634 0.1502 0.1409 0.1042
30  0.1146 0.0000 0.0751 0.1409 0.2085 0.2058
31  0.1634 0.0751 0.0000 0.1042 0.2058 0.2293
32  0.1502 0.1409 0.1042 0.0000 0.1146 0.1634
33  0.1409 0.2085 0.2058 0.1146 0.0000 0.0751
```

```
34  0.1042 0.2058 0.2293 0.1634 0.0751 0.0000
ratio   3.053133

0.5168346
0.5013439
0.3297131
0.6697809
0.7140642
0.7843630
0.8275298
0.8005663
0.7311445
0.6878421
0.8197653
0.9187382
0.9309915
0.8497087
0.7509524
0.7325141
0.5319006
0.6202481
0.6796764
0.6691324
0.5956778
0.5169901
0.3156753
0.3034965
0.3254029
0.3460691
0.3608167
0.3519293
in the group   7.5086616E-02
in the group   0.1042383
in the group   0.1146309
in the group   7.5087868E-02
in the group   0.1042395
in the group   0.1146324
0.4650539
0.5580047
0.5651314
0.4794129
0.3743489
0.3668361
0.9149598
 1.026898
 1.076047
 1.017449
0.9026543
0.8485794
0.7219574
0.7392245
0.7252344
0.6856784
0.6671527
0.6896936
```

```
     35      36      37      38      39      40
35  0.0000 0.1142 0.1803 0.1911 0.1700 0.1088
36  0.1142 0.0000 0.0956 0.1700 0.2176 0.2015
37  0.1803 0.0956 0.0000 0.1088 0.2016 0.2284
38  0.1911 0.1700 0.1088 0.0000 0.1142 0.1803
39  0.1700 0.2176 0.2016 0.1142 0.0000 0.0956
40  0.1088 0.2015 0.2284 0.1803 0.0956 0.0000
ratio   2.390187

0.3006300
0.5133337
0.3259191
0.3989295
0.2856721
0.3523889
0.4174896
0.4167873
0.3607572
0.2969200
0.4489002
0.5478930
0.5807254
0.5294275
0.4422754
0.3922562
0.7682474
0.8635427
0.8725270
0.7947966
0.6964788
0.6776365
0.6178031
0.6756165
0.6964551
0.6568285
0.5996138
0.5817055
0.4539264
0.5524763
0.5665824
0.4827816
0.3799284
0.3648415
in the group   9.5567234E-02
in the group   0.1087767
in the group   0.1142161
in the group   9.5562533E-02
in the group   0.1087848
in the group   0.1142102
0.6540306
0.7412436
0.7588633
0.6869891
0.5892524
0.5739740
```

0.6253147
0.7047697
0.7224546
0.6568166
0.5708718
0.5574501

	41	42	43	44	45	46
41	0.0000	0.1160	0.1977	0.2064	0.1615	0.0955
42	0.1160	0.0000	0.1032	0.1615	0.1910	0.1857
43	0.1977	0.1032	0.0000	0.0955	0.1857	0.2320
44	0.2064	0.1615	0.0955	0.0000	0.1160	0.1977
45	0.1615	0.1910	0.1857	0.1160	0.0000	0.1032
46	0.0955	0.1857	0.2320	0.1977	0.1032	0.0000

ratio 2.429042

0.6057661
0.6133400
0.6436676
0.3020320
0.5889291
0.6653641
0.7271731
0.7147459
0.6434230
0.5796993
0.3589895
0.3375008
0.3781908
0.4379539
0.4698505
0.4323780
0.8729894
0.9237661
0.8579059
0.7332855
0.6748018
0.7499402
0.8939756
1.002381
1.045657
0.9843845
0.8754086
0.8277283
0.9284191
1.040986
1.074769
1.000189
0.8883347
0.8498302
0.5974039
0.6054062
0.6776252
0.7347047
0.7318283
0.6673279

```
in the group    0.1031749
in the group    9.5500655E-02
in the group    0.1159861
in the group    0.1031759
in the group    9.5500983E-02
in the group    0.1159874
0.3859472
0.4816814
0.5393289
0.5065330
0.4167521
0.3534404
     47      48      49      50      51      52
47  0.0000 0.1110 0.1782 0.1696 0.1289 0.0853
48  0.1110 0.0000 0.0848 0.1289 0.1706 0.1789
49  0.1782 0.0848 0.0000 0.0853 0.1789 0.2220
50  0.1696 0.1289 0.0853 0.0000 0.1110 0.1782
51  0.1289 0.1706 0.1789 0.1110 0.0000 0.0848
52  0.0853 0.1789 0.2220 0.1782 0.0848 0.0000
ratio   2.618008

0.5751342
0.3828247
0.4013503
0.3191126
0.7334319
0.8357884
0.9035051
0.8748468
0.7770073
0.7029162
0.5947101
0.6402128
0.6606217
0.6414452
0.6076587
0.5813889
0.4874850
0.5220048
0.4539543
0.3392665
0.2985591
0.3811291
0.5520599
0.6623901
0.7259642
0.6883720
0.5854591
0.5117896
0.6677458
0.7619602
0.7944201
0.7364668
0.6466391
0.6100587
```

```
0.5511804
0.6343986
0.7162639
0.7228460
0.6551696
0.5666568
0.3520308
0.4585432
0.5448735
0.5371329
0.4462354
0.3480879
 in the group   8.4805638E-02
 in the group   8.5300371E-02
 in the group   0.1110153
 in the group   8.4805682E-02
 in the group   8.5300423E-02
 in the group   0.1110154
 weight
.65
   1  16  10   5
   2  28  22   3
   3  40  34  28
   4  52  46  16
   5  10   9   6
   6  10   5   1
   7  10   5   1
   8  10   5   1
   9  10   5   1
  10   5   1   9
  11  16  15  12
  12  11  10   5
  13  11  10   5
  14  16  15  11
  15  16  11  10
  16  15  11  10
  17  22  21  20
  18  22  17   2
  19  22  21  17
  20  21  17   2
  21  22  20  17
  22  21  17   2
  23  28  27  24
  24  23  22   2
  25  23  22   2
  26  28  23  22
  27  28  23  22
  28  23  22   2
  29  34  33  30
  30  29  23   3
  31  29  28  23
  32  34  29   3
  33  34  29   3
  34  33  29   3
  35  40  39  36
```

```
 36  35  10   5
 37  36  35  10
 38  40  39  35
 39  40  35   1
 40  39  35   3
 41  46  45   4
 42  46  45  41
 43  45  44  41
 44  45  41  11
 45  46  41   4
 46  45  41   4
 47  52  51  48
 48  47  21   4
 49  48  47  21
 50  51  47  21
 51  52  47   4
 52  47   4   2
1    5 10  6  7  8  9
2   15 16  1 11 12 13 14
3   21 22 17 18 19 20
4   23 28  2 24 25 26 27
5   33 34 29 30 31 32
6   39 40  3 35 36 37 38
7   45 46 41 42 43 44
8   47 52  4 48 49 51 50

 Groups found so far ...
 1    5 10  6  7  8  9
 2   15 16 11 12 13 14
 3   21 22 17 18 19 20
 4   23 28 24 25 26 27
 5   33 34 29 30 31 32
 6   39 40 35 36 37 38
 7   45 46 41 42 43 44
 8   47 52 48 49 51 50

 These are not in a group
 1    1
 2    2
 3    3
 4    4
MONICA> run findp
 Data from external file? If so, enter file
 name, else enter a semicolan and hit return.
mh01
         28
 weight
.65
   1   5   3   2
   2  11   4   1
   3   5   1  10
   4  11   2   1
   5  10   6   1
   6   5   1  10
   7   5   1   6
```

```
  8   5   1  10
  9  10   5   3
 10   5   3   1
 11  12   2   1
 12  11   2   1
 13  12  11   2
 14  11   2   1
 15  16  11   2
 16  11   2   1
 17  18  10   5
 18  17  10   5
 19  18  17  10
 20  19  18  17
 21  17  10   5
 22  17  10   5
 23  24  11   2
 24  23  11   2
 25  23  11   2
 26  23  11   2
 27  23  11   2
 28  23  11   2
1    5 10  1  3  6  7  8  9
2   11 12  2  4 13 14 16 15
3   17 18 19 20 21 22
4   23 24 25 26 27 28

Groups found so far ...
1    5 10  6  7  8  9
2   11 12 13 14 16 15
3   17 18 19 20 21 22
4   23 24 25 26 27 28

These are not in a group
1    3
2    1
3    2
4    4
MONICA> run findp
 Data from external file? If so, enter file
 name, else enter a semicolan and hit return.
mh06
%FOR-F-FILNOTFOU, file not found
  unit 1  file SYS$SYSDEVICE:[SHAPES.DAT]MH06.DAT;
  user PC 0000FE7E
-RMS-E-FNF, file not found
%TRACE-F-TRACEBACK, symbolic stack dump follows
module name     routine name                    line       rel PC    abs PC

                                                        00043D9E  00043D9E
                                                        00043CAB  00043CAB
                                                        0003EA38  0003EA38
                                                        0003D49A  0003D49A
FINDP$MAIN      FINDP$MAIN                        74    0000007E  0000FE7E
MONICA> run findp
 Data from external file? If so, enter file
```