PSYCHOMETRIKA—VOL. 29, NO. 2
JUNE, 1964

NONMETRIC MULTIDIMENSIONAL SCALING: A NUMERICAL METHOD

J. B. KRUSKAL
BELL TELEPHONE LABORATORIES

We describe the numerical methods required in our approach to multidimensional scaling. The rationale of this approach has appeared previously.

1. *Introduction*

We describe a numerical method for multidimensional scaling. In a companion paper [7] we describe the rationale for our approach to scaling, which is related to that of Shepard [9]. As the numerical methods required are largely unfamiliar to psychologists, and even have elements of novelty within the field of numerical analysis, it seems worthwhile to describe them.

In [7] we suppose that there are n objects $1, \cdots, n$, and that we have experimental values δ_{ij} of dissimilarity between them. For a configuration of points $x_1, \cdots, x_n$ in t-dimensional space, with interpoint distances d_{ij}, we defined the *stress* of the configuration by

$$S = \sqrt{\frac{S^*}{T^*}} = \sqrt{\frac{\sum (d_{ij} - \hat{d}_{ij})^2}{\sum d_{ij}^2}},$$

where the values of $\hat{d}_{ij}$ are those numbers which minimize S subject to the constraint that the $\hat{d}_{ij}$ have the same rank order as the δ_{ij}. More precisely, the constraints are that $\hat{d}_{ij} \leqq \hat{d}_{i'j'}$ whenever $\delta_{ij} < \delta_{i'j'}$.

The stress is intended to be a measure of how well the configuration matches the data. More fully, it is supposed that the "true" dissimilarities result from some unknown monotone distortion of the interpoint distances of some "true" configuration, and that the observed dissimilarities differ from the true dissimilarities only because of random fluctuation. The stress is essentially the root-mean-square residual departure from this hypothesis.

By definition, the best-fitting configuration in t-dimensional space, for a fixed value of t, is that configuration which minimizes the stress. The primary computational problem is to find that configuration. A secondary computational problem, of independent interest, is to find the values of $\hat{d}_{ij}$ from the fixed given values of d_{ij}; this is the computational problem of "monotone regression." This latter computation constitutes one step of the main computation.

2. *Missing Entries*

In some cases not all dissimilarities will be observed. Frequently the self-dissimilarities δ_{ii} are either meaningless or unobserved. Sometimes there is no distinction experimentally between δ_{ij} and δ_{ji} so that only a half-matrix of dissimilarities is obtained. Sometimes certain individual dissimilarities may simply fail to be observed. If the number n of objects is large (say 40 or 50), the experimenter may very wisely decide in advance to observe only a fraction of the dissimilarities for reasons of cost. Whatever the reason, we adapt to the situation by a very simple change in the definition of the stress: namely, both sums which appear in that definition are restricted to run over those pairs (i, j) for which δ_{ij} is observed. Thus we accommodate missing observations without loss of elegance. Throughout this paper all similar sums will be understood in the same sense unless otherwise indicated.

Of course, if there are not enough dissimilarities observed our method will break down. What this means is that there will be a zero-stress configuration which has no real relationship to the data. One important case in which this occurs is when the objects are split into two groups and the only dissimilarities observed are those between objects in different groups. In this case there is a simple zero-stress configuration in one dimension, namely two distinct points, where each point represents all the objects in one group.

On the other hand, it is not merely a question of how many dissimilarities are observed, but depends on which ones are observed. In many cases of practical importance, one-half or one-quarter of the dissimilarities or fewer are quite sufficient if they are properly distributed in the matrix of all possible dissimilarities.

3. *Non-Euclidean Distance*

Of major interest is the ordinary case in which the distances are Euclidean. If the point x_i has (orthogonal) coordinates $x_{i1}, \cdots, x_{it}$, then the Euclidean (or Pythagorean) distance from x_i to x_j is given by

$$d_{ij} = \left[\sum_{l=1}^{t} (x_{il} - x_{jl})^2 \right]^{1/2}.$$

However, the theory is applicable to much more general distance functions. The numerical methods and formulas given in this paper cover a class of distance functions most often called the L_p or l_p metrics, but occasionally known as Minkowski r-metrics (the term we use). The Minkowski r-metric distance is given by

$$d_{ij} = \left[\sum_{l=1}^{t} |x_{il} - x_{jl}|^r \right]^{1/r}.$$

For $r \geqq 1.0$, this metric is a genuine distance function because it satisfies

the triangle inequality. (For proof see ([6], pp. 19–22) or ([4], pp. 30–33).) We restrict ourselves to these cases.

For $r = 2.0$, this metric becomes Euclidean distance. For $r = 1.0$, this metric becomes the so-called "city-block distance" or "Manhattan metric"

$$d_{ij} = \sum_{l=1}^{t} |x_{il} - x_{jl}|.$$

For $r = \infty$, it becomes a familiar metric,

$$d_{ij} = \max_{l} |x_{il} - x_{jl}|,$$

which is sometimes called the l_∞-metric but is widely referred to in colloquial mathematics as the "sup" metric ("sup" is short for supremum).

4. *The Method of Steepest Descent*

Let us now focus on the computational problem that faces us. We restrict our attention to a fixed number of dimensions t and a fixed metric, that is, a fixed value of r. The determination of their values is a matter of judgment, and in many cases can be properly decided only after making analyses with several different values. Thus for the computational question we may assume fixed values of t and r.

An entire configuration can be described as a single vector (or a single point—we use the terms interchangeably) in nt-dimensional space, whose coordinates x_{il} for $i = 1$ to n and $l = 1$ to t are all the coordinates of all the points of the configuration. We refer to nt-dimensional space as "configuration space" and to t-dimensional space as "model space." We emphasize the fact that while a configuration has previously been viewed as n points in model space, we may with equal validity consider it as a single point

$$(x_{11}, \cdots, x_{1t}, \cdots, x_{n1}, \cdots, x_{nt})$$

in configuration space.

Suppose now that the values of δ_{ij} are given. Then for any point in configuration space, that is, for any configuration, there is a definite stress value S. In other words, S is a function

$$S = S(x_{11}, \cdots, x_{1t}, \cdots, x_{n1}, \cdots, x_{nt})$$

defined on the points of configuration space. Our problem is to find that point which minimizes S. Thus we are faced with a standard problem of numerical analysis: to minimize a function of several variables.

For a general review of methods used to solve this problem, see Spang [10], which includes a good bibliography.

To solve this problem we adopt a widely used method of numerical

analysis. It is called the "method of steepest descent" or the "method of gradients." We start by picking a more or less arbitrary point in configuration space. In other words, we start with an arbitrary configuration. We then wish to improve the configuration a bit by moving it around slightly. The method of steepest descent calls for this to be done by ascertaining in which direction in configuration space S is decreasing most quickly, and moving a short step in that direction. This direction is called the (negative) gradient and is determined by evaluating the partial derivatives of the function S. In fact,

$$\left(-\frac{\partial S}{\partial x_{11}}, \cdots, -\frac{\partial S}{\partial x_{1t}}, \cdots, -\frac{\partial S}{\partial x_{nt}}\right)$$

is the (negative) gradient. After arriving at a new, slightly better point in configuration space, we again determine the gradient, which is different at different points, and move along it. After many repetitions we arrive at a point from which no improvement is possible, in other words, at a minimum value of S. This is what we are looking for. We can tell when this happens, for at a minimum all the partial derivatives are zero, that is, the gradient vector is zero.

5. *The Difficulty of Local Minima*

A point in configuration space from which no small movement is an improvement is by definition a local minimum. However, a local minimum may or may not be an overall minimum. Fig. 1 shows a function of one

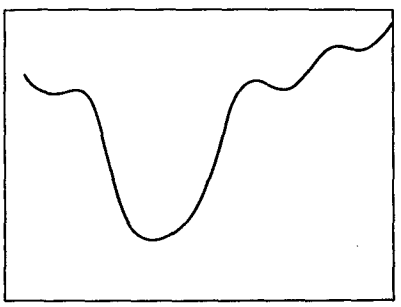

FIGURE 1

variable with four local minima. Only one of them is the overall minimum. If we seek a minimum by the method of steepest descent or by any other method of general use, there is nothing to prevent us from landing at a local minimum other than the true overall minimum. This is a widely known difficulty—in fact, it could be called a standard difficulty of such problems.

In certain important minimization problems the only local minimum is the true overall minimum, so this difficulty does not arise. With this important exception, there are few minimization problems (in numerical analysis) in which the local minimum difficulty can truly be vanquished. At best, we can hope for reasonable confidence that we have the true minimum.

In our minimization problem the difficulty is quite mild. In most cases of interest it need not be a serious concern, for these reasons.

First, we can easily start the method of steepest descent from a variety of different initial configurations. In principle, each initial configuration could lead to a different local minimum. While only the smallest of these could possibly be the true minimum, we would wonder about other still smaller local minima. Usually in our minimization problems, most initial configurations lead to the same local minimum, and this local minimum is much smaller than the few other local minima we find. It seems needless to worry when this is so.

Second, the local minimum configuration which we suppose to be the true overall minimum is not in itself the final end product of the analysis, which must be accepted blindly. In most cases this configuration is of interest only if it makes sense, only if it can be interpreted or is useful in giving the experimenter insight. If a configuration does this, it is unlikely to be seriously at fault.

Third, unless the stress of the supposed true minimum configuration is sufficiently small, we will not be interested anyhow. A minimum-stress configuration whose stress is above 20% is unlikely to be of interest. Above 15% we must still be cautious; from 10% to 15% we wish it were better; from 5% to 10% is satisfactory; below 5% is impressive.

Fourth, many checks are possible by detailed comparison of the configuration and the data, and by separate analysis of parts of the data.

6. *Numerical Technique*

In principle the iterative technique we use to minimize the stress is not difficult. It requires starting from an arbitrary configuration, computing the (negative) gradient, moving along it a suitable distance, and then repeating the last two steps a sufficient number of times. In this section we discuss some computational aspects which are entirely independent of which computer and which programming language are used.

Since the stress is invariant under translation and uniform stretching and shrinking, we always normalize a configuration by first placing its centroid (center of gravity) at the origin and then by stretching or shrinking so that the root-mean-square distance of the points from the origin equals one. (In our program we have arbitrarily chosen to use Euclidean distance for this purpose, regardless of which Minkowski distance is being used for

the interpoint distances.) A configuration which has these properties is said to be normalized.

If ordinary Euclidean distance is used, then the stress is invariant under all rotations, so it becomes possible, and for some purposes desirable, to normalize the angular attitude of the configuration. A natural way to do this is to rotate the configuration so that its so-called principal axes coincide with the coordinate axes (in the natural order). On the other hand, using Minkowski r-metric distance for $r \neq 2$, the only rotations which leave stress invariant are those which transform coordinate axes into coordinate axes. In this case it is possible, and perhaps desirable, to normalize the angular attitude by rotating so that the so-called one-dimensional variance decreases from one coordinate axis to the next. While these normalizations are not difficult, they can easily be left as a separate operation. Therefore we do not discuss them further here.

If a fairly good configuration is conveniently available for use as the starting configuration, it may save quite a few iterations. If not, an arbitrary starting configuration is quite satisfactory. Only two conditions should be met: no two points in the configuration should be the same, and the configuration should not lie in a lower-dimensional subspace than has been chosen for the analysis. If no configuration is conveniently available, an arbitrary configuration must be generated. One satisfactory way to do this is to use the first n points from the list

$$(1, 0, 0, \cdots, 0, 0),$$
$$(0, 1, 0, \cdots, 0, 0),$$
$$\cdots$$
$$(0, 0, 0, \cdots, 0, 1),$$
$$(2, 0, 0, \cdots, 0, 0),$$
$$(0, 2, 0, \cdots, 0, 0), \text{ etc.}$$

Another way would be to generate the points by use of a pseudorandom number generator. In either case the resulting configuration should be normalized.

Suppose we have arrived at the configuration x, consisting of the n points $x_1, \cdots, x_n$ in t dimensions. Let the coordinates of x_i be $x_{i1}, \cdots, x_{it}$. We shall call all the numbers x_{is}, with $i = 1, \cdots, n$ and $s = 1, \cdots, t$, the coordinates of the configuration x. Suppose the (negative) gradient of stress at x is given by g, whose coordinates are g_{is}. Then we form the next configuration by starting from x and moving along g a distance which we call the *step-size* α. In symbols, the new configuration x' is given by

$$x'_{is} = x_{is} + \frac{\alpha}{\text{mag } (g)} g_{is}$$

for all i and s. Here mag (g) means the relative magnitude of g and is given by

$$\text{mag } (g) = \sqrt{\sum_{i,s} g_{is}^2} \Big/ \sqrt{\sum_{i,s} x_{is}^2} .$$

If we assume that x is normalized, then a simpler formula is valid:

$$\text{mag } (g) = \sqrt{\frac{1}{n} \sum_{i,s} g_{is}^2} .$$

We give the formulas for g in another section. Of course, x' should be normalized before further use.

The step-size α is varied from one iteration to the next. The step sizes used do not affect the solution ultimately obtained. However, they profoundly affect the number of iterations required to reach the solution, and are an important computational consideration.

The step-size procedure given here is the result of considerable numerical experimentation. No claim is made that it is optimal in any sense. However, it seems to be reasonably fast, it is robust, and it avoids many pitfalls which we discovered in earlier procedures. It provides large steps during the early stages of calculation and small steps at the end. It is capable of providing a very exact solution when desired.

The initial value of α with an arbitrary starting configuration should be about 0.2. For a configuration that already has low stress, a smaller value should be used. (A poorly chosen value results only in extra iterations.) Thereafter the step size is determined by the following formula.

$$\alpha_{\text{present}} = \alpha_{\text{previous}} \cdot (\text{angle factor}) \cdot (\text{relaxation factor}) \cdot (\text{good luck factor}),$$

where

$$\text{angle factor} = 4.0^{(\cos\theta)^3 \cdot 0},$$

θ = angle between the present gradient and the previous gradient,

$$\text{relaxation factor} = \frac{1.3}{1 + (\text{5-step-ratio})^{5.0}},$$

$$\text{5-step-ratio} = \min\left[1, \left(\frac{\text{present stress}}{\text{stress 5 iterations ago}}\right)\right],$$

$$\text{good luck factor} = \min\left[1, \left(\frac{\text{present stress}}{\text{previous stress}}\right)\right].$$

If five iterations have not yet been calculated, the "stress five iterations ago" may be taken as the first stress computed. Similar artifices may be used in connection with "previous stress" and θ on the very first iteration.

If g is the present gradient and g'' the previous gradient, $\cos\theta$ may be calculated by

$$\cos\theta = \frac{\sum_{i,s} g_{is} g''_{is}}{\sqrt{\sum_{i,s} g_{is}^2} \sqrt{\sum_{i,s} g''^2_{is}}}.$$

As the computation proceeds, successively smaller values of stress are achieved. (Occasionally, an iteration may increase the stress rather than decrease it.) Eventually the stress "levels off," and further iterations cause little or no improvement. When is it time to stop and consider the configuration then obtained as sufficiently accurate? There is no really good answer to this question. However, the following rough guide, sensibly used, provides a practical answer.

As the computation proceeds, the relative magnitude mag (g) of the successive gradients decreases. At a configuration which is precisely a minimum, the gradient and its magnitude are zero. Subject to the following qualifications, we suggest that when the magnitude mag (g) reaches a value of approximately 2 per cent of its value for a typical arbitrary configuration, then the iteration may be terminated. This value of mag (g) at which we stop will be called the *local minimum criterion*. For data with large statistical variations, larger values are appropriate, and conversely. For large values of n and t (say $n = 40$, $t = 2$ or $n = 15$, $t = 5$), a larger local minimum criterion is appropriate, and for small values of n and t (say $n = 9$, $t = 2$ or $n = 15$, $t = 1$), a smaller value is appropriate. A larger value of the criterion could mean 5 per cent or conceivably as high as 10 per cent. A smaller value could mean 0.5 per cent or anything down to 0 per cent. If it appears possible and desirable to achieve a stress of zero, then of course the computation should continue until the gradient is zero.

Any iterative minimization procedure is in danger of converging to a local minimum which is not the overall minimum, that is, a solution from which no *small* change is an improvement and yet which is not the best solution. In our situation, experience shows that this is not a serious difficulty because a solution which appears satisfactory is unlikely to be merely a local minimum. The following simple technique may be used to investigate the situation.

After reaching a solution which we fear is only a local minimum, we apply a violent motion to create a new arbitrary configuration, and we start all over again. We do this repeatedly, and obtain several solutions. We may take the best of these to be the true best solution. A handy way to apply a violent motion is simply to use a very large value of α. It is also reasonable to use the present g (after normalization) as the new x. (In effect this makes α infinite.) If we intend to compute several solutions as described, it is appropriate to relax the local minimum criterion to a fairly large value,

and later to continue iterative convergence with a more stringent criterion starting from the best solution.

7. *Programming Technique*

Since the procedure described in this paper is entirely impractical without the aid of an automatic computer, it seems desirable to describe the procedure in sufficient detail that an experienced programmer can easily program it.

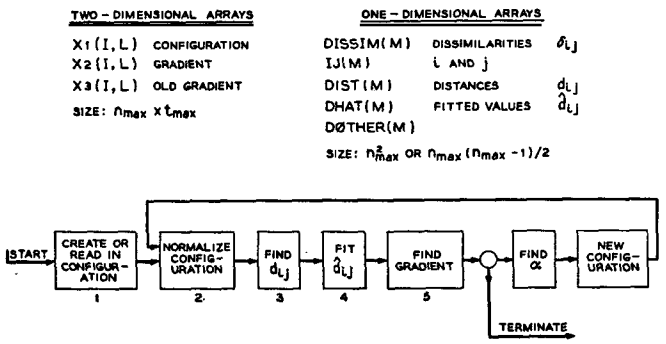

FIGURE 2

A block diagram of the procedure appears in Fig. 2. We start by creating or reading in a configuration. After normalizing the configuration, we calculate the distances d_{ij} . Then we fit the numbers $\hat{d}_{ij}$. (The rank order of the dissimilarities is used only at this stage of the iteration.) From d_{ij} and $\hat{d}_{ij}$, we calculate the stress and the gradient of the present configuration. Then we decide whether we have found a (local) minimum of the stress yet, or whether the normal iterative process should be continued. (It is also desirable to have other termination rules, notably a limit on the number of iterations.) If a local minimum has been reached, then the configuration, the stress, and other useful information should be printed. The configuration should be punched out or saved in some way. Also, printing a history of the more important variables is desirable. If the calculation is to continue, then the step size is determined, and the new configuration is calculated. This starts a new step of the iteration.

Let $n_{\max}$ be the greatest number of objects and $t_{\max}$ the greatest number of dimensions which the program is meant to handle. The major blocks of storage needed are two-dimensional arrays X1, X2, X3 and one-dimensional arrays DISSIM, IJ, DIST, DHAT (*d*-hat), and DØTHER (*d*-other) as shown in Fig. 2. At the start of an iteration, X1 holds the configuration and X3 holds the old gradient which was used to find it. (Thus X1 (I, L) holds the value x_{il} , and X3 (I, L) holds the previous value of g_{il}.) The gradient

at the present configuration is put into X2. Next cos θ is calculated, where θ is the angle between the two gradients. The new configuration is calculated and placed in X1. Then everything is ready for the next step of the iteration.

DISSIM contains the original dissimilarities (or similarities) δ_{ij} , as shown in Fig. 3. Each cell of IJ contains (packed together in one cell) the

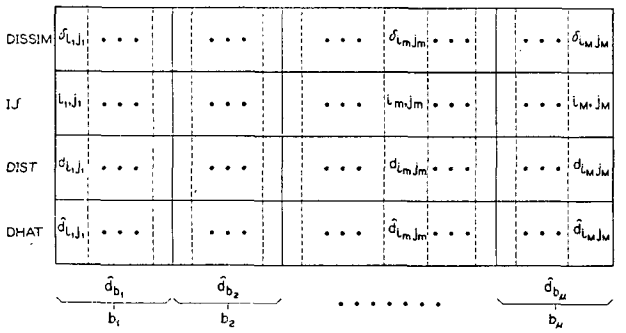

FIGURE 3

values of i and j for the dissimilarity in the corresponding cell of DISSIM. When the dissimilarities are first read, they are placed in DISSIM without any gaps (that is, the cells of DISSIM are filled one by one in order, with no intermediate cells remaining empty). If the dissimilarities are inherently symmetric, or have previously been made symmetric by a separate calculation, then only the entries from one-half the matrix are put into DISSIM. If a dissimilarity is missing (presumably this fact is signalled by a very special artificial value of δ_{ij}), then no entry is made in DISSIM nor is any space reserved.

At the same time that each entry is placed in DISSIM, the corresponding values of i and j are packed together in the corresponding cell of IJ. Thus, although the dissimilarities are put into DISSIM in a manner which ignores their subscripts, this essential information is still present in IJ. Let the number of entries actually placed in DISSIM be M.

After DISSIM and IJ have been filled, then the M entries in DISSIM are sorted in order of increasing algebraic value. (An efficient sorting procedure should be used, such as the radix-exchange method of Hildebrandt and Isbitz [5].) However, if the measurements in DISSIM are similarities instead of dissimilarities, decreasing order is used. (This is the *only* way in which similarities and dissimilarities are treated differently.) During the sorting procedure the M entries in IJ are simultaneously rearranged so as to preserve the correspondence between cells in corresponding positions of DISSIM and IJ. After the sorting is complete, the values in DISSIM

are no longer strictly needed, as their rank order is now available in IJ. However, it is convenient for several reasons to retain the original data.

In cases of ties (equal dissimilarities), the order in which they occur does not matter. However, the IJ cell corresponding to the first dissimilarity of a tie-block (that is, a block of equal dissimilarities) should contain the number of dissimilarities in that tie-block. (This number must be packed in together with i and j.) Also, these cells must be distinguished from other cells, so that the presence of the tie-block can be noted later.

The first stage of each iteration is to find the distances d_{ij} . For Minkowski r-metric distance, use the formula

$$d_{ij} = \left[\sum_{s=1}^{t} |x_{is} - x_{js}|^r \right]^{1/r}.$$

In the case of ordinary Euclidean distances, $r = 2$ and

$$d_{ij} = \sqrt{\sum_{s=1}^{t} (x_{is} - x_{js})^2}.$$

In the case of city-block metric, $r = 1$ and

$$d_{ij} = \sum_{s=1}^{t} |x_{is} - x_{js}|.$$

One very important point concerns the order in which the distances are computed. They should *not* be computed using a double loop on i and j. Instead they should be computed using a single loop in which m runs from 1 to M, corresponding to the entries in IJ. Thus at the mth pass through the loop, the mth entry in IJ is consulted, its values of i and j are used, and the resulting value of d_{ij} is placed in the mth position of DIST.

The next stage of each iteration is to fit the numbers $\hat{d}_{ij}$. We describe how to do this in another section. After fitting, we calculate the stress. It is convenient to set

$$S^* = \sum (d_{ij} - \hat{d}_{ij})^2,$$
$$T^* = \sum d_{ij}^2 ,$$
$$S = \sqrt{S^*/T^*}.$$

It is best to calculate S^* and T^* by a single loop on m from 1 to M. On the mth pass through the loop we use i and j from the mth entry of IJ. We add to the partially accumulated values of S^* and T^* the quantities $(d_{ij} - \hat{d}_{ij})^2$ and d_{ij}^2 . At the end of the loop, S is calculated from S^* and T^*.

To calculate the (negative) gradient we use the following formulas. For Minkowski r-metric, component g_{kl} , which is to be placed in X2 (K, L), is given by

$$g_{kl} = S \sum_{i,j} (\delta^{ki} - \delta^{kj}) \left[\frac{d_{ij} - \hat{d}_{ij}}{S^*} - \frac{d_{ij}}{T^*} \right] \frac{|x_{il} - x_{jl}|^{r-1}}{d_{ij}^{r-1}} \text{ signum } (x_{il} - x_{jl}).$$

Here δ^{ki} and δ^{kj} denote the Kronecker symbols ($\delta^{ki} = 1$ if $k = i$, $\delta^{ki} = 0$ if $k \neq i$) and must not be confused with dissimilarities δ_{ij}. Signum is $+1$ for a positive number, -1 for a negative number, and 0 for 0. In case of Euclidean distance $r = 2$ and this becomes

$$g_{kl} = S \sum_{i,j} (\delta^{ki} - \delta^{kj}) \left[\frac{d_{ij} - \hat{d}_{ij}}{S^*} - \frac{d_{ij}}{T^*} \right] \frac{(x_{il} - x_{jl})}{d_{ij}}.$$

In the case of city-block distance, $r = 1$ and it becomes

$$g_{kl} = S \sum_{i,j} (\delta^{ki} - \delta^{kj}) \left[\frac{d_{ij} - \hat{d}_{ij}}{S^*} - \frac{d_{ij}}{T^*} \right] \text{signum } (x_{il} - x_{jl}).$$

To calculate the gradient use a single iterative loop on m from 1 to M. On the mth pass through this loop, use i and j from the mth cell of IJ. If $i = j$, then $\delta^{ki} - \delta^{kj} = 0$ for all k, so the corresponding term in the formula vanishes, and we may skip to the next value of m. If $i \neq j$, then for $l = 1$ to t, add the following term into g_{il} (that is, X2 (I, L)) and subtract it from g_{jl} (that is, X2 (J, L)):

$$\left[\frac{S}{S^*} (d_{ij} - \hat{d}_{ij}) - \frac{S}{T^*} d_{ij} \right] \frac{1}{d_{ij}^{r-1}} |x_{il} - x_{jl}|^{r-1} \text{ signum } (x_{il} - x_{jl}).$$

At the end of the loop, the gradient g has been accumulated in X2.

Once the gradient has been calculated, it is time to decide whether or not a local minimum has been reached. If it has, suitable output is created, and either the calculation terminates, or else it continues after applying a violent motion to create a new arbitrary configuration. If a local minimum has not been reached, the new step size is calculated, the new configuration is calculated and normalized, and the iteration is ready to start over again.

8. *Algorithm for Fitting*

We describe our algorithm for calculating the numbers $\hat{d}_{ij}$. We first describe it supposing that there are no ties (equal dissimilarities). Afterwards we describe the simple modification needed in case ties are present.

Algorithms for essentially the same purpose, though more general because weights are permitted, may be found in Miles ([8], pp. 319–320), Barton and Mallows ([1], pp. 426–427), and Bartholomew ([2], pp. 37–38) and ([3], pp. 242–244). Algorithms and useful facts for the very much more general situation in which the dissimilarities are only partially ordered, not linearly ordered, and for which the function being minimized is much more general then a sum of squares may be found in van Eeden ([11], pp. 134–136) and ([12], pp. 508–512). Our algorithm is essentially the same as algorithm $\mathfrak{A}_1$ of Miles ([8], p. 539). However, we feel for several reasons that it is worthwhile to describe our algorithm. First, Miles' algorithm involves many arbitrary choices, and how these are made affects the efficiency of the com-

putation. Our algorithm is fully explicit; in effect, we make these arbitrary choices in an intelligent manner. Second, none of these algorithms are described in sufficient detail or in simple enough notation to make easy their use on an automatic computer. Third, the efficiency of these algorithms varies very widely. We believe ours to be as efficient as any of those mentioned.

Imagine the dissimilarities $\delta_{i_m j_m}$ arranged from smallest to largest in DISSIM, as in Fig. 3. The subscript pairs (i_m , j_m) are arranged in the same order in IJ. The correct values $\hat{d}_{ij}$ can be described in this way. There is a partition of the dissimilarities into consecutive blocks $b_1 , \cdots , b_\mu$ such that within each block b the value of $\hat{d}_{ij}$ is constant, and this common value $\hat{d}_b$ is the average of the d_{ij} values in the block. As this is true, it is only necessary to find the correct partition in order to calculate the numbers $\hat{d}_{ij}$.

Our algorithm starts with the finest possible partitions into blocks, and joins the blocks together step by step until the correct partition is found. The finest possible partition consists naturally of M blocks, each containing only a single dissimilarity.

Suppose we have any partition into consecutive blocks. We shall use $\hat{d}_b$ to denote the average of the d_{ij} in block b. If b_- , b, b_+ are three adjacent blocks in ascending order, then we call b *up-satisfied* if $\hat{d}_b < \hat{d}_{b_+}$ and *down-satisfied* if $\hat{d}_{b_-} < \hat{d}_b$. We also call b up-satisfied if it is the highest block, and down-satisfied if it is the lowest block.

At each stage of the algorithm we have a partition into blocks. Furthermore, one of these blocks is *active*. The active block may be *up-active* or *down-active*. At the beginning, the lowest block, consisting of $d_{i_1 j_1}$, is up-active. The algorithm proceeds as follows. If the active block is up-active, check to see whether it is up-satisfied. If it is, the partition remains unchanged but the active block becomes down-active; if not, the active block is joined with the next higher block, thus changing the partition, and the new larger block becomes down-active. On the other hand, if the active block is down-active, do the same thing but upside-down. In other words, check to see whether the down-active block is down-satisfied. If it is, the partition remains unchanged but the active block becomes up-active; if not, the active block is joined with the next lower block into a new block which becomes up-active. Eventually this alternation between up-active and down-active results in an active block which is simultaneously up-satisfied and down-satisfied. When this happens, no further joinings can occur by this procedure, and we transfer activity up to the next higher block, which becomes up-active. The alternation is again performed until a block results which is simultaneously up-satisfied and down-satisfied. Activity is then again transferred to the next higher block, and so forth until the highest block is up-satisfied and down-satisfied. Then the algorithm is finished and the correct partition has been obtained.

After the final partition has been found, then for every block b the

value $\hat{d}_b$ is placed in every DHAT cell of b. This completes the fitting computation.

In case there are ties among the dissimilarities, the algorithm for fitting only needs to be modified by preprocessing. If we adopt the primary approach to ties described in [7], that is, if the only constraints on the $\hat{d}_{ij}$ are those in Section 1, then this preprocessing simply consists of arranging the dissimilarities within each tie-block in such a way that the distances d_{ij} within that block form an increasing sequence. After this preprocessing, the algorithm is carried out as before.

In case we adopt the secondary approach to ties, that is, if we further constrain the $\hat{d}_{ij}$ to be equal when the corresponding δ_{ij} are equal, the preprocessing is still simpler. Instead of starting the algorithm with the finest possible partition, we start it with the partition into tie-blocks. More specifically, the block containing δ_{ij} consists of all dissimilarities which equal δ_{ij} . (In case no other dissimilarities happen to be tied with δ_{ij} , the block contains only one dissimilarity.)

In programming the above algorithm, it is convenient to keep track of the blocks of the partition in the following way. If a block b starts with the mth dissimilarity and contains ν dissimilarities, with $\nu \geqq 2$, then the first DØTHER cell should contain ν and the last DØTHER cell should contain m; also, the first DHAT cell should contain $\hat{d}_b$ and the second DHAT cell should contain $\sum d_{ij}$, where the sum is over all d_{ij} in the block. (Of course,

$$\hat{d}_b = \frac{1}{\nu} \sum d_{ij} ,$$

so we are storing redundant information.) If a block contains only one dissimilarity, then the DØTHER cell should be recognizably blank, and the DHAT cell should contain $\hat{d}_b = d_{ij} = \sum d_{ij}$.

This structure makes it easy to check whether b is up-satisfied or down-satisfied and makes it easy to join two adjacent blocks together. When joining takes place, the joined $\sum d_{ij}$ should be formed by adding the two separate sums, and the new $\hat{d}_b$ formed by dividing by ν. This minimizes round-off error.

If the primary approach to ties is adopted, the sorting of the d_{ij} in each tie-block (during preprocessing) must of course be accompanied by a simultaneous identical rearrangement of the corresponding cells in IJ. If large tie-blocks are anticipated, then the sorting should be done by an efficient procedure such as the radix-exchange technique of Hildebrandt and Isbitz [5].

9. *Summary*

We have described the numerical methods necessary to use our approach to multidimensional scaling. We have included sufficient detail so

that an experienced programmer should not have difficulty in creating a program to perform these computations.

REFERENCES

[1] Barton, D. E. and Mallows, C. L. The randomization bases of the amalgamation of weighted means. *J. roy. statist. Soc.*, Series B, 1961, **23**, 423–433.

[2] Bartholomew, D. J. A test of homogeneity for ordered alternatives. *Biometrika*, 1959, **46**, 36–48.

[3] Bartholomew, D. J. A test of homogeneity of means under restricted alternatives (with discussion). *J. roy. statist. Soc.*, Series B, 1961, **23**, 239–281.

[4] Hardy, G. H., Littlewood, J. E., and Polya, G. *Inequalities*. (2nd ed.) Cambridge, Eng.: Cambridge Univ. Press, 1952.

[5] Hildebrandt, P. and Isbitz H. Radix-exchange—An internal sorting method for digital computers. *J. Assoc. computing Machinery*, 1959, **6**, 156–163.

[6] Kolmogorov, A. N. and Fomin, S. V. *Elements of the theory of functions and functional analysis*. Vol. 1. *Metric and normed spaces*. Translated from the first (1954) Russian Edition by Leo F. Boron, Rochester, N. Y., Graylock Press, 1957.

[7] Kruskal, J. Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *Psychometrika*, 1964, **29**, 1–28.

[8] Miles, R. E. The complete amalgamation into blocks, by weighted means, of a finite set of real numbers. *Biometrika*, 1959, **46**, 317–327.

[9] Shepard, R. N. The analysis of proximities: Multidimensional scaling with an unknown distance function. *Psychometrika* (I and II), 1962, **27**, 125–139, 219–246.

[10] Spang, H. A. III. A review of minimization techniques for nonlinear functions. *SIAM Rev.*, 1962, **4**, 343–365.

[11] van Eeden, C. Maximum likelihood estimation of partially or completely ordered parameters, I. *Proc. Akademie van Wetenschappen*, Series A, 1957, **60**, 128–136.

[12] van Eeden, C. Note on two methods for estimating ordered parameters of probability distributions. *Proc. Akademie van Wetenschappen*, Series A, 1957, **60**, 506–512.

Manuscript received 4/11/63
Revised manuscript received 8/22/63