Journal of Mathematical Psychology **46**, 731–745 (2002)
doi:10.1006/jmps.2002.1423

Identifying a Reordering of Rows and Columns for Multiple Proximity Matrices Using Multiobjective Programming

Michael J. Brusco

Florida State University, Tallahassee, Florida

This paper is concerned with a problem where K $(n \times n)$ proximity matrices are available for a set of n objects. The goal is to identify a single permutation of the n objects that provides an adequate structural fit, as measured by an appropriate index, for each of the K matrices. A multiobjective programming approach for this problem, which seeks to optimize a weighted function of the K indices, is proposed, and illustrative examples are provided using a set of proximity matrices from the psychological literature. These examples show that, by solving the multiobjective programming model under different weighting schemes, the quantitative analyst can uncover information about the relationships among the matrices and often identify one or more permutations that provide good to excellent index values for all matrices.

Key Words: combinatorial data analysis; matrix permutation; multiobjective programming; three-way two-mode data.

1. INTRODUCTION

Many approaches for fitting a structure to a single proximity matrix (with rows and columns corresponding to the same set of objects) require the identification of a reordering (or permutation) of the rows and columns of the matrix that optimizes some type of index (Baker & Hubert, 1977; Brusco & Stahl, 2001; Hubert, 1976, 1978; Hubert & Baker, 1977; Hubert & Golledge, 1981b; Hubert & Schultz, 1976; Hubert, Arabie, & Meulman, 2001). Such problems are inherently combinatorial in nature and can frequently be described within the context of some broader framework, such as the quadratic assignment paradigm (Hubert & Schultz, 1976; Hubert *et al.*, 2001). When multiple proximity matrices are available for the same set of objects, the combinatorial nature of the matrix reordering problem is still present, but the situation is complicated by the fact that multiple indices (one for each matrix) are relevant.

Address correspondence and reprint requests to Michael J. Brusco, Marketing Department, College of Business, Florida State University, Tallahassee, FL 32306-1110. E-mail: mbrusco@cob.fsu.edu.

0022-2496/02 $35.00

As noted by Hubert (1979, 1987, Chap. 6), there are a number of important problems related to the analysis of multiple proximity matrices. These problems include: (a) the development of measures of concordance among multiple proximity matrices, (b) relating several proximity matrices to a single proximity matrix of some hypothesized structure, (c) establishing relationships within and between subsets of a set of proximity matrices, and (d) fitting some type of structural model to a set of proximity matrices. Analytical methods for these problems can take many forms depending upon the characteristics of the proximity data, and there are a number of available statistical procedures (Hubert, 1979, 1985; Hubert & Golledge, 1981a; Kendall, 1970; Manly, 1986; Mantel, 1967; Oden & Sokal, 1992; Smouse, Long, & Sokal, 1986). The identification of a matrix reordering within the context of multiple proximity matrices, which is the focus of this current paper, is perhaps most closely related to the fourth problem area suggested by Hubert (1979).

When multiple proximity matrices are of interest, a particular permutation of objects might provide optimal (or near-optimal) index values for one or more of the matrices, but rather poor values for others. Such a permutation is likely to be unacceptable to the quantitative analyst, particularly if one or more of the matrices for which the fit is poor is deemed especially important. The analyst might therefore seek a permutation that provides reasonably good index values for each of the relevant proximity matrices, perhaps with particular concern for the index values of a select subset of the matrices.

The problem of establishing a single permutation of the n objects that provides a reasonable fit to multiple proximity matrices is posed as a multiobjective combinatorial optimization problem. In this regard, the problem can be viewed as an extension of a recent multiobjective programming approach to combinatorial data analysis (Brusco & Stahl, 2001). The key difference is that, whereas Brusco and Stahl were concerned with identification of a permutation for a *single proximity matrix* that provided good values for multiple structural indices, the focus herein is on finding a permutation for *multiple proximity matrices* each with their own relevant structural index.

The multiobjective programming approach is applicable for a wide range of alternative indices associated with either symmetric or asymmetric matrices. We focus the presentation and examples on an anti-Robinson index (Robinson, 1951) for symmetric matrices and a well-known dominance index (Baker & Hubert, 1977; DeCani, 1969; Hubert, 1976; Hubert *et al.*, 2001) for asymmetric matrices. In the next section, we present the multiobjective programming model. Section 3 provides two numerical examples that demonstrate the utility of the model. The paper concludes with a brief summary in Section 4.

2. METHODOLOGY

2.1. Relevant Indices for Symmetric and Asymmetric Matrices

We define $S = \{o_1, o_2, \ldots, o_n\}$ as a set of n objects, $A = \{\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_K\}$ as a set of $k = 1, \ldots, K$ nonnegative $n \times n$ proximity matrices corresponding to the objects

in S, $\Psi = \{\psi^1, \psi^2, \ldots, \psi^{n!}\}$ as the set of all possible permutations of the n objects, and $\psi(l)$ as the object in sequence position l of a given permutation ψ. Elements of the matrices in A, q_{ijk}, represent the proximity between objects o_i and o_j in matrix $\mathbf{Q}_k$, and it is assumed that the diagonal element q_{iik} is irrelevant (or ignored) for all $i = 1, \ldots, n$ and $k = 1, \ldots, K$. A well-established index for seriation of asymmetric matrices is the maximization of the sum of the elements above the diagonal of the reordered matrix (Baker & Hubert, 1977; DeCani, 1969; Hubert, 1976, 1978; Hubert & Golledge, 1981b; Hubert *et al.*, 2001). Using the above definitions, this index can be mathematically stated as

$$f_k(\psi) = \sum_{l=1}^{n-1} \sum_{m=l+1}^{n} q_{\psi(l),\psi(m),k} \qquad \text{for} \quad k = 1, \ldots, K. \tag{1}$$

For any particular matrix $\mathbf{Q}_k$, a reordering of the rows and columns that maximizes (1) can be obtained using branch-and-bound (Flueck & Korsh, 1974), dynamic programming (Hubert *et al.*, 2001; Hubert & Golledge, 1981b), or integer linear programming (DeCani, 1969). The computational viability of the dynamic programming approach, which is used throughout the remainder of this paper, is primarily based on available computer memory. For microcomputers with 128 to 512 MB of RAM, dynamic programming can successfully be applied to matrices with values of n ranging from about 22 to 27. The dynamic programming code used herein, which is written in FORTRAN, can be obtained from the Classification Society of North America (website: http://www.pitt.edu/~csna).

For symmetric dissimilarity matrices, in which larger elements indicate greater dissimilarity between objects, we consider an anti-Robinson index (Hubert *et al.*, 2001; Robinson, 1951). A matrix exhibiting perfect anti-Robinson structure will have nondecreasing entries in each row of the matrix when moving away from the main diagonal (Hubert, 1987, Chapter 4). This anti-Robinson index is mathematically defined as

$$g_k(\psi) = \left(\sum_{m=1}^{n-2} \sum_{l=m+1}^{n-1} \sum_{h=l+1}^{n} u_{mlhk} + \sum_{m=3}^{n} \sum_{l=1}^{m-2} \sum_{h=l+1}^{m-1} v_{mlhk} \right), \tag{2}$$

where $u_{mlhk} = 1$ if $q_{\psi(m),\psi(l),k} \leqslant q_{\psi(m),\psi(h)k}$, 0 otherwise; and $v_{mlhk} = 1$ if $q_{\psi(m),\psi(h),k} \leqslant q_{\psi(m),\psi(l),k}$, 0 otherwise.

The first summation triple in (2) corresponds to the index contributions above the main diagonal, whereas the second corresponds to contributions below the main diagonal. Optimal solutions for (2) can be obtained using the methods identified for (1). We used dynamic programming for the results in this paper.

2.2. The Multiobjective Programming Model

We denote an optimal permutation of objects obtained by dynamic programming (based on the appropriate asymmetric or symmetric index) for matrix $\mathbf{Q}_k$ as ψ_k^*, for $k = 1, \ldots, K$. The optimal index value for matrix $\mathbf{Q}_k$ is defined as f_k^* for asymmetric matrices and g_k^* for symmetric matrices. If $\psi_1^* = \psi_2^* = \ldots \psi_K^*$, then there is no problem because a single permutation is optimal for each of the matrices in A.

However, this occurrence is rather unlikely from a practical standpoint (particularly for larger values of n and K), and thus there is a need for a method that can find a single reordering that provides good values of $f_k(\psi)$ (or $g_k(\psi)$) for $k = 1, \ldots, K$. One such approach is to define $\mathbf{Q}_{sum} = \mathbf{Q}_1 + \mathbf{Q}_2 + \cdots + \mathbf{Q}_K$ and apply dynamic programming directly to $\mathbf{Q}_{sum}$. The primary limitation of this approach is that a subset of the matrices in A are apt to dominate the solution for $\mathbf{Q}_{sum}$. (This observation is true even if a normalization procedure based on ranks or z-scores is applied to each of the matrices in A prior to the analysis.) Whereas the optimal matrix reordering for $\mathbf{Q}_{sum}$ might provide very good index values for the matrices in this subset, the reordering is also likely to yield inferior (in some cases, very poor) index values for other matrices in A.

Rather than aggregating the proximity matrices in A into a single matrix ($\mathbf{Q}_{sum}$), we propose a multiobjective approach that incorporates the index values of each of the proximity matrices in a single weighted objective function. Defining model parameters $w_k (k = 1, \ldots, K)$ as the objective function weight for matrix $\mathbf{Q}_k$, the multiobjective programming problem for asymmetric matrices can be stated as

$$\text{Maximize: } F(\psi) = \sum_{k=1}^{K} w_k \left(\frac{f_k(\psi)}{f_k^*} \right) \tag{3}$$

subject to

$$\psi \in \Psi \tag{4}$$

$$\sum_{k=1}^{K} w_k = 1 \tag{5}$$

$$w_k > 0 \tag{6}$$

The objective function (3) of the multiobjective problem is a weighted function of ratios related to each of the K asymmetric proximity matrices in A. For symmetric proximity matrices, $f_k(\psi)$ and f_k^* would be replaced by $g_k(\psi)$ and g_k^*, respectively. For each matrix, the ratios represent the percentage of the maximum attainable index value that is achieved by permutation ψ. The weighted objective function is maximized subject to a restriction ensuring a feasible permutation (4) and constraints (5) and (6), which guarantee a convex combination of weights summing to one. The fact that the ratio terms in (3) are bounded above by one, in conjunction with the fact that the weights must sum to one, ensures that $F(\psi)$ is also bounded above by one. It should also be noted that additional constraints establishing threshold index values for one or more of the proximity matrices could be added to the multiobjective programming model. However, such threshold values are difficult to establish à priori, and we have found such constraints of limited value in this context.

Consistent with Brusco and Stahl (2001), dynamic programming is the recommended solution procedure for providing an optimal solution to the multiobjective programming problem when computationally feasible. An excellent coverage of the dynamic programming paradigm for seriation and other combinatorial data analysis problems is provided by Hubert *et al.* (2001). A brief description of the

multiobjective dynamic programming model presented herein, which is consistent with the general paradigm described by Hubert *et al.*, begins with the definition of Ω_y as the set of all subsets ($R_y \in \Omega_y$) consisting of exactly y of the n objects in S. Thus, the power set (or set of all subsets) of S is given by $P(S) = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_n$. If the objects in R_y are assumed to be placed in the first y positions of a sequence, then it is straightforward to calculate their contribution, $F(R_y)$, to the weighted index in (3). The crux of the approach is that for any $R_{y-1} \subset R_y$ (i.e., $R_{y-1} = R_y - \{o_x\}$, where $o_x \in R_y$), a recursive equation relating the incremental contribution to the weighted index in (3) can be established as

$$F(R_y) = \underset{o_x \in S \setminus R_{y-1}}{\text{Max}} \left\{ F(R_{y-1}) + \sum_{k=1}^{K} w_k d_k(R_{y-1}, o_x) \right\}, \tag{7}$$

where $d_k(R_{y-1}, o_x)$ is the effect on the appropriate index value (e.g., (1) for the asymmetric case or (2) for the symmetric case) for matrix $\mathbf{Q}_k$ when appending object o_x to R_{y-1} to form R_y, such that o_x occupies the yth position in the sequence. The summation term in (7) therefore represents the weighted effect across all matrices (for the appropriate index) that would result from adding object o_x to R_{y-1}. The stages of the dynamic programming algorithm proceed by implementing the recursive equation (7) for $y = 1$ to n (assuming that $F(R_0 = \{\varnothing\}) = 0$), and the optimal sequence is obtained by backtracking through the recursion.

The resulting optimal solution for a given set of weights is an *efficient* (non-dominated) solution for the multiobjective problem. By solving the multiobjective problem for a number of different weighting schemes, the quantitative analyst can investigate a portion of the *efficient set* of solutions in a particular area of interest. For problems that are too large for dynamic programming, the multi-operation local search procedure developed by Hubert and Arabie (1994) is highly recommended. Although optimal solutions to the multiobjective programming problem are not guaranteed when this heuristic is used, computational evidence suggests that it performs very well (see Brusco & Stahl, 2001).

The multiobjective programming model provides a powerful tool for exploring tradeoffs among the index values for the matrices in A. This goal is accomplished by solving the model for different sets of weights. For example, suppose that the solution for a particular set of weights provides good index values for some subset ($A1$) of the matrices in A, but not for another subset, ($A2 = A \setminus A1$). The analyst can prepare another run of the model after increasing the weights for the matrices in $A2$ and correspondingly decreasing weights for matrices in $A1$. By iteratively manipulating the weights, the analyst is often able to investigate rapidly a portion of the efficient set that provides desirable index values for all proximity matrices.

2.3. Comparing Index Values to Their Distributions

A potential criticism of the ratio measures in the objective function of the multiobjective programming model is that, in some cases, they might not correspond well to the actual distributions of index values across all permutations. In other words, for a given matrix $\mathbf{Q}_k$, the index value for some permutation, ψ, might be

80% of the optimal index value for that matrix, yet that same index value might only fall in the 40th percentile of the distribution of index values across all $n!$ permutations. It is also possible that the index value might be 80% of the optimal index value, yet fall in the 98th percentile of the distribution. For this reason, we also recommend collecting information regarding the distributions of index values for each matrix and comparing the multiobjective solutions to those distributions.

TABLE 1

Confusion Matrices from Morgan *et al.* (1973)

	Response									
	1	2	3	4	5	6	7	8	9	
Stimulus										
1	188	9	12	17	150	8	21	16	119	Q1
2	23	117	161	33	16	83	51	32	21	Recognition
3	33	60	143	37	20	37	71	60	62	(Male Voice)
4	55	16	20	300	58	10	29	18	29	
5	21	2	5	7	445	0	5	6	51	
6	20	15	29	21	11	346	52	27	14	
7	31	23	31	30	41	38	274	30	29	
8	21	30	44	36	32	56	59	219	34	
9	56	6	5	15	113	2	7	11	324	
1	770	41	56	63	254	30	41	64	242	Q2
2	96	385	438	89	66	110	189	87	96	Recognition
3	121	224	480	125	81	98	168	139	119	(Female Voice)
4	98	48	54	1023	119	38	64	58	60	
5	128	35	35	67	937	16	26	28	292	
6	77	94	100	63	64	805	191	111	44	
7	122	79	85	100	83	110	807	109	65	
8	114	91	141	177	59	242	154	513	55	
9	218	60	49	61	324	22	51	48	720	
1	2340	23	34	35	48	11	33	41	63	Q3
2	4	2533	28	12	6	12	21	6	6	Memory
3	16	54	2465	13	8	19	24	22	7	(Voice #1)
4	13	42	32	2421	42	11	34	21	12	
5	25	24	26	56	2315	17	58	34	73	
6	4	11	19	7	10	2506	45	22	4	
7	4	14	5	3	8	7	2570	9	8	
8	13	13	12	14	7	19	21	2508	21	
9	49	28	21	10	72	11	25	25	2387	
1	2505	28	28	34	40	10	42	31	32	Q4
2	11	2609	31	16	11	24	19	15	14	Memory
3	7	35	2612	16	13	24	19	14	10	(Voice #2)
4	25	31	23	2530	54	23	24	13	27	
5	23	23	35	43	2426	22	51	37	90	
6	0	9	14	11	4	2654	37	17	4	
7	6	6	11	7	5	14	2692	4	5	
8	16	17	13	14	18	23	24	2600	25	
9	42	21	13	9	44	9	28	18	2566	

3. NUMERICAL EXAMPLES

Both of our examples were conducted using a set of $K = 4$ confusion matrices that were originally reported by Morgan, Chambers, and Morton (1973, pp. 376–377). These 9×9 matrices, which were also analyzed by Hubert and Baker (1977, pp. 242–245), are associated with acoustical memory and recognition tasks for digits from one to nine. The first two matrices correspond to recognition tasks for male and female voices, respectively. The third and fourth matrices correspond to memory tasks for two different female voices. Whereas Morgan *et al.* (1973, pp. 376–377) originally reported the matrices with the rows representing the responses and the columns representing the stimuli, our results are based on rows denoting the stimuli and the columns denoting the responses. The matrices are shown, in this form, in Table 1.

3.1. Example 1: The Symmetric Case

For the first numerical example, the asymmetric confusion matrices ($\mathbf{C}_k$, $k = 1, \ldots, K$) in Table 1 were transformed into symmetric dissimilarity matrices ($\mathbf{Q}_k$, $k = 1, \ldots, K$) based on the following relationship:

$$q_{ijk} = q_{jik} = \phi_k - (c_{ijk} + c_{jik}), \qquad \text{for} \quad i < j \quad \text{and} \quad k = 1, \ldots, K \tag{8}$$

where

$$\phi_k = \underset{i<j}{\text{Max}} (c_{ijk} + c_{jik}) \qquad \text{for} \quad k = 1, \ldots, K. \tag{9}$$

Similar approaches for converting asymmetric confusion matrices into symmetric dissimilarity matrices have been used in previous studies (Hubert, Arabie, & Meulman, 1997).

Total enumeration of all 9! permutations was conducted to identify an optimal reordering and corresponding anti-Robinson index values (2), as well as a distribution of index values, for each of the four symmetric dissimilarity matrices. The resulting optimal index values (2) for $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$, and $\mathbf{Q}_4$ are 148, 142, 126, and 128, respectively. For each of the four optimal permutations and each of the four matrices, Table 2 presents the corresponding raw index values (2), the raw index value expressed as a percentage of the optimal index value, and the percentage of the total number of permutations that have index values that are less than or equal to the raw index value. The optimal permutations for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ provide index values for all four matrices that are at least 80% of the respective optimal index values. Perhaps more importantly, these permutations also yield, for each of the four matrices, index values that are above the 92nd percentile of their respective distributions. The optimal permutations for $\mathbf{Q}_3$ and $\mathbf{Q}_4$ provide somewhat inferior index values for $\mathbf{Q}_1$ and $\mathbf{Q}_2$.

Table 3 presents the results for the multiobjective programming model under several different weighting schemes. Under an equal weighting scheme, index values for each of the four matrices are above the 96th percentile for their respective distributions. Because the index values for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ were both above the 99.9th percentile, the next run was made with slightly lower weights for these matrices

TABLE 2

A Comparison of Index Values Corresponding to Optimal Permutations for the Symmetric Case[a]

Optimized matrix	Resulting objective function value (2)			
	$g_1(\psi)$	$g_2(\psi)$	$g_3(\psi)$	$g_4(\psi)$
$\mathbf{Q}_1$	148 (1.0000) [1.0000]	133 (0.9366) [0.9997]	101 (0.8016) [0.9206]	104 (0.8125) [0.9542]
$\mathbf{Q}_2$	136 (0.9189) [0.9995]	142 (1.0000) [1.0000]	103 (0.8175) [0.9452]	103 (0.8047) [0.9455]
$\mathbf{Q}_3$	101 (0.6824) [0.9346]	92 (0.6479) [0.7968]	126 (1.0000) [1.0000]	123 (0.9609) [0.9999]
$\mathbf{Q}_4$	84 (0.5676) [0.5751]	83 (0.5845) [0.5049]	117 (0.9286) [0.9990]	128 (1.0000) [1.0000]

[a] The table reports the raw anti-Robinson index value (2) for the reordered matrix, the proportion of the optimal value (in parentheses below the raw figure), and the proportion of permutations that result in an index value less than or equal to the corresponding raw value [in brackets below the raw figure].

TABLE 3

A Comparison of Index Values Corresponding to Optimal Permutations under Different Sets of Objective Function Weights for the Symmetric Case[a]

Objective weights				Resulting objective function values				
w_1	w_2	w_3	w_4	$g_1(\psi)$	$g_2(\psi)$	$g_3(\psi)$	$g_4(\psi)$	$F(\psi)$
0.2500	0.2500	0.2500	0.2500	146 (0.9865) [> 0.9999]	134 (0.9437) [0.9998]	107 (0.8492) [0.9770]	105 (0.8203) [0.9620]	— (0.8999)
0.2000	0.2000	0.3000	0.3000	141 (0.9527) [0.9999]	126 (0.8873) [0.9982]	110 (0.8730) [0.9891]	112 (0.8750) [0.9920]	— (0.8924)
0.1000	0.3000	0.3000	0.3000	134 (0.9054) [0.9993]	132 (0.9296) [0.9996]	112 (0.8889) [0.9938]	108 (0.8438) [0.9796]	— (0.8892)
0.1000	0.2000	0.2000	0.5000	134 (0.9054) [0.9993]	121 (0.8521) [0.9952]	111 (0.8810) [0.9918]	115 (0.8984) [0.9966]	— (0.8864)

[a] The table reports the raw anti-Robinson index values (2) for the reordered matrix, the proportion of the optimal value (in parentheses below the raw figure), and the proportion of permutations that result in an index value less than or equal to the corresponding raw value [in brackets below the raw figure].

TABLE 4

The Optimal Permutations for $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$, $\mathbf{Q}_4$, and Four Multiobjective Problems with Different Weights for the Symmetric Case

ψ_1^*	{5, 9, 1, 4, 7, 3, 8, 2, 6}
ψ_2^*	{5, 9, 1, 4, 3, 2, 7, 8, 6}
ψ_3^*	{6, 3, 2, 7, 4, 5, 9, 1, 8}
ψ_4^*	{3, 2, 4, 1, 9, 5, 7, 8, 6}
ψ^* (0.25, 0.25, 0.25, 0.25)	{5, 9, 1, 4, 7, 8, 3, 2, 6}
ψ^* (0.2, 0.2, 0.3, 0.3)	{9, 1, 5, 4, 7, 8, 3, 2, 6}
ψ^* (0.1, 0.3, 0.3, 0.3)	{9, 1, 5, 4, 8, 3, 2, 7, 6}
ψ^* (0.1, 0.2, 0.2, 0.5)	{9, 1, 5, 4, 7, 8, 2, 3, 6}

($w_1 = w_2 = 0.2$) and, accordingly, slightly greater weights for $\mathbf{Q}_3$ and $\mathbf{Q}_4$ ($w_3 = w_4 = 0.3$). These weights provided a permutation that yields outstanding index values that are above the 98th percentile of the index distributions for each of the four matrices. Using weights of $w_1 = 0.1$, $w_2 = w_3 = 0.2$, and $w_4 = 0.5$, the resulting optimal permutation provides index values above the 99th percentile of the index distributions for each of the four matrices.

The optimal permutations for each of the four matrices, as well as the four multiobjective solutions, are presented in Table 4. The optimal permutations for the multiobjective solutions tend to resemble more closely the optimal permutations for $\mathbf{Q}_1$ and $\mathbf{Q}_2$, which is not surprising because these latter permutations provide good index values for all matrices. For example, the optimal multiobjective permutation under equal weights is nearly the same as the optimal permutation for $\mathbf{Q}_1$ (the difference is that objects o_3 and o_8 are reversed in the two permutations).

As a whole, the results of the analysis generally support the findings of Hubert and Baker (1977, p. 245) that "…it is reasonable to conclude that similar proximity information is being provided in all four of the original frequency matrices." Whereas those authors' conclusion was based on complete-link hierarchical clustering, the conclusion here is primarily based on the ability to identify permutations that provide anti-Robinson index values that are above the 99th percentile of the index distributions for each of the four matrices.

3.2. Example 2: The Asymmetric Case

For the second example, the asymmetric confusion matrices ($\mathbf{C}_k, k = 1, \ldots, K$) in Table 1 directly served as the relevant proximity matrices ($\mathbf{Q}_k, k = 1, \ldots, K$). Total enumeration of all 9! permutations was conducted to identify an optimal reordering and corresponding index values, as well as a distribution of the dominance index values (1), for each of the four matrices. The resulting optimal index values (1) for $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$, and $\mathbf{Q}_4$ are 1751, 4737, 1090, and 1029, respectively. For each of the four optimal permutations and each of the four matrices, Table 5 presents the corresponding raw index values (1), the raw index value expressed as a percentage of the optimal index value, and the percentage of the total number of permutations

TABLE 5

A Comparison of Index Values Corresponding to Optimal Permutations for the Asymmetric Case[a]

Optimized matrix	Resulting objective function value (1)			
	$f_1(\psi)$	$f_2(\psi)$	$f_3(\psi)$	$f_4(\psi)$
$\mathbf{Q}_1$	1751 (1.0000) [1.0000]	2837 (0.9983) [> 0.9999]	965 (0.5890) [0.0512]	891 (0.6463) [0.1040]
$\mathbf{Q}_2$	1725 (0.9852) [> 0.9999]	4737 (1.0000) [1.0000]	639 (0.5862) [0.0475]	656 (0.6375) [0.0832]
$\mathbf{Q}_3$	1035 (0.5957) [0.1164]	3303 (0.6969) [0.0387]	1090 (1.0000) [1.0000]	981 (0.9631) [0.9976]
$\mathbf{Q}_4$	1198 (0.6840) [0.4340]	3509 (0.7408) [0.1593]	1060 (0.9725) [0.9993]	1029 (1.0000) [1.0000]

[a] The table reports the raw dominance index value (1) for the reordered matrix, the proportion of the optimal value (in parentheses below the raw figure), and the proportion of permutations that result in an index value less than or equal to the corresponding raw value [in brackets below the raw figure].

that have index values that are less than or equal to the raw index value. Of particular interest is the relationship between these latter two pieces of information. For example, the optimal permutation for $\mathbf{Q}_1$ provides an index value for $\mathbf{Q}_3$ that is 58.9% of the optimal index value for $\mathbf{Q}_3$. A far more dismal picture is obtained from the fact that this index value is greater than or equal to only about 5% of the index values corresponding to the remaining permutations. In some cases, a better picture is provided by the comparison to the distribution. The optimal permutation for $\mathbf{Q}_3$ provides an index value for $\mathbf{Q}_4$ that is 96.4% of the optimal index value for $\mathbf{Q}_4$, but an even better picture is portrayed by the fact that this index value is greater than or equal to nearly 99.9% of the index values corresponding to the remaining permutations.

The most striking aspect of Table 5 is that the optimal permutations for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ provide excellent index values for each other, but poor index values for $\mathbf{Q}_3$ and $\mathbf{Q}_4$. Similarly, the optimal permutations for matrices $\mathbf{Q}_3$ and $\mathbf{Q}_4$ each provide extremely good index values for each other, but rather poor values for $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Again, this finding is consistent with the results of Hubert and Baker (1977, p. 245), who observed that the hierarchies for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ and the hierarchies for $\mathbf{Q}_3$ and $\mathbf{Q}_4$ were most similar. The multiobjective programming approach was applied to the matrix reordering problem in an effort to identify a permutation that yields reasonably good index values for all four matrices.

The results of the multiobjective programming approach are presented in Table 6. Under an equal weighting scheme, the index values for all four matrices

TABLE 6

A Comparison of Index Values Corresponding to Optimal Permutations under Different Sets of Objective Function Weights for the Asymmetric Case[a]

Objective weights				Resulting objective function values				
w_1	w_2	w_3	w_4	$f_1(\psi)$	$f_2(\psi)$	$f_3(\psi)$	$f_4(\psi)$	$F(\psi)$
0.2500	0.2500	0.2500	0.2500	1439 (0.8218) [0.9103]	3880 (0.8191) [0.6387]	972 (0.8917) [0.9583]	1000 (0.9718) [0.9996]	— (0.8761)
0.3000	0.3000	0.2000	0.2000	1682 (0.9606) [0.9996]	4541 (0.9586) [0.9983]	769 (0.7055) [0.3752]	787 (0.7648) [0.5392]	— (0.8698)
0.2750	0.2750	0.2250	0.2250	1427 (0.8150) [0.8963]	4001 (0.8446) [0.7871]	957 (0.8780) [0.9396]	992 (0.9640) [0.9990]	— (0.8708)
0.2800	0.2800	0.3000	0.1400	1529 (0.8732) [0.9754]	4158 (0.8778) [0.9152]	913 (0.8376) [0.8574]	894 (0.8688) [0.9056]	— (0.8632)

[a] The table reports the raw dominance index value (1) for the reordered matrix, the proportion of the optimal value (in parentheses below the raw figure), and the proportion of permutations that result in an index value less than or equal to the corresponding raw value [in brackets below the raw figure].

were more than 80% of their respective optimal values. However, whereas the index values for $\mathbf{Q}_1$, $\mathbf{Q}_3$, and $\mathbf{Q}_4$ were each above the 90th percentile for their respective distributions, the index value for $\mathbf{Q}_2$ was only in the 64th percentile of its distribution. In an effort to improve the index values for $\mathbf{Q}_1$ and $\mathbf{Q}_2$, the next run was made using weights of $w_1 = w_2 = 0.3$ and $w_3 = w_4 = 0.2$. Although the optimal permutation associated with these weights yielded a significant improvement for $\mathbf{Q}_1$ and $\mathbf{Q}_2$, the sacrifice with respect to $\mathbf{Q}_3$ and $\mathbf{Q}_4$ was considerable. The last two sets of results in Table 6 provide somewhat better compromises. In particular, the weighting scheme of $w_1 = w_2 = 0.28$, and $w_3 = 0.30$, and $w_4 = 0.14$ provides a permutation with index values that are at least 83.7% of the optimal values for each of the four matrices and, at the same time, the index values are above the 85th percentile for all four of the respective distributions.

Defining $A1 = \{\mathbf{Q}_1, \mathbf{Q}_2\}$ and $A2 = \{\mathbf{Q}_3, \mathbf{Q}_4\}$, it is clear that there is general agreement in the optimal permutations for matrices *within* these two subsets, but a somewhat antagonistic relationship *between* the two subsets. Some additional insight regarding this rather perplexing result may be gleaned from the optimal permutations for each of the four matrices, as well as the four multiobjective solutions, which are presented in Table 7. For example, the *last* four objects in the optimal permutations for both $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are $\{o_1, o_4, o_5, o_9\}$, which are the *first* four objects in the optimal reordering for $\mathbf{Q}_3$ (and four of the first five objects in the optimal reordering for $\mathbf{Q}_4$).

TABLE 7

The Optimal Permutations for $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$, $\mathbf{Q}_4$, and Four Multiobjective Problems with Different Weights for the Asymmetric Case

ψ_1^*	{2, 3, 8, 6, 7, 4, 1, 9, 5}
ψ_2^*	{8, 2, 6, 3, 7, 4, 1, 9, 5}
ψ_3^*	{1, 5, 4, 9, 6, 3, 8, 2, 7}
ψ_4^*	{1, 4, 5, 8, 9, 3, 2, 6, 7}
ψ^* (0.25, 0.25, 0.25, 0.25)	{4, 1, 5, 2, 3, 8, 6, 9, 7}
ψ^* (0.3, 0.3, 0.2, 0.2)	{2, 3, 8, 6, 4, 1, 9, 5, 7}
ψ^* (0.275, 0.275, 0.225, 0.225)	{1, 8, 4, 5, 2, 3, 6, 9, 7}
ψ^* (0.28, 0.28, 0.30, 0.14)	{1, 8, 4, 2, 3, 6, 9, 5, 7}

Closer inspection of the raw confusion frequencies provides some direct explanation for the reversal of the orderings. Consider the 4×4 submatrices corresponding to the first four digits in each of the matrices in Table 1. For the recognition tasks ($\mathbf{Q}_1$ and $\mathbf{Q}_2$), observe that $c_{21k} > c_{12k}$, $c_{31k} > c_{13k}$, $c_{41k} > c_{14k}$, $c_{32k} < c_{23k}$, $c_{42k} < c_{24k}$, and $c_{43k} < c_{34k}$, for $k = 1$ and 2. For the memory tasks ($\mathbf{Q}_3$ and $\mathbf{Q}_4$), the situation is completely reversed because $c_{21k} < c_{12k}$, $c_{31k} < c_{13k}$, $c_{41k} < c_{14k}$, $c_{32k} > c_{23k}$, $c_{42k} > c_{24k}$, and $c_{43k} > c_{34k}$, for $k = 3$ and 4. Other stimulus pairs also exhibit similar reversal of the asymmetry pattern when moving from recognition to memory tasks. Observe that, in the recognition tasks, '5' was much more apt to be given as a mistaken response to the stimulus '6', than '6' was mistakenly given as a response to the stimulus '5', whereas the reverse was true for the memory tasks.

This degree of "reversal" in the optimal permutations is, to say the least, interesting and suggests at least two possibilities. One possibility is that recognition tasks exhibit confusion asymmetries that are roughly opposite of those exhibited by memory tasks. However, in the absence of any theoretical justification for such a result, this scenario seems rather unlikely. A second possibility is that the matrices in one of the two subsets ($A1$ or $A2$) might have been mistakenly reported, and actually represent the *transposes* of the true confusion matrices. Although there is certainly no conclusive evidence to suggest that such a problem exists with the matrices as originally reported by Morgan *et al.* (1973, pp. 376–377), a transposition would have been less detectable in the studies of Morgan *et al.* (1973), Hubert and Baker (1977), and Example 1 herein because, in each of these cases, the original confusion matrices were transformed to symmetric matrices prior to analysis.

Following up on the speculation of a transposition, we re-ran the multiobjective model (with equal weights) after replacing $\mathbf{Q}_3$ and $\mathbf{Q}_4$ (as originally reported by Morgan *et al.*, 1973) with their transposes. Under an equal weighting scheme, the resulting optimal permutation yields index values that are at least 90.7% of the optimal values for each of the four matrices and, at the same time, the index values are above the 97th percentile for all four of the respective distributions. Using the weights $w_1 = w_2 = w_3 = 0.1666$ and $w_4 = 0.5$, the optimal reordering yields index values that are at least 92.3% of the optimal values for each of the four matrices and, at the same time, the index values are above the 99th percentile for all four of

the respective distributions. This result clearly demonstrates that the replacement of $\mathbf{Q}_3$ and $\mathbf{Q}_4$ (or, alternatively, $\mathbf{Q}_1$ and $\mathbf{Q}_2$) with their transposes enables the identification of permutations that simultaneously provide an excellent fit to all four matrices, as was observed for the symmetric case in Example 1.

Regardless of whether or not there was any transposition of matrices in the original data, the analysis of the confusion matrices in their original asymmetric form has certainly revealed some fascinating findings. Neither of the two symmetric transformations used by Morgan *et al.* (1973) were as successful at uncovering such striking differences between the memory and recognition matrices. This not only suggests that the multiobjective programming approach is a useful tool for uncovering important relationships among proximity matrices, but also that considerable care is necessary when converting raw confusion frequencies to symmetric proximity data. This latter observation is evident from the fact that the differences between the memory and recognition matrices were not revealed in the multiobjective analysis of the symmetric data.

3.3. Application of the Multiobjective Programming Approach to Larger Proximity Matrices

We also applied the multiobjective programming method to a set of proximity matrices associated with a much larger set of objects. In particular, the method was applied (using the dominance index (1)) to a set of $K = 3$ stimulus-response matrices reported by Cho, Yang, and Hallett (2000, pp. 750–751), which correspond to subjects' visual judgements of $n = 20$ different textured materials at three different distances. Specifically, the 20×20 confusion matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$, and $\mathbf{Q}_3$ are associated with distances of 8.2, 15.5, and 22.9 m, respectively.

For the sake of parsimony, detailed results for the Cho *et al.* (2000) proximity data are not reported, however, the results are available upon request. It should be noted that, because of the larger object size, total enumeration of all permutations was not possible. Dynamic programming was used to obtain all the optimal permutations, and the distributions of index values could only be estimated using a large number of randomly generated permutations.

The results of the analysis showed that total confusion increased markedly when moving from 8.2 to 15.5 m, and slightly when moving from 15.5 to 22.9 m. Perhaps more interesting, however, is that the pattern of confusion also seemed to change as distance was increased. For example, 'pressed cork' and 'grass lawn' move up significantly in the optimal ordering as distance is increased. At all distances, these two textures are frequently misidentified when they are presented, however, the pattern of misidentification changes. At the 8.2 m distance, 'grass lawn' is, by far, the most frequent incorrect response for the 'pressed cork' stimuli. When moving to 15.5 m, 'herringbone weave' replaces 'grass lawn' as the most frequent incorrect response for the 'pressed cork' stimuli, and, at 22.9 m, 'straw matting' supplants 'herringbone weave.' In spite of the changing patterns, an excellent compromise permutation was identified that provides index values for each of the three matrices that are more than 95% of their respective optimal values.

4. SUMMARY

This paper extends the multiobjective programming approach to combinatorial data analysis proposed by Brusco and Stahl (2001) to the analysis of multiple proximity matrices. In this regard, the paper also builds on the work of Hubert (1979, 1987, Chapter 6) by providing another tool for investigating relationships among multiple matrices. By solving the multiobjective programming model for different objective function weights, the quantitative analyst can explore the efficient set of the multiobjective problem and often identify a matrix permutation that provides good index values for all relevant matrices. Two numerical examples were provided using a set of matrices from the psychological literature.

REFERENCES

Baker, F. B., & Hubert, L. J. (1977). Applications of combinatorial programming to data analysis: Seriation using asymmetric proximity measures. *British Journal of Mathematical and Statistical Psychology*, **30**, 154–164.

Brusco, M. J., & Stahl, S. (2001). An interactive multiobjective approach to combinatorial data analysis. *Psychometrika*, **66**, 5–24.

Cho, R. Y., Yang, V., & Hallett, P. E. (2000). Reliability and dimensionality of judgments of visually textured materials. *Perception and Psychophysics*, **62**, 735–752.

DeCani, J. S. (1969). Maximum likelihood paired comparison ranking by linear programming. *Biometrika*, **56**, 537–545.

Flueck, J. A., & Korsh, J. F. (1974). A branch search algorithm for maximum likelihood paired comparison ranking. *Biometrika*, **61**, 621–626.

Hubert, L. (1976). Seriation using asymmetric proximity measures. *British Journal of Mathematical and Statistical Psychology*, **29**, 32–52.

Hubert, L. J. (1978). Generalized proximity function comparisons. *British Journal of Mathematical and Statistical Psychology*, **31**, 179–192.

Hubert, L. J. (1979). Generalized concordance. *Psychometrika*, **44**, 135–142.

Hubert, L. J. (1985). Combinatorial data analysis: Association and partial association. *Psychometrika*, **50**, 449–467.

Hubert, L. J. (1987), *Assignment methods in combinatorial data analysis*. New York: Dekker.

Hubert, L., & Arabie, P. (1994). The analysis of proximity matrices through sums of matrices having (anti-) Robinson forms. *British Journal of Mathematical and Statistical Psychology*, **47**, 1–40.

Hubert, L., Arabie, P., & Meulman, J. (1997). Linear and circular unidimensional scaling for symmetric proximity matrices. *British Journal of Mathematical and Statistical Psychology*, **50**, 253–284.

Hubert, L., Arabie, P., & Meulman, J. (2001). *Combinatorial data analysis: Optimization by dynamic programming*. Philadelphia: SIAM.

Hubert, L. J., & Baker, F. B. (1977). The comparison and fitting of given classification schemes. *Journal of Mathematical Psychology*, **16**, 233–253.

Hubert, L. J., & Golledge, R. G. (1981a). A heuristic method for the comparison of related structures. *Journal of Mathematical Psychology*, **23**, 214–226.

Hubert, L. J., & Golledge, R. G. (1981b). Matrix reorganization and dynamic programming: Applications to paired comparisons and unidimensional seriation. *Psychometrika*, **46**, 429–441.

Hubert, L. J., & Schultz, J. V. (1976). Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, **29**, 190–241.

Kendall, M. G. (1970). *Rank correlation methods* (fourth ed.). New York: Hafner.

Manly, B. F. J. (1986). Randomization and regression methods for testing associations with geographical, environmental, and biological distances between populations. *Researches in Population Ecology*, **28**, 201–218.

Mantel, N. (1967). The detection of disease clustering and a generalized regression approach. *Cancer Research*, **27**, 209–220.

Morgan, B. J. T., Chambers, S. M., & Morton, J. (1973). Acoustic confusion of digits in memory and recognition. *Perception and Psychophysics*, **14**, 375–383.

Oden, N. L., & Sokal, R. R. (1992). An investigation of three-matrix permutation tests. *Journal of Classification*, **9**, 275–290.

Robinson, W. S. (1951). A method for chronologically ordering archaeological deposits. *American Antiquity*, **16**, 293–301.

Smouse, P. E., Long, J. C., & Sokal, R. R. (1986). Multiple regression and correlation extensions of the Mantel test of matrix correspondence. *Systematic Zoology*, **35**, 627–632.

Received: March 26, 2001