Ranking a set of objects on the basis of a preference matrix

W. van Doorne

Report 51

DLO The Winand Staring Centre, Wageningen (The Netherlands), 1991

ABSTRACT

W. van Doorne, 1991. *Ranking a set of objects on the basis of a reference matrix.* Wageningen (The Netherlands), DLO The Winand Staring Centre. Report 51. 27 pp.; 7 Tables; 4 Refs.

Occasionally it is required to rank a set of objects in an order of preference. When criteria for evaluating the objects are present, multicriteria techniques could be used for the purpose of ranking. However, it may be a problem to find suitable criteria and/or to weigh them properly. In such cases one often decides to use pair-wise comparison of the objects as a basis for ranking them. The full matrix of comparisons is then used to evaluate the objects. But what is the meaning of comparing an object with itself, and how to assign a value to the corresponding matrix element ? As no satisfactory answers were found in literature, a method avoiding these questions was elaborated. Much attention is paid to methods for speeding up the ranking algorithm.

Keywords: preference matrix, paired comparisons, permutation method

ISSN 0924-3062

Project KGI2 [496RM/10.91]

CONTENTS

PREFACE

In research and practice occasionally ranking problems arise. The objects to be ranked do not always depend on well defined criteria on the basis of which the objects can be evaluated. Moreover the user of the rankings may impose restrictions on the results. An example is comparing a set of choropleth maps in order to find the 'best' one. Then the criteria for judging the maps generally are vague and there may be more than one best map. The ultimately preferred map may, for example, be chosen by the requirement that some particular information is preserved in the map and/or that a small number of classes is used.

Ranking not always serves to find 'best' objects. One may wish to detect groups of equivalent objects within the set that is examined. An other question may be: find the second best object in case the best ones are not accepted due to restrictions. Yet an other question: establish what rank numbers a specified object may have.

A ranking method based on intuitive comparison of objects and coping with the above kind of problems, is presented.

SUMMARY

It is described how a preference matrix containing the paired comparisons in a set of objects can be used to arrange the objects by means of a simple and natural evaluation criterion for sequences of objects. The algorithm is essentially a permutation method and finds the optimal rankings by maximizing the criterion value. It avoids many non-promising arrangements by calculating bounds for certain statistics and dropping permutations which do not satisfy these bounds. The evaluation criterion also provides a means of finding near-optimal rankings. The calculation efforts required by permutation methods tend to be exponentially related to the number of objects. The same is true for the algorithm that is proposed here. But there is one essential difference: the trend is increasingly less steep for an increasing number of objects when compared to simpler permutation methods. About 25 objects can be arranged on a VAX-4300 computer within a few minutes of run time whereas with other permutation methods this might take hours.

1 INTRODUCTION

Kendall and Babington Smith (1939) gave an illustration of the type of problem this report deals with. To a dog six kinds of food (objects) are to be offered in order to obtain a ranking of the objects. The question then is how to make the dog to rank the objects. We assume that the dog has no notion of 'criteria' and 'evaluation', so ranking has to be done in an intuitive way; in this case the dogs taste must guide us. If the objects are offered simultaneously to the animal judge, it first has to select one object to find the food that it likes best. After that, one of the residual objects has to be selected as the second best food, etc. These multiple choice problems might be too hard to tackle. Therefore the objects are offered in pairs to faciliate choosing. From each pair the dog will choose its preferred food by eating it as the first one.

After carrying out such an experiment, the preference matrix can be formed. Its elements p_{ij} equal 0 or 1; if p_{ij} =1 object i is preferred to object j, otherwise p_{ij} =0. In the above experiment the matrix of Table 1 was obtained. The quantities listed to the right of it will be explained in the next section.

Table 1 A preference matrix and its associated quantities

Objects							**Row totals**		**Pairs of 1's in each row**
j	**1**	**2**	**3**	**4**	**5**	**6**	R_i	R_i^2	
i									
1	-	**1**	**1**	**0**	**1**	**1**	**4**	**6**	**16**
2	**0**	-	**0**	**1**	**1**	**0**	**2**	**4**	**1**
3	**0**	**1**	-	**1**	**1**	**1**	**4**	**16**	**6**
4	**1**	**0**	**0**	-	**0**	**0**	**1**	**1**	**0**
5	**0**	**0**	**0**	**1**	-	**1**	**2**	**4**	**1**
6	**0**	**1**	**0**	**1**	**0**	-	**2**	**4**	**1**
							Total 15	**45 = Q_n**	**15 = T_n**

There are many ways to rank objects on the basis of a preference matrix. Taking row totals is one method; it uses an unweighted linear combination of the matrix columns to evaluate the objects. Wei (1952) uses a linear combination with unequal weights. As his procedure is based on matrix multiplication, the main diagonal has to be given values. It seems natural but not imperative to take p_{ii} =1/2. Wei more subtly takes into account the separate elements of the preference matrix as compared with row totals, but two drawbacks remain, as in other methods. First, the main diagonal must be defined and secondly, the columns are linearly combined to evaluate the objects. Unfortunately, the only reason for choosing that particular kind of function seems to be simplicity. That is why a ranking method will be proposed in which not the objects themselves, but sequences of objects are evaluated. In its basic form the method is part of a multicriteria technique suggested by Jacquet-Lagrèze (Bernard et Besson, 1971; De Jong, 1982). Here, it is a starting point to develop an algorithm for ranking objects solely on the basis of a preference matrix having an empty main diagonal.

Applying permutation methods requires the aid of a computer. Therefore the main objective is to find methods to speed up calculations to make the permutation principle applicable in case a somewhat larger number of objects has to be ranked.

2 PROPERTIES AND NOTIONS CONNECTED TO THE PREFERENCE MATRIX

2.1 Row totals and transitive triads

We now introduce Q_n, the sum of squares of row totals of an n by n preference matrix (see Table 1). It is linearly related to T_n, the number of 'transitive triads'. A triad is transitive if it allows a permutation i,j,k of its objects, such that both i is preferred to j, j preferred to k and i preferred to k. It will be shown that

$$Q_n = 2T_n + n(n-1)/2 \qquad (1)$$

To this end a function T_{ijk} is devised which equals 1 in transitive cases and 0 otherwise. Then, as triples can be arranged in six ways, T_n is equal to 1/6 the sum of T_{ijk} over i,j and k. Table 2 gives all cases of triads; here it is used to find a suitable function:

$$T_{ijk} = p_{ij} \cdot p_{ik} + p_{jk} \cdot p_{ji} + p_{ki} \cdot p_{kj}$$

Table 2 Transitive and non-transitive triads

Case	p_{ij}	p_{jk}	p_{ik}	T_{ijk}	
1	**1**	**1**	**1**	**1**	**transitive**
2	**0**	**0**	**0**	**1**	**transitive**
3	**0**	**1**	**1**	**1**	**transitive**
4	**1**	**0**	**0**	**1**	**transitive**
5	**1**	**0**	**1**	**1**	**transitive**
6	**0**	**1**	**0**	**1**	**transitive**
7	**1**	**1**	**0**	**0**	**non-transitive**
8	**0**	**0**	**1**	**0**	**non-transitive**

Summing the first term over the column indexes (j and k) yields twice the number of pairs of 1's in a matrix row (i), see Table 1. By summing over the rows, denoting the row totals by R_i, we get

$$\sum_i R_i(R_i-1) = Q_n - n(n-1)/2$$

The other terms each yield the same sum. From this (1) follows.

2.2 Extrema for Q_n and T_n

From (1) we see that the maxima of Q_n and T_n occur simultaneously; the same applies to the minima. The sum of the row totals is a constant. So only when the row totals are all different, Q_n attains its maximum. In that case the row totals equal 0,1,...,n-1;

ranking the objects according to descending row totals, causes the upper triangle of the corresponding preference matrix to be filled with 1's. So Q_n and T_n attain their maximum if a renumbering of objects exists, such that $p_{ij} = 1$ for $i<j$. Using (1), we find

$$Q_{max} = \sum_{o}^{n-1} j^2 = n(n-1)(2n-1)/6$$

$$T_{max} = n(n-1)(n-2)/6 \qquad (2)$$

Now the number of transitive triads equals the number of possible triads, so each triad is transitive. In this situation the objects can be ranked without inconsistencies occurring when comparing them. The matrix for such a 'perfect ranking', obtained by descending row totals, will be named a 'perfect (n by n preference) matrix'.

Minima for Q_n and T_n occur when the row totals are as equal as can be. A preference matrix of that kind occurs when the [n/2] diagonals adjacent to and above the main diagonal contain 1's only, whereas each other upper element is 0; [...] indicates integer truncation. In that case all row totals equal (n-1)/2 for odd n; for even n, the row totals equal either n/2 or n/2-1, each value occurring n/2 times. Using (1) we get from this case

$$Q_{min} = n\{(n-1)(n-2) + 2[n/2]\}/4$$
$$T_{max} = n\{(n-1)(n-4) + 2[n/2]\}/8 \qquad (3)$$

3 OPTIMAL RANKINGS OF OBJECTS

3.1 Optimal ranking

In a perfect ranking, each triad of objects is transitive; arranging the objects according to descending row totals produces a perfect matrix. So the perfect ranking is found by permuting the objects in such a way that the associated row and column permutations yield an upper triangle containing a maximum number of 1's. Now let the sum of the upper triangle elements of a preference matrix be named the 'upper sum' of that matrix. Then the perfect ranking is obtained by permuting the objects in such a way that the associated row/column permutation yields a maximum upper sum.

In general in paired comparisons inconsistencies do occur, i.e., in some triads the transitivity rule is violated and a perfect ranking cannot be found. Therefore we introduce 'optimal rankings'. Optimal rankings are those arrangements of objects that maximize the upper sum; near-optimal rankings correspond to the second largest upper sum, etc.

3.2 Measuring the consistency of ranking

Let us again consider Table 2. The first six cases represent the transitive triads. The more 1's there are in the upper triangle of the preference matrix, the more likely the occurrence of a case 1, 3, 5 or 7 is. The more 0's occur in the upper triangle, the more the even numbered cases tend to occur. When there are more 0's than 1's in the upper triangle, this can be reversed by reversing the order of the objects as this has the effect of interchanging 0's and 1's in the matrix. So larger maximum upper sums generally imply a more frequent occurrence of odd numbered cases. As case 7 is outnumbered by the cases 1, 3 and 5 the number of transitive triads tends to be larger at larger maximum upper sums.

Taking into account the results of Section 2, we find that both Q_n, T_n and the maximum upper sum can serve to measure the consistency of ranking.

3.3 Near-optimal rankings

In Table 1 the upper sum equals 10. By permuting the objects, this sum can be maximized to 13. Three optimal rankings can accomplish this: 1,3,2,5,6,4; 1,3,5,6,2,4 and 1,3,6,2,5,4.

Possibly they can not be accepted because of restrictions imposed by the user of the results. Then, it is easy to obtain near-optimal solutions. One way is interchanging two neighbours in an optimal ranking as this reduces the maximum upper sum by the

minimum quantity of 1.

The objects 1 and 3 can be considered the best objects, object 4 the worst one. The other objects have no definite rank number: to some extent they can be permuted without affecting the upper sum.

3.4 Why and how speeding up the search for optimal rankings ?

When searching for optimal rankings of objects, in principle the upper sum has to be calculated for each ranking. Using fast permutation algorithms, it was found that generating all rankings yields too much calculation efforts, even for $10<n<20$. So speeding up the search for optimal rankings is often required.

Suppose we construct rankings by adding and deleting objects. Each time a ranking of all objects is obtained, we check whether the upper sum is not smaller than the largest upper sums of the complete rankings found earlier. If it does, the ranking is kept as a temporarily optimal one. Otherwise it is broken down until the remaining ranking can be extended by an object satisfying conditions yet to be specified. These conditions ensure that there is a positive chance that the by one object extended remaining ranking can be extended to an optimal ranking of all objects. Each time all conditions are satisfied, an object is added until the resulting ranking must be broken down, the remaining ranking extended, etc. There will be two kinds of conditions. Suppose a sequence of objects coded 1,2,...,m has been formed and that an object, m+1, is selected to extend it. Then the first kind of condition is necessary in order that 1,2,...,m+1 can be an optimal ranking. If the condition does not hold, the latter sequence does not contribute a maximum to the upper sum of a completed sequence. In that case 1,2,...,m+1 is not extendible to any optimal ranking of all objects and m+1 is not accepted.

What was described here, is looking back in a sequence to ascertain whether the last object, m+1, can be linked. On the other hand it is possible to look ahead. This leads to a second kind of condition. We first observe that the upper sum of a complete sequence consists of three parts. The first part is the upper sum of the partial sequence already obtained, the second part is the sum of the p that compare the objects of the partial sequence to the residual objects. The third part, unknown during searching, is the upper sum of the residual objects. In the sequel a general method of finding a non-trivial upper bound of the upper sum will be developed. We intend to apply the method to the residual objects: by adding the result to the first and second part, an upper bound for the upper sum of all objects is found, in each stage of the construction of a complete sequence.

The above conditions prevent the forming of many non-promising sequences when searching for optimal rankings. Looking back, looking ahead and calculation speed will be treated in the sequel.

4 CONDITIONS WHEN LOOKING BACK

4.1 A set of formulas

Suppose that a partial sequence 1,2,...,m has been formed. Under what conditions may it be useful to add an object when searching for optimal rankings of all n objects ? To answer this, we select two permutation actions, I and II, and see how they affect the upper sum.

I: 1,2,...,m,m+1; m+2,...,n $\rightarrow$ m+1,1,2,...,m; m+2,...,n
II: 1,2,...,m;m+1, m+2,...,n $\rightarrow$ 1,2,...,m; m+2,m+3,...,n,m+1

As $p_{ij} + p_{ji} = 1$ the decrease of the upper sum due to action I is

$$\sum_{i=1}^{m} (p_{i,m+1} - p_{m+1,i}) = m - 2 \sum_{i=1}^{m} p_{m+1,i} \qquad (4)$$

When considering the effect of II, the row total R_{m+1} is splitted into two terms, comprising the row elements below and above the diagonal, respectively. Then II causes an upper sum decrease of

$$\sum_{i=m+2}^{n} (p_{m+1,i} - p_{i,m+1} = 2R_{m+1} - n + m + 1 - 2 \sum_{i=1}^{m} p_{m+1,i} \qquad (5)$$

If 1,2,...,m+1 is an optimal ranking (4) and (5) are non-negative values while each subsequence ending in m+1 is optimal too. From this, necessary conditions for the last object follow. They read:

> object j, $1 \le j \le n$, is accepted to extend a sequence of m objects, $1 \le m \le n-1$, when searching for optimal rankings of n objects, only if
>
> $$\sum_{i=1}^{k} p_{ji} \le \min ([k/2], R_j - [(n-k)/2]) \text{ for } k=1,2,...,m \qquad (6)$$
>
> where i points to the k objects directly preceding j; R_j is the the row total for object j

The conditions (6) are the 'looking back conditions'. They apply to m=0 too; by taking k=0 in (6) we see that for each possible first object, j, of an optimal ranking

$$R_j \ge [n/2] \qquad (7)$$

For m=1 conditions (6) reduce to $p_{ij} = 1$, meaning that object j can directly follow i in an optimal ranking, only if $p_{ij} = 1$. So the objects and their possible direct successors can be listed prior to searching. To the list we also add the objects satisfying (7), i.e., the possible leaders of optimal rankings. The list of successors will

be a means of linking objects during searching.

4.2 An example of looking back and reducing searching efforts

Now the foregoing is applied to the example of Table 3 where n=6. Condition (7) says that row totals no smaller than [n/2]=3 correspond to possible leaders of optimal rankings (1,2 and 4). During searching, one of the sequences to be tested is 1,5,2,6,3.

Table 3 A preference matrix and its list of successors

j	1	2	3	4	5	6	R_j	Object	Successors
1	-	1	1	0	1	0	3	1	2,3,5
2	0	-	1	1	0	1	3	2	3,4,6
3	0	0	-	1	1	0	2	3	4,5
4	1	0	0	-	1	1	3	4	1,5,6
5	0	1	0	0	-	1	2	5	2,6
6	1	0	1	0	0	-	2	6	1,3
								Leaders:	1,2,4

Then the list of successors shows that object j=4 is the successor of 3 yet available to extend the sequence. Table 4 shows that the test, performed according to (6), ends in the rejection of object 4. Here, about fifty percent of the sequences was excluded in advance by (7), after which (6) left only six percent to be examined.

Table 4 Applying the looking back conditions (6) to the matrix in Table 3

k	R_4	$\sum_{i=1}^{k} p_{ji}$		Min (,)	Object 4 accepted ?
1	3	$p_{43}=0$	$\leq$	Min (0,1)	yes
2	3	.. + $p_{46}=1$	$\leq$	Min (1,1)	yes
3	3	..... + $p_{42}=1$	$\leq$	Min (1,2)	yes
4	3	..... + $p_{45}=2$	$\leq$	Min (2,2)	yes
5	3	..... + $p_{41}=3$	$>$	Min (2,3)	no

5 AN UPPER BOUND OF THE UPPER SUM

Earlier we found that Q_n, the sum of squares of row totals, is a measure of the consistency of ranking. It can also be shown that Q_n also plays a role in methods for speeding up the search for optimal rankings. For that purpose we first establish how Q_n is affected when changing one element of the matrix. Let R_r be the total of row r, p_{ij} the element to be changed and d the resulting decrease of Q . Then, when p_{ij} is reduced from 1 to 0, we get

$$p_{ij} \rightarrow 0 \; ; \; p_{ji} \rightarrow 1 \; ; \; R_i \rightarrow R_i\text{-}1 \; ; \; R_j \rightarrow R_j\text{+}1 \tag{8}$$

and so

$$d = R_i^2 - (R_i\text{-}1)^2 + R_j^2 - (R_j\text{+}1)^2 = 2(R_i\text{-}R_j\text{-}1) \tag{9}$$

Now let us consider a perfect preference matrix. Among the n by n preference matrices this matrix has both the largest Q_n value and the largest upper sum S_n . In such a matrix the maximum d value occurs when the element (1,n) of the first row is made 0. After this the next largest decreases of Q_n occur when reducing (1,n-1) and (2,n). The next lower d values occur in (1,n-2), (2,n-1) and (3,n). In this way one passes through diagonals, parallel to the main diagonal. In diagonal k, k equal decreases of Q_n occur; they will be denoted by d_k. The example for n=8 is given in Table 5, which was calculated using (8) and (9). It shows (the changes of) the row totals R_r, the sum of squares Q_n and the upper sum S_n. In the columns headed 'k=...' pairs of rows are marked. Each pair (i,j) indicates the element that is made 0; the d_k value involved is listed in the same column, the resulting R_r, Q_n and S_n are in the next column. From the systematic way the marks occur, we find

$$d_k = 2(n\text{-}2k), \quad k=1,2,...,[(n\text{-}1)/2] \tag{10}$$

For the final k, Q_n attains its minimum, the reduction of diagonals having generated the kind of matrix described near (3).

Each time an element in diagonal k is made 0, the upper sum decreases by 1 and Q_n by the amount d_k, which is the largest decrease of Q_n after clearing the diagonals 1,...,k-1. It is felt that, after a number of reduction steps, the sum of the involved d_k's gives a lower bound of Q_n at the S_n value attained. To prove this by induction, suppose Q_n is a minimum after performing some reduction steps. Then, at the next step, d_k occurs in the same or next diagonal. Anyhow, as d_k decreases with k, the next d_k value is the maximum decrease of Q_n yet available. Hence after one more reduction, Q_n again is a minimum at the S_n value attained. Because Q_n starts at a fixed value, a lower bound for Q_n is obtained.

Before making the reduction technique operational we show how to utilize the lower bound of Q_n. In Table 5 four steps are used to descend from the point (S_{max}, Q_{max}) of perfect ranking to the point (S,Q), Q being marked by *. Then S = S_{max}-4 and

$Q = Q_{max}-d_1-2d_2-d_3$. According to the above, when S_n is fixed at S, Q is a lower bound of Q_n. Therefore S is an upper bound of S_n when Q_n is fixed at Q. For an arbitrary n, a table like Table 5 could be constructed and Q_n calculated from the preference matrix. Using this value to enter the table an upper bound for S_n could be found by interpolation and upward integer rounding. For n=8, interpolating and rounding at Q_n=114 gives S_n=26. For Q_n=104, S_n=23 is found directly.

Table 5 Reducing a perfect preference matrix along its upper diagonals k; the changes of the row totals, sum of squares and upper sum due to reduction

Row number	Row totals changing by reduction						Final row total
	k=1	k=2	k=2	k=3	k=3	k=3	
1	7-	6-	5	5-	4	4	4
2	6	6	6-	5	5-	4	4
3	5	5	5	5	5	5-	4
4	4	4	4	4	4	4	4
5	3	3	3	3	3	3	3
6	2	2	2	2+	3	3	3
7	1	1+	2	2	2+	3	3
8	0+	1	1+	2	2	2+	3
d_k=	12	8	8	4	4	4	
Q_n=	140	128	120	112	108*	104	100
S_n=	28	27	26	25	24	23	22

6 FORMULATING AND APPLYING AN UPPER BOUND OF THE UPPER SUM

6.1 A set of formulas

Adopting the foregoing technique to find an upper bound for the residual upper sum, we must be able to calculate this bound for every matrix size. Therefore formulas for an upper bound S of the upper sum of an n by n preference matrix were designed. They read:

$$
\begin{aligned}
&A = (n^2 + 4/3)^{1/2} \\
&B = 24Q_n - n(7n^2 - 12n+8) \\
&C = 6Q_n + n(n-1)(4n+1) \\
&\\
&U = 1/3.\text{Arc cos}(B/A^3) \\
&\\
&V = 2A.\cos U \qquad \text{if } B \geq 0 \\
&V = 2A.\cos(U-2\pi/3) \qquad \text{if } B<0 \\
&\\
&k = [(V+n+2)/4] \qquad \text{if } Q_n < Q_{max} \\
&k = 0 \qquad \text{if } Q_n = Q_{max} \\
&\\
&S = \frac{C - 4k\{3n(n-1)-k^2+1\}}{12(n-2k)} \\
&\\
&\text{where S is rounded upward.}
\end{aligned}
\tag{11}
$$

A proof based on reducing diagonals is outlined now. Starting from a perfect matrix, the minimum number (k) of diagonals needed to reduce the sum of squares to a value no larger than the given Q_n is determined first. After that, interpolation in diagonal k takes place to find the maximum number of reduction steps that do not reduce the sum of squares to a value below Q_n. By subtracting this number from the upper sum for perfect ranking, the upper bound S is found. Table 5 illustrates the procedure: it takes three diagonals to arrive below the given $Q_n = Q$ (marked by *). Therefore, we find S in diagonal 3.

To elaborate the above into a set of formulas we put $D = Q_{max}-Q_n$; then k is the smallest natural number that satifies

$$\sum_{j=1}^{k} j.d_j \geq D \tag{6}$$

Because of (10) this can be rewritten

$$k(k+1)(3n-4k-2)/3 \geq D$$

Finding k is the same as finding the smallest positive root of the associated cubic equation, followed by upward integer rounding. After that, the expression for S is found by solving the equation

$$D - \sum_{j=o}^{k-1} j.d_j = (S_{max} - S - k(k-1)/2).d_k$$

because both sides equal the sum of the decreases of Q_n due to the reductions in diagonal k.

6.2 Looking ahead and reducing searching efforts; an example

Referring to the preference matrix in Table 3 it is now demonstrated how the upper bound is used during searching. The row totals give the initial sequence 1,2,4,3,5,6 (upper sum=10). Searching yields the better, complete sequence 1,2,3,4,5,6 (upper sum=11). One of the next partial sequences found is 1,2,4. Is it worthwile to extend it ? Let us list the matrix (Table 6), ranking the residual objects 3,5,6 arbitrarily.

Table 6 Looking ahead during searching

j	1	2	4	3	5	6	Residual row total
1	-	1	0	1	1	0	
2	0	-	1	1	0	1	
4	1	0	-	0	1	1	
3	0	0	1	-	1	0	1
5	0	1	0	0	-	1	1
6	1	0	0	1	0	-	1

The sum of those upper sum elements that occur in the rows corresponding to the objects of the sequence 1,2,4 already found, is known (=8). An upper bound of the residual part is found by applying (11) to the residual matrix:
n = 3; Q_n = 3; A = 3.215; B = -33; C = 96
U = 1/3.Arc cos(-0.9935) = 1.01
V = 6.43 cos(1.01 - 2π/3) = 3.0
k = 2 ; S = 2

Adding S to the known part (=8) of the complete upper sum yields a bound of 10. As this is less than the previous sum of 11, all rankings starting by 1,2,4 can be skipped. According to the list of successors the searching process jumps to a sequence 1,2,6,...

Almost trivial as this small sized example is, it clearly shows the power of the upper bound test. In computing practice the test is useful when a larger number of objects (ten or more) is involved. In that case the extra run time needed for testing is negligible when compared with the amount of time gained by the reduction of the number of sequences to be examined. If a few objects have to be ranked the extra

time is relatively large but small in an absolute sense. Therefore the test is recommended in automatic calculations for both a small and a larger number of objects.

7 CALCULATION SPEED

7.1 Calculation speed in two cases of a preference matrix

In the foregoing, tests for speeding up the search for optimal rankings were developed. We will now establish what kind of relation may exist between the number of objects and the computer run time needed to find optimal rankings. Two cases will be studied, a perfect matrix and a random one.

Let the perfect ranking be 1,2,...,n. Then the direct successors of object i are numbered higher than i. Because of (7) the objects numbered no higher than [(n+1)/2], are the possible leaders when forming sequences. As a measure of calculation efforts we take the number of times that an object is added when forming sequences. Beginning by object s, the frequency of adding j objects succesively, is the number of ways one can place j-1 objects between s and n. As this is equal to the binomial coefficient $\binom{n-s-1}{j-1}$ the number of times an object is added to the leading object s is

$$\sum_{j=1}^{n-s} j.\binom{n-s-1}{j-1} = (n-s+1).2^{n-s-2}$$

By summing over s the frequency of adding objects is found to be

$$\sum_{s=1}^{m} (n-s+1).2^{n-s-2} = \{(n-1).2^n - (E-1).2^E\}/4$$

where m = [(n+1)/2] and E=[n/2]. As $(n-1).2^n$ dominates the term comprising E, the computer run time needed for finding the optimal ranking in a perfect matrix, is roughly proportional to

$$(n-1).2^n \qquad (12)$$

Next a random preference matrix is considered. When sequences are formed, a particular sequence i,j,k,... occurs if both $p_{ij} = 1$, $p_{jk} = 1$ and $p_{kl} = 1$ for $l \neq i,j$ or k. So the expected number of sequences is $2.2.2^{n-3} = 2^{n-1}$ for a random matrix. All sequences are equally likely, so the average sequence length equals (n-1)/2. Multiplying this by the expected number of sequences we again find that the run time is roughly proportional to (12) when applying the list of successors.

7.2 Calculation speed in general

In case of a perfect matrix the sum of squares attains its maximum, see (2); random matrices tend to produce equal row totals and thus small Q_n and T_n values according to (3). In both cases computer run time is proportional to $(n-1).2^n$. It is therefore expected that, when searching is based on the list of successors only, this

proportionality holds almost independently of Q_n.

As a final test the algorithm was applied to random preference matrices: sequences of objects were formed by adding and deleting objects (Section 3) while applying both the list of successors and the methods of looking back (6) and ahead (11). In Table 7 some statistics on computer run time are collected. A statistically very significant trend was found: the run time tends to be proportional to $(n-1)p^n$, where p is a value of no more than 1.6 whereas with the method of solely applying the list of successors p is around 2. Moreover the proportionality factor for the simpler technique is smaller than the factor for the searching method. So the searching method reduces the expected computer run time by a factor of less than $(1.6/2)^n$ as compared with simpler techniques. For n=20 objects the reduction factor is about .01, for n=25 it is .004.

Table 7 Computer run time for the searching algorithm when applied to random preference matrices of different sizes (numbers of objects); the Fortran program was run on a VAX-4300 computer

Number of objects	Run time for searching (sec)				Number of matrices
	min.	max.	average	median	
12	.0	.2	.1	.1	25
16	.1	11	2	1.4	25
20	.1	50	7	2.6	25
24	.2	388	35	6.9	25

REFERENCES

BERNARD, G. et M.L. BESSON, 1971. "Douze Méthodes d'Analyse Multicritère". *Revue Française d'Informatique et de Recherche opérationnelle*, no.V-3.

JONG, J.L.T. DE, 1982. "Multicriteria-analyse: een toepassing van de permutatie-methode van Jaquet-Lagrèze". *Beleidsanalyse* '82-3.

KENDALL, M.G. and B. BABINGTON SMITH, 1939. "On the Method of Paired Comparisons". *Biometrika*, Vol.31.

WEI, T.H., 1952. *The algebraic foundations of ranking theory*. Cambridge University, England. Microfilm, exp. 1-146.