A PROCEDURE AND PASCAL PROGRAM FOR CLIQUE DETECTION.

F.W.Wilmink

Summary.

Several algorithms for clique-detection have been proposed. Among these, the algorithms of Needham, Moody and Hollis (Jardine and Sibson, 1971) and of Peay (1974) are rather well-known for their efficiency. In this article, it is shown that both procedures can be derived from the same basic consideration. A new algorithm is presented, which is also based on this consideration, and which is shown to be more efficient than the mentioned procedures. This algorithm has been programmed in PASCAL; as an example, it used 42 seconds computation time to detect 680 cliques from a set of 114 elements. Finally, a somewhat different approach is suggested to the detection of structure in a set of elements over which a symmetric relation is defined.

1. Introduction.

Given a set E of n elements, a clique (maximal complete subgraph, ML-set) is defined as a subset of E with the property than between each pair of elements of this subset there exists some specified relationship; moreover, it is not contained in another subset of E having the same property.

Algorithms for the detection of cliques ("clique-detection procedures") have been proposed by Harary and Ross (1957); Needham, Moody and Hollis (appendix 5 of: Jardine and Sibson, 1971); Rattinger (1973); and Peay (1970; 1974) [x]). The last three algorithms have been programmed in FORTRAN IV (the Needham - Moody - Hollis - algorithm by Shafto, 1974). The method of Harary and Ross is very cumbersome and does not lend itself to the solution of large problems. Rattingers method, like that of Harary and Ross, is recursive; in both methods, the concept of clique is restricted to subsets containing at least three elements. For a problem with n = 60 Rattingers method needed six minutes computation time (Rattinger, p. 10). Its capacity is limited, however, because it requires computation of a square (nxn)-matrix.

[x]) As starting material for further statistical analyses, cliques are computed in a program written by Alba (1972). How this is done is not known to this author; Peay (1974, p. 60) calls Alba's method "quite lengthy and complex".

The other two mentioned algorithms do not require computation of such a matrix; nevertheless, Shafto's program is limited to sets with $n \leq 60$, whereas Peay (1974, p. 63) says that when $n \geq 35$, the problem should be broken down into subproblems. No information is given about their computational efficiency.

In the next section, we shall show that the methods of Needham, Moody and Hollis and of Peay are based on the same principle. Further, we shall present a new clique-detection procedure, also based on this principle, which we developed unaware of the Needham-Moody-Hollis and Peay algorithms, and which is much more efficient then these. Lastly, we shall make a remark about the usefulness of clique-detection, and suggest a somewhat different approach to the problem of cliques.

2. The new algorithm for clique-detection.

Notation.

Logically, one should make a distinction between a set, like $\{1,3,4\}$, and a symbolic representation of the membership of some elements of E of this set, like (101100). (101100) is a binary bector, indicating that of six elements in total, elements 1,3 and 4 belong to the set $\{1,3,4\}$, whereas $\{1,3,4\}$ is the set itself. Indeed, Jardine and Sibson make this distinction. We feel, however, that this distinction, though correct, is rather confusing, so we shall have (101100) to be a binary vector, indicating the set $\{1,3,4\}$.
Further, we shall speak of "verbs" instead of "elements", and of "synonymous with" instead of "having some speci-

fied relation with". This is only to simplify propositions. "Synonimity" is assumed to be a symmetric relationship.
A symmetric (nxn)-matrix S represents the synonimities between the verbs of E; if $s_{ij} = 1$, then verbs i and j are synonyms, if $s_{ij} = 0$, they are not.
We shall indicate the Needham-Moody-Hollis algorithm by "method 1", the algorithm of Peay, specifically his "downward procedure", by "method 2".

The algorithm is based on the consideration that two verbs i and j cannot be members of the same clique if they are not synonymous with each other. (This consideration also forms the basis for methods 1 and 2, as will become apparent). Hence, given some (sub)set A, which includes verbs i and j, we have to construct two subsets A_i and A_j, with $A_i = A - \{i\}$, and $A_j = A - \{j\}$, by which A is replaced. (thus, A_i contains all verbs of A except verb i, and A_j contains all verbs of A except verb j). This is the core step in method 2.

This basic consideration has two important implications. First, all the information necessary to detect the cliques in E is contained in the zero entries of S. To use this information most efficiently, our first step[x] is to permutate rows and columns of S so that in the resulting matrix T rowtotals increase from the first to the n'th row.(This step does not limitate the maximum size of n, as is explained in the next section). Second, because S, and therefore T, is symmetric, all essential information is contained in the above-diagonal part of T, so we only have to consider entries t_{ij} of T with $j > i$. (this implica-

[x]) following a suggestion of drs F.B.Brokken

tion also follows from the second of two theorems we shall present below). This also means that we do not have to consider the n'th row of T.

Before we continue the description of the algorithm, we introduce another notational convention. If A is some subset of E (A may be identical to E), $A_{a,b,\ldots,g}$ will denote the subset $A - \{a,b,\ldots,g\}$. This means that $A_{a,b,\ldots,g}$ contains all verbs of A except the verbs a, b,...,g, as far as these verbs are contained in A. So if $\{a,b,\ldots,g\} \cap A = \emptyset$, then $A_{a,b,\ldots,g} = A$.

As long as we have not used information from T, we may assume that all verbs are synonymous. That is, there is only one clique: E.(Method 1 and method 2 (at the lowest criterion level) have the same start). However, inspection of the first row of T may reveal that, for instance, $t_{1,3} = 0$. This means that we have to replace E by E_1 and E_3. Suppose, further, that $t_{1,5} = 0$. It is clear that this new information has no implication for E_1; however, E_3 is replaced by $E_{3,1}$ and $E_{3,5}$, so that now three subsets have been formed: E_1, $E_{3,1}$ and $E_{3,5}$. Going on the same way, working through the above-diagonal part of T, we would end up with a number of subsets of E, which would indeed meet the requirements, posed by the zero entries in T. After checking each subset on containment in other subsets, we would have all the cliques in E (called the clique-set of E).

Obviously, the procedure described above is not efficient. The efficiency comes from two theorems. Formal proves are given in Wilmink (1976), and will be sent on request. The first theorem states that if in row i of T entries $k_1, k_2, \ldots, k_m$ are zero ($k_1 > i$, $m-1 \leq n-k_1$), the effect of

replacing a subset A by the subsets A_i and $A_{k_1,k_2,\ldots,k_m}$ is the same as first replacing A by A_i and A_{k_1}; then A_{k_1} by $A_{k_1,i}$ and A_{k_1,k_2}; then A_{k_1,k_2} by $A_{k_1,k_2,i}$ and A_{k_1,k_2,k_3}; etcetera. It is not difficult to verify that the subsets resulting from the latter procedure are all contained in A_i or in $A_{k_1,k_2,\ldots,k_m}$, and that A_i and $A_{k_1,k_2,\ldots,k_m}$ are endresults of that procedure.

This first theorem allows us to inspect complete rows of T instead of single elements. As the reader may already have noted, we now are very close to method 1: deleting element i from a subset - as we do - is of course the same as logically intersect that subset (represented by a binary vector) with a binary vector having 1's in all places except the i'th place - as is done in method 1; similarly, deleting from a subset those elements of a row of T, for which the entry is zero - as we do - is the same as logically intersect that subset (represented by a binary vector) with that row of T (which is a binary vector, too) - as is done in method 1. That we only look at the above-diagonal part of T does not affect the truth of these statements, because every entry t_{ij} with $j < i$ has already been treated as entry t_{ji}, with $i > j$, as is easy to verify.

It has now been shown that method 1 and method 2 can both be derived from the same basic consideration; we have formulated this consideration at the beginning of this section. Apart from the formation of T from S,

which greatly increases the efficien y of the algorithm, our gain over Peay's method is the usage of complete rows of T instead of single entries. The gain of our method over the Needham-Moody-Hollis algorithm stems from our second theorem.

The second theorem states that, upon inspection of row i of T, with entries $k_1, k_2, \ldots, k_m$ being zero, ($k_1 > i$, $m-1 \leq n-k_1$), a subset A only needs to be replaced by A_i and $A_{k_1,k_2,\ldots,k_m}$ if verb i is contained in A and at least one of the verbs $k_1, k_2, \ldots, k_m$ is contained in A. It is easily seen that if one of these conditions is not satisfied, one of the two resulting subsets is identical to the original one, in which of course the other resulting subset is contained; so that replacement is unnecessary. The same is true, of course, if both conditions are not satisfied. If m = 0, which means that the above-diagonal part of the i-th row does not comtain zero entries, then this row contains no information, so no replacement has to take place. - In fact, this theorem also states that one only has to look at the above-diagonal part of T, as is easy to verify.

If, when replacement of a subset A is necessary, always first A_i and then $A_{k_1,k_2,\ldots,k_m}$ are inserted into the list of subsets on the positon of A, then no subset can be contained in subsets with a previous position in the list (see also the example below). This consequence of the followed procedure reduces the number of checks on containment of subsets in other subsets considerably.

We now have given a complete description of the algorithm. We want to make three futher notes. First, when

the number of newly constructed subsets exceeds a fixed number DIFMAX (chosen by the user of the algorithm), the list of subsets is checked on the presence of subsets contained in other subsets; these are eliminated.- Second, when $n > 59$, only parts of two subsets can be compared (see next section). If these parts show no containment, then the rests of the two subsets need not be compaired, which increases the efficiency of the procedure.- Third, our method is suited to the construction of hierarchical clique structures in the same way Peay's method is, using the cliqueset at a certain level as point of depature for the analysis at the next level. Adaptation of the computerprogram, which is written to operate at the lowest criterion level only, to this possibility requires only slight modifications.

Example.

$$T = \begin{pmatrix} 1000101 \\ 0100110 \\ 0011011 \\ 0011011 \\ 1100111 \\ 0111111 \\ 1111111 \end{pmatrix}$$

; we have put the diagonal elements equal to 1. Harary and Ross (1957,p. 205) and Rattinger (1973,p. 5) use zero's, "by convention", Jardine and Sibson (1971,p. 238) 1's, also "by convention". Of course, the matter is trivial.

We start with E: (1111111).
According to row 1, we replace E by the subsets (0111111) and (1000101).
According to row 2, we replace (0111111) by (0011111) and (0100110); (1000101) needs not to be replaced.
According to row 3, we replace (0011111) by (0001111) and (0011011); (0100110) needs not to be replaced,

nor does (1000101) .

According to row 4, we replace (0001111) by (0000111) and (0001011); (0011011) needs no replacement, nor do (0100110) or (1000101).

Because rows 5 and 6 do not contain zero entries, we are now ready with the replacement procedure, and we have got the subsets $\begin{pmatrix} 0000111 \\ 0001011 \\ 0011011 \\ 0100110 \\ 1000101 \end{pmatrix}$. Note the ordering of the subsets. Subset 2 is contained in subset 3, so subset 3 is deleted from the list. The remaining subsets are the cliques in E. These are the sets $\{5,6,7\}$, $\{3,4,6,7\}$, $\{2,5,6\}$ and $\{1,5,7\}$.

3. The computerprogram.

The program has been written in PASCAL (Jensen and Wirth, 1975[11]). This language is already available at many computerinstallations.

We defined type "binaryvector" as:

type binaryvector = array[1..a] of set of 0..maxbit;

(cf. Jensen and Wirth, p.53-54).

In this definition, maxbit is 1 less than the number of bits in one memoryword of the computer (implementation-dependent; we take this number to be 59), and a is the number of parts that result from dividing a vector of length n into parts of 59 (a = n div (maxbit +1) + 1). Each part is stored in one memoryword, thus allowing a very efficient storage of information. Moreover, on sets (as the parts are called) several operations are possible, like checks on set-inclusion, and these operations are relatively fast, compared to other

types of operations.

We shall now give a stepwise description of the program.

1. Permutationstep. This step is not included into the program because usually such routines do already exist. We note, however, that a matrix of practically any order n can be permutated by using an array [1..n] of files, each file containing one row of the matrix to be permutated. By using the type "binaryvector", matrices up to order n = 1700 can be permutated in a memory of $200{,}000_8$ words (about 60,000 decimal).
2. Replacementstep. Two files are used. After a row of T has been read, all subsets on one file are written to the other file, if necessary after replacement. Then the next row of T is read, and the subsets are copied back to the first file, if necessary after replacement. A new row of T is read , and the subsets are again written to the other file, if necessary after replacement; etcetera. We start with a vector (11...1) of length n (representing E) on file 1.
3. Eliminationstep. If, after completion of the writing of subsets from one file to the other, the total number of replacements since the last eliminationstep (or the start of the program) exceeds the value of DIFMAX as defined by the user, every vector (subset) which is contained in another vector (subset) is eliminated from the list of already formed vectors (subsets). Then the program returns to step 2.
4. The program terminates when the replacementstep for the (n-1)-th row of T has been executed. If necessary, an eliminationstep is executed. The remaining vectors are copied to a file RES, after transformation from "binaryvector"-form to a readable form. RES may be copied to OUTPUT and/or stored on disk, tape or cards.

5. Output. After every replacement- and eliminationstep the most recent number of vectors and the computation-time already used are recorded. At the end of the program, the number of detected cliques and the greatest number of vectors used are recorded.

Example.

In a set of 114 verbs (all synonyms of "to deceive" or of synonyms of "to deceive"), our program detected 680 cliques. The greatest number of vectors during execution was 696, the computation time 42 seconds. The matrix S contained 10098 zero entries on a total of $114^2 = 12996$ entries. DIFMAX was set equal to 100. Instead of two files, two arrays were used, because n was small enough.

4. Suggestion for a different approach.

In our example we found 680 cliques. This cliqueset was not at all informative, because of enormous overlap. Maybe a hierarchical clique structure would have been more informative, but we only had our binary matrix S. Therefore, we constructed the (nxn)-matrix Q, with q_{ij} = the number of cliques of which verbs i and j both were members. It is interesting to note that Q and S determine each other completely: because every S has its own, unique, cliqueset (as is not hard to prove), Q is completely determined by S; on the other hand, $s_{ij} = 0$ if $q_{ij} = 0$, $s_{ij} = 1$, otherwise, so S is completely determined by Q. Of course, the question arises whether there exists a relatively simple, direct function f such that Q = f(S). If such a function exists, clique-detection would become superfluous. Moreover, the ordinal data of Q (which represent the same information as the binary data of S!), can be "worked up" to interval-level by some nonmetric multidi-

mensional scaling procedure. Following this approach, we were able to define, in a 7-dimensional Euclidian space, a criterion distance d at which the dichotomized distance matrix resembled the original matrix S quite well (Wilmink, 1976).

Litterature.

Alba,R.D., COMPLT-A program for analyzing sociometric data and clustering similarity matrices. Behavioral Science, 17, p.566-567, 1972.

Harary,F. and Ross, I.C., A procedure for clique detection using the group matrix. Sociometry, 20, p.205-215, 1957.

Jardine, N. and Sibson, R., Mathematical Taxonomy. New York: Wiley, 1971.

Jensen, K. and Wirth, N., PASCAL User manual and report. New York: Springer, 1975 (second edition).

Peay, E.R., Nonmetric grouping: clusters and cliques. Michigan Mathematical Psychology Program Technical Report MMPP 70-5. Ann Arbor, Michigan, University of Michigan, 1970.

Peay, E.R., Hierarchical clique structures. Sociometry, 37, 1, p.54-65, 1974.

Rattinger, H., Eine einfache Methode und ein FORTRAN-Programm zur Ermittlung von Cliquen. Zeitschrift für Sozialpsychologie, 4, p. 5-14, 1973.

Shafto, M., CLIQUE: A FORTRAN IV program for the Needham-Moody-Hollis clusterlisting algorithm. Behavior Research Methods & Instrumentation, 6, 1, p. 58-59, 1974.

Wilmink, F.W., Analyse van een synonimiteitenmatrix. Heymans Bulletin 76 HB 215 EX. Groningen (Netherlands): Psychologische Instituten Heymans, 1976.