Reconstruction of An Account's Past

M.S. Waterman and V.W. Lowe
University of California
Los Alamos Scientific Laboratory
Los Alamos, New Mexico 87545

ABSTRACT

Historical records and data frequently have been reduced by combination of accounts. The problem we address is that of finding all ways a given set of accounts could have been combined to give some given set of reduced accounts. An algorithm has been developed to accomplish this task and a computer code in FORTRAN is given. An example with some MUF data is also presented.

1. Introduction

Recent interest in accountability and safeguards has prompted investigation into historical records and historical data. These records and data have been reduced by combination of accounts. In this situation, MUFs from various accounts were added to form a reduced number of MUFs associated with new accounts. Records on which accounts were combined are frequently missing. Sometimes it is not even clear what accounts are candidates for forming the new accounts.

Thus the problem of current interest is retracing the (paper) tracks of an account. Our formulation of the problem is that we are given a possible set of m accounts $\{x_1, x_2, \ldots, x_m\}$ which may have been combined to give a new set of n accounts $\{y_1, \ldots, y_n\}$. The solution is a program which produces as output all possibilities (if any exist) for producing the desired result.

Another situation to which our results might be applied is when the set $\{x_1, \ldots, x_m\}$ are measurements of m initial accounts and the set $\{y_1, \ldots, y_n\}$ are measurements of n accounts which are measured after the initial accounts were physically combined. This situation motivates us to find all ways of combining x's to give y's with an error bound.

In the next section we give a careful and precise description of the problem and discuss its solution.

2. Algorithm

To make a formal statement of the problem, some notation must be introduced. Let

$$\underset{\sim}{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_m \end{pmatrix},$$

$$\underset{\sim}{y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{pmatrix},$$

and

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdot & \cdot & & \\ \cdot & \cdot & & \\ \cdot & \cdot & & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$$

*Work done under the auspices of the U.S. Energy Research and Development Administration.

where $a_{ij} = 0$ or 1 and $\sum_{i=1}^{n} a_{ij} \leq 1$ for all j. Our object is to find all A satisfying the above conditions such that

$$\text{(I)} \quad A\underset{\sim}{x} = \underset{\sim}{y}$$

or

$$\text{(II)} \quad |(A\underset{\sim}{x})_j - y_j| \leq e_j \text{ for all j, } 1 \leq j \leq n,$$

where $0 \leq e_j$ is some given error bound on the $j\underline{th}$ entry. Of course, I is equivalent to II if $e_1 = e_2 = \ldots = e_n = 0$.

The program given in section 4 is for the more general situation II but, for ease of discussion, we consider case I here. To force the program to do case I, simply set the error vector (E) there equal to 0.

Next, we discuss the meaning of the restrictions on A. When $a_{ij} = 1$, this means x_j was used in the sum to obtain y_i. Of course, $a_{ij} = 0$ means x_j was not used to obtain y_i. The restriction $\sum_{i=1}^{n} a_{ij} \leq 1$ is interpreted as x_j can be used at most once in all sums for the y's. Clearly, any x_j should not be used more than once. Allowing x_j not to be used at all makes it possible to search over a larger set of x's than were actually used to produce $\underset{\sim}{y}$.

The most naive algorithm to find possible A's would be to try each subset of $\{x_1, x_2, \ldots, x_m\}$ to obtain each y_j and reject any collections that were inconsistent or failed to yield $\underset{\sim}{y}$. With this algorithm, $(2^m)^n = 2^{mn}$ cases must be considered. For m = n = 10, this amounts to approximately 1.27 x 10^{30} cases. Clearly, no existing computer can handle this case.

To make a faster algorithm, we now consider only A's satisfying the restrictions $a_{ij} = 0$ or 1 and $\sum_{i=1}^{n} a_{ij} \leq 1$ for all j. There are $(n + 1)^m$ such cases. For m = n = 10, $(n + 1)^m = 2.59 \times 10^{10}$, still a large number of cases.

The algorithm we employ makes an important modification of the scheme mentioned in the preceding paragraph. If a candidate A is found where a collection of rows such that the corresponding x sums do not yield the corresponding y values, no other A's which contain these rows will be considered. This procedure, known in computer science as backtracking, greatly speeds up the running time.

Another modification we have failed to make is the elimination of A's where the only difference is permutation of x values that are equal. Frequently, ± 1 and 0 are repeated values of x so that this would be an important improvement. However, we are unable to retain backtracking and include this modification.

3. Example

Sometimes a declared MUF is obtained by adding several intermediate or temporary MUFs kept by production people on the shop floor. If this declared MUF is thought to be the only MUF of interest, the record relating the temporary MUFs to the declared MUF may be destroyed. However, since the temporary MUFs are part of the production records, they are usually still available.

To make these ideas specific, suppose the following information is available from the production people:

Intermediate MUF (grams)	Throughput (kilos)
10	.1
-50	7.0
75	1.2
-15	1.0
-100	.9
20	1.5
5	.1
5	6.0
60	.9
-10	1.4

Each MUF is that declared for the operation of a given process in a given month. The declared MUFs are given below.

Declared MUF (grams)
10
25
-95
60

When examining the declared MUFs, the 60 gram MUF is singled out for further investigation. One part of this investigation is to determine the throughput associated with this MUF. Unfortunately the throughput is not part of the record, and an attempt must be made to reconstruct the events that led to this MUF.

Here m = 10, n = 4,

$$\underset{\sim}{x} = \begin{pmatrix} 10 \\ -50 \\ 75 \\ -15 \\ -100 \\ 20 \\ 5 \\ 5 \\ 60 \\ 10 \end{pmatrix}, \text{ and } \underset{\sim}{y} = \begin{pmatrix} 10 \\ 25 \\ -95 \\ 60 \end{pmatrix}$$

Define the throughput vector $\underset{\sim}{t}$ by

$$\underset{\sim}{t} = \begin{pmatrix} .1 \\ 7.0 \\ 1.2 \\ 1.0 \\ .9 \\ 1.5 \\ .12 \\ 6.0 \\ .9 \\ 1.4 \end{pmatrix}$$

Running our program, we obtain 50 A matrices. That is, there are 50 distinct ways of combining $\underset{\sim}{x}$ to obtain $\underset{\sim}{y}$. If A is a typical matrix, then our interest is in the last component of

$$\underset{\sim}{s} = A\,\underset{\sim}{t} = (s_1, s_2, s_3, s_4).$$

For the first output matrix A^1,

$$\underset{\sim}{s} = A^1\,\underset{\sim}{t} = \begin{pmatrix} 0&0&0&0&0&0&1&1&0&0 \\ 0&1&1&0&0&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \end{pmatrix} \begin{pmatrix} .1 \\ 7.0 \\ 1.2 \\ 1.0 \\ .9 \\ 1.5 \\ .1 \\ 6.0 \\ .9 \\ 1.4 \end{pmatrix}$$

$$= (6.1,\ 8.2,\ 3.4,\ .9).$$

Thus, $s_4^1 = .9$.

Continuing this procedure, we find 14 throughputs of 2.2 kilos and 36 throughputs of .9 kilos. This information can then be given to the group investigating the 60 gram MUF.

4. Program

```
      SUBROUTINE TRACE(X,M,Y,N,E)
      DIMENSION IA(3Ø,3Ø), X(3Ø), Y(3Ø), E(3Ø)
*********************************************
*  M X-VALUES ARE SEARCHED TO GIVE SUMS     *
*  EQUAL TO N Y-VALUES WITHIN ERROR E       *
*********************************************
      WRITE (5,1ØØ)
      WRITE (5,11Ø)((I,X(I)),I=1,M)
      WRITE (5,12Ø)
      WRITE (5,13Ø)((I,Y(I)),I=1,N)
      WRITE (5,14Ø)
      DO 1Ø I=1,N
      DO 1Ø J=1,M
   1Ø IA(I,J)=Ø
      I=Ø
   2Ø I=I+1
      IF (I.EQ.N+1) GO TO 7Ø
   3Ø CALL ADD (IA,I,IEND,M)
      IF (IEND.EQ.1) GO TO 5Ø
      SS=-Y(I)
      DO 4Ø J=1,M
   4Ø SS=SS+X(J)*IA(I,J)
      CHK=ABS(SS)
      IF (CHK.LE.E(I)) GO TO 2Ø
      GO TO 3Ø
   5Ø IF ((IEND.EQ.1).AND.(I.EQ.1)) GO TO 9Ø
      DO 6Ø L=I,N
      DO 6Ø J=1,M
   6Ø IA(L,J)=Ø
      I=I-2
      GO TO 2Ø
   7Ø WRITE (5,15Ø)
      DO 8Ø L=1,N
   8Ø WRITE (5,16Ø)(IA(L,J),J=1,M)
      I=N
      GO TO 5Ø
   9Ø RETURN
C
  1ØØ FORMAT (6X,*LIST OF ALL A SATISFYING*,
     */,16X,*AX=Y*,/,6X,*WHERE*,/)
  11Ø FORMAT (16X,*X(*,I2,*)=*,F1Ø.5,/)
  12Ø FORMAT (6X,*AND*,/)
  13Ø FORMAT (16X,*Y(*,I2,*)=*,F1Ø.5,/)
  14Ø FORMAT (/,6X,*IF A(I,J)=1,THEN X(J)*,
     **WAS USED TO MAKE Y(I)*)
  15Ø FORMAT (/,/,/,/,/,6X,***A***)
  16Ø FORMAT (X,3Ø(X,I1))
      END
      SUBROUTINE ADD(IA,L,IEND,M)
      DIMENSION IA(3Ø,3Ø)
      IEND=Ø
      J=M+1
   1Ø J=J-1
      IF (J.EQ.Ø) IEND=1
      IF (J.EQ.Ø) GO TO 5Ø
      IT=Ø
      IF (L.EQ.1) GO TO 3Ø
      LL=L-1
      DO 2Ø K=1,LL
   2Ø IT=IT+IA(K,J)
   3Ø IF (IT.NE.Ø) GO TO 1Ø
      IF (IA(L,J).EQ.Ø) GO TO 4Ø
      IA(L,J)=Ø
      GO TO 1Ø
   4Ø IA(L,J)=1
   5Ø RETURN
      END
```

REPRINT

This article is reprinted from the Winter 1977-1978 (Volume VI, No. 4) issue of NUCLEAR MATERIALS MANAGEMENT which was issued February 1, 1978. The Journal is published by the Institute of Nuclear Materials Management, Inc., with publications offices at Kansas State University, Seaton Hall, Manhattan, Kansas 66506 U.S.A. (Phone: 913-532-5837).