REFINEMENTS IN COMPUTERIZED ITEM SERIATION

By

William Bert Craytor

and

Le Roy Johnson Jr.

BULLETIN NO. 10

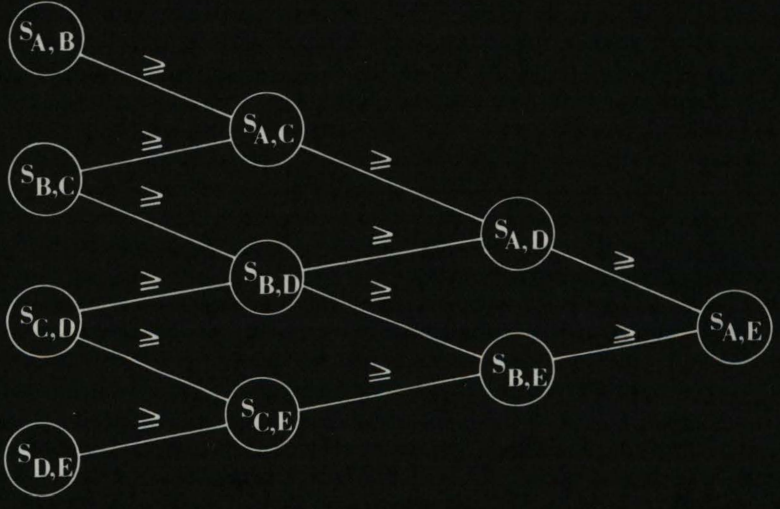
of the

MUSEUM OF NATURAL HISTORY

University of Oregon

Eugene, Oregon

March 1968



The *Bulletin* of the Museum of Natural History of the University of Oregon is published to increase the knowledge and understanding of the Natural History of Oregon. Original articles in the fields of Archaeology, Botany, Ethnology, Geology, Paleontology and Zoology appear irregularly in consecutively numbered issues. Contributions arise primarily from the research programs and collections of the Museum of Natural History and the Oregon State Museum of Anthropology. However, in keeping with the basic purpose of the publication, contributions are not restricted to these sources and are both technical and popular in character.

J. ARNOLD SHOTWELL, *Director*
Museum of Natural History
University of Oregon

Communications concerning manuscripts and purchase of copies of the *Bulletin* should be addressed in care of the Museum of Natural History, University of Oregon, Eugene, Oregon 97403. A list of previous issues of the *Bulletin* will be found inside the back cover.

REFINEMENTS IN COMPUTERIZED ITEM SERIATION

Funds for publication of Bulletin No. 10 provided by the National Science Foundation, grant GS 1413.

REFINEMENTS IN COMPUTERIZED ITEM SERIATION

by

William Bert Craytor

and

LeRoy Johnson, Jr.

Bulletin No. 10

Museum of Natural History
University of Oregon
March 1968

REFINEMENTS IN COMPUTERIZED ITEM SERIATION

by

William Bert Craytor
and
LeRoy Johnson, Jr.

INTRODUCTION

Our purpose in this paper is to refine certain aspects of computerized matrix seriation so as to provide a more precise and efficient automatic technique for determining and displaying interrelationships between compared items. A new program for matrix analysis, called PROGRAM SERIATE, is presented and explained.

The writing of a computer program for seriation was originally undertaken by the senior author in order to seriate archaeological collections chronologically by the Brainerd-Robinson method (Brainerd 1951, Robinson 1951), using as a model the program description published by Kuzara *et al.* (1966). Dissatisfaction with important aspects of previous work led, however, to the formulation of a better method for comparing trial orderings of matrix similarity scores, and to the writing of an improved seriation program.

Nonsystematic procedures of multiple-item analysis often lack rigor and efficiency, and need to be replaced with increasingly more efficient and systematic methods of comparison which have the advantages of replicability and, to a degree, objectivity in determining item relations. It is possible for analysts to make more nearly identical estimates of relationships of affinity within the same corpus of data, and reach agreement about the kind and degree of item relations, if they use effective automatic techniques. With such advantages in mind, we hope that the refinements on matrix seriation presented in the following pages will be useful for comparative analyses and will have relevance for the topics of chronological seriation, as in archaeology, and cluster analysis, as in biological taxonomy, cultural typology, and related studies.

The computer time necessary to develop PROGRAM SERIATE was kindly provided by the Statistical Laboratory and Computing Center of the University of Oregon, Fred C. Andrews, Director. Laurence R. Kittleman, Museum of Natural History, and Joseph G. Jorgensen, Department of Anthropology, U. O., read a draft of the present paper and made appreciated suggestions for changes in form and content.

MATRIX SERIATION

The present paper is concerned with a method of analysis which can be applied to a set of individuals scored on a set of characters. Both *Q*-technique and *R*-technique studies are allowed (Stephenson 1953). The former considers the correlations of pairs of individuals present in a population in terms of their characters, while the latter considers the correlation of pairs of characters present in a population. Seriation is the placement of items, whether individuals or characters, in a series, and as such is one form of scale analysis. Once properly seriated, however, items may further be tested for clustering.

Seriation arranges, as well as possible, a number of items into a vector array such that each item pair (except the outermost) is surrounded by less similar or equally similar item pairs. The scores (measures of similarity) for all item pairs are placed in a matrix table and then analyzed. If we let n represent the number of items, there are $n\ (n-1)/2$ different similarity scores for the $(\frac{n}{2})$ pairs of items. (It is common, however, to represent the matrix graphically in its double, or symmetric, form.) Analysis of the matrix scores can be a long and involved process beginning with an overall study of the matrix and, then when needed, perhaps ending with an analysis of particular subsets of scores within the matrix.

Suppose the items of the vector array (A, B, C, D, E) are perfectly seriated. Letting $S_{X,Y}$ symbolize the similarity between items X and Y, one could then say, for example, that $S_{A,B} \geqslant S_{A,C}$, and $S_{B,D} \geqslant S_{A,E}$. However, nothing could be said about the difference between the similarities of such item pairs as AB and CE. The inequalities between the adjoining similarity scores for this item array are diagrammed in Fig. 1, and a possible similarity matrix for the five items is shown in Fig. 2.

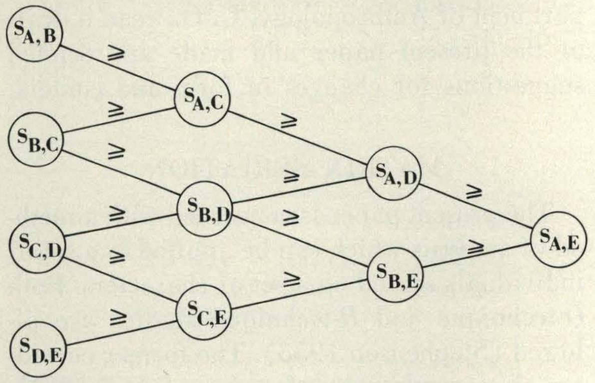


Figure 1. *Diagram of the Inequalities between the Similarity Scores for Items of the Array (A, B, C, D, E)*

ITEMS	*A*	*B*	*C*	*D*	*E*
A		27	25	21	8
B			31	24	11
C				25	12
D					14
E					

Figure 2. *Hypothetical Similarity Matrix for Items A, B, C, D, and E.*

Notice, however, that many inequalities besides those for adjoining similarity scores are implicitly specified. If we let $S_{i,j}$ represent the similarity score between items i and j, and $S_{k,l}$ the score between items k and l (where the subscripts denote the position of the item in the vector array), the inequalities requisite for the perfect seriation of any group of n items are the following:

$$S_{i,j} \geqslant S_{k,l} \qquad \begin{array}{l} \text{for } j=2,3,\ldots,n \\ i=1,2,\ldots,j-1 \\ k=1,2,\ldots,i \\ l=j,j+1,\ldots,n. \end{array}$$

The number of these inequalities, excluding the identity scores for items with themselves, can be determined by inspection of a matrix of similarity scores for a set of n items (*e.g.*, Fig. 1) to be the sum

$$\sum_{i=2}^{n-1} (n-\mathrm{i}) \sum_{j=2}^{i} j$$

which is equal to

$$\frac{n^4+2n^3-13n^2+10n}{24}. \qquad \text{(derivation: Appendix I.1)}$$

Thus an immense number of inequalities must be satisfied by a perfectly seriated group, even by one of only moderate size. For example, 269,500 inequalities must be satisfied by a perfectly seriated group of 50 items. For a group of 250 items, 164,091,205 inequalities must be satisfied for perfect seriation.

Seriating an item group requires arranging the items so that their similarity scores will satisfy, as nearly as possible, all the component inequalities requisite for perfect seriation. Several criteria, called ordering coefficients, have been proposed to differentiate the better from the less well seriated permutations of an item group, but they are only approximate measures of seriation and will not be discussed here. The following are two possible ordering coefficients:

Let n represent the number of items,

S^i_j represent the similarity score between the item in row i and the item in column j in the similarity matrix,[1] and

S^k_l represent the similarity score between the item in row k and the item in column l;

1) *The number of inequalities which should be met for perfect seriation and are not met; namely, the number of negative results obtained from the subtractions*

$$S^i_j - S^k_l \quad \text{for } \begin{array}{l} j=2, 3, \ldots, n \\ i=1, 2, \ldots, j-1 \\ k=1, 2, \ldots, i \\ l=j, j+1, \ldots, n; \end{array}$$

2) *the sum of all the coefficient differences (which should be positive or equal to zero for complete seriation), that is*

$$H = \sum_{j=2}^{n} \sum_{i=1}^{j-1} \sum_{k=1}^{i} \sum_{l=j}^{n} S^i_j - S^k_l.$$

In the process of seriating an item group, the first coefficient would be minimized or the second coefficient would be maximized. The difference between the two coefficients is that the second takes account of the magnitude of the similarity score differences whereas the first does not. If scores could be considered precise measures of similarity between items, the first ordering coefficient would perhaps be the more appropriate for item seriation. The degree, however, to which inequalities are satisfied is important, since similarity scores are seldom precise. Small differences between them are often statistically insignificant and may be due to sampling error.

Another advantage of the second ordering coefficient over the first is that its value can be acquired by a more efficient method. As shown in Appendix I.2, the sum of the coefficient differences is acquired without separately testing each of the score inequalities, as is required in the computation of the first coefficient. Thus the second ordering coefficient, termed *Coefficient H*, is used here.

Coefficient H is effective for comparing trial seriations of the same set of items, but its values are not comparable between different sets of items. In order to establish a somewhat more general criterion for determining the degree of seriation in different item sets, the ordering coefficient is standardized into a general seriation coefficient, traditionally called a matrix coefficient. Two such coefficients are already in use, termed Matrix Coefficient A and Matrix Coefficient B (Robinson 1951, Kuzara *et al.* 1966). Both are strongly biased measures of seriation. Our standardized matrix coefficient is termed *Matrix Coefficient C*, and is calculated by dividing Coefficient H by the number of inequalities tested, that is, by $(n^4+2n^3-13n^2+10n)/24$, to get the average inequality difference, and then dividing by the standard deviation of the $n(n-1)/2$ similarity scores.

Matrix Coefficient C has an approximate value of zero for a randomly arranged set of items, but in general the value of C becomes larger as the matrix is more perfectly seriated. It has been empirically determined that a C value of approximately 2 is indicative of excellent seriation; *viz.*, 2 is approached as the requisite inequalities are satisfied.

ITEM MANIPULATION

Since successful seriation requires arranging the items so that their similarity scores will satisfy, as nearly as possible, all the component inequalities necessary for perfect seriation, permutations of items in a set must be tested by Coefficient H. The only way that one can be certain of finding the best seriated permutation of a set of items is to examine all essentially different permutations and select the one with the highest H value. Such an examination is not feasible because of the prohibitive amount of time which would be required in computing the ordering coefficients for the $n!/2$ essentially different permutations for an item set,[2] even with a high-speed digital computer.

[1] $S_{i,j}$ is equivalent to S^i_j except that the former is the similarity score between items in a vector array, while the latter is the similarity score between items in the similarity matrix.

[2] Of the $n!$ permutations, those that are the reverse of the others represent the same item sequences; this is to say, the direction of the permutation is unimportant.

For example, a set of only 9 items involves 181,440 essentially different item permutations.

To solve this problem, it is first necessary to consider the relations between the ordering coefficients and their successive permutations. Fig. 3 is a graph of a hypothetical group of item permutations plotted against their respective ordering coefficients. The permutations are ordered along the horizontal axis on the basis of similarity of successive arrays to all others.[3] In this example there are three groups of relatively well seriated item permutations corresponding to three local modes or maxima represented by points *A*, *B*, and *C*. Point *B* is the absolute maximum corresponding to the best seriated permutation of the item group, and represents the kind of ordering sought after by archaeologists wishing to seriate artifact collections chronologically. For purposes of cluster analysis, as in numerical taxonomy, one would not only desire the item permutation corresponding to *B*, but also the permutations corresponding to the local maxima *A* and *C*, since these permutations might

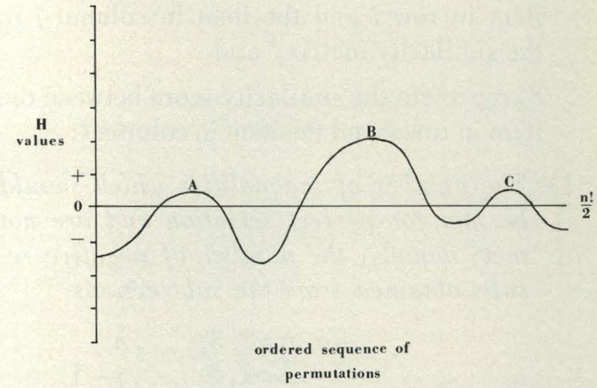


Figure 3. *Ordered Sequence of Essentially Different Permutations for a Set of Imaginary Items.*

enable one to hypothesize further on the nature of the item group under consideration.

The ideal procedure would be to generate a random sequence of the items and then manipulate the items by changing their positions in the item series until the permutation corresponding to the closest local maximum had been found. Essential to such a procedure would be the use of some criterion to determine if a permutation corresponded to a local maximum.

No ideal criterion has been established to enable an accurate determination of the permutations corresponding to local maxima. Nevertheless, Richard Kuzara (Kuzara *et al.* 1966) has described an item manipulation technique which provides the needed criterion for determining if an item permutation is in the close vicinity of a local maximum. This will therefore be called *Kuzara's Criterion.* Importantly, the criterion can be effectively approached and satisfied on a digital computer (see Appendix I.3 and Appendix III), and thus the problem of not being able to test all essentially different item permutations, mentioned above, can be solved.

Kuzara's Criterion is the following. An item permutation whose Coefficient *H* is in the close vicinity of a local maximum possesses an *H* value greater than, or equal to, any other Coefficient *H* corresponding to permutations

[3] In Fig. 3, the vertical axis is an axis of real values, while the horizontal axis consists of equally spaced points representing the $n!/2$ essentially different permutations seriated on the basis of the intersimilarities of their item ranks. Although there is no standard index of similarity between series corresponding to the rank-order correlation coefficients, the similarities between the permutations can be measured as the complements to their interpoint Eucildean distances in a space defined by the n positions in the series compared. Let s_{xy} represent the item for position y in some permutation x; let $D(ij)$ represent the Euclidean distance between two points, i and j, on the basis of the positions of the items in their permutations; and let c represent any real number. Thus the Euclidean distance between two permutations $(s_{i1}, s_{i2}, \ldots, s_{in})$ and $(s_{j1}, s_{j2}, \ldots, s_{jn})$ is

$$D(ij) = \sqrt{\sum_{l=1}^{n} (s_{il} - s_{jl})^2}$$

and the similarity of the series consequently is some constant c minus $D(ij)$, that is

$$S^i_j = c - D(ij).$$

Since all $n!/2$ essentially different permutations cannot be tested except for very small items sets, a certain number would have to be randomly selected from the above population of permutations, compared, seriated, and then plotted against their corresponding H values.

which might be formed by shifting any item, but only one item at a time, through the positions of the item permutation being considered.

In a computer analysis, seriation based on Kuzara's Criterion proceeds in the following manner. A randomly generated item sequence is stored in a static array, an array in which the components are never interchanged, only referred to. The similarity matrix for the static array is formed. The similarity matrix is a nonstatic array in which every row-column is eventually interchanged with every other to produce a seriated item sequence. Evolution of the item sequence in the similarity matrix commences with the testing of the first item from the static array in the first, second, . . . , and nth positions of the similarity matrix, that is, row-columns 1, 2, . . . , and n, respectively. The shifted item is then returned to the position corresponding to the highest ordering coefficient. In the same manner, the second, third, . . . , and nth items from the static array are subsequently tested in each position of the evolving similarity matrix and then returned to their most favorable location. This procedure, termed as one pass, is repeated at least once by replacing the contents of the static array with the item sequence produced in the manipulation of the similarity matrix, and then testing continues as above. Kuzara's Criterion for reaching the vicinity of the highest local degree of seriation is fulfilled by repeating these passes until the ordering coefficient for the item permutation produced by one pass is equal to the ordering coefficient for the item permutation produced by the pass previous to it.

Each seriation of a randomly generated item sequence may be referred to as an ordering. Whether or not there is more than one local maximum for an item group can usually be determined in one ordering. If an ordering produces a nearly perfectly seriated item permutation (as would be indicated by a value of approximately 2 for Matrix Coefficient C), then there is very little chance of getting different or dissimilar permutations from more orderings. But a poorly seriated item permutation indicates an inherent instability within the item set, *viz.*, the presence of complex multidimensional relationships between the items. In such cases several local maxima may occur for the degrees of seriation of the different permutations.

THE BASIC SERIATION ALGORITHM

The notation used in the algorithm is taken largely from K. E. Iverson's *A Programming Language* (1962) because of its conciseness and relative simplicity. Unless otherwise indicated by the branch instructions described below, the instructions of the algorithm are executed sequentially from the first to the last step. The notation used is briefly described as follows:

"$a \leftarrow b$" means that a accepts the value of b and b retains its value.

"$a \leftrightarrow b$" means that the values of variables a and b are interchanged.

"$y \downarrow (x_1, x_2, \ldots, x_n)$" means that the values of the variables in the vector are rotated to the right by y positions. (Similarly, "$\uparrow$" signifies a left rotation.) For instance, $1 \downarrow (7, 6, 5, 4) = (4, 7, 6, 5)$ and $2 \downarrow (4, 2, 3) = (2, 3, 4)$.

"$qRb, \rightarrow m$" means that after the instruction qRb is executed, directly branch to step m.

"$a{:}c, (>, =) \rightarrow (l, m)$" means that if $a > c$ then branch to step l; if $a = c$ then branch to step m; otherwise proceed to the following step,

"$a \leftarrow x \iota y$" means that the rank of the values of y in x are placed in a. For instance, if $x = (3, 9, 10)$ and $y = (10, 3, 9)$, then $a = (3, 1, 2) = x \iota y$, and $a = 1 = x \iota y_2$.

"$\iota^x(y)$", termed the interval vector with origin x and length y, is a vector of y integers progressing from left to right, beginning with integer x. For example $\iota^0(5) = (0, 1, 2, 3, 4)$ and $\iota^5(3) = (5, 6, 7)$.

The algorithmic procedure begins at a point indicated by the *entry arrow*, and terminates wherever indicated by an *exit arrow*.

Verbal instructions are inserted where simple functions would otherwise require complex algorithmic instructions unnecessary for the understanding of the method.

In order to construct an algorithm for seriation it is simplest to redefine S so that

$$S^{i+1} \leftarrow S^{i} \qquad S_{i+1} \leftarrow S_{i} \qquad i=(n, n-1, n-2, \ldots, 1)$$

and place an interval vector of integers to identify the items in the input similarity matrix in the first row-column of S as follows:

$$S^{1} \leftarrow \iota^{0}(n+1) \text{ and } S_{1} \leftarrow S^{1}.$$

The following algorithm is supplied to give a simple description of the method of seriating a set of items used in PROGRAM SERIATE (Appendix III). Although the algorithm could be applied for the purpose of seriation, it would be much too inefficient to use on more than a dozen items. As with the development of nearly any mechanism, complexity increases with efficiency and flexibility; this is the case in the development of PROGRAM SERIATE from the Basic Seriation Algorithm (see Appendix I.3).

THE ALGORITHM

```
1→  Read $S, n, w$
2   Print $S$ (labeled "Data Matrix")
3   Randomly interchange the row-columns of $S$
4   $b \leftarrow S^1$
5   $c \leftarrow$ Coef. $H$
6   $i \leftarrow 1$
7   $i \leftarrow i+1$
8   $i{:}n+1, (>) \rightarrow (25)$
9   $a \leftarrow S^1 \iota b_i$
10  $(S^2, S^3, \ldots, S^a) \leftarrow 1\downarrow(S^2, S^3, \ldots, S^a)$
11  $(S_2, S_3, \ldots, S_a) \leftarrow 1\downarrow(S_2, S_3, \ldots, S_a)$
12  $d \leftarrow$ Coef. $H$
13  $l \leftarrow 2$
14  $j \leftarrow 1$
15  $j \leftarrow j+1$
16  $j{:}n+1, (>) \rightarrow (23)$
17  $S^j \leftrightarrow S^{j+1}$
18  $S_j \leftrightarrow S_{j+1}$
19  $h \leftarrow$ Coef. $H$
20  $h{:}d, (\leqslant) \rightarrow (15)$
21  $d \leftarrow h$
22  $l \leftarrow j+1, \rightarrow 15$
23  $(S^l, S^{l+1}, \ldots, S^{n+1}) \leftarrow 1\downarrow (S^l, S^{l+1}, \ldots, S^{n+1})$
24  $(S_l, S_{l+1}, \ldots, S_{n+1}) \leftarrow 1\downarrow (S_l, S_{l+1}, \ldots, S_{n+1}), \rightarrow (7)$
25  $d{:}c, (>) \rightarrow (4)$
26  Print $S$ (labeled "Ordered Matrix")
27  $w \leftarrow w-1$
28  $w{:}0, (>) \rightarrow (3)$
29  Print execution time →
```

$$\text{Coef. } H = \sum_{j=3}^{n+1} \sum_{i=2}^{j-1} \sum_{k=2}^{i} \sum_{l=j}^{n+1} S_j^i - S_l^k$$

n—the no. of items
S—the similarity matrix
w—the no. of orderings to be executed
S^x—row x of S
S_x—column x of S

Legend

REFERENCES CITED

Brainerd, George W., 1951. The place of chronological ordering in archaeological analysis. *American Antiquity*, v. 16, no. 4, pp. 303-313. Salt Lake City.

Iverson, E. E., 1962. *A programming language.* John Wiley and Sons, New York.

Kuzara, Richard S., George R. Mead, and **Keith A. Dixon**, 1966. Seriation of anthropological data: a computer program for matrix ordering. *American Anthropologist*, v. 68, no. 6, pp. 1442-1455. Menasha.

Robinson, W. S., 1951. A method for chronologically ordering archaeological deposits. *American Antiquity*, v. 16, no. 4, pp. 239-301. Salt Lake City.

Stephenson, W., 1953. *The study of behavior: Q-technique and its methodology.* University of Chicago Press, Chicago.

APPENDIX I

Computational Aids and Refinements

1. The derivation of the final expression for the sum of the inequalities is

$$\sum_{i=2}^{n-1}(n-i)\sum_{j=2}^{i} j$$

$$=\tfrac{1}{2}\sum_{i=1}^{n-1}(n-i)\,(i^2+i-2)$$

$$=\tfrac{1}{2}\sum_{i=1}^{n-1}[-i^3+(n-1)i^2+(n+2)i-2n]$$

$$=\frac{(n-1)n}{2}\Big[-\frac{(n-1)}{4}n+\frac{(n-1)\,(2n-1)}{6}+\frac{(n+2)}{2}-2\Big]$$

$$=\frac{(n-1)n}{48}(14n^2+6n-20)$$

$$=\frac{n^4+2n^3-13n^2+10n}{24}.$$

2. For purposes of computation, Coefficient H (as its notation is changed on p. 19), can be simplified as follows:

$$H=\sum_{j=3}^{n+1}\sum_{i=2}^{j-1}\sum_{k=2}^{i}\sum_{l=j}^{n+1}S_j^i-S_l^k$$

$$=\sum_{j=3}^{n+1}\sum_{i=2}^{j-1}\Big[(i-1)\,(n+2-j)\,S_j^i-\sum_{k=2}^{i}\sum_{l=j}^{n+1}S_l^k\Big]$$

$$=\sum_{j=3}^{n+1}\sum_{i=2}^{j-1}S_j^i\Big[\frac{-j^2+j-i^2+i(2n+5)-2(n+2)}{2}\Big].$$

The factors (that is, the elements which form a product when multiplied together) for the components of S are constant for different permutations of an item group. Therefore, previous to ordering an item group, these factors can be calculated and placed in a *model matrix* with the same structure (no. of row-columns) as the similarity matrix. All that is then required in calculating Coefficient H for some item permutation is to multiply the similarity scores above the primary diagonal of the similarity matrix by the corresponding factors from the model matrix. If we let M represent the model matrix, this can be formulated as

$$M_j^i=\frac{-j^2+j-i^2+i(2n+5)-2(n+2)}{2}$$

$$\text{for } j=3,4,\ldots,n+1 \qquad i=2,3,\ldots,j-1.$$

Consequently, Coefficient H is reduced to the following:

$$H=\sum_{j=3}^{n+1}\sum_{i=2}^{j-1}S_j^i\,M_j^i.$$

3. Development of PROGRAM SERIATE

No matter how theoretically sound any algorithm may be, it is useless if it cannot be operationalized within the financial means of the user and the limitations of the computer available to him. If an algorithm can be modified for computational purposes so that it becomes more efficient and flexible, then the scope of its application is usefully expanded.

For the following reasons the Basic Seriation Algorithm is rather inflexible and almost intolerably inefficient for computer calculation: (1) the entire similarity matrix of $(n+1)^2$ elements must be stored, while the symmetry of the matrix requires that only $(n+2)\,(n+1)/2$ essential elements be stored; (2) the elements of the row-columns of S must be interchanged, requiring a relatively large amount of computer time; (3) the ordering coefficient must be computed for every item permutation tested whereas, in actuality, only the changes in the ordering coefficients need to be computed; (4) the calculation of the ordering coefficient by actually computing the score differences demands an impossible amount of labor; and (5) subsets of an item set cannot be seriated without repunching the similarity scores for each subset on separate sets of data cards.

PROGRAM SERIATE improves on the Basic Seriation Algorithm in respect to all five foregoing points. A step by step explanation of the program is not necessary, but the techniques and short cuts which make the program function efficiently and give it a degree of flexibility not possessed by the Basic Seriation Algorithm are discussed below. In appropriate places, cross-reference is made to the specific subroutines of PROGRAM SERIATE listed in Appendix III.

General Refinements

(a) Compression of the Similarity Matrix into an Array: In PROGRAM SERIATE the elements of the upper right section of the similarity matrix S are stored column-wise in vector a in a sequential manner from the first to the last column, that is

$$a\left[i+\sum_{k=0}^{j-1}k\right]=S^i_j \quad (i\leqslant j).$$

The summations in the subscript for vector a are constant, and are calculated as S is read and placed in a vector q, that is

$$q_j=\sum_{k=0}^{j-1}k \quad (j=1, 2, \ldots, n, n+1).$$

Therefore, any element S^i_j of S is

$$a\left[\min(i,j)+q_{\max(i,j)}\right].$$

The two principal advantages of this type of storage compression are (1) that the amount of storage required for the similarity matrix is nearly halved, and (2) that the computer time required to reference similarity scores is minimized.

(b) Eliminating the Manipulation of Similarity Scores within Storage: In the Basic Seriation Algorithm, the row-columns of S were interchanged in order to calculate the ordering coefficient for a particular item sequence (which was correspondingly formed in the first row-column of S). In PROGRAM SERIATE, neither the item identification numbers (which are $a_{i+q_i}=i-1$ for an item in row-column i of S as punched on the data cards) nor the similarity scores are interchanged. Instead, the components of the item sequence vector m to be tested are used as the subscripts of vector a as if the similarity matrix corresponding to vector a had its row-columns arranged according to the sequence in vector m. The connection between S, m, and a is as follows:

$$a\left[\min(m_i,m_j)+q_{\max(m_i,m_j)}\right]=S^{m_i}_{m_j}$$

where initially $m_i=i=a_{i+q_i}+1$ if the entire similarity matrix is to be seriated. However, as will be mentioned below, the initial components of m may be optionally specified by the user.

As a consequence of the introduction of the item sequence vector m, only the components of m are interchanged in PROGRAM SERIATE. It might appear that the computation of the subscript for vector a would offset any other advantages of the notation presented, but in the end the total reference to vector a in the program is relatively infrequent.

(c) Symbols Used to Deal with Subsets of S: The input similarity matrix punched on cards is labeled S. In certain cases the user may desire to seriate only a specific number of the n items of an item set, rather than the whole set. Any similarity matrix formed from a subset of the items in S shall be referred to as ${}^iS$ for the ith specified subset of S in the input sequence for PROGRAM SERIATE. Correspondingly, the number of items in S shall be referred to as n, the number of items in ${}^iS$ as n_i, and the item sequence vector for ${}^iS$ as m^i. The number of subsets of S to be seriated in one execution of the program will be symbolized by N. If the entire item set S is to be seriated in one execution, then N equals zero since, in this case, S need not be specified as a subset of itself. (If, however, the user desires to have the similarity matrix and corresponding coefficients printed for some particular sequence of all the items of S that is different from the original punched sequence, then N is set equal to 1 and the combination to be printed is specified as indicated in Appendix II.4.)

Calculation of H

In Appendix I.2 the calculation of Coefficient H was refined by the introduction of a model matrix M. Although the calculation of H using M requires the least amount of computer time, M requires as much storage as S. Therefore, to avoid storing M another method is used in PROGRAM SERIATE to calculate H, one which involves the storage of only one vector $g = M^2$. Calculation of H progresses columnwise by multiplying similarity scores in the second row of column i by g_i, then sequentially multiplying the scores in the third, fourth, . . . , and $(i-1)$th rows by factors which are calculated as accumulations of changes based on the value g_i for the column. The changes are simply $M^k_i - M^{k+1}_i = n+2-k$ between any two appropriate rows k and $k+1$ for column i. The following is the algorithmic procedure used in PROGRAM SERIATE, whose FORTRAN IV version appears as subroutine MCOEF.

```
1→  $H \leftarrow 0$
2   $i \leftarrow 2$
3   $i \leftarrow i+1$
4   $j \leftarrow n+1$
5   $k \leftarrow g_i - (n+1)$
6   $p \leftarrow 1$
7   $p \leftarrow p+1$
8   $k \leftarrow k+j$
9   $j \leftarrow j-1$
10  $H \leftarrow H + k \times S^i_p$
11  $p: i-1, (<) \rightarrow (7)$
12  $i: n+1, (<) \rightarrow (3)$                →
```

Changes in Coefficient H and Their Accumulation

It will be noted by the reader that in steps 12 through 20 of the Basic Seriation Algorithm, the second row-column of S is shifted through row-columns 3, 4, . . . , and $(n+1)$ of S with H being calculated after each shift. However, to determine the best position for the item, it is only necessary that the changes in H, produced by the row-column shifts, be recorded into a vector and then accumulated. The proper calculation of the changes in H and their accumulation, as shown below, is a great short cut in item seriation.

Suppose the index groups of row-columns k and $k+1$ are interchanged within S. Symbolizing the corresponding change in H by element m_{k+1} of vector m, the change is formulated as directed by the following derivation:

$$\begin{aligned}
\mathrm{m}_{k+1} = {} & \sum_{i=2}^{k-1} (S^i_k - S^i_{k+1})\,(M^i_k - M^i_{k+1}) \\
& [\text{for } k>2] \\
& + \sum_{i=k+2}^{n+1} (S^k_i - S^{k+1}_i)\,(M^k_i - M^{k+1}_i) \\
& [\text{for } k<n] \\
= {} & k \sum_{i=2}^{k-1} (S^i_k - S^i_{k+1}) \quad [\text{for } k>2] \\
& -(n+2-k) \sum_{i=k+2}^{n+1} (S^k_i - S^{k+1}_i) \\
& [\text{for } k<n] \\
= {} & k\,\Big(\sum_{i=2}^{k-1} S^i_k - \sum_{i=2}^{k-1} S^i_{k+1}\Big) \quad [\text{for } k>2] \\
& -(n+2-k)\,\Big(\sum_{i=k+2}^{n+1} S^k_i - \sum_{i=k+2}^{n+1} S^{k+1}_i\Big) \\
& [\text{for } k<n].
\end{aligned}$$

Several preliminary steps are necessary before computing the $(n-1)$ changes in H. The row and column sums of the similarity scores above the primary diagonal and below the second row in S are placed in two separate arrays V^1 and V^2 according to corresponding row or column positions; scores from the second row-column are placed into an array V^3 (corresponding to their positions in S^2); and the sum of the smiliarity scores from the second row (excluding S^2_2) is assigned to two variables, x and y. These quantities can be written as

$$V^1_i = \sum_{k=i+1}^{n+1} S^i_k \qquad (i=3,4,\ldots,n)$$

$$V^2_i = \sum_{k=3}^{i-1} S^k_i \qquad (i=4,5,\ldots,n+1)$$

$$V^3_i = S^2_i \qquad (i=3,4,\ldots,n+1)$$

$$x = y = \sum_{k=3}^{n+1} S^2_k.$$

The $(n-1)$ changes in H are then computed in the following manner: V^1 is transformed so that it contains the changes in H effected by the shifting of row elements; V^2 is transformed into a vector of changes in the value of H wrought by column manipulations; the changes in V^2 are added onto the changes in V^1; then the changes are accumulated in V^1, simultaneously placing in variable r the position of maximum change, and in variable j the maximum change up to each accumulation. This may be stated algorithmically as follows:

```
1→  $i \leftarrow 2$
2   $i \leftarrow i+1$
3   $i:n, (>) \rightarrow (9)$
4   $x \leftarrow x - V^3_i$
5   $V^1_i \leftarrow (x - V^1_i)\ (n+3-i)$
6   $k \leftarrow n+4-i$
7   $y \leftarrow y - V^3_k$
8   $V^2_k \leftarrow (V^2_k - y)\ (k-1), \rightarrow (2)$
9   $V^1_2 \leftarrow 0$
10  $r \leftarrow 2$
11  $j \leftarrow 0$
12  $i \leftarrow 2$
13  $i \leftarrow i+1$
14  $i:n+1, (>) \rightarrow (19)$                  →
15  $V^1_i \leftarrow V^1_i + V^1_{i-1} + V^2_i$
16  $V^1_i:j, (\leqslant) \rightarrow (13)$
17  $r \leftarrow i$
18  $j \leftarrow V^1_i, \rightarrow (13)$
```

corresponds to statements 910+2 through 252 in subroutine PLACE

It can be verified by the reader that after the above procedure is executed, V^1_i equals the change in H that would result if row-column 2 were rotated to any row-column i (where $i>1$).

The statements of the above algorithmic procedure are incorporated within subroutine PLACE in equivalent FORTRAN IV statements. However, as will be described in the next paragraph, the procedure of summing the row and column scores into vectors need be executed only once as one of the initial steps in the ordering of an item set, since the changes in the sums caused by shifting an item to row-column r by the above algorithmic procedure can be speedily calculated after each shift.

Vector Storage of Row and Column Sums

Two sets of vectors are required for the storage of the row and column sums of S since the elements of V^1 and V^2 are transformed into other values. Thus two vectors I^1 and I^2 are set aside for the sole purpose of storing the row and column sums. In PROGRAM SERIATE, subroutine LASR places the initial row and column sums for an ordering into vectors I^1 and I^2, respectively, and subroutine PLACE transfers the contents of I^1 and I^2 into V^1 and V^2, respectively, initial to the execution of the algorithm for finding the best position r for the row-column l being tested. Once r has been determined, a subset of the row-columns of S and the corresponding columns of I are rotated left or right depending on whether l is to left or right of r. Then the following modifications are made on the shifted elements of I so that they equal the new row and column sums of S. The elements of V^3 are either added to, or subtracted from, the elements in the corresponding columns of I, depending on the sign of $d=l-r$ and on the particular row of I for a given element in I, except for column I_r in which the quantity

$$q = (\operatorname{sgn} d) \sum_{i=\min(l,r)+1}^{\max(l,r)} V^3_i$$

must be added to I^2_r and subtracted from I^1_r. The algorithmic procedure for rotating row-column l to r and modifying I follows.

```
1→  $d:0, (=,>) \rightarrow (16,5)$                                    →
2   $(S_r,S_{r+1},\ldots,S_l) \leftarrow 1\uparrow (S_r,S_{r+1},\ldots,S_l)$
3   $(S^r,S^{r+1},\ldots,S^l) \leftarrow 1\uparrow (S^r,S^{r+1},\ldots,S^l)$
4   $(I_r,I_{r+1},\ldots,I_l) \leftarrow 1\uparrow (I_r,I_{r+1},\ldots$
    $\ldots,I_l), \rightarrow (8)$
5   $(S_r,S_{r+1},\ldots,S_l) \leftarrow 1\downarrow (S_r,S_{r+1},\ldots,S_l)$
6   $(S^r,S^{r+1},\ldots,S^l) \leftarrow 1\downarrow (S^r,S^{r+1},\ldots,S^l)$
7   $(I_r,I_{r+1},\ldots,I_l) \leftarrow 1\downarrow (I_r,I_{r+1},\ldots,I_l)$
```

```
8    $i \leftarrow 0$
9    $i \leftarrow i+1$
10   $i{:}d,\ (>) \rightarrow (14)$
11   $j \leftarrow l+1-i$
12   $I^1_j \leftarrow I^1_j - (\mathrm{sgn}\ d) \times V^3_j$
13   $I^2_j \leftarrow I^2_j + (\mathrm{sgn}\ d) \times V^3_j, \rightarrow (9)$
14   $I^1_r \leftarrow I^1_r - (\mathrm{sgn}\ d) \times q$
15   $I^2_r \leftarrow I^2_r + (\mathrm{sgn}\ d) \times q$                    $\rightarrow$
```

corresponds to statements 888 through 521+3 in subroutine PLACE

Immediately after $I^i \rightarrow V^i (i=1, 2)$ the elements of V^1 and V^2 are manipulated so that they equal the appropriate row and column sums which would result if row-column l, the row-column being tested, were moved to the second row-column. Also, the elements of row-column l are placed in V^3 as if they occupied the second row-column of S. The algorithmic statement of this procedure of subroutine PLACE is

```
1 →  $l{:}2,\ (=) \rightarrow (4)$
2    $(V^3_3, V^3_4, \ldots, V^3_l) \leftarrow (S^2_l, S^3_l, \ldots, S^{l-1}_l)$
3    $l{:}n+1,\ (=) \rightarrow (5)$
4    $(V^3_{l+1}, V^3_{l+2}, \ldots, V^3_{n+1}) \leftarrow (S^l_{l+1}, S^l_{l+2}, \ldots, S^l_{n+1})$
5    $x \leftarrow I^1_2$
6    $y \leftarrow x$
7    $l{:}2,\ (\leqslant) \rightarrow (15)$
8    $x \leftarrow I^1_l + I^2_l$
9    $y \leftarrow x$
10   $i \leftarrow 2$
11   $i \leftarrow i+1$
12   $i{:}l,\ (>) \rightarrow (15)$
13   $j \leftarrow l+3-i$
14   $V^1_j \leftarrow V^1_{j-1} - V^3_j, \rightarrow (11)$
15   $l{:}n,\ (>) \rightarrow (20)$
16   $i \leftarrow l$
17   $i \leftarrow i+1$
18   $i{:}n+1,\ (>) \rightarrow (20)$
19   $V^2_i \leftarrow V^2_i - V^3_i, \rightarrow (17)$
20   $l{:}4,\ (<) \rightarrow (22)$
21   $(V^2_4, V^2_5, \ldots, V^2_l) \leftarrow 1{\downarrow}\ (V^2_4, V^2_5, \ldots, V^2_l)$
22   $V^1_{n+1} \leftarrow 0$
23   $V^2_3 \leftarrow 0$                                  $\rightarrow$
```

corresponds to statements 668+7 through 910+1 in subroutine PLACE

Flow Chart for PROGRAM SERIATE

An abbreviated flow chart for PROGRAM SERIATE is given in Fig. 4. The numbered steps of the flow chart are described as follows:

(1) n, w, and N are read from the header card, as well as other values of minor importance described in Appendix II.1.

(2) Subroutine MATIN reads S.

(3) If the last item combination has been read, then the execution of the program is ended.

(4) Subroutine INPUT reads the ith item combination specification m^i in the input sequence. The presence of errors in m^i sets an indicator variable to a particular value which causes a branch to step 3 from step 5 and, as a consequence, the seriation of m^i is skipped.

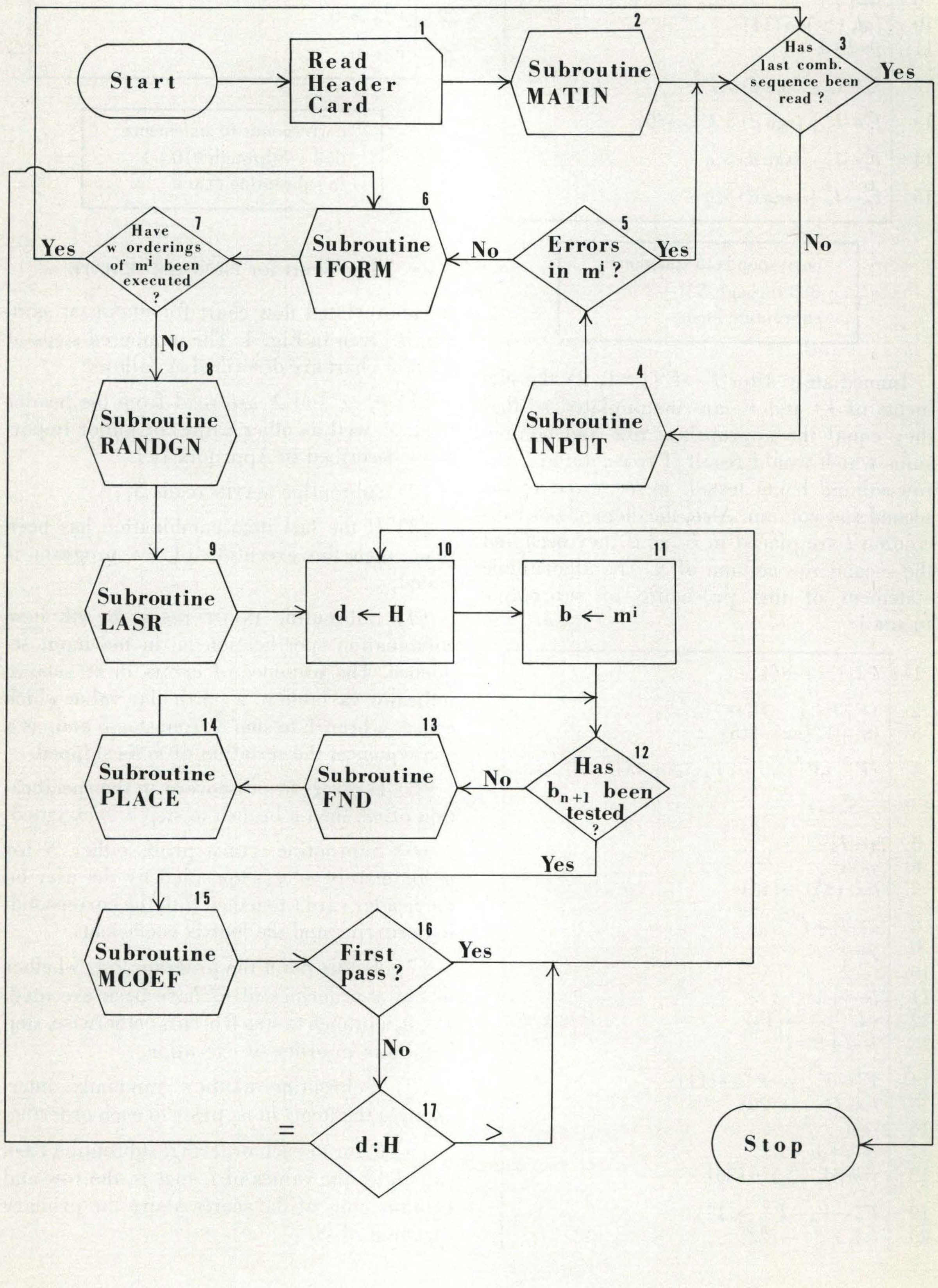
(5) If errors are discoverd in the specification of m^i, then a branch to step 3 is executed.

(6) Subroutine IFORM prints either ${}^iS$ for m^i or merely m^i (as specified by the user on the header card) together with the correpsonding ordering and the matrix coefficients.

(7) At this point the program tests whether or not w orderings of m^i have been executed. If not, a branch to step 8 occurs; otherwise, step 3 follows in order of execution.

(8) Subroutine RANDGN randomly interchanges the items in m^i prior to each ordering.

(9) Prior to each ordering, subroutine LASR calculates the values of I, that is, the row and column sums of the scores above the primary diagonal of ${}^iS$.


Start
1
Read Header Card
2
Subroutine MATIN
3
Has last comb. sequence been read ?
Yes
No
Stop
4
Subroutine INPUT
5
Errors in m^i ?
Yes
No
6
Subroutine IFORM
7
Have w orderings of m^i been executed ?
Yes
No
8
Subroutine RANDGN
9
Subroutine LASR
10
$d \leftarrow H$
11
$b \leftarrow m^i$
12
Has b_{n+1} been tested ?
No
Yes
13
Subroutine FND
14
Subroutine PLACE
15
Subroutine MCOEF
16
First pass ?
Yes
No
17
d:H
=
>

(10) Prior to each pass, the value of Coefficient *H* for m^i is stored in *d* for comparison (in step 17) with the *H* value calculated (in step 15) for m^i after the pass.

(11) Since the elements of m^i are interchanged during each pass, their initial order must be retained in another vector *b* in order that the same item will not be tested more than once during the pass.

(12) The items $b_2, b_3, \ldots,$ and b_{n+1} are sequentially tested; therefore, if b_{n+1} has been tested the pass is completed.

(13) Subroutine FND finds the rank *l* of item b_j in m^i ($j=2,3,\ldots,n+1$ within the pass).

(14) Subroutine PLACE finds the position *r* of m^i_l in m^i where *H* is maximum, rotates item m^i_l to m^i_r, and correspondingly modifiies *I*.

(15) Subroutine MCOEF calculates Coefficient *H*.

(16) At least two passes are required per ordering.

(17) If the last pass has produced no change in the ordering coefficient, then the ordering is ended; otherwise, another pass is executed.

APPENDIX II

INPUT FORMAT FOR PROGRAM SERIATE

1. Header Card

Cols. 1-3	No. of items (n) in the input similarity matrix ($n_{\max}=250$).
Cols. 4-5	No. of orderings (w) for each combination (max. 99).
Cols. 6-10	A five digit odd integer supplied to the random number generator.
Cols. 11-13	Lowest index value to be underlined in the matrix printouts.[4]
Cols. 14-16	No. of printout copies.[5]
Cols. 17-19	Three underlines.
Cols. 20-22	Three blanks.
Cols. 23-24	Either 1 or 2: 1 to print the similarity matrices, 2 to print only the item sequences.
Cols. 25-26	Zero, if all the items in the input similarity matrix are to be seriated; otherwise the number of combinations of items from the input similarity matrix to be seriated (N).

Figure 4. *Abbreviated Flow Chart for* PROGRAM SERIATE.

2. Format Card for the Input Similarity Matrix

Cols. 1-80	The format by which the rows of the upper right half of the symmetric similarity matrix are read; the format (2X, 26I3) is standard.

3. Similarity Matrix (assuming use of above standard format)
   Only the similarity scores of the upper right portion of the input similarity matrix are punched. Begin on each row with the diagonal identity score and continue to the right through the similarity score in the last matrix column in 26 three-column, right-justified fields from col. 3 through col. 80 for each data card (see Fig. 5). The scores for each item should begin on a new data card. Consequently, all the scores for the first row of the input matrix will be punched in the first data card(s) and only the diagonal identity score of the last row will be punched in the last data card (in cols. 3-5). The

[4] The user has the option of underlining similarity scores greater than, or equal to, a specified similarity value. This procedure may, for example, help delineate clusters (as in taxonomic studies) of similarity scores along the primary diagonal having a desired range of probability and/or strength. The computer time required to underline the index values is equal to the time required to print the indices; therefore, neglecting to underline decreases the time required to print out martices by as much as 50 per cent. For large matrices (*i.e.*, matrices with more than 80 items) this might be important from a financial point of view. Underlining can be avoided if so desired, by specifying an integer larger than any within the similarity matrix.

[5] The number specified increases the printout time by the same factor.

	Item Scores				
ITEMS	1	2	3	4	5
1	S_1^1	S_2^1	S_3^1	S_4^1	S_5^1
2		S_2^2	S_3^2	S_4^2	S_5^2
3			S_3^3	S_4^3	S_5^3
4				S_4^4	S_5^4
5					S_5^5

		Card Cols.				
ITEMS		1	2	3	4	5
card 1:	1	3-5	6-8	9-11	12-14	15-17
card 2:	2		3-5	6-8	9-11	12-14
card 3:	3			3-5	6-8	9-11
card 4:	4				3-5	6-8
card 5:	5					3-5

Figure 5. *Example Input Matrix with Column Positions for Scores on Data Cards.*

similarity scores must be integers between —99 and 999.

4. Specification of Item Combinations to Be Seriated from the Input Similarity Matrix. *N* subsets of the items of the input similarity matrix may be seriated if specified on the header card. If the *i*th combination of n_i items to be seriated contains any repetitions of item identification numbers, the repeated items will be listed as errors and the seriation of the combination will be skipped. The format for combination specifications follows.

Cols. 1-4	No. of items in combination (n_1).
Cols. 5-8 Cols. 9-12 . . . . Cols. 77-80	The numeric positions of the items in the input similarity matrix constituting the combination to be seriated. The position numbers are continued in successive four-column fields through as many cards as are required. Each combination specification, however, must begin on a new card.

Cols. 1-4	n_N
Cols. 5-8	
.	
.	
.	

APPENDIX III

Listing of PROGRAM SERIATE, Estimation of Run Time

1. PROGRAM SERIATE is written in FORTAN IV programming language for the IBM 360 Model 50 computer. A copy of the program deck may be purchased from the Museum of Natural History, University of Oregon, Eugene 97403.
2. An estimation of run time for PROGRAM SERIATE is given in Fig. 6. The estimates are for time involved when (1) the similarity matrix is printed, or (2) only the seriated item sequence is printed. Two orderings are assumed for each set of data, with two passes per ordering. Use of the IBM 360/50 computer and an IBM 1403 printer is assumed.

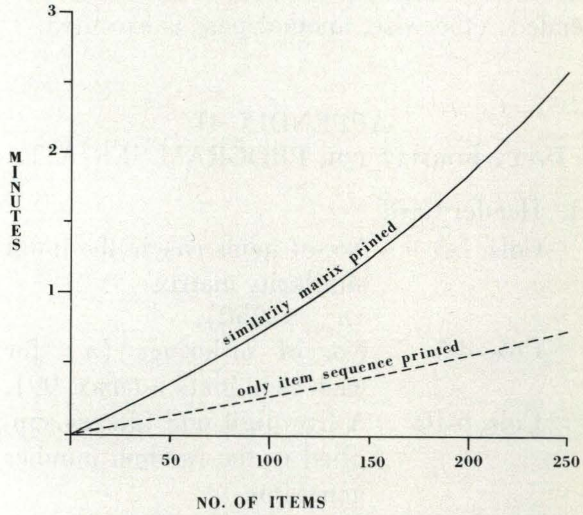


Figure 6. *Estimated Minimum Computation Time per Ordering (assuming 2 passes/ordering) in Minutes on the IBM 360/50 computer with an IBM 1403 Printer.*

```
C     MAIN PROGRAM
      WRITE (3,24)
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256),ISVM,N,NP1
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2,NCOM,JNCOM
  278 FORMAT (1H0,'PROCESSING TIME:',F8.3,' MINUTES')
   24 FORMAT ('1',T3,'ITEM SERIATION',/' ',T3,'W.B. CRAYTOR',/' ',T3,
     1'UNIVERSITY OF OREGON')
      CALL TIMER (Y3)
      COMMON /XX/FMT(18)
    1 FORMAT (I3,I2,I5,2I3,2A3,2I2)
    5 FORMAT (18A4)
      JNCOM=0
      READ (1,1) N,NNN,IY,NVAL,INOS,UNDLN,BLNK,ISVM,NCOM
      READ (1,5) FMT
      CALL MATIN
  900 CALL INPUT
      GO TO (302),I
  300 CALL IFORM (1)
  301 DO 131 IUTR=1,NNN
      CALL RANDGN (IY)
      MN1=0
      CALL LASR
   95 D1=COEFB
   98 DO 81 LC2=2,KE
   81 B(LC2)=MSH(LC2)
      DO 935 LIO =2,KE
  777 CALL FND(LIO,LIM)
  935 CALL PLACE (LIM)
      MN1=MN1+1
      CALL MCOEF
      GO TO (95),MN1
      IF (D1-COEFB)95,131,131
  131 CALL IFORM (3)
  302 IF(NCOM-JNCOM)1144,1144,900
 1144 CALL TIMER (Y2)
      T=(Y2-Y3)/60.
      WRITE (3,278) T
 7777 STOP
      END
      SUBROUTINE MATIN
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /XX/FMT(18)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256),ISVM,N,NP1
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2
      NP1=N+1
      IMMX=0
      DO 794 IMX=1,NP1
      A(IMMX+1)=IMX-1
      SQ(IMX)=IMMX
  794 IMMX=IMMX+IMX
      DO 233 IA=2,NP1
  233 READ (1,FMT) (A(IA+SQ(IB)),IB=IA,NP1)
      RETURN
       END
```

```
      SUBROUTINE INPUT
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256),ISMN,N
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2,NCOM,JNCOM
   21 FORMAT (T20,'ERROR - RELATIVE ',I3,' REPEATED IN COMBINATION ',I3)
  890 FORMAT (20I4)
      JNCOM=JNCOM+1
      I=2
      IF(NCOM-0)901,901,900
  901 NCOM=1
      KEE=N
       KET=KEE-1
      KE=KEE+1
      DO 950 ISL=1,KE
  950 MSH(ISL)=ISL
      GO TO 902
  900 READ (1,890) KEE,(ISQ(IRS),IRS=1,KEE)
      KET=KEE-1
      DO 29 IT=1,KET
      ITP1=IT+1
      DO 9 IS=ITP1,KEE
      IF(ISQ(IT)-ISQ(IS))9,10,9
   10 WRITE (3,21) ISQ(IT),JNCOM
      I=1
      GO TO 29
    9 CONTINUE
   29 CONTINUE
      GO TO (400),I
      DO 794 IMX=2,KE
  794 MSH(IMX)=ISQ(IMX-1)+1
      MSH(1)=1
  902 KE=KEE+1
      KEM3=KE-3
      KEP1=KE+1
       KEP3=KE+3
      KEP2=KE+2
      TEK=KET
      EEE=KEE
      OTZ123=(((EEE+2.)*EEE-13.)*EEE+10.)*EEE/24.
      EY=(KEE*KEE-KEE)/2
      SUM=0
      SUMSQ=0
      DO 97 IX=3,KE
      IXS=SQ(IX)
      IX1=IX-1
      DO 97 IY=2,IX1
      INTS=A(IY+IXS)
       SUM=SUM+INTS
   97 SUMSQ=SUMSQ+INTS*INTS
      STDV=SQRT(ABS((SUMSQ-SUM*SUM/EY   )/(   EY     -1.0)))
      OTZ123=OTZ123*STDV
      KE1=KE+1
      KE2=2*KEE+5
      KE4=4*KEE+6
      DO 399 IX=3,KE
  399 ICF(IX)=(IX-IX*IX+KE4)/2-KE1
  400 RETURN
      END


      SUBROUTINE RANDGN(IY)
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256)
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2
      EK=KE
      DO 5 I=2,KE
    4 IX=IY
      CALL RANDU (IX,IY,OYF)
      J=OYF*EK+1.
    3 IF(J-1)4,4,7
    7 IH=MSH(I)
      MSH(I)=MSH(J)
    5 MSH(J)=IH
      RETURN
      END


      SUBROUTINE LASR
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256)
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2
      IMT(2)=0
      IMS(KE)=0
  303 DO 247 IA=2,KEE
      IA1=IA+1
      IQ=0
      KR=MSH(IA)
      DO 244 IB=IA1,KE
      KC=MSH(IB)
      IF(KR-KC)500,500,510
  500 ML=KR
      MG=KC
      GO TO 244
  510 ML=KC
      MG=KR
  244 IQ=IQ+A(ML+SQ(MG))
  245 IMS(IA)=IQ
      IB=IA+1
      IQ=0
      KC=MSH(IB)
      DO 248 IAR=2,IA
      KR=MSH(IAR)
      IF(KR-KC)600,600,610
  600 ML=KR
      MG=KC
      GO TO 248
  610 ML=KC
      MG=KR
  248 IQ=IQ+A(ML+SQ(MG))
  247 IMT(IB)=IQ
      RETURN
      END
```

```
      SUBROUTINE FND (LIO,LIM)
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256)
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2
      DO 1 LAFF=2,KE
      IF(B(LIO)-MSH(LAFF))1,2,1
    2 LIM=LAFF
      GO TO 7
    1 CONTINUE
    7 RETURN
      END
```

```
      SUBROUTINE PLACE (LIM)
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256)
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2
      INTEGER*4 MS(256),MT(256)
      DO 507 IX=2,KE
      MS(IX)=IMS(IX)
  507 MT(IX)=IMT(IX)
  668 COEFB=0
      LIMP3=LIM+3
      LIMM1=LIM-1
      LIMP1=LIM+1
      LIMP4=LIM+4
      ISQM=SQ(LIM)
      IQ=1
      KF=MSH(LIM)
      DO 63 IX=3,KE
   70 IQ=IQ+1
      KV=MSH(IQ)
      IF(KV-KF)61,70,62
```

```
 61 KL=KV
    KG=KF
    GO TO 63
 62 KL=KF
    KG=KV
 63 ISQ(IX)=A(KL+SQ(KG))
303 MXX=IMS(2)
    MMX=MXX
    IF (LIM-2)932,932,901
901 MXX=MT(LIM)+MS(LIM)
    MMX=MXX
    DO 779 IMR=3,LIM
    ITS=LIMP3-IMR
779 MS(ITS)=MS(ITS-1) -ISQ(ITS)
932  IF(LIM-KEE)903,903,933
903 DO 905 LZE=LIMP1,KE
905 MT(LZE)=MT(LZE)-ISQ(LZE)
933 IF (LIM-4)910,920,920
920 DO 906 JIK=4,LIM
    IW=LIMP4-JIK
906 MT(IW)=MT(IW-1)
910 MS(KE)=0
    MT(3)=0
    DO 246 ILC=3,KEE
    MXX=MXX-ISQ(ILC)
    MS(ILC)=(MS(ILC)-MXX)*(KEP2 -ILC)
    KE4=KEP3-ILC
    MMX=MMX-ISQ(KE4)
246 MT(KE4)=(MMX-MT(KE4))*(KE4-1)
    MS(2)=COEFB
    IRIC4=2
    MSL=MS(2)
    DO 252 ILC=3,KE
251 MS(ILC)=MS(ILC)+MS(ILC-1)  +MT(ILC)
    IF(MS(ILC)-MSL)253,252,252
253 IRIC4=ILC
    MSL=MS (ILC)
252 CONTINUE
    COEFB=MSL
888 IF(IRIC4-LIM)500,7,501
501 IL=LIM
    IG=IRIC4
    GO TO 502
500 IL=IRIC4
    IG=LIM
502 IGP1=IG+1
    IGM1=IG-1
    ILP1=IL+1
    IQ=0
    DO 588 IXS=ILP1,IG
588 IQ=IQ+ISQ(IXS)
    IAS=IMS(LIM)
    IAT=IMT(LIM)
    IF (LIM-IRIC4)520,520,530
530 LGD=IG-IL
    IH=MSH(LIM)
    DO 531 IX=1,LGD
    IMR=IGP1-IX
    IRL=IMR-1
    MSH(IMR)=MSH(IRL)
```

```
      IR=ISQ(IMR)
      IMS(IMR)=IMS(IRL)-IR
  531 IMT(IMR)=IMT(IRL)+IR
      MSH(IL)=IH
      IMT(IL)=IAT-IQ
      IMS(IL)=IAS+IQ
      GO TO 7
  520 IH=MSH(LIM)
      DO 521 IX=IL,IGM1
      IXP1=IX+1
      MSH(IX)=MSH(IXP1)
      IXT=ISQ(IXP1)
      IMS(IX)=IMS(IXP1)+IXT
  521 IMT(IX)=IMT(IXP1)-IXT
      MSH(IG)=IH
      IMT(IG)=IAT+IQ
      IMS(IG)=IAS-IQ
    7 IMT(2)=0
      IMS(KE)=0
      RETURN
      END


      SUBROUTINE MCOEF
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256)
      INTEGER*2 A(32761),SQ,B,ISQ(256)
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2
      COEFB=0
      DO 740 L=3,KE
      LB=KE
      LR=ICF(L)-KE
      KC=MSH(L)
      LM1=L-1
      DO 740 MC=2,LM1
      KR=MSH(MC)
      IF(KR-KC)60,60,61
   60 KL=KR
      KG=KC
      GO TO 74
   61 KL=KC
      KG=KR
   74 LR=LR+LB
      LB=LB-1
  740 COEFB=COEFB+A(KL+SQ(KG))*LR
      RETURN
      END


      SUBROUTINE IFORM (IC)
      INTEGER*2 A(32761),SQ,B,NA (256)
      COMMON /CANL/IMS(256),IMT(256),MSH(256),ICF(256),ISVM
      COMMON A/EXTRA/SQ(256),B(256),COEFB,OTZ123,KE,KEE,KET,NVAL,INOS,I,
     1J,IUTR,BLNK,UNDLN,CP,KEM3,KEP1,KEP3,KEP2,NCOM,JNCOM
      DIMENSION C(80)
   87 FORMAT (T10,30(1X,A3))
```

```
 14 FORMAT (1H+,T10,30I4)
 66 FORMAT(T52,'D A T A  M A T R I X ',2X/1H+,T52,'_ _ _ _  _ _ _ _ _
   1_')
 67 FORMAT (T52,'I N P U T  M A T R I X ' ,I2/1H+,T52,'_ _ _ _ _  _ _
   1_ _ _ _  _')
 68 FORMAT (T52,'O R D E R E D  M A T R I X ',I2/1H+,T52,'_ _ _ _ _
   1_  _ _ _ _ _ _  _')
 79 FORMAT (T5,4X,1H|,A3,30(1X,A3))
 89 FORMAT (T10,41A3)
 23 FORMAT (' ',T46,'ORDERING COEFFICIENT =',E12.5)
 22 FORMAT (1H ,T46,'MATRIX COEFFICIENT C=',E13.6)
 24 FORMAT (1H0,5X,'REL.')
 10 FORMAT (5H+REL.,I3,31I4)
 25 FORMAT (1H1,3X)
 11 FORMAT (1H+,3X,32I4)
568 FORMAT (T52,'C O M B I N A T I O N ',I2/1H+,T52,'_ _ _ _ _ _ _ _
   1 _ _  _')
    CALL MCOEF
    CO=COEFB/OTZ123
511 FORMAT (' ',T46,'SERIATED RELATIVE SEQUENCE NO.',I4)
512 FORMAT (' ',25I5)
    GO TO (500),ISVM
    WRITE(3,568)JNCOM
    WRITE (3,511) IUTR
    WRITE (3,23)   COEFB
    WRITE          (3,22) CO
    DO 520 INT=2,KE
520 NA (INT)=MSH(INT)-1
    WRITE (3,512) (NA (INT),INT=2,KE)
    GO TO 4444
500 DO 4 IIII=1,INOS
    I76=0
    IG3Z=0
    K2N=1
    I90=0
  3 I91=I90+1
    I90=I90+30
    I66=30
    IG3Z=41
    IF(I90-KE)9,13,13
  9 WRITE (3,25)
    WRITE (3,568) JNCOM
    GO TO (60,61,62),IC
 60 WRITE (3,66)
    GO TO 400
 61 WRITE (3,67) IUTR
    GO TO 400
 62 WRITE (3,68) IUTR
400 WRITE (3,23) COEFB
    WRITE (3,22) CO
    WRITE (3,24)
    DO 82 L99=1,IG3Z
 82 C(L99)=UNDLN
    WRITE (3,89) (C(IO),IO=1,IG3Z)
    DO 800 IR1=I91,I90
800 NA(IR1)=MSH(IR1) -1
    GO TO (30),I91
    WRITE (3,14)  (NA(IZ13),IZ13=I91,I90)
    GO TO 31
 30 WRITE (3,10)  (NA(IZ1),IZ1=I91,I90)
```

```
  31 DO 7 I6=2,KE
     MI6=MSH(I6)
     DO 804 IR2=I91,I90
     MIR2=MSH(IR2)
     IF(MI6-MIR2)803,803,802
 802 LI1=MIR2
     LI2=MI6
     GO TO 804
 803 LI1=MI6
     LI2=MIR2
 804 NA(IR2)=A(LI1+SQ(LI2))
     ILINOF=0
     DO 19 ILIN=I91,I90
     ILINOF=ILINOF+1
     IF (NA(ILIN ) -NVAL)17,18,18
  17 C(ILINOF)=BLNK
     GO TO 19
  18 C(ILINOF)=UNDLN
  19 CONTINUE
     I76=I76+1
     GO TO (38),I91
  39 WRITE (3,87) (C(IFC),IFC=1,I66)
     WRITE (3,14) (NA(LI),LI=I91,I90)
     GO TO 902
  38 WRITE (3,79) (C(IFC),IFC=2,I90)
     WRITE (3,11) (NA(LI),LI=I91,I90)
 902 IF(I76/50-I76)7,901,7
 901 WRITE (3,25)
   7 CONTINUE
     WRITE (3,25)
     GO TO (3,4),K2N
  13 K2N=2
     IF (KE-I90) 243,245,4
 243 I66=KE-KE/30*30
     IG3Z=I66*4/3
 245 I90=KE
     GO TO 9
   4 CONTINUE
4444 RETURN
     END
```