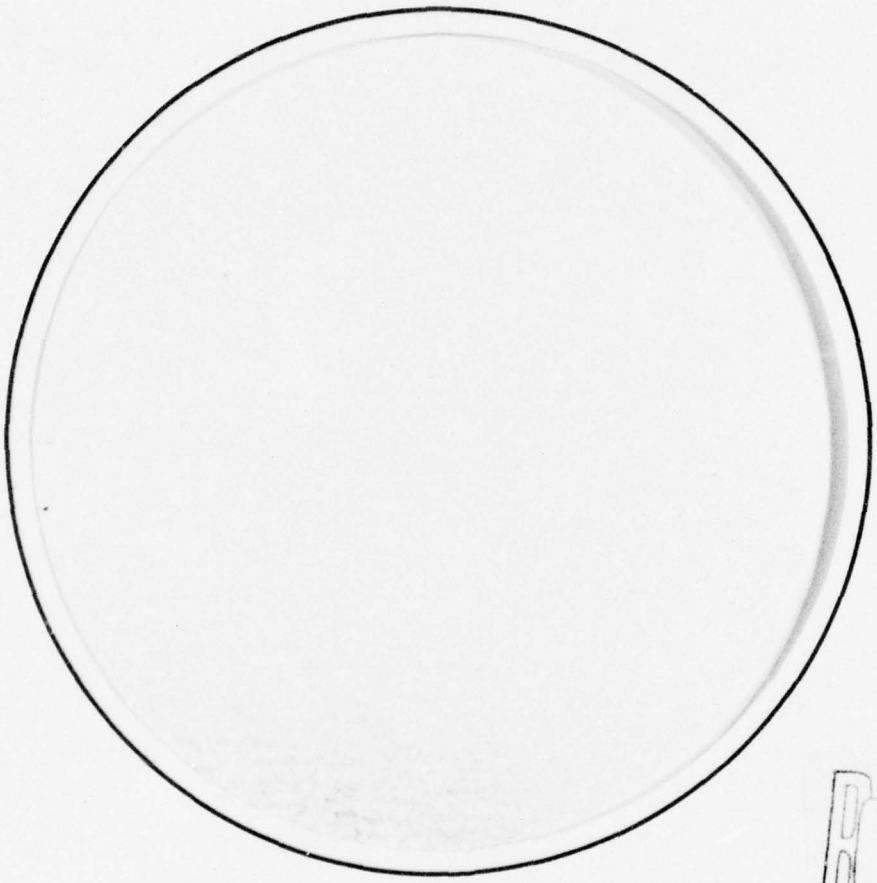
AD-A044 759 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MASS C--ETC F/G 12/1
SOLVING DECOMPOSITION PROBLEMS: ALTERNATIVE TECHNIQUES AND DESC--ETC(U)
SEP 77 R C ANDREU N00039-77-C-0255
UNCLASSIFIED CISR-P010-01-02 NL

| OF |
AD
A044759

END
DATE FILMED
10-77
DDC

(12)

B.S.



Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

Contract No. N00039-77-C-0255

Internal Report No. P010-01-02

Deliverable No. A003

TECHNICAL REPORT #2

SOLVING DECOMPOSITION PROBLEMS: ALTERNATIVE TECHNIQUES AND DESCRIPTION OF SUPPORTING TOOLS

R. C. Andreu

September, 1977

Principal Investigator:

Prof. Stuart E. Madnick

Prepared for:

Naval Electronics Systems Command
Washington, D.C. 20360

LB

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Technical Report #2	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Solving decomposition problems: Alternative techniques and description of supporting tools	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER: P010-01-02
7. AUTHOR(s): Rafael C. Andreu	8. CONTRACT OR GRANT NUMBER(s): N00039-77-C-0255
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center for Information Systems Research, M.I.T. Sloan School of Management - E53 -330, Cambridge, MA, 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Electronic Systems Command, Washington, D.C. 20360	12. REPORT DATE: September 1977
	13. NUMBER OF PAGES: 47
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 54p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

14 CISR-P010-01-02, CISR-TR-2

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DDC SEP 30 1977 C

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Clustering algorithms; Graph decomposition algorithms; Cluster analysis software; Software systems design.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An interactive software system specially designed to solve the kind of set decomposition problems that arise in the investigation of a systematic approach for the early phases of the system development process is presented. Traditional cluster analytic algorithms are available in the system, as well as other more recent approaches.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LB

409 590

PREFACE

The Center for Information Systems Research (CISR) is a research center located and managed in M.I.T.'s Sloan School of Management; it consists of a group of Management Information Systems specialists, including faculty members, full-time research staff and part-time students. The Center's general research thrust is to devise better means for designing, generating and maintaining applications software, information systems and decision support systems.

Within the context of the research effort sponsored by the Naval Electronics Systems Command under contract N00039-77-C-0255, CISR proposed to conduct basic research on a systematic approach to the early phases of complex software systems design, one of the main goals being the development of a well defined methodology aimed at explicitly filling the gap between system requirements and program specifications that characterizes most traditional system design strategies. At the heart of such a methodology is the structuring of the initial set of requirements so as to make apparent the design trade-offs existing among its elements; the decomposition of that set into subsets of strongly interdependent requirements which would define a meaningful framework for system design is the main focus of the proposed methodology. The research project is organized so as to investigate the following four areas:

1) Graph-like representation of requirements sets and suitable decomposition techniques,
2) Design and development of a set of software tools to support the set decomposition activity,
3) Identification of a methodology for the assessment of interdependencies among requirements, as well as guidelines for the interpretation of the obtained decompositions and for the coordination of design subproblems, and
4) Experimental application of the methodology and supporting tools to a specific case, with emphasis on recommendadtions for their practical use and comparison with more traditional design approaches.

This document focuses on the activities carried out at CISR to investigate the second area outlined above.

CONTRACT N00039-77-C-0255

Technical Report #2

EXECUTIVE SUMMARY

This Report describes an interactive software package which incorporates the decomposition techniques discussed in Technical Report #1, along with a number of facilities that permit checking the initial set for consistency, convenient user interfacing, evaluating partitions, saving and restoring problems, and performing traditional cluster analysis with three different algorithms.

The different capabilities of the package are illustrated by means of simple examples, and its usage in the context of a non trivial decomposition problem discussed in some detail.

The package was written in FORTRAN and is currently operational on the Sloan School's PRIME minicomputer.

TABLE OF CONTENTS

SOLVING DECOMPOSITION PROBLEMS: ALTERNATIVE TECHNQIUES AND DESCRIPTION OF SUPPORTING TOOLS

I. Introduction

This report has two main purposes:

a) To describe a software package which incorporates most of the algorithms presented in [Andreu 77] and allows its users to work interactively; and

b) To illustrate the usage of this package for solving a non-trivial graph decomposition problem.

2. The Package

In this section we describe the facilities available in a "graph decomposition" software package that incorporates most of the algorithms described in [Andreu 77] and permits their interactive usage. It has been implemented on the PRIME minicomputer at the Sloan School. Here we emphasize the basic available facilities; their use is illustrated by means of simple examples in Section 3. A non-trivial graph decomposition problem is then solved using the package in Section 4, which thus illustrates its usefulness in more realistic settings.

2.1. Basic facilities

The package was designed to incorporate the algorithms described in [Andreu 77]. Basically, it allows its users to perform the following operations:

(a) Enter the structure of a graph in a convenient manner; the associated adjacency matrix is generated by the package.

(b) Compute a distance matrix for the graph under analysis, using the strategy described in [Andreu 77 -- Section 5.2.2]. A value of p = 1 is assumed, and preclustering (collapsing nodes) is automatically performed as needed. If preclustering occurs, the user has two options for computing a distance matrix: either treating the groups of collapsed nodes as single nodes, or not. The resulting distance matrix can be used in traditional cluster analytic algorithms to partition

the graph. Three agglomerative cluster analytic algorithms are provided in the package.

(c) Compute a similarity matrix from a distance matrix as described in [Andreu 77 -- Appendix IV].

(d) Use the algorithm described in Section 5.3.3 of [Andreu 77] to identify an initial partition of the graph under study, given a percentage parameter.

) Compute a distance matrix from a non-binary similarity matrix (such as that derived in (d) above), and use this iteratively, as proposed in [Andreu 77 -- Sections 5.3.2 and 5.3.2.1].

(f) Evaluate a given graph partition by means of the measure introduced in [Andreu & Madnick 77].

(g) Update a given graph by adding or deleting links among nodes, or by deleting nodes.

(h) Save and restore graphs, partitions and distance matrices to and from secondary storage.

Usage of the package is interactive; a total of 28 commands are currently available to the user to request the operations listed above, to print the different matrices or graph partitions as required, and to perform certain checks on the data defining the problem under analysis (e.g., to check for isolated nodes in a graph).

A brief summary of these commands is presented in Section 2.3 below.

2.2. Structure of the package

The package is characterized by a basic design feature which must be understood in order to be able of using it effectively. It was designed to work with a decomposition problem at a time; this implied a concept of "currency" regarding the different entities that may exist at any point in time, and to which the user can refer to.

The most fundamental entity with which the package deals is a graph. At any point in time, there is a "current graph" on whose decomposition that package is working. A number of commands are available to change the current graph as needed. This currency concept is extended to other entities that may be generated by the package at the user request. Most notably, a current adjacency matrix, distance matrix, similarity matrix and partition may exist. Any user request which requires one of these entities is processed using its current version if it exists; an error message is issued otherwise. For example, a request to print the similarity matrix results in its current version being printed. The current version of any entity is the one last computed or specified.

Furthermore, since the package allows the usage of the different available analytic tools in no predefined order, a notion of "current status" is also needed. The reason is that some requests can't be processed unless others have been processed previously for the current graph. For example, a cluster analytic algorithm can't be used unless a distance matrix is available. In other words, the order in which

the user can issue requests is not completely arbitrary; rather, the requests that can be processed at a given point in time are a function of what has been done so far. The notion of "current status" is useful to summarize what is currently available to the package and, consequently, which commands can be meaningfully processed. Changes in the current status may come about after the processing of certain user requests that change the entities with which the package deals (e.g., distance matrix, etc.). The "current status" can be saved and later re-activated; this allows the user to explore different solutions to a given decomposition problem while minimizing processing.

For the purpose of the discussion here, it is sufficient to think of the current status as a variable that can take five values (status levels), with the following associated meanings:

- Level 0: Nothing is available; the package has just been invoked and no graph has been specified yet;
- Level 1: A graph has been specified and the associated adjacency matrix is available;
- Level 2: In addition to what is available at level 1, a graph partition has been computed or specified;
- Level 3: A distance matrix is also available;
- Level 4: A similarity matrix is available.

In the description of the 28 available requests contained in the next section, we indicate, for each of them, both the minimum status level required for the request to be processable and the status level resulting after its execution. Wherever a request is issued which can't be processed at the time, the message "UNABLE TO

DO IT" is printed in the user's terminal.

2.3. Request command summary

The package is invoked by entering the command "RUN *GRDEC"[1]. Upon invocation, the user is asked if he needs a description of the available command's codes (codes are more ore less mnemonic four letter words). If the answer is affirmative, these are displayed and a prompt ("REQ:") issued for the first request, as shown below.

```
RUN *GRDEC
GO
 DO YOU NEED THE REQUEST CODES (YES OR NO),
YES

 ENGR: ENTER GRAPH DATA; FORGET CURRENT GRAPH,
 REST: READ IN A STATUS PREVIOUSLY SAVED,
 PRNO: PRINT CURRENT NUMBER OF NODES,
 PRAD: PRINT ADJACENCY MATRIX,
 ISOL: CHECK FOR ISOLATED NODES,
 NOLK: PRINT LINKS ASSOCIATED WITH EACH NODE,
 DELK: DELETE GRAPH LINKS,
 DENO: DELETE GRAPH NODES (NODES RENUMBERED),
 ADLK: ADD GRAPH LINKS,
 CLON: COMPUTE PRECLUSTERING; P = 1 ASSUMED,
 NWPA: SPECIFY A PARTITION,
 DIMS: COMPUTE DISTANCE MATRIX AND PERFORM PRECLUSTERING,
       CLUSTERS CONSIDERED AS SINGLE NODES; P=1 ASSUMED,
 DIMN: AS "DIMS" BUT CLUSTERS NOT CONSIDERED SINGLE NODES,
 SIDI: COMPUTE DISTANCE MATRIX FROM CURRENT SIMILARITY MATRIX,
 ITER: ITERATE ON DISTANCE MATRIX,
 SIMA: COMPUTE SIMILARITY MATRIX FROM CURRENT DISTANCE MATRIX,
 INPA: COMPUTE AN INITIAL PARTITION FROM CURRENT SIMILARITY MATRIX;
       PERCENTAGE VALUE WILL BE REQUESTED,
 HCM1: HIERARCHICAL CLUSTERING (METHOD 1),
 HCM2: HIERARCHICAL CLUSTERING (METHOD 2),
 HCM3: HIERARCHICAL CLUSTERING (METHOD 3),
 EVAL: EVALUATE CURRENT CLUSTERING,
 PRLK: PRINT LINKS AMONG CLUSTERS,
 PRCL: DISPLAY CURRENT CLUSTERING,
 PRDI: PRINT CURRENT DISTANCE MATRIX,
 PRSI: PRINT CURRENT SIMILARITY MATRIX,
 SAVE: SAVE CURRENT STATUS,
 DEFI: DELETE A FILE PREVIOUSLY SAVED,
 QUIT: QUIT,

 REQ:
```

---

[1] In the Sloan School's PRIME minicomputer.

At this point, the current status level is 0.

Each command is described at some length below. Examples of their usage can be found in Section 3.

1) ENGR.

Minimum status level required (MSL): 0.

Resulting status level (RSL): 1.

Upon typing "ENGR" in response to the "REQ:" prompt, the system prompts the user to enter graph data by printing ":". Graph data must be entered according to the following format:

*N/ni;nj_1,nj_2, ... / .../$ (embedded blanks allowed),

where "N" is the number of nodes in the graph, and a group of the form "/ni;nj_1, ... /" indicates that there are links between node number "ni" and nodes "nj_1", "nj_2", etc. "$" is used to indicate end of data. Data can be entered in more than one line as long as the last character in each line is "/" (except for the last line, which must end with "$"; the system will keep prompting the user after a line ending with "/" is entered). Input data is checked for correct format and to ensure that no node number is greater than the specified number of nodes (the system assumes that nodes are assigned the consecutive numbers 1, ... , N); if an error is detected, the user is notified and the input process must begin starting at the first line.

Note that since the package deals with undirected graphs whose adjacency matrices are symmetric, links need to be specified only in one direction. An orderly and often convenient way of entering data

is to specify "/ni;nj_1,nj_2, ... /" groups in increasing order of "ni" values; only "nj" values such that nj > ni need then to be included in each group (main diagonal entries of the adjacency matrix are automatically set to 1).

2) REST.

MSL: 0.

RSL: 1, 2 or 3. (See "SAVE" below.)

The user will be asked to specify a file name from where to read graph data; such a file must have been specified previously in a "SAVE" command.

3) PRNO.

MSL: 1.

RSL: Unchanged.

The dimension of the current graph is printed.

4) PRAD.

MSL: 1.

RSL: Unchanged.

5) ISOL.

MSL: 1.

RSL: Unchanged.

Any isolated node in the current graph (a node with no links associated with it) is printed.

6) NOLK.

MSL: 1.

RSL: Unchanged.

A listing is produced where each node in the current graph is shown with the nodes it is linked to. A measure of the "link density" of the current graph, the "average number of links per node" is also printed.

7) DELK.

MSL: 1.

RSL: 1.

The user is prompted (":") to enter the links to be deleted. Input data format is as that for "ENGR", with the exception that no "*N" part is required. Note that the resulting status level is 1, since the adjacency matrix is modified.

8) DENO.

MSL: 1.

RSL: 1.

The user is prompted (":") to enter the node numbers to be deleted. Input data formal should be of the form

$$/n_1,n_2,\ldots,n_i/ \quad ,$$

where $n_1,\ldots,n_i$ are the node numbers to be deleted. Embedded blanks are allowed and data can be entered in more than one line as long as each one ends with "," (the last one should end with "/"). If the nodes being deleted are other than the last i nodes, the remaining ones are renumbered; the newly assigned numbers are printed out. Attempts to remove all the nodes or nodes whose number is greater than the current number of nodes will fail.

9) ADLK.

MSL: 1.

RSL: 1.

System prompt and input data format as in "DELK.".

10) CLON.

MSL: 1.

RSL: 2.

A "partition" is identified where members of each subgraph are collapsed nodes. Usually this partition won't have practical interest as an end result, but it will give insight about the structure of the graph under analysis.

11) NWPA.

MSL: 1.

RSL: 2.

The user is prompted to enter a partition. Input data format should be of the form

$$/n_i,.../n_j,.../.../\$ \quad ,$$

where the node numbers between a pair of consecutive "/" indicate a subgraph. Embedded blanks are allowed as well as multi-line input, as long as each line ends with "/" or ",". Every node should appear in the input data, otherwise it won't be accepted. Note that the resulting status level is 2: otherwise, since the partition is externally specified, it might create inconsistencies with preclustering derived from collapsing nodes. Note also that this means that any available distance or similarity matrix will be lost after executing this command; these should be saved for future reference (see "SAVE").

12) DIMS.

MSL: 1.

RSL: 3.

13) DIMN.

As DIMS.

14) SIDI.

MSL: 4.

RSL: 3.

15) ITER.

MSL: 3.

RSL: 3.

16) SIMA.

MSL: 3.

RSL: 4.

17) INPA.

MSL: 4.

RSL: 4.

18) HCM1, HCM2, HCM3.

MSL: 3.

RSL: Unchanged.

These commands all use the current distance matrix to perform agglomerative cluster analysis. They generate a "clustering trace" in the form of a hierarchical tree which shows how nodes are successively

clustered together; the tree may be optinally printed. The successive partitions produced by this trace are evaluated and the best one retained as "current partition".

The commands differ on the cluster analytic method employed, regarding the criteria used, at a given point in the node merging process, to decide what clusters to merge next, as follows:

- Method 1 (HCM1) merges the "closest" pair of clusters, measuring the distance between two clusters A and B by the mean of the distances between nodes of A and nodes of B; that is,

$$d(A,B) = \frac{1}{n_A n_B} \Sigma s(a,b) \quad ,$$

where n_A and n_B are the number of elements in A and B respectively, and the summation is over all the elements $a \in A$ and $b \in B$.

- Method 2 (HCM2) merges the pair of clusters that lead to the minimum mean of the distances between all pairs of nodes in the cluster resulting from the merger. That is, at each step it chooses to merge the two clusters that, once merged, produce the minimum $\underline{x}$:

$$x = \frac{1}{n_A^2} \Sigma s(a,a') \quad ,$$

where n_A is the number of nodes in the merger A; $a,a' \in A$ and the summation is over <u>all</u> pairs of nodes (main diagonal entries of the distance matrix S are included in it).

- Method 3 (HCM3), finally, merges the two clusters A and B that lead to the minimum y:

$$y = \frac{1}{n_A n_B} \Sigma s(a,a') - \frac{1}{n_A} \Sigma s(a,a') - \frac{1}{n_B} \Sigma s(b,b') ,$$

where the first summation is over all pairs of nodes in A and B, the second over all pairs in A, and the third over all pairs in B.

19) EVAL.

MSL: 2.

RSL: Unchanged.

The current partition is measured and the result printed out, along with its components, "strength" and "coupling".

20) PRLK.

MSL: 2.

RSL: Unchanged.

The links joining nodes belonging to different subgraphs in the current partition are printed out. This is helpful to figure out how the components of the current partition interact.

21) PRCL.

MSL: 2.

RSL: Unchanged.

22) PRDI.

MSL: 3.

RSL: Unchanged.

23) PRSI.

MSL: 4.

RSL: Unchanged.

24) SAVE.

MSL: 1.

RSL: Unchanged.

The user is prompted to specify a file name where the current status will be saved. The file name may be up to 6 characters long, beginning with a letter. Depending upon the current status level, the following is saved away in the specified file:

- Status level 1: Current adjacency matrix.
- Status level 2: Current adjacency matrix plus current partition.
- Status level 3 or 4: Current adjacency matrix, plus current distance matrix, plus any preclustering that might have occurred while computing the distance matrix. The similarity matrix is never saved because its recomputation ("SIMA") is very fast.

25) DEFI.

MSL: 0.

RSL: Unchanged.

The specified file is deleted.

```
26) QUIT.

      MSL:  0.

Work session is ended.  Current status is lost if not previously saved.
```

3. A Sample Session

This section illustrates the usage of the package with a simple graph. We work with the graph depicted in Figure 1.

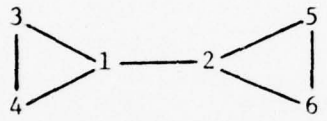


Figure 1

The first thing we do is to enter the graph structure through a "ENGR" command:

```
ENGR
:
*6/1;2,3,4/2;5,6/3;4/5;6/$

 NEW GRAPH ENTERED.
```

At this point, a current adjacency matrix is available. It is often safe to display it ("PRAD") and to request information about the current graph so as to check that the entered graph is indeed the correct one:

```
REQ:
PRAD

CURRENT ADJACENCY MATRIX:

     1  2  3  4  5  6

1:   1  1  1  1  0  0
2:   1  1  0  0  1  1
3:   1  0  1  1  0  0
4:   1  0  1  1  0  0
5:   0  1  0  0  1  1
6:   0  1  0  0  1  1
```

```
 REQ:
PRNO

 CURRENT NUMBER OF NODES:    6.

 REQ:
NOLK

 RECORDED LINKS.
 FROM NODE      TO NODE(S):

      1    (  3)  2,  3,  4,
      2    (  3)  1,  5,  6,
      3    (  2)  1,  4,
      4    (  2)  1,  3,
      5    (  2)  2,  6,
      6    (  2)  2,  5,

 (AVERAGE NO. OF LINKS PER NODE:      2.333).

 REQ:
ISOL

 NO ISOLATED NODES.
```

Using "CLON", we next specify that we wish to perform preliminary clustering (a value of p = 1 is assumed by the system):

```
 REQ:
CLON

* PRECLUSTERING PERFORMED WITH P = 1.
```

To display the current clustering (partition), we use "PRCL":

```
 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 --------------------
    1     ( 1)  1
    2     ( 1)  2
    3     ( 2)  3  4
    4     ( 2)  5  6
```

We can also evaluate it ("EVAL"):

```
 REQ:
EVAL

 STRENGTH:   0.0000,
 COUPLING:   3.0000,
 MEASURE:   -3.000,

 REQ:
```

Next, we can compute a distance matrix treating collapsed nodes not as single nodes ("DIMN"), display it ("PRDI"), compute a similarity matrix from it ("SIMA"), display this matrix ("RPSI"), compute an initial partition with a percentage parameter of 80% ("INPA") which can then be evaluated ("EVAL"):

```
DIMN
 (PRECLUSTERING COMPLETE)

 PRECLUSTERING PERFORMED AND DISTANCE MATRIX COMPUTED WITH P = 1.
 CLUSTERS NOT TAKEN AS SINGLE NODES.

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

        1    2    3    4

  1: 0.00 0.67 0.25 0.83
  2: 0.67 0.00 0.83 0.25
  3: 0.25 0.83 0.00 1.00
  4: 0.83 0.25 1.00 0.00

 REQ:
SIMA

 SIMILARITY MATRIX COMPUTED.

 REQ:
PRSI

 CURRENT SIMILARITY MATRIX:

        1    2    3    4

  1: 1.00 0.33 0.75 0.17
  2: 0.33 1.00 0.17 0.75
  3: 0.75 0.17 1.00 0.00
  4: 0.17 0.75 0.00 1.00

 REQ:
INPA
 ENTER PERCENTAGE PARAMETER:
80.

 INITIAL PARTITION COMPUTED WITH P = 80.00 %.
```

```
 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 ------------------------
    1     ( 3)  1  3  4
    2     ( 3)  2  5  6

 REQ:
EVAL

 STRENGTH:   0.6667,
 COUPLING:   0.1111,
 MEASURE:    0.556.

 REQ:
```

To save the current status, we use "SAVE" (note that a checking is performed so as not to destroy an existing file if the user doesn't want to):

```
 SAVE
  ENTER FILE NAME:
 SIXNOD

  FILE SIXNOD ALREADY EXISTS.
  DO YOU WANT TO DESTROY IT ?
 NO
  ENTER FILE NAME:
 NO6ALL

  STATUS SAVED IN FILE NO6ALL
```

A distance matrix can also be computed by taking collapsed nodes as single nodes ("DIMS"):

```
 REQ:
DIMS
 (PRECLUSTERING COMPLETE)

 PRECLUSTERING PERFORMED AND DISTANCE MATRIX COMPUTED WITH P = 1,
 CLUSTERS TAKEN AS SINGLE NODES.

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

        1     2     3     4

  1: 0.00 0.50 0.33 0.75
  2: 0.50 0.00 0.75 0.33
  3: 0.33 0.75 0.00 1.00
  4: 0.75 0.33 1.00 0.00
```

We now use the distance matrix saved previously to perform cluster analysis. Fisrt, we restore the status saved above ("REST"):

```
 REQ:
REST
 ENTER FILE NAME:
NO6ALL
 ADJACENCY MATRIX, PRECLUSTERING AND DISTANCE MATRIX READ FROM FILE NO6ALL

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

        1     2     3     4

  1: 0.00 0.67 0.25 0.83
  2: 0.67 0.00 0.83 0.25
  3: 0.25 0.83 0.00 1.00
  4: 0.83 0.25 1.00 0.00
```

With the distance matrix shown above, we perform hierarchical cluster analysis by means of the commands "HCM1", "HCM2" and HCM3". Note that the system informs the user of the measure associated with the best identified partition, and gives him a chance to print the hierarchical clustering trace (tree); the best identified partition becomes the "current partition", which can therefore be printed through "PRCL":

```
 REQ:
HCM1
 BEST PARTITION MEASURE:     0.556
 DO YOU WANT TO PRINT THE TREE?
NO

 REQ:

PRCL

 CLUSTER (NO) OBJECTS
 ----------------------
    1     ( 3)  1  3  4
    2     ( 3)  2  5  6

 REQ:
HCM2
 BEST PARTITION MEASURE:     0.556
 DO YOU WANT TO PRINT THE TREE?
NO

 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 ----------------------
    1     ( 3)  1  3  4
    2     ( 3)  2  5  6

 REQ:
HCM3
 BEST PARTITION MEASURE:     0.556
 DO YOU WANT TO PRINT THE TREE?
YES
 SET PAPER AND PRESS RETURN:
```

If the user wants to print the tree, it comes out as shown below:

```
LEVEL:     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
=============================================================================================================
     2  ---¦-------------------------------------------------------------------------------------------------
 (* 1*) ---*
     1  ---------------¦-------------------------------------------------------------------------------------
 (* 2*) ---------------*
=============================================================================================================
```

COLLAPSED OBJECTS:

(* 1*): 5 6
(* 2*): 3 4

MEASURES:

-1.000 0.556 0.133

Note that objects (nodes) collapsed while computing the distance matrix are treated as single nodes during the clustering process, as determined by the fact that only one entry in the distance matrix exists for each group of nodes, but that information is given regarding what nodes belong to each group.

The measures shown below the tree correspond to the partitions resulting at each clustering step, from left to right in the printed tree. In the example above, for instance, -1.00 is the measure associated with the partition {2,5,6},{1},{3,4}; 0.556 with {2,5,6},{1,3,4} and 0.133 with {1,2,3,4,5,6}.

At this point, the following observation must be made regarding the format of the printed tree. Asterisks ("*") are printed to indicate when a cluster ends, so as to facilitate the reading of the partition associated with a given level in the clustering trace, as follows: Given a level in the trace (say level 10 in the tree shown above), the partition associated with this level has as many components (subgraphs) as horizontal lines would be crossed by a vertical line joining the level numbers at the top and bottom of the diagram (two in the case above); to read off the nodes belonging to each subgraph, one backtracks from the point at which each horizontal line is crossed (from right to left) and looks for the bottom-most asterisks in any previous cluster. This strategy would result in the partition {2,5,6},{1,3,4} for the example above.

Another observation regarding the printed tree is that only 25 clustering levels are printed in any case, due to space constraints. Level values determined by the clustering level employed are rounded off and assigned to the closest level of the 25 printed. This can result in clusterings involving more than two subclusters for output purposes, which would seem to contradict the basic assumption of any hierarchical clustering method; it should be understood that such contradiction is only apparent, the clustering process proceeds always one step at a time.

"PRLK" allows to print the links existing among subgroups in the current partition:

```
 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 ---------------------
    1     ( 3)  1  3  4
    2     ( 3)  2  5  6

 REQ:


PRLK

 LINKS BETWEEN CLUSTERS    1 &    2 :
                    1 -    2
```

(See Figure 1 and check that the printed link -- from node 1 to node 2 -- is the only one between the subgraphs {1,3,4} and {2,5,6}).

To illustrate the use of "SIDI" we recompute the similarity matrix associated with the current distance matrix (recall that the similarity matrix is never saved):

```
 REQ:
SIMA

 SIMILARITY MATRIX COMPUTED.

 REQ:
SIDI

 DISTANCE MATRIX COMPUTED FROM SIMILARITY MATRIX.
 (NO CLUSTERING PERFORMED).

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

       1    2    3    4

 1! 0.00 0.71 0.33 0.81
 2! 0.71 0.00 0.81 0.33
 3! 0.33 0.81 0.00 0.90
 4! 0.81 0.33 0.90 0.00
```

The iterative procedure proposed in [Andreu 77] can be performed by means of "ITER", and clustering performed:

```
 REQ:
ITER
 ENTER NUMBER OF ITERATIONS:
100
 EPSILON:
0.001

 CLUSTERING AT ITERATION  18:

 CLUSTER (NO) OBJECTS
 ---------------------
    1     ( 3)  1  3  4
    2     ( 3)  2  5  6

 CONTINUE?
NO

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

        1     2

  1: 0.00 0.43
  2: 0.43 0.00
```

We now save the current status in file "SIXNOD". Since this file flready exists (see above), we first delete it through "DEFI" (alternatively, we could specify that we wish to destroy its current version after issuing a "SAVE" command, see above):

```
 REQ:
DEFI
 ENTER FILE NAME:
SIXNOD

 FILE SIXNOD DELETED.

 REQ:
SAVE
 ENTER FILE NAME:
SIXNOD

 STATUS SAVED IN FILE SIXNOD

 REQ:
```

We now illustrate how the structure of the current graph can be changed. Recall that our current graph is that depicted in Figure 1. To transform it to that shown in Figure 2,

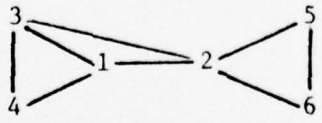


Figure 2

We add the link 2 — 3 by means of "ADLK":

```
ADLK
:
/2:3/$

 GRAPH UPDATED.

 REQ:
NOLK

 RECORDED LINKS.
 FROM NODE      TO NODE(S):

      1    (  3)  2,  3,  4,
      2    (  4)  1,  3,  5,  6,
      3    (  3)  1,  2,  4,
      4    (  2)  1,  3,
      5    (  2)  2,  6,
      6    (  2)  2,  5,

 (AVERAGE NO. OF LINKS PER NODE:      2.667).
```

Next, we investigate how the resulting distance matrix differs from that obtained above through "ITER". Note that since we updated the current graph, the initial distance matrix must be recomputed; we do it through "DIMN" below and see that a different preclustering than that obtained before results:

```
 REQ:
ITER
 UNABLE TO DO IT.

 REQ:
DIMN
 (PRECLUSTERING COMPLETE)

 PRECLUSTERING PERFORMED AND DISTANCE MATRIX COMPUTED WITH P = 1,
 CLUSTERS NOT TAKEN AS SINGLE NODES.

 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 --------------------
    1     ( 2)  1  3
    2     ( 1)  2
    3     ( 1)  4
    4     ( 2)  5  6

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

       1    2    3    4

 1: 0.00 0.50 0.25 0.83
 2: 0.50 0.00 0.67 0.40
 3: 0.25 0.67 0.00 1.00
 4: 0.83 0.40 1.00 0.00

 REQ:
```

Now we can use "ITER" (note that intermediate clusterings differ from those obtained previously, but that the end partition is the same and that the final distance matrix reflects the fact that the "coupling" between the two subgraphs in the final partition is now higher):

```
ITER
 ENTER NUMBER OF ITERATIONS:
100
 EPSILON:
0.001

 CLUSTERING AT ITERATION  15:

 CLUSTER (NO) OBJECTS
 ----------------------
    1     ( 3)  1  3  4
    2     ( 1)  2
    3     ( 2)  5  6

 CONTINUE?
YES

 CLUSTERING AT ITERATION  16:

 CLUSTER (NO) OBJECTS
 ----------------------
    1     ( 3)  1  3  4
    2     ( 3)  2  5  6

 CONTINUE?
NO

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

        1    2

  1: 0.00 0.13
  2: 0.13 0.00
```

Next, we illustrate the use of "DELK" and "DENO" to transform the current graph (Figure 2) into that shown in Figure 3.

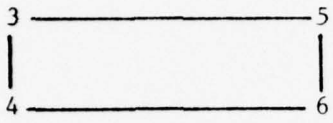


Figure 3

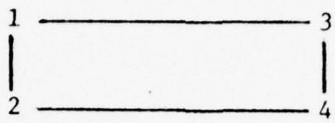


Figure 4

First, we delete the unneeded links and add the new ones:

```
 REQ:
DELK
:
/1;2,3,4/2;3,5,6/$

 GRAPH UPDATED.

 REQ:
ADLK
:
/3;5/4;6/$

 GRAPH UPDATED.
```

To obtain the graph in Figure 3, we must delete nodes 1 and 2. We first check that they are currently isolated nodes ("ISOL"), and then delete them ("DENO"):

```
 REQ:
ISOL

 ISOLATED NODES:
   1
   2

 REQ:
```

```
DENO
:
/1,2/

 THE FOLLOWING NODES HAVE BEEN REMOVED:
    1     2

 NODES HAVE BEEN RENAMED AS FOLLOWS:
 OLD NO.  NEW NO.
 ---------------
    3        1
    4        2
    5        3
    6        4
```

Note that the remaining nodes were re-numbered; the system always numbers nodes starting at 1. So, our current graph is now that shown in Figure 4, which is structurally equivalent to that in Figure 3 except for node labelling. The structure can be visually checked through "PRAD" and "NOLK":

```
 REQ:
PRAD

 CURRENT ADJACENCY MATRIX:

      1  2  3  4

 1:   1  1  1  0
 2:   1  1  0  1
 3:   1  0  1  1
 4:   0  1  1  1

 REQ:
NOLK

 RECORDED LINKS.
 FROM NODE      TO NODE(S):

      1    (  2)  2,  3,
      2    (  2)  1,  4,
      3    (  2)  1,  4,
      4    (  2)  2,  3,

 (AVERAGE NO. OF LINKS PER NODE:      2.000).
```

Any available algorithm can now be applied to this graph. For example, we can compute a distance matrix:

```
 REQ:
DIMN
 (PRECLUSTERING COMPLETE)

 NO PRECLUSTERING PERFORMED; DISTANCE MATRIX COMPUTED WITH P = 1.

 REQ:
PRDI

 CURRENT DISTANCE MATRIX:

       1     2     3     4

 1: 0.00  0.50  0.50  0.50
 2: 0.50  0.00  0.50  0.50
 3: 0.50  0.50  0.00  0.50
 4: 0.50  0.50  0.50  0.00
```

This sample session has illustrated the use of all the available commands. We thus quit ("QUIT"):

```
 REQ:
QUIT

 QUIT.

OK.
```

4. Solving a Non-trivial Graph Decomposition Problem

In this section we attack a less trivial graph decomposition problem. The graph employed is that presented in [Andreu & Madnick 77], which for convenience is reproduced in Figure 5.

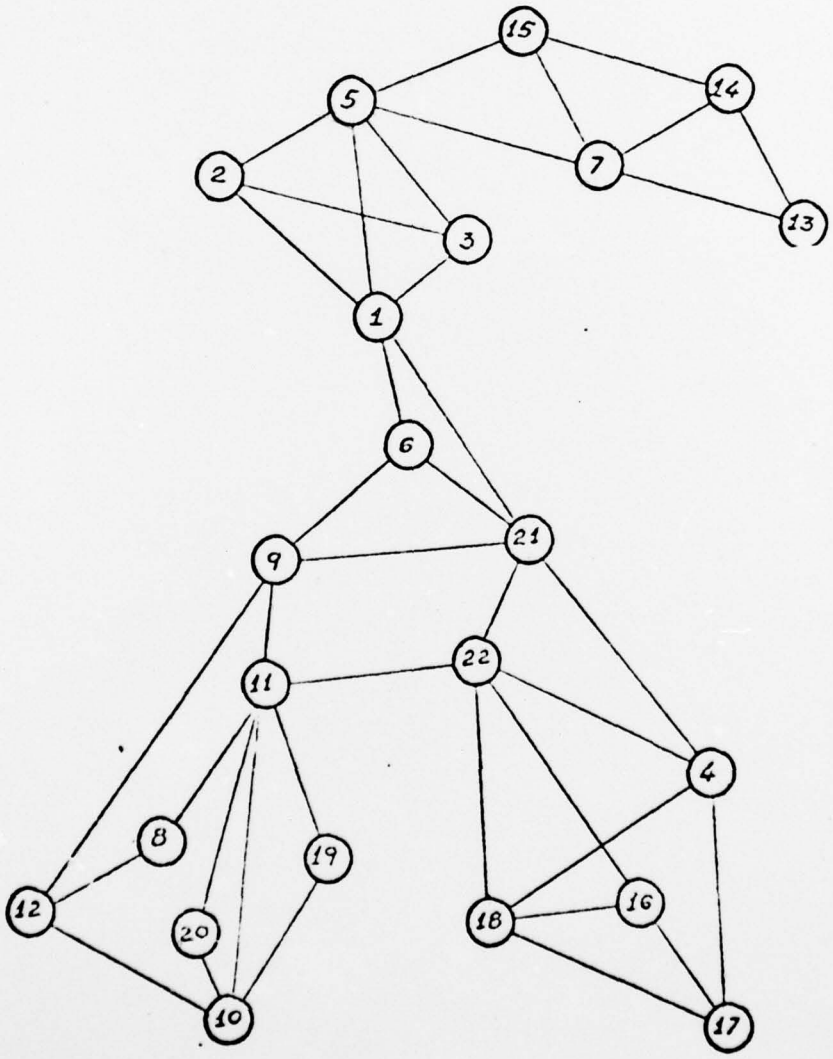


Figure 5

With the aid of the package introduced above, we computed a distance matrix as shown below:

```
 REQ:
ENGR
:
*22/1;2,3,5,6,21/2;3,5/3;5/4;17,18,21,22/5;7,15/6;9,21/
:
7;13,14,15/8;11,12/9;11,12,21/10;11,12,19,20/11;19,20,22/
:
13;14/14;15/16;17,18,22/17;18/18;22/21;22/$

 NEW GRAPH ENTERED.

 REQ:
NOLK

 RECORDED LINKS.
 FROM NODE      TO NODE(S):

      1   (  5)  2,  3,  5,  6, 21,
      2   (  3)  1,  3,  5,
      3   (  3)  1,  2,  5,
      4   (  4) 17, 18, 21, 22,
      5   (  5)  1,  2,  3,  7, 15,
      6   (  3)  1,  9, 21,
      7   (  4)  5, 13, 14, 15,
      8   (  2) 11, 12,
      9   (  4)  6, 11, 12, 21,
     10   (  4) 11, 12, 19, 20,
     11   (  6)  8,  9, 10, 19, 20, 22,
     12   (  3)  8,  9, 10,
     13   (  2)  7, 14,
     14   (  3)  7, 13, 15,
     15   (  3) .5,  7, 14,
     16   (  3) 17, 18, 22,
     17   (  3)  4, 16, 18,
     18   (  4)  4, 16, 17, 22,
     19   (  2) 10, 11,
     20   (  2) 10, 11,
     21   (  5)  1,  4,  6,  9, 22,
     22   (  5)  4, 11, 16, 18, 21,

 (AVERAGE NO. OF LINKS PER NODE:     3.545).
```

```
 REQ:
DIMN
 (PRECLUSTERING COMPLETE)

 PRECLUSTERING PERFORMED AND DISTANCE MATRIX COMPUTED WITH P = 1,
 CLUSTERS NOT TAKEN AS SINGLE NODES.

 REQ:
```

```
PRCL
```

CLUSTER (NO)	OBJECTS
1 (1)	1
2 (2)	2 3
3 (1)	4
4 (1)	5
5 (1)	6
6 (1)	7
7 (1)	8
8 (1)	9
9 (1)	10
10 (1)	11
11 (1)	12
12 (1)	13
13 (1)	14
14 (1)	15
15 (1)	16
16 (1)	17
17 (1)	18
18 (1)	19
19 (1)	20
20 (1)	21
21 (1)	22

```
 REQ:
```

```
PRDI

 CURRENT DISTANCE MATRIX:

      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21
 1: 0.00 0.33 0.90 0.50 0.57 0.90 1.00 0.78 1.00 1.00 1.00 1.00 1.00 0.89 1.00 1.00 1.00 1.00 1.00 0.67 0.91
 2: 0.33 0.00 1.00 0.33 0.86 0.87 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.86 1.00 1.00 1.00 1.00 1.00 0.89 1.00
 3: 0.90 1.00 0.00 1.00 0.87 1.00 1.00 0.89 1.00 0.91 1.00 1.00 1.00 1.00 0.50 0.50 0.33 1.00 1.00 0.62 0.43
 4: 0.50 0.33 1.00 0.00 0.89 0.62 1.00 1.00 1.00 1.00 1.00 0.87 0.75 0.57 1.00 1.00 1.00 1.00 1.00 0.91 1.00
 5: 0.57 0.86 0.87 0.89 0.00 1.00 1.00 0.50 1.00 0.90 0.86 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.33 0.89
 6: 0.90 0.87 1.00 0.62 1.00 0.00 1.00 1.00 1.00 1.00 1.00 0.40 0.20 0.20 1.00 1.00 1.00 1.00 1.00 1.00 1.00
 7: 1.00 1.00 1.00 1.00 1.00 1.00 0.00 0.67 0.67 0.75 0.60 1.00 1.00 1.00 1.00 1.00 1.00 0.80 0.80 1.00 0.87
 8: 0.78 1.00 0.89 1.00 0.50 1.00 0.67 0.00 0.75 0.80 0.71 1.00 1.00 1.00 1.00 1.00 1.00 0.86 0.86 0.62 0.78
 9: 1.00 1.00 1.00 1.00 1.00 1.00 0.67 0.75 0.00 0.50 0.71 1.00 1.00 1.00 1.00 1.00 1.00 0.40 0.40 1.00 0.90
10: 1.00 1.00 0.91 1.00 0.90 1.00 0.75 0.80 0.50 0.00 0.62 1.00 1.00 1.00 0.90 1.00 0.91 0.57 0.57 0.82 0.82
11: 1.00 1.00 1.00 1.00 0.86 1.00 0.60 0.71 0.71 0.62 0.00 1.00 1.00 1.00 1.00 1.00 1.00 0.83 0.83 0.89 1.00
12: 1.00 1.00 1.00 0.87 1.00 0.40 1.00 1.00 1.00 1.00 1.00 0.00 0.25 0.60 1.00 1.00 1.00 1.00 1.00 1.00 1.00
13: 1.00 1.00 1.00 0.75 1.00 0.20 1.00 1.00 1.00 1.00 1.00 0.25 0.00 0.40 1.00 1.00 1.00 1.00 1.00 1.00 1.00
14: 0.89 0.86 1.00 0.57 1.00 0.20 1.00 1.00 1.00 1.00 1.00 0.60 0.40 0.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
15: 1.00 1.00 0.50 1.00 1.00 1.00 1.00 1.00 1.00 0.90 1.00 1.00 1.00 1.00 0.00 0.40 0.20 1.00 1.00 0.89 0.57
16: 1.00 1.00 0.50 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.40 0.00 0.20 1.00 1.00 0.89 0.57
17: 1.00 1.00 0.33 1.00 1.00 1.00 1.00 1.00 1.00 0.91 1.00 1.00 1.00 1.00 0.20 0.20 0.00 1.00 1.00 0.78 0.43
18: 1.00 1.00 1.00 1.00 1.00 1.00 0.80 0.86 0.40 0.57 0.83 1.00 1.00 1.00 1.00 1.00 1.00 0.00 0.50 1.00 0.87
19: 1.00 1.00 1.00 1.00 1.00 1.00 0.80 0.86 0.40 0.57 0.83 1.00 1.00 1.00 1.00 1.00 1.00 0.50 0.00 1.00 0.87
20: 0.67 0.89 0.62 0.91 0.33 1.00 1.00 0.62 1.00 0.82 0.89 1.00 1.00 1.00 0.89 0.89 0.78 1.00 1.00 0.00 0.67
21: 0.91 1.00 0.43 1.00 0.89 1.00 0.87 0.78 0.90 0.82 1.00 1.00 1.00 1.00 0.57 0.57 0.43 0.87 0.87 0.67 0.00

 REQ:
```

The rows and columns of the distance matrix on page 38 are numbered according to the assigned cluster nodes shown on page 37.

We use two main approaches for partitioning the graph, as follows:

1) The three hierarchical cluster analysis methods (available in the package through "HCM1", "HCM2" and "HCM3") are used, and

2) We employ the iterative procedure available through "ITER" to successively partition the graph.

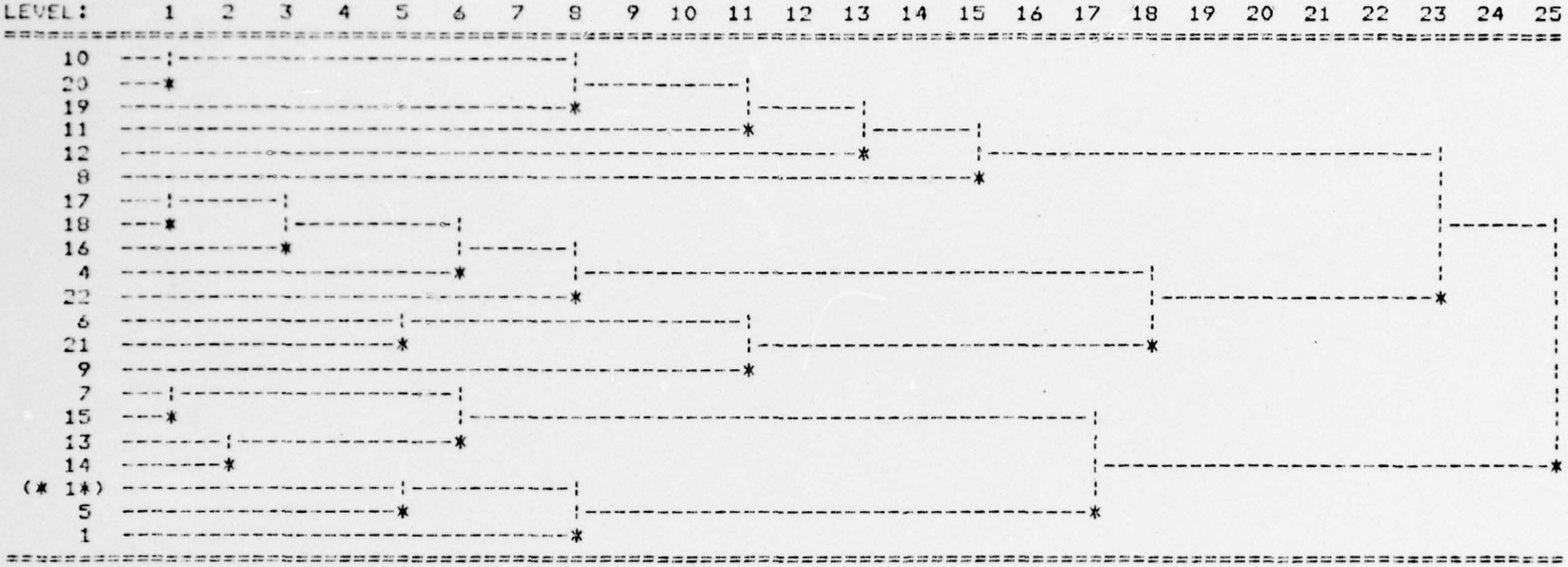
The results are presented in the next two sections.

4.1. Cluster analysis approach.

The results obtained with "HCM1", "HCM2" and "HCM3" are shown in the next three pages, where the hierarchical clustering trees (traces) are depicted.

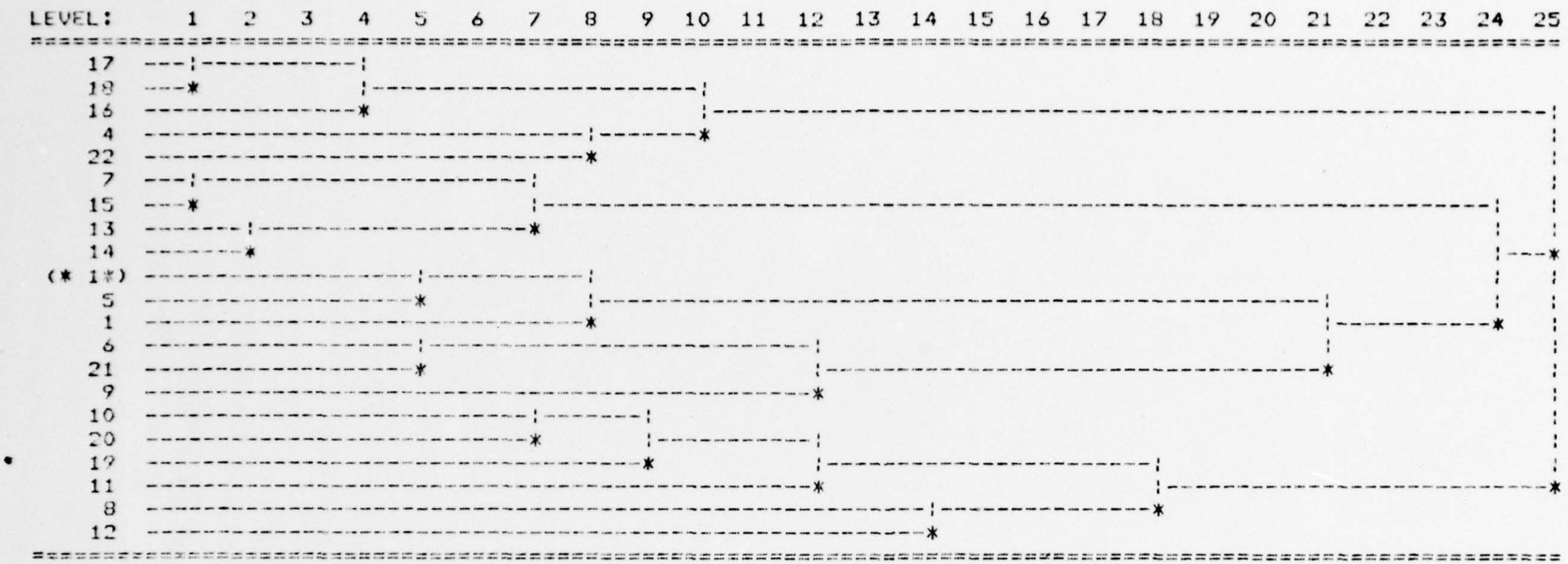
As can be seen, all three methods produced the same graph partition (Figure 6), which coincides with that presented in [Andreu & Madnick 77].

```
 REQ:
HCM1
 BEST PARTITION MEASURE:    1.197
 DO YOU WANT TO PRINT THE TREE?
YES
 SET PAPER AND PRESS RETURN:
```



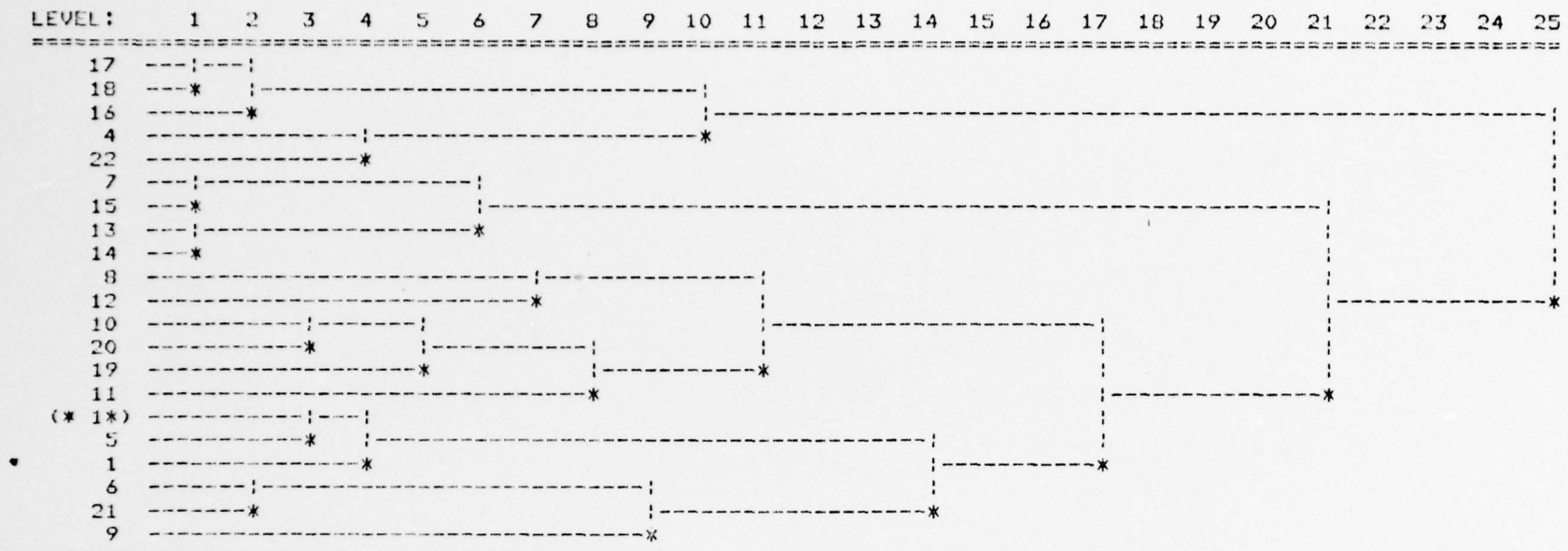

```
 COLLAPSED OBJECTS:

  (* 1*):    2    3


 MEASURES:

    -33.000   -29.500   -26.000   -24.250   -21.750   -17.750   -14.750   -13.500   -11.542    -9.583
     -7.917    -5.875    -2.808    -0.492     0.768     1.197     0.787     0.535     0.317     0.078

 REQ:
```

```
HCM2
 BEST PARTITION MEASURE:    1.197
 DO YOU WANT TO PRINT THE TREE?
YES
 SET PAPER AND PRESS RETURN:
```




```
 COLLAPSED OBJECTS:

  (* 1*):    2    3


 MEASURES:

   -32.500   -29.000   -27.250   -24.750   -20.750   -17.750   -16.500   -13.500   -11.542    -8.875
    -7.208    -5.875    -2.808    -0.492     0.925     1.197     0.962     0.645     0.277     0.078

 REG:
```

```
HCM3
 BEST PARTITION MEASURE:    1.197
 DO YOU WANT TO PRINT THE TREE?
YES
 SET PAPER AND PRESS RETURN:
```




```
COLLAPSED OBJECTS:

 (* 1*):    2    3


MEASURES:

  -32.500   -29.000   -27.250   -24.750   -20.750   -17.750   -14.750   -12.083   -10.083    -8.417
   -7.208    -5.042    -2.167    -0.083     0.925     1.197     0.962     0.764     0.453     0.078

REQ:
```

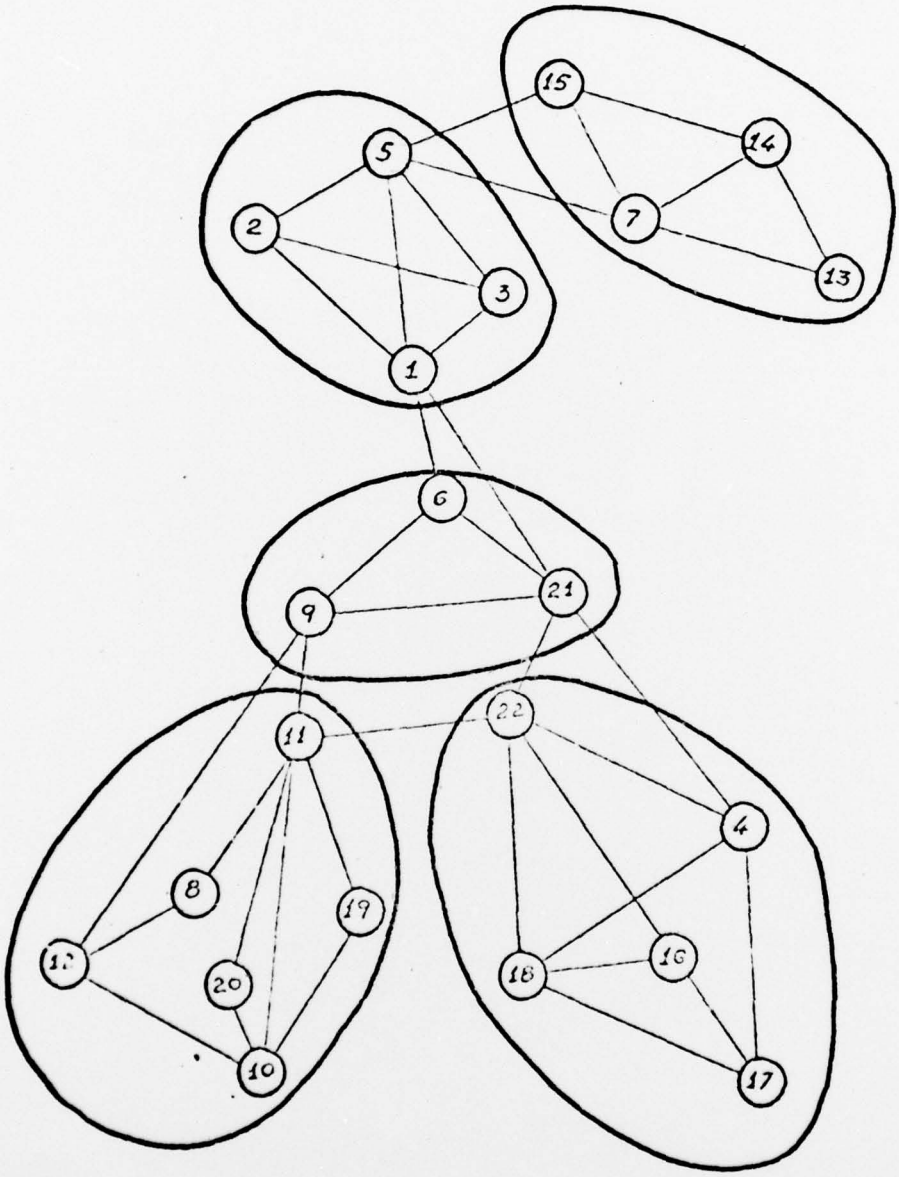


Figure 6

4.2. Iterative approach.

Starting with the distance matrix shown on page 38, we used "ITER" and "DENO" to successively partition the graph. The first steps of this procedure are illustrated below:

```
ITER
 ENTER NUMBER OF ITERATIONS:
100
 EPSILON:
0,01

 CLUSTERING AT ITERATION    4:

 CLUSTER (NO) OBJECTS
 ----------------------

    1     ( 1)  1
    2     ( 2)  2  3
    3     ( 1)  4
    4     ( 1)  5
    5     ( 1)  6
    6     ( 1)  7
    7     ( 1)  8
    8     ( 1)  9
    9     ( 1) 10
   10     ( 1) 11
   11     ( 1) 12
   12     ( 1) 13
   13     ( 1) 14
   14     ( 1) 15
   15     ( 1) 16
   16     ( 1) 17
   17     ( 1) 18
   18     ( 2) 19 20
   19     ( 1) 21
   20     ( 1) 22

 CONTINUE?
YES
```

.
.
.

```
 CLUSTERING AT ITERATION  18:

 CLUSTER (NO) OBJECTS
 --------------------
    1     (18)  1  4  6 21 22  9 16 18 17  8 11 12 10 19 20  2  3  5
    2     ( 4)  7 13 14 15

 CONTINUE?
NO

 REQ:
DENO
:
/7,13,14,15/

 THE FOLLOWING NODES HAVE BEEN REMOVED:
    7   13   14   15

 NODES HAVE BEEN RENAMED AS FOLLOWS:
 OLD NO. NEW NO.
 ---------------
    1       1
    2       2
    3       3
    4       4
    5       5
    6       6
    8       7
    9       8
   10       9
   11      10
   12      11
   16      12
   17      13
   18      14
   19      15
   20      16
   21      17
   22      18


 REQ:
DIMN
 (PRECLUSTERING COMPLETE)

 PRECLUSTERING PERFORMED AND DISTANCE MATRIX COMPUTED WITH P = 1,
 CLUSTERS NOT TAKEN AS SINGLE NODES.

 REQ:
```

```
 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 --------------------
    1     ( 1)  1
    2     ( 3)  2  3  5
    3     ( 1)  4
    4     ( 1)  6
    5     ( 1)  7
    6     ( 1)  8
    7     ( 1)  9
    8     ( 1) 10
    9     ( 1) 11
   10     ( 1) 12
   11     ( 1) 13
   12     ( 1) 14
   13     ( 1) 15
   14     ( 1) 16
   15     ( 1) 17
   16     ( 1) 18

 REQ:
ITER
 ENTER NUMBER OF ITERATIONS:
100
 EPSILON:
0.01

          .

          .

          .

 CLUSTERING AT ITERATION  13:

 CLUSTER (NO) OBJECTS
 --------------------
    1     (12)  1  2  3  5  6 17 18  4  8 12 14 13
    2     ( 6)  7 10 11  9 15 16

 CONTINUE?
NO
```

Figure 7 summarizes the successive "cuts" made to the graph; the numbers next to each cut indicate the chronological order in which they were made. The resulting partition coincides with the one obtained previously (Figure 6).

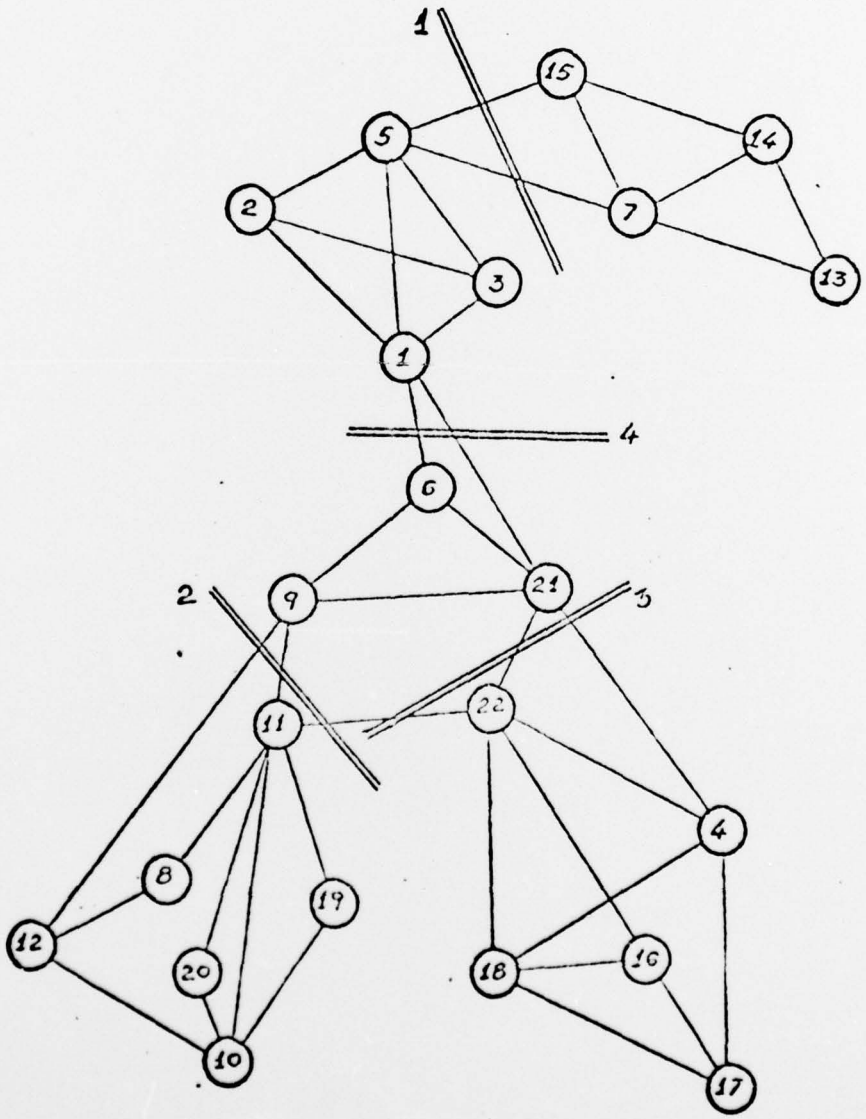


Figure 7

REFERENCES

[Andreu 77]:
Andreu, R.C.: "Set decomposition: Cluster analysis and graph decomposition techniques", internal report, M.I.T. Sloan School,Technical Report #1, Sept. 1977.

[Andreu & Madnick 77]:
Andreu, R.C. and Madnick, S.E.: "A systematic approach to the design of complex systems: Application to DBMS design and evaluation", C.I.S.R. report #32, M.I.T., Sloan School, March, 1977.