# Integer Programming and the Theory of Grouping

**Article** *in* Journal of the American Statistical Association · June 1969

DOI: 10.1080/01621459.1969.10500990

Hrishikesh D. Vinod
Fordham University
193 PUBLICATIONS 3,085 CITATIONS

# INTEGER PROGRAMMING AND THE THEORY OF GROUPING

HRISHIKESH D. VINOD[1]

*Mathematica*

This paper is written with three objectives in mind. First, to point out that the problem of grouping, where a larger number of elements $n$ are combined into $m$ mutually exclusive groups ($m<n$) should be recognized as a problem in Integer Programming, and that such recognition can help us in avoiding complete enumeration of stages in grouping from $n$ to $n-1 \cdots$ to $m$, and of alternative possibilities in each stage. Second, to formulate mathematically some simple versions of the relevant Integer Programming Problem, so that the available computer codes can solve it. When the grouping attempts to minimize the within groups sums of squares, the so-called *string property* is proved to be necessary for the minimum (Lemma 1). It is shown that the string property can be exploited to write the non-linear within group sums of squares as a linear function (Lemma 2). An attempt is made to generalize the string property for the higher dimensional case. Third, to give some numerical examples to clarify the mathematical models and to illustrate the advantages of Integer Programming formulation by comparing with the numerical examples appearing in the literature [8], [13]. Zero-one programming where the integer variables can take values of zero or one only appears to be most efficient from the limited experience of the author. The paper is divided into three sections which closely correspond to the three objectives noted above.

## 1. CURRENT GROUPING PRACTICE, COMPLETE ENUMERATION AND INTEGER PROGRAMMING

A LARGE number of objects, persons, variables, symbols, etc. have often to be grouped into a smaller number of mutually exclusive groups so that members within a group are similar to each other in some sense. Such grouping does ordinarily result in certain loss of information or cost which can be quantified. This problem of grouping has received some attention in the literature of various natural as well as social sciences. It would be impossible to give an exhaustive list.[2]

In the currently used techniques for grouping, a point of view based on Integer Programming is absent. Ward and associates [5 and 13] have developed a "hierarchical grouping procedure" which is similar to what Fisher [8] calls

[1] The problem of grouping arose in connection with my work at Mathematica, Princeton, for the Office of High Speed Ground Transportation, Contract No. 7-35120. I am indebted to Joel Cord, Francis Sand and Kan Young of Mathematica for reading the earlier versions and to the anonymous referees and the editor for several valuable suggestions.

[2] The following is taken partially from Ward [13] and associates [5].

(a) Grouping of persons, for example, with respect to their sociological characteristics (Ball and Hall [3]) so that they are as much alike each other as possible. (b) Grouping of jobs to minimize the cross training time when men are reassigned according to established policies. (c) Grouping of job descriptions to minimize the errors in describing a large number of jobs with a small number of descriptions. (d) Grouping of regression equations to minimize the loss of predictive efficiency resulting from reductions in the number of equations. (e) Establishing the taxonomy of plants. (f) Grouping tropical fish in terms of the similarity of their eating habits. (g) Grouping scientists in terms of the similarity of their reading interests. (h) Profile analysis. (i) Simplication of Economic models (Fisher [8]) has also been considered as a grouping problem. For practical work in econometrics, the problem of simplifying a large-scale model for heuristic as well as analytical purposes is quite important. If there are too many variables, the models become incomprehensible, especially to the policy makers.

"progressive merger technique." The idea is to reduce the number of groups from $n$ to $n-1$ at each stage of the iteration until a small enough number of groups (say $m$) is obtained. The procedures advocated by Ward and Fisher involve complete enumeration of all stages of reduction from $n$ to $m$, and complete enumeration of the alternative ways of grouping at each stage of reduction. These algorithms canbe improved by using branch and bound techniques. An Integer Programming formulation of the grouping problem is useful[3] precisely because it does not require complete enumeration.

The ISODATA algorithm developed by Ball and Hall [3] is an iterative technique with steps similar to the ones taken in mathematical programming. Ball and Hall start with an arbitrary choice of what they call "typical" cluster points. In mathematical programming, we start with a basic feasible solution, and an approach to the optimum solution is guaranteed by some well-known theorems proved by G. B. Dantzig and others. The question of convergence of the iterative procedure to the optimum cluster is omitted by Ball and Hall [3]. The choice of the cluster center and the ultimate grouping of the elements are parts of one and the same basic Integer Programming problem, and the separation of these aspects may sometimes give incorrect solutions.

It should be possible to bring the tool kit developed for Integer Programming using the associated linear programming solutions, cutting planes, branch-bounds, warehouse location problems, etc. to bear on obtaining an efficient solution to the grouping problem. Balinski's [2] survey article gives an excellent account of the recent advances in Integer Programming. In the next section the analogy of the grouping problem to the warehouse location problem is shown. Clearly, new advances in Integer Programming merit serious consideration, although in some applications to behavioral sciences the *ad hoc* procedures may still be the most practical and cheapest way to proceed.

## 2. MATHEMATICAL FORMULATION OF THE RELEVANT INTEGER PROGRAMS

In this section we shall construct two mathematical formulations of the grouping problem. Both formulations are based on integer variables that can take values 0 or 1 only. The first formulation uses a constraint set similar to that of the warehouse location problems. The second formulation discusses the problem of minimization of the within-group sum of squares and gives alternative methods of converting the non-linear problem into a linear setup as required by Integer Programming codes. The main thrust of this section is to show that the Integer Programming formulation of the grouping problem is more general and flexible than it appears at first sight. It is hoped that the techniques for converting the non-linear problems will be of interest in themselves.

The reason for choosing formulations that use variables taking values 0 and 1 only is that some special computer codes are available[4] for them. Running times for these codes can be many times shorter than for other Integer Pro-

[3] It was pointed out to me by one of the referees that G. B. Dantzig had already recognized the relevance of an Integer Programming type of formulation in 1958 in private correspondence with Fisher. Of course the expression Integer Programming was not widely used at that time.

[4] I am indebted to K. Spielberg for making available to me his Warehouse Location and other codes which are extremely efficient. One is called SPLT 1 and inquiries concerning it should be addressed to him at IBM New York Scientific Center, 590 Madison Avenue, New York, New York 10022. Spielberg and Lemke's Direct Search Zero-One Integer Programming 1-DZIP 1 was also tried.

gramming codes where the integer variable can be larger than 1. Work in this area was stimulated by Balas [1], Glover [9], Lemke, Spielberg [11, 12] and others.

*Formulation 1*

We shall use the following set of notations, some of which will have to be slightly changed in the second formulation. First, we select a certain measurable characteristic, with respect to which we wish to make the grouping so that each element bears a certain value.

$n$ = total number of elements or objects to be grouped.

$p_i$ = value of the $i$th element: $i=1, 2, \cdots, n$.

$n_j$ = number of elements in $j$th group: $j=1, 2, \cdots, m$. Also

$$n = \sum_{j=1}^{m} n_j .$$

$m_o$ = the largest number of elements that can belong to a single group. If this is not specified, $m_o = n$.

$c_{ij}$ = loss of information, or cost involved in placing $i$th element into $j$th group. In the simplest case, $c_{ij} = |p_i - p_j|$ or $(p_i - p_j)^2$. The more general cases appear in (2.6.1) to (2.6.4) below. Always $c_{ii} = 0$ and $c_{ij} = c_{ji}$.

$x_{ij}$ = 1 if $i$th element belongs to the $j$th group and 0 otherwise.

$$n_j = \sum_{i=1}^{n} x_{ij}$$

is the column sum of the elements of the $x_{ij}$ matrix.

(0, 1) denotes a variable which can equal either zero or one but takes no other value.

Since there are $n$ elements and $m$ groups, one would expect that the matrices of elements $c_{ij}$ and $x_{ij}$ both have dimensions $n \times m$. This arrangement is not convenient because we do not know in advance the nature of the $m$ groups. We, therefore, pretend that we want to arrange the $n$ elements into $n$ groups, but some of the groups may be empty. When there are $m$ groups, $n-m$ of the column sums $n_j$ are zero. To identify the groups and to express the idea that there should be no more than $m$ groups we devise the idea of a group "leader"[5] and define the $j$th element as the leader of the $j$th group. We shall allow no more than $m$ leaders. Thus $y_j = 1$ if $j$th element is a leader (i.e., $n_j \neq 0$) and 0 otherwise.

For the first formulation of the grouping problem as an Integer Programming problem, it helps to consider an analogy with the well-known warehouse (plant) location problem in Integer Programming literature, because it has been well studied and some efficient computer programming algorithms are available to solve it. The analogy is admittedly artificial and should not be

[5] Notice, for example, that when the 2nd, 9th and 25th elements belong to the same group in the final solution, any one and only one of these three elements can be the leader. The scheme which chooses group leaders should not violate this requirement. If the 9th element is the leader, $x_{2,9} = x_{9,9} = x_{25,9} = 1$ and $x_{2,2} = x_{25,25} = x_{9,25} = x_{9,2} = x_{25,2} = x_{2,25} = 0$.

taken literally. Grouping of the elements with leaders is analogous to the customers being supplied from different warehouses. The warehouse location algorithm is expected to choose the group leaders and the grouping, at the same time, in an optimum fashion. We assume that the cost matrix $c_{ij}$ is known, constant and now has the interpretation that $c_{ij}$ is the cost of placing the $i$th element in that group where the $j$th element is the group leader. The analogy gives rise to the following formulation of the grouping problem which is not natural or completely general.

Now the Integer Programming problem is:[6]

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} c_{ij}, \tag{2.1}$$

which is the total cost of any scheme of grouping, subject to (2.2) to (2.5).

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all } i = 1, 2, \cdots, n, \tag{2.2}$$

which means that an $x_i$ cannot be put in more than one group at the same time.

$$\sum_{j=1}^{n} y_j = m, \tag{2.3}$$

which means that there are exactly $m$ groups (warehouses).

$$y_j \geq x_{1j},\ y_j \geq x_{2j},\ \cdots,\ y_j \geq x_{nj} \tag{2.4}$$

for all $j = 1, 2, \cdots, n$. This says that before the $i$th element can belong to a group where the $j$th element is the leader, we must make sure that it is indeed a leader. The information contained in the constraints (2.4) of which there are $n^2$ in number can be contained in the following set of only $n$ constraints[7] assuming (2.5).

$$n y_j \geq \sum_{i=1}^{n} x_{ij} \quad \text{for all } j = 1, 2, \cdots, n. \tag{2.4'}$$

$$x_{ij} \text{ and } y_j \text{ with } i \text{ and } j = 1, 2, \cdots, n \text{ are } (0, 1) \text{ variables.} \tag{2.5}$$

For some computer codes $y_j \leqq 1$ may have to be explicitly introduced as a constraint. Since (2.2) will not permit $x_{ij}$ larger than 1, $x_{ij} \leqq 1$ need not be explicitly introduced.

This formulation is applicable to cases where the $p_i$ are vectors with $h$ components, $(p_{i1}, p_{i2}, \cdots, p_{ih})$. The procedure is somewhat arbitrary. We define $c_{ij}$ after taking the higher dimensionality into account. The algorithm will minimize (2.1) with respect to the new definition of the cost matrix. The following definitions are possible. Let $\sigma_l$ denote the standard deviation of the $p_{il}$

[6] I am indebted to Michel Balinski for this formulation of the problem.

[7] If $y_j = 0$ so that the $j$th element is not a leader, (2.4′) will automatically ensure that $x_{1j}, x_{2j}, \cdots, x_{nj}$ are all identically $=0$. If $y_j = 1$, the $j$th element is one of the leaders and it is up to the Integer Programming algorithm to allocate the elements to this group. Although (2.4′) includes all information contained in (2.4) it does not exploit all the features of our problem. It works on the notion that conceivably all the elements may be attached to only one group. If we know in advance that no more than $m_o$ elements can conceivably belong to the same group, we should have $m_o y_j$ where $m_o < n$ as the left side of (2.4′).

values for $i=1, 2, \cdots, n$ and let $w_l$ denote some weights for $l=1, 2, \cdots, h$, specified in advance. We have:

$$c_{ij} = \sum_{l=1}^{h} (p_{il} - p_{jl})^2 w_l, \tag{2.6.1}$$

$$c_{ij} = \sum_{l=1}^{h} [(p_{il} - p_{jl})^2 w_l/\sigma_l], \tag{2.6.2}$$

$$c_{ij} = \sum_{l=1}^{h} | p_{il} - p_{jl} | w_l, \tag{2.6.3}$$

and

$$c_{ij} = \sum_{l=1}^{h} [| p_{il} - p_{jl} | w_l/\sigma_l]. \tag{2.6.4}$$

*Formulation 2*

Now we shall turn to the second formulation which is more general than the first. It is also based on (0, 1) variables. A familiar criterion is used, namely, minimization of the within-group sum of squares (WGSS). We shall use the same set of notations as before, wherever possible.

First, we rank the scalars, $p_1, p_2, \cdots, p_n$ in increasing order of magnitude of $p_i$. To emphasize the change we shall change the notation from $p_i$ to $q_k$ and define $q_k$ as the rank ordered values of $p_i$ with $k=1, 2, \cdots, n$ so that the following inequalities hold:

$$q_1 \leq q_2 \leq q_3 \cdots \leq q_n. \tag{2.7}$$

Without loss of generality, we can discuss the problem in terms of partitioning $q_1, q_2, \cdots, q_n$. Let us denote by $G_j$ that group in which $q_j$ is the smallest element. This gives a convenient scheme for identifying the groups. In the terminology of the previous formulation we make the smallest element the group "leader."

Define $x_{ij}=1$ if $q_i$ belongs to $G_j$ and $x_{ij}=0$ otherwise. From this definition we have $x_{ij}=0$ for $j>i$. In other words $q_i$ cannot belong to $G_{i+1}, G_{i+2}, \cdots, G_n$ because of (2.7). Thus the elements of the $x_{ij}$ matrix above the diagonal will be zero. This can be stated as the constraint

$$\sum_{i=1}^{n} \sum_{j>1}^{n} x_{ij} = 0. \tag{2.8}$$

Clearly, if (2.8) holds, with $x_{ij} \geq 0$, every element above the diagonal will have to be zero.

Now it is possible to write the minimization of WGSS as:

minimize

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}(q_i - \bar{q}_j)^2, \tag{2.9}$$

where

$$\bar{q}_j = \sum_{i=1}^{n} x_{ij}q_i/n_j \text{ is the group mean}$$

and

$$\sum_{i=1}^{n} x_{ij} = n_j.$$

If we can find $x_{ij}$ to minimize (2.9) subject to certain constraints, we would obtain the parition of $q_i$ elements into $m$ groups so that WGSS will be minimized.

Now we are ready to define what we call the *string property* (SP) of the $x_{ij}$ matrix for elements on or below the diagonal. It will be proved that the SP is necessary for the minimum of (2.9) to be attained, although it is not sufficient. The proof will use (2.7). By the SP we mean the requirement that $x_{ij}$ matrix should have an uninterrupted string of ones *below* the diagonal in its columns. Of course, some columns may have all zeroes. Since $m_o$ is the largest number of elements permitted to belong to a group, the longest string can have $m_o$ ones. The following set of linear constraints on the elements of $x_{ij}$ matrix below the diagonal can ensure that we have the uninterrupted string of ones.

$$x_{jj} \geq x_{j+1,j}, \cdots, \geqq x_{nj}, \quad \text{for all } j = 1, 2, \cdots, n. \tag{2.10}$$

Thus, (2.10) states the SP for the elements below the diagonal.

*Lemma 1.* The SP of (2.10) is necessary for the minimum of (2.9).

The SP is not new. It merely restates the known result [7] that in the one-dimensional case the groups should form separate intervals without any overlap. If there is an overlap, i.e., if the SP fails, the WGSS can be reduced (i.e., the between group sum of squares can be increased) by switching elements so as to avoid the overlap. The new group mean of the smaller set of elements will be smaller while the other group mean will be larger and obviously the two will be further apart increasing the between group sum of squares. The following proof clarifies the idea in our notations.

We can prove that for any column of $x_{ij}$, the SP must hold. The grouping $(q_j, q_{j+1}, \cdots, q_{j+l-1}, q_{j+l+1})$, $(q_{j+l})$ which violates the SP of (2.10) is suboptimal, because the value of the WGSS of (2.9) can always be reduced by regrouping these elements as $(q_j, q_{j+1}, \cdots, q_{j+l-1}, q_{j+l})$, $(q_{j+l+1})$. Let $Q_j = q_j + q_{j+1} + \cdots + q_{j+l-1}$. Now (former WGSS$-$later WGSS) can be shown to be

$$(q_{j+l+1} - q_{j+l})[(lq_{j+l+1} - Q_j) + (lq_{j+l} - Q_j)]. \tag{2.11}$$

If we can prove that (2.11) is non-negative, we can establish the Lemma. The first term of (2.11) is non-negative by (2.7). Now we have to show that the second term is also non-negative. (2.7) implies the following well-known inequality.

$$\sum_{i=1}^{k} w_i q_i \Big/ \sum_{i=1}^{k} w_i \leqq q_{k+1} \leqq q_{k+2} \cdots \leqq q_n, \tag{2.12}$$

where $w_i$ are some non-negative weights. Choose $w_1 = w_2 = \cdots = w_{j-1} = 0$ and $w_j = w_{j+1} = \cdots w_{j+l-1} = 1$. Thus $(Q_j/l) \leqq q_{j+l} \leqq q_{j+l+1}$ proves the Lemma for the $j$th column of $x_{ij}$. It pays to close the gap in the string of ones in the $j$th column.

If we start from the top left side of the $x_{ij}$ matrix and close the gaps from the top to the bottom in each column in that sequence, we shall finally satisfy the SP. Q.E.D.

It may be recalled that our main objective has been to reformulate (2.9) as (2.1) with appropriate definition of $x_{ij}$ and $c_{ij}$. We have so far defined and studied the structure of $x_{ij}$. Now, we may turn to the definition of $c_{ij}$, which measures the cost of grouping, i.e., the loss of information in placing $q_i$ in a group where $q_j$ is the "leader" (or the smallest element). We denoted this group as $G_j$. The definition of $c_{ij}$ is made in such a way that Lemma 2 below holds. A numerical example appears in Section 3, where (3.3) gives the cost matrix for a simple problem posed by Ward [13].

In finding the proper definition of the $c_{ij}$ matrix, we need not worry about the elements above the diagonal. Their value will not appear in the objective function because the corresponding $x_{ij}$ are specified to be zero by (2.9). We may arbitrarily specify that such elements are all equal to some large number. This point will be discussed later. Regarding the diagonal elements, $c_{ii}=0$ for all $i=1, 2, \cdots, n$ will hold as before. Our task now is simply to define elements of the $c_{ij}$ matrix below the diagonal, i.e., where $i>j$. We shall compute $c_{ij}$ as the *net* addition to the WGSS by including $q_i$ in the group having the elements $q_j, q_{j+1}, \cdots, q_{i-1}$. The answer obviously depends on $i$, $j$ and the data. We have

$$\begin{aligned} c_{ij} &= q_i^2 - (q_j + \cdots + \cdots + q_i)^2/(i-j+1) \\ &\quad + (q_j + \cdots + q_{i-1})^2/(i-j). \end{aligned} \tag{2.13}$$

*Lemma 2.* For $x_{ij}$ as defined above and satisfying the SP of (2.10) and $c_{ij}$ defined in (2.13), we have the following exact equality:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}c_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}(q_i - \bar{q}_j)^2. \tag{2.14}$$

We need only prove this Lemma for the $j$th column of the $x_{ij}$ matrix. The proof is trivial if the corresponding group is vacuous or $n_j=0$. Assume $n_j>0$. Now, by the SP, the elements in this group must be $q_j, q_{j+1}, q_{j+2}, \cdots, q_{j+n_j-1}$. Now the right hand side of (2.14) is easily shown to be

$$\sum_{l=j}^{j+n_j-1} q_l^2 - \left(\sum_{l=j}^{j+n_j-1} q_l\right)^2 \Big/ n_j. \tag{2.15}$$

The left hand side of (2.14) for the $j$th column must be

$$c_{jj} + c_{j+1,j} + \cdots + c_{j+n_j-1,j}. \tag{2.16}$$

Again, because of SP the corresponding $x_{ij}$'s for the remaining $c_{ij}$'s will be zero and will not appear here. Now, use the definition (2.13) of $c_{ij}$ and substitute in (2.16). There will be cancellations and the remaining terms will be the same as in (2.15). This proves the Lemma. Q.E.D.

Lemma 2 shows that by appropriate definition of the $c_{ij}$ values, thanks to the SP we can replace a non-linear function (2.9) by a simple linear function (2.1) using (0, 1) variables $x_{ij}$. The definition of $x_{ij}$ in this formulation is more restrictive than the formulation 1 and should not be confused with the latter.

Note that the $c_{ij}$ matrix is a part of the data for our algorithm, to be fed in before the solution is computed. The knowledge of $m_o < n$ can reduce the burden of this computation to a small extent. It is unnecessary to compute more than $m_o$ elements in any column of the matrix. We may specify the remaining elements of $c_{ij}$ to be some arbitrarily large number. Or, we can consider this as a constraint on the $x_{ij}$ values and specify that they are zero after $m_o$ units below each diagonal. For the last $m_o$ columns of $x_{ij}$ matrix this constraint will not be required because there cannot be more than $m_o$ elements below the diagonal. If these elements have to be zero individually, their sum also has to be zero, giving the constraint

$$\sum_{l=0}^{n-m_0-j} \sum_{j=1}^{n-m_0} x_{n-l,j} = 0. \tag{2.17}$$

We may make a few comments regarding the role of the elements in the $c_{ij}$ matrix in the context of the constraints on the corresponding $x_{ij}$ matrix. Clearly, as far as the integer programming algorithm is concerned, having any $x_{ij}=1$ is equivalent to making that $c_{ij}$ a part of the objective function. Thus if $c_{ij}$ is large, the algorithm will not make that $x_{ij}=1$ unless the constraints demand it. It is, therefore, possible to ensure that a certain $x_{ij}=0$ by making corresponding $c_{ij}$ very large. This gives an alternative way of including the constraints (2.8) and (2.17) by making all elements of the $c_{ij}$ above the diagonal and those more than $m_o$ units below the diagonal elements in each column, very large. Also, if the cost matrix obeys strict inequalities

$$c_{nj} > c_{n-1,j} > c_{n-2,j} > \cdots > c_{j+1,j} > c_{jj} \qquad \text{for all } j = 1, 2, \cdots, n, \tag{2.18}$$

the $x_{ij}$ values in the solution will indeed satisfy the SP as desired and the constraints in (2.10) need not be explicitly included. Whether to include (2.8), (2.10) and (2.17) as constraints or not, will depend on the nature of the $p_i$ values and the flexibility of the computer algorithm used. In general, it is safe to include them. In any case, once the cost matrix is constructed, one can check whether (2.18) holds[8] and then decide.

In the multivariate case, the condition (2.7) does not hold, and hence we need an entirely new definition of $x_{ij}$, $c_{ij}$ matrices and the constraints.

*Multivariate Case:*

When $p_1, p_2, \cdots, p_n$ are points in $h$-dimensional space, we can compute the Euclidian distance matrix of order $n \times n$ given by

$$d_{ij} = \sqrt{\sum_{l=1}^{h} (p_{il} - p_{jl})^2}. \tag{2.19}$$

Consider the $j$th column of $d_{ij}$. We can easily reorder these in the increasing order of magnitude so that

[8] The relation (2.18) does not always hold. For example, $q_1=1$, $q_2=2$ and $q_3=2.1$ has $c_{21}=.5$, $c_{31}=.24$ violating (2.18).

$$d_{jj} = 0 \leqq d_{i_1j} \leqq d_{i_2j} \leqq \cdots \leqq d_{i_{n-1}j}. \tag{2.20}$$

This can be compared to (2.7).

The generalized SP is given by the requirement that the string of ones should now follow a pattern given by $i_1, i_2, \cdots, i_{n-1}$ etc. Thus (2.10) generalizes to (2.21). For the $j$th column:

$$x_{jj} \geq x_{i_1j} \geq x_{i_2j} \geq \cdots \geq x_{i_{n-1}j}. \tag{2.21}$$

Again, if $m_o$ is fixed this string need not proceed beyond $m_o$ and no more than $m_o$ $c_{ij}$ values need be computed in each column.

When the WGSS is to be minimized, the generalized SP of (2.21) is necessary for the minimum (not sufficient). The proof is similar to that of the Lemma 1 above. To save space we shall not give a proof. The structure of the $x_{ij}$ is different now, and the constraints (2.8) and (2.17) should not be imposed. Now, consider the structure of $c_{ij}$. One should record which $i$ corresponds with which $i_k$ because our matrix $c_{ij}$ has $i$ as the index variable. Contribution of the $j$th column to the WGSS when $p_j$ is grouped with $p_{i_1}$ is given by $(d_{i_1j})^2/2$. For the group $(p_j, p_{i_1}, p_{i_2})$, the WGSS is $d^2_{i_1j}+d^2_{i_2j}-(d_{i_1j}+d_{i_2j})^2/3$. The *net* addition to the WGSS is given by $d^2_{i_2j}-(d_{i_1j}+d_{i_2j})^2/3+d^2_{i_1j}/2$. Thus a generalization of (2.13) for multivariate case when $i_k \neq j$ is:

$$c_{i_kj} = d^2_{i_kj} - \left(\sum_{l=1}^{k} d_{i_lj}\right)^2 \bigg/ (k+1) + \left(\sum_{l=1}^{k-1} d_{i_lj}\right)^2 \bigg/ k, \tag{2.22}$$

where $k=1, 2, \cdots, n-1$. Of course, $c_{jj}=0$ for all $j$. For the $c_{ij}$ matrix computed using (2.22) and the $x_{ij}$ satisfying the generalized SP of (2.21) we can show that WGSS $=\Sigma\Sigma c_{ij}x_{ij}$ as required by the Integer Programming algorithm.

From Lemma 1 and 2 and the above discussion we conclude that the nonlinearity of the WGSS (2.9) can be eliminated for all practical purposes by a suitable definition of the $c_{ij}$ and $x_{ij}$ matrices The linearization holds in the multivariate case also. It is exact, i.e., it does not use any Taylor series kind of approximative technique.

## 3. NUMERICAL EXAMPLES

In this section an attempt is made to illustrate the mathematical formulations of the previous section by numerical examples.

Simplification of economic models was given as one of the areas of application of the theory of grouping and was listed as (i) in footnote 2. Fisher's [8] paper is a pioneering work in this area.

Fisher's numerical example is taken from the Klein-Goldberger model [10]. There are four equations and fourteen variables. We want to simplify the model by grouping the fourteen explanatory variables into seven groups. The data used consist of (a) a matrix $P$ of size $4\times 14$ giving the regression coefficients in the four equations and (b) the variance covariance matrix $M$ of size $14\times 14$.

Every row of the matrix $P$ gives regression coefficients of a different equation, and we expect that the optimal grouping will be different for different equations. Now, consider the first row of the matrix $P$ giving the estimates of the regression coefficients $(b_1, b_2, \cdots, b_{14})$ of the first equation (consumption function). We shall denote the explanatory variables as $x_1, x_2, \cdots, x_{14}$. The problem is to derive the best partition of these fourteen variables into seven groups. According to the solution given by Fisher, the following grouping is optimal in the sense that it minimizes a certain criterion: $(x_1)$, $(x_2, x_3, x_4, x_5)$, $(x_6, x_8, x_{12})$, $(x_7, x_9, x_{13})$, $(x_{10})$, $(x_{11})$, $(x_{14})$.

This was arrived at by his progressive merger technique. The criterion chosen by him is quadratic.

$$r = \operatorname{tr} PMP' - \operatorname{tr} \overline{P}\overline{M}\overline{P}', \tag{3.1}$$

where tr denotes the trace of the matrix product and $\overline{P}$ and $\overline{M}$ mean matrices obtained by grouping rows and columns. Fisher [8] does not give adequate interpretation of his quadratic criterion (3.1). A different linear criterion also may be appropriate where absolute values are used rather than squares. The usual tests of significance in statistics are made by using the following criterion:

$$t_{ij} = \frac{b_i - b_j}{\sqrt{\operatorname{Var}(b_i - b_j)}}, \tag{3.2}$$

where $b_i$ and $b_j$ are the least square estimates of the regression coefficients of $x_i$ and $x_j$ respectively. Denote the elements of the inverse of $M$ by superscripts as $m^{ij}$. Now from elementary regression analysis we know that Var $(b_i - b_j) = m^{ii} + m^{jj} - 2m^{ij}$. Hence we can easily construct the $14 \times 14$ cost matrix defined by $c_{ij} = |t_{ij}|$.

A comparison of my grouping[9] with Fisher's [8] for the four equations is given below. We give merely the subscripts of variables $x$ for convenience. Some variable numbers are underlined to show where Fisher's grouping differs from mine.

*Equation 1:* Consumption

my: (2, 6, 8, 12)(3, 4, 5)(7, 9, 13)(10)(11)(14)

Fisher: (1)($\underline{2}$, 3, 4, 5)(6, 8, 12)(7, 9, 13)(10)(11)(14)

*Equation 2:* Depreciation

my: (1)(2)(3)(4, 5, 10)(6, 8, 11, 12)(7, 9, 13)(14)

Fisher: (1)(2)(3)(4, 5, 10)(6, 8, 11, 12)(7, 9, 13)(14)

*Equation 3:* Private Wage Bill

my: (1, 3, 14)(2)(4, 5)(6, 8, 12)(7, 13)(9, 10)(11)

Fisher: (1, 3, 14)(2)(4)($\underline{5}$, 9, 10)(6, 8, 12)(7, 13)(11)

*Equation 4:* Gross National Product

my: (1)(2, 6, 8, 12)(3, 4, 5)(7, 10, 13)(9)(11)(14)

Fisher: (1, $\underline{14}$)($\underline{2}$)(6, 8, 12)(3, 4, 5)(7, 9, 13)(10)(11)

Fisher draws four tree diagrams showing the optimal groups when the de-

[9] See footnote 4 for details about the code and how to obtain it.

sired number of groups ($m$) is respectively 7,6,5,4,3,2 by progressive merger. The comparison with the cases where $m=6,5,4$ etc. shows that the discrepancies in the two ways of grouping diminish as $m$ becomes smaller in most cases. Our criterion is similar to Fisher's. The main difference is that we compute absolute values rather than squares. Fisher's progressive merger technique cannot be adjusted to take any other criterion function. By contrast, our technique can accept any linear criterion. The computer time required for solving our Integer Programming problems was no more than 1 second on the IBM 360 machine time, while the time for Fisher's algorithm is 20 seconds. One should be careful and not conclude from this that Fisher's method is inferior for all situations. If all stages of the grouping tree are required, Fisher's method may be the best way to proceed. The above numerical example illustrates the first model in (2.1) to (2.5).

The second formulation in the previous section accepts a nonlinear criterion function. As an illustration of this technique the numerical example given by Ward [13] was solved where $n=5$ and we have $p_1=1$, $p_2=7$, $p_3=2$, $p_4=9$, $p_5=12$. Hence $q_1=1$, $q_2=2$, $q_3=7$, $q_4=9$, $q_5=12$. It gives rise to the following cost matrix using the definition (2.13) for the elements below the diagonal.

$$\begin{bmatrix} 0 & 10000 & 10000 & 10000 & 10000 \\ .5 & 0 & 10000 & 10000 & 10000 \\ 20.17 & 12.50 & 0 & 10000 & 10000 \\ 24.08 & 13.50 & 2 & 0 & 10000 \\ 42.05 & 27 & 10.67 & 4.50 & 0 \end{bmatrix} \tag{3.3}$$

This cost matrix does obey the inequality in (2.18). Thus, we do not need to explicitly introduce the constraint (2.10). The numbers 10000 can be defined to be very large for Spielberg's computer code. Hence the explicit introduction of constraint (2.8) is unnecessary. The constraint (2.17) is unnecessary because $m_o=n=5$ for the Ward's problem. The following solution was obtained for different values of $m$.

$$\begin{aligned} m &= 4: \quad (q_1, \underline{q_2}), (\underline{q_3}), (\underline{q_4}), (\underline{q_5}) \\ m &= 3: \quad (q_1, \underline{q_2}), (q_3, \underline{q_4}), (\underline{q_5}) \\ m &= 2: \quad (q_1, \underline{q_2}), (q_3, q_4, \underline{q_5}) \end{aligned}$$

The group leaders as chosen by the algorithm have been underlined. This solution is identical to the one given by Ward as it should be. (He quotes them in terms of $p$ variables.)

The Integer Programming formulation was applied to solve medium sized problems for grouping commodities with respect to their transportation rate characteristics. The results of that study are encouraging.[10]

[10] These results cannot be quoted here before they are reviewed by the Department of Transportation, Washington, D. C. My problems had $m=6$ and $n=46$. The typical machine time on IBM 360 was about 450.33 seconds. In one case it took 814.950 seconds. I did not use all the features of the Spielberg's program that may save time, such as specification of the tolerances. I believe the program can be adapted for any computer having the capabilities. of IBM 7094 or IBM 360. Spielberg has used his code for problems of size 92×92 which take about 12 minutes. According to Spielberg the code can be improved and extended to solve much larger problems if there is demand for the improved code.

In conclusion, we note that Integer Programming is a more flexible and efficient method of attacking the grouping problem. The numerical examples given by Ward [13] and Fisher [8] were quickly and cheaply solved by our method. The solutions are identical to Ward's where the cost criterion is the same. The solution is comparable to Fisher's where the criterion is slightly different. The Integer Programming formulation should be looked upon as an alternative worthy of consideration. It requires linearity of the objective function, and that is its main limitation. This paper gives new ways of converting exactly, some quadratic objective functions into linear. Our technique involves no approximation such as in the case of Taylor series or other linearizations. We exploit the special features of the grouping problem, namely, the *string property*. More programming effort is necessary to solve very large problems.

## REFERENCES

[1] Balas, E., "An additive algorithm for solving linear programs with zero-one variables," *Operations Research,* Vol. 13, No. 4 [July–August 1965] 517–46.

[2] Balinski, M. L., "Integer programming: methods, uses and computation," *Management Science,* 12 [November 1965] 253–313.

[3] Ball, G. H. and Hall, D. J., "A clustering technique for summarizing multivariate data," *Behavioral Science,* Vol. 12, No. 2 [March 1967] 153–55.

[4] Benders, J. F., "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik,* 4 [February 1962] 238–52.

[5] Christal, R. E. and Ward, J. H., Jr., "The Maxof clustering model," see reference [6]. Chapter 11, pp. 11.01–11.45.

[6] Conference on clustering analysis of multivariate data, New Orleans, La., Report, Clearinghouse, Department of Commerce, Springfield, Virginia, Document #AD 653–722, June 1967.

[7] Fisher, W. D., "On grouping for maximum homogeneity," *Journal of the American Statistical Association,* 53 [1958] 789–98.

[8] Fisher, W. D., "Simplification of economic, models," *Econometrica* Vol. 34, No. 3 [July 1966] 563–84.

[9] Glover, F., "A multiphase-dual algorithm for the zero-one integer programming algorithm," *Operations Research,* Vol. 13, No. 6 [November–December 1965] 879–919.

[10] Goldberger, A. S. *Impact multipliers and dynamic properties of the Klein-Goldberger model.* Amsterdam: North-Holland Publishing Company, 1959.

[11] Lemke, C. E., and Spielberg, K., "Direct search algorithms for zero-one and mixed integer programming," *Operations Research,* Vol. 15, No. 5 [September–October 1967] 892–914.

[12] Speilberg, K., "On the fixed charge transportation problem," *Proceedings of the 19th National Conference,* Association for Computing Machinery, 211 East 43rd Street, New York, August 1964.

[13] Ward, J. H., Jr., "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association,* 58 [1963] 236–44.

## APPENDIX

In some practical applications, for example for computer match-making, for assignment of individual pupils to courses, etc., we know in advance the exact number of elements that may belong to a group. This case gives rise to a very simple computation of the $c_{ij}$ matrix even in the multivariate case.

For this formulation, we have to re-define the $x_{ij}$ matrix and adopt a new convention. The reader is warned against confusing the notation with that of the formulations 1 and 2. Here there is no "leader" of a group. All are leaders. This involves greater repetition of ones in the $x_{ij}$ matrix.

$x_{ij}=1$ if $i$ and $j$ belong to the same group and 0 otherwise.
$m'=$ the fixed number of elements that can and must be grouped together.
$p_i$, $n$ and $m$ have the same meaning as in the previous formulation but $n \neq \Sigma n_j$. Also $\Sigma_i x_{ij} = \Sigma_j x_{ij} = m'$ and $m'm = n$.

The non-linear objective function in (2.9) can be re-written[1] as:

$$\sum_j \sum_i x_{ij}(p_i - p_j)^2/2 \sum_i x_{ij}. \tag{A.1}$$

The non-linearity of (A.1) arises because $\Sigma_i x_{ij}$ in the denominator is not known before the solution to the Integer Programming problem is found. But, if we impose the restriction that $\Sigma_i x_{ij}=m'$ and is known in advance, we can simply define

$$c_{ij} = (p_i - p_j)^2/2m', \tag{A.2}$$

so that we have $\Sigma\Sigma x_{ij}c_{ij}$ as the objective function which is linear.

Note that the constraint (2.2) should no longer be imposed: instead we should have:

$$\sum_j x_{ij} = m' \qquad \text{for all } i = 1, 2, \cdots, n. \tag{A.3}$$

There is no need to define $y_j$ and the constraints involving its values. We should have

$$\sum_i x_{ij} = m' \qquad \text{for all } j = 1, 2, \cdots, n. \tag{A.4}$$

Notice that with the new convention making all elements as group leaders, the $x_{ij}$ matrix is symmetric and this fact should appear as a constraint.

$$x_{ij} = x_{ji} \qquad \text{for all } i \text{ and } j = 1, 2, \cdots, n. \tag{A.5}$$

It is not necessary to impose the fact that there are $m$ groups as a constraint. Knowledge of $m'$ and $n$ determines $m$ from the identity $n=m'm$.

$$x_{ij} \text{ are } (0, 1) \text{ variables.} \tag{A.6}$$

This may or may not have to be explicitly introduced, depending on the algorithm used. This formulation of the Integer Program appears to be suitable

[1] To see this result, consider a simple example where $n=6$, $m=2$, $m'=3$ where $(p_1, p_4, p_4)$, $(p_2, p_3, p_6)$ are the two groups. In new notation, where all elements are leaders, $3=\sum_i x_{ij} = \sum_j x_{ij}$,

$$[x_{ij}] = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

and the WGSS $=\frac{1}{3}[3\sum_1^6 p_j^2 - (p_1+p_4+p_5)^2 - (p_2+p_3+p_6)^2]$.

for the case where $p_i$ are higher order vectors, say $h$-order with components $p_{i1}, p_{i2}, \cdots, p_{ih}$. If these are $h$ observations on an element and if we insist on minimizing within-group sum of squares as the criterion, this is a convenient way of making the objective function linear, so that it is possible to obtain the numerical solution. Now we have:

$$c_{ij} = \sum_{l=1}^{h} (p_{il} - p_{jl})^2/2m'. \tag{A.7}$$

The main advantage of this formulation lies in the simplicity with which $c_{ij}$ can be computed in the multivariate case. The main limitation is the fact that $m'$ is assumed fixed and known.