many authors published different proposals. But up to then, the lack of an explicit control structure will be a permanent source of concern for many APL programmers. ■

References

[BA1] F. H. D. van Batenburg, 1982; *"New Control Structures in APL?"* APL Quote Quad Vol. 13, No. 2, pp. 15 – 20

[BA2] F. H. D. van Batenburg, 1991; *"L-One-Two-Three (L1 :...L2 :...L3 :) Considered Harmful,"* APL Quote Quad Conference Proceedings 1991, pp. 330 – 337

[BA3] F. H. D. van Batenburg, V. Bos, J. J. M. Riethoven, J. P. Abrahams, C. Pley, 1992; *"Porting and Optimising STAR: A Case Study of Suffering and Surfacing,"* APL Quote Quad Conference Proceedings, 1992, pp. 265 – 274

APL2 Implementation of a New Definition of Consistency of Pairwise Comparisons

Michael W. Herman
and
Waldemar W. Koczkodaj
Department of Mathematics and Computer Science
Laurentian University
Sudbury, Ontario P3E 2C6 Canada
Fax: 705-673-6532
E-mail: icci@nickel.laurentian.ca
or waldemar@ramsey.cs.laurentian.ca

Keywords:
APL2 functions, consistency, pairwise comparison, decision making

Abstract

A set of APL2 functions is presented for a new definition of consistency of pairwise comparisons. By calculating an inconsistency for each comparison, these functions locate the source of inconsistency and can thus be used to improve relative judgements.

Introduction

In the decision-making process, many factors must be considered simultaneously and with about the same degree of importance. It has been shown by numerous examples [1,2] that the Pairwise Comparison Method introduced by Thurstone [3], can always be used to make a final decision in a comparatively straightforward manner. Yet, despite its practicality and its use in some important applications such as, for example, decisions about the use of nuclear power in Holland [2], the Pairwise Comparison Method is not a tool that is widely used by decision makers. According to [4], the failure of the Pairwise Comparison Method to become more popular is due to deficiencies in the old definition of consistency. Saaty's definition [5] is based on the largest eigenvalue of the matrix of comparative judgements. An eigenvalue is a global attribute of a matrix and says nothing about the location of any inconsistencies. Furthermore, even though small changes in a matrix lead to correspondingly small changes in the eigenvalues, there is no proof that a large change will necessarily produce a large variation in the eigenvalues.

In the next section, we provide a motivation for the new definition of consistency and in the following section, an example is presented. We use APL2 notation throughout. The definitions of all the functions used are gathered together in an Appendix at the end of the paper.

The New Definition

In the Pairwise Comparison Method, we deal with a reciprocal matrix *A* where each element `A[I;J]>0` and expresses the relative importance of two attributes *I* and *J*. Each item `A[I;J]` has the reciprocal property

```
A[I;J] = ÷A[J;I]
```

so that the entire matrix *A* satisfies the relation

```
A≡÷⍉A
```

For a consistent reciprocal matrix [6], we also have

```
A[I;K] = A[I;J]×A[J;K]
```

for all *J* in `⍳↑⍴A`, which expresses the transitive nature of consistent relative comparisons. This relation can be displayed by the weighted digraph in Figure 1 on page 38, which indicates that for consistent judgements, the relative importance of *I* over *J* multiplied by the relative importance of *J* over *K* should be equal to the relative importance of *I* over *K*. Note that the weights are replaced by their reciprocals if the direction of an arc is reversed.

In the particular case of three attributes, the pairwise comparison matrix *A*3 is

$$\begin{array}{ccc} 1 & a & b \\ \div a & 1 & c \\ \div b & \div c & 1 \end{array}$$

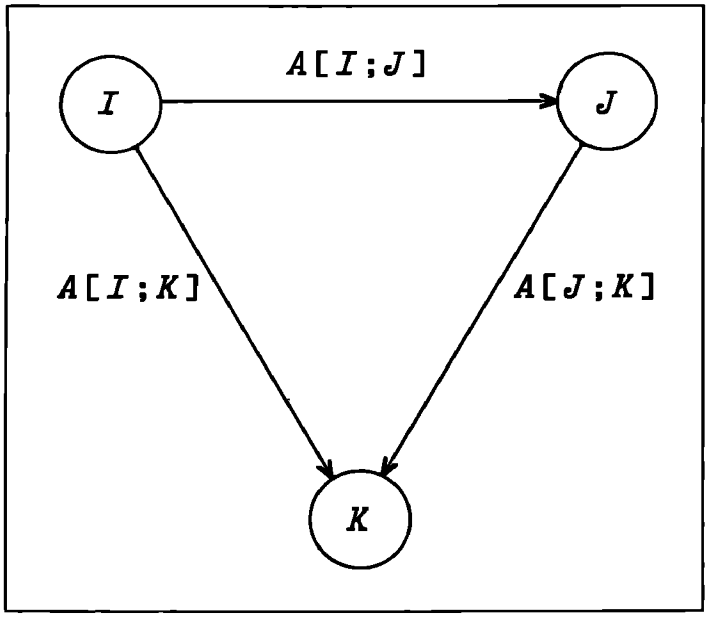

Figure 1. Weighted digraph representation of one triad in a reciprocal matrix.

where *a* expresses an expert's relative preference of attribute 1 over attribute 2, *b* expresses his preference of attribute 1 over 3, and *c* is the relative preference of attribute 2 over attribute 3. *A*3 is called a basic reciprocal matrix; *A*1 is a trivial case, and *A*2 is always consistent. Matrix *A*3 is consistent if `b = a×c`.

The new definition of consistency introduced in [4] involves a measure of the deviation from the nearest consistent reciprocal matrix. The interpretation of the consistency measure becomes apparent if we represent a basic reciprocal matrix by a vector of the three elements

```
a b c
```

Since for a consistent basic reciprocal matrix,

```
b = a×c
```

we can produce three consistent basic reciprocal matrices (represented by three vectors) by computing one element from the remaining two elements. These three vectors are:

```
V1← (b÷c)     b       c

V2←   a     (a×c)     c

V3←   a       b     (b÷a)
```

The inconsistency measure is then defined in terms of the distance to the nearest consistent basic reciprocal matrix represented by one of the above three vectors. Using either the Euclidean or Chebysheff metrics, the three distances are given by

```
V1 V2 V3 DIST ⊂ a b c
```

where *DIST* is one of the distance functions. This simplifies to

```
| (a-b÷c) (b-a×c) (c-b÷a)
```

Dividing by the vector `a b c` for normalization and then taking the minimum yields

```
CM←⌊/|(a-b÷c) (b-a×c) (c-b÷a)÷a b c
```

for the consistency measure. This expression can be simplified to

```
CM←⌊/|(1-b÷a×c) (1-a×c÷b)
```

or

```
CM←1-(b⌊a×c)÷b⌈a×c
```

We can easily extend the above definition to matrices of higher order. First we note that the above expression is of the form

```
CM←1-(A[I;K]⌊A[I;J]×A[J;K])÷
         A[I;K]⌈A[I;J]⌈A[I;J]×A[J;k]
```

Thus for a given matrix element the consistency can be defined as the maximum of the *CM*s for all possible triads which include this element

```
A[I; ] A[I;K] A[ ;K]
```

Hence, for a higher order matrix *A* we have

```
CM←⌈/¨¨1-(A⌊B)÷A⌈B←A,.×A
```

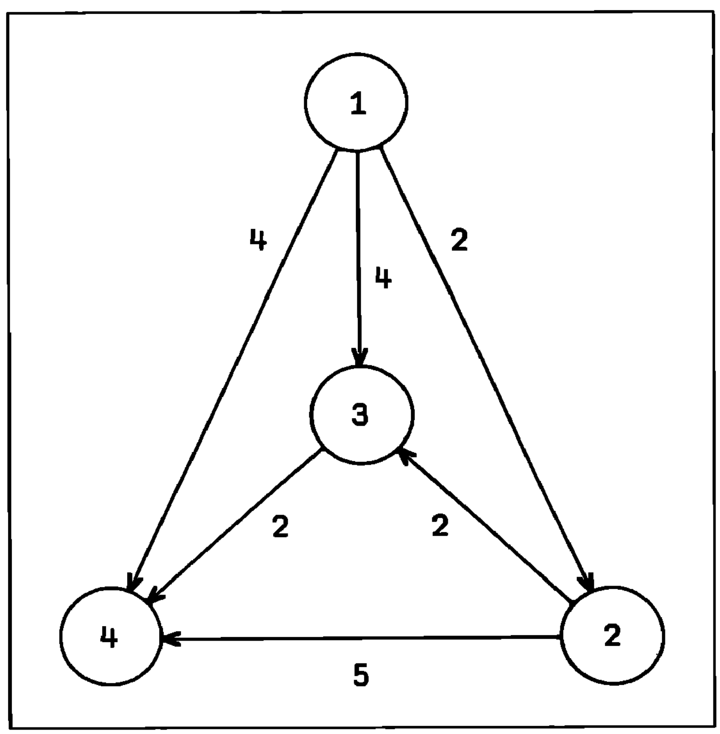

Figure 2. Weighted digraph, judgements used in the example.

An Example

Let us consider the graph of judgements in Figure 2 on page 38. The expression

```
3 RND A (ICM A←SETUP 4)
```

yields the following reciprocal and inconsistency matrices:

```
1     2    4    4    0    0.6  0.5  0.6
0.5   1    2    5    0.6  0    0.2  0.6
0.25  0.5  1    2    0.5  0.2  0    0.5
0.25  0.2  0.5  1    0.6  0.6  0.5  0
```

Changing the 1 4 entry to an 8 with the expression

```
3 RND A (ICM A←A CHNG 1 4)
```

gives

```
1      2    4    8    0    0.2  0    0.2
0.5    1    2    5    0.2  0    0.2  0.2
0.25   0.5  1    2    0    0.2  0    0.2
0.125  0.2  0.5  1    0.2  0.2  0.2  0
```

which shows that judgement 1 3 is now consistent, whereas the other five judgements still display an inconsistency of 0.2. If the 2 4 entry is changed to a 4 with the expression

```
3 RND A (ICM A←A CHNG 2 4)
```

we get

```
1      2    4    8    0  0  0  0
0.5    1    2    4    0  0  0  0
0.25   0.5  1    2    0  0  0  0
0.125  0.5  0.5  1    0  0  0  0
```

All the entries in the second matrix are zero, indicating that the judgements are now totally consistent.

Conclusion

We have presented a new definition of consistency of pairwise comparisons in APL2. Since the functions are all executable in TryAPL2, this work should be accessible to anyone with a PC. Only the core functions needed to perform the calculations have been considered in this paper. A complete system would require a command shell to allow interaction with non-APL users, and an improved display of the results.

We hope that the new definition will refocus the attention of researchers from trying to find better approximations (in the form of heuristics) to solutions of inconsistent matrices, to devising heuristics that can influence judgements to be more consistent (but by no means totally consistent). To change the inconsistency, we need to know not only its value but also its location and this is what our definition is designed to do. It gives a judge the necessary feedback and opportunity to reconsider his judgements. Note that it is not advisable to allow complete flexibility since attempting to achieve total consistency may result in unbiased opinions becoming biased ones. Thus, we may want to restrict a judge to changing only a fixed number of opinions by a fixed total. For example, in the case of a matrix of order 4 with six judgements, we may allow a maximum of only three modifications such that the total of all the changes does not exceed three, say.

Hopefully the diversity of interests in the APL community will result in further research in this area.

Acknowledgments

This project was partially supported by a grant from the Natural Science and Engineering Council of Canada under Grant OGP 0036838. ■

References

[1] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making*, Springer-Verlag, Berlin (1981)

[2] P. Nijkamp, *Multicriteria Evaluation in Physical Planning*, Springer-Verlag, Berlin (1991)

[3] L. L. Thurstone, *A Law of Comparative Judgements*, Psychological Review 34, 273 − 286 (1927)

[4] W. W. Koczkodaj, *A New Definition of Consistency of Pairwise Comparisons*, Mathematical and Computer Modelling 18(7), 79 − 84 (1993)

[5] T. L. Saaty, *A Scaling Method for Priorities in Hierarchical Structure*, Journal of Mathematical Psychology 15, 234 − 281 (1977)

[6] H. A. Davis, *The Method of Pairwise Comparisons*, Griffin, London (1963)

Appendix: APL2 Functions for the Pairwise Comparison Method

```
     ∇
[0]    I←ICM A; B
[1]   ⍝Calculate inconsistency measures
       for pairwise comparison
[2]   ⍝              matrix <A>
[3]    I←⌈/¨¯1-(A⌊B)÷A⌈B←A,.×A
     ∇
```

```
     ∇
[0]    R←SETUP N; I; S
[1]   ⍝Setup N by N pairwise comparison
       matrix <R>
[2]    R←, N Nρ1
[3]    S←GET¨I←IX N
[4]    R[1+N N⊥⊃¨1+I,⌽¨I]←S,÷S
[5]    R←N NρR
     ∇

     ∇
[0]    I←IX N
[1]   ⍝Generate indices of elements in
       upper triangle of N by N matrix
[2]    I←(,N∘.<N)/,N∘.,N←⍳N
     ∇

     ∇
[0]    W←GET I
[1]   ⍝Display indices <I> and get
       corresponding weight <W>
[2]    ⍞←I←(⍕I),': '
[3]    W←C2N (ρI)↓⍞
     ∇

     ∇
[0]    N←C2N C
[1]   ⍝Convert numeric character string
       <C> to number <N>
[2]   ⍝Include input error checking here
       if desired
[3]    N←⍎C
     ∇

     ∇
[0]    R←A CHNG I;T
[1]   ⍝Change entry <I> of matrix <A>
[2]    R←A
[3]    (I⌷R)←T←GET I
[4]    ((⌽I)⌷R)←÷T
     ∇

     ∇
[0]    R←N RND X
[1]   ⍝Round <X> to <N> decimal places
[2]    R←(10*-N)×⌊0.5+X×10*N
     ∇

     ∇
[0]    D←X E_DIST Y
[1]   ⍝Euclidean distance between <X>
       and <Y>
[2]    D←(+/(X-Y)*2)*0.5
     ∇

     ∇
[0]    D←X C_DIST Y
[1]   ⍝Chebysheff distance between <X>
       and <Y>
[2]    D←⌈/|X-Y
     ∇
```

APL and Coroutines

Norman Thomson
Mail Point 17H
IBM UK Ltd.
P. O. Box 30
Greenock, Scotland PA18 6AP
Tel: +44-475-895165

The article on *Functional Programming with APL2* in the December 1993 issue of Quote Quad (Vol. 24, No. 2) stimulated a note from Nick Beaumont of the Syme Faculty of Business, Monash University, Australia, asking whether I had considered the possible association with APL2 of another computer science concept, namely coroutines.

A coroutine is a member of a set of routines in which **A** calls **B** which in turn calls **A** in a resumptive rather than a recursive fashion, that is instead of creating a new invocation of **A**, the previous invocation is resumed immediately after the point where it was interrupted by the call to **B**. This facility might be necessary for example in writing an editor which allowed the data to be split and the various parts edited separately in different windows. A pair of simple coroutines might look like:

```
coroutine A;
  begin
    resume B;
    action A1;
    resume B;
    action A2;
 end;

coroutine B;
  begin
    resume A;
    action B1;
    resume A;
    action B2;
 end;
```

Although the "resume" statement in the above has a superficial resemblance to the form of an APL2 operator, maturer consideration shows that the two are not alike, in that the essence of "resume" is the retention of a prior environment, which is not part of the way in which an APL2 operator modifies a function. APL2 does not provide a means of passing values between successive invocations of the same function, and hence coroutines have to be *modelled* rather than *implemented* in APL2. One tool which APL2 has to assist in this process is ⎕*LC*, and an example of how a set of two simple coroutines could be modelled in APL2 is: