Programming Techniques

R. M. McCLURE, Editor

An Algorithm for the Probability of the Union of a Large Number of Events

G. D. MILLER
*University of Canterbury,**
Christchurch, New Zealand

An algorithm is presented which efficiently evaluates the probability for the union of n independent and not mutually exclusive events. The problem is that of evaluating the sums of the products of all possible combinations of n variables in minimum time and storage space.

KEY WORDS AND PHRASES: algorithm, probability, optimum, storage vs. time compromise, set union, mutually exclusive events
CR CATEGORIES: 5.12, 5.5, 5.6

1. Introduction

The probability of the union of n events, E_α ($\alpha = 1, 2, \cdots, n$), [1] is

$$P\left(\bigcup_{\alpha=1}^{n} E_\alpha\right) = \sum_{\alpha=1}^{n} P(E_\alpha) - \sum_{\beta>\alpha=1}^{n} P(E_\alpha \cap E_\beta) + \cdots + (-1)^{n-1} P(E_1 \cap \cdots \cap E_n). \tag{1}$$

If the n events are independent then

$$P\left(\bigcup_{\alpha=1}^{n} E_\alpha\right) = \sum_{\alpha=1}^{n} P(E_\alpha) - \sum_{\beta>\alpha=1}^{n} P(E_\alpha)P(E_\beta) + \cdots + (-1)^{n-1} P(E_1) \cdots P(E_n). \tag{2}$$

Denoting $P(E_\alpha)$ by P_α, eq. (2) becomes

$$P\left(\bigcup_{\alpha=1}^{n} E_\alpha\right) = \sum_{\alpha=1}^{n} P_\alpha - \sum_{\beta>\alpha=1}^{n} P_\alpha P_\beta + \sum_{\gamma>\beta>\alpha=1}^{n} P_\alpha P_\beta P_\gamma + \cdots + (-1)^{r-1} \sum_{\eta>\cdots>\beta>\alpha=1}^{n} P_\alpha P_\beta \cdots P_\eta + \cdots + (-1)^{n-1} P_1 P_2 \cdots P_n. \tag{3}$$

For the large values of n the evaluation of eq. (3) in minimum time while using minimum storage space becomes a problem. To achieve minimum storage space it is necessary to determine each product in eq. (3) separately and add it to the sum. For the rth term this entails $(r \times {}_nC_r) - 1$ operations and for $P(\bigcup_{\alpha=1}^{n} E_\alpha)$ a total of

$$2(n-1) + \sum_{r=2}^{n} [(r \times {}_nC_r) - 1]$$

operations. The time required would be prohibitive for large n.

On the other hand, by storing each of the ${}_nC_r$ products in the rth term for use in evaluating the $(r+1)$-th term a minimum of ${}_nC_{n/2}$ storage spaces would be required.

The algorithm presented here is an optimal compromise in that it is nonrepetitive in evaluating products while at the same time it requires only $3n$ storage spaces in evaluating $P(\bigcup_{\alpha=1}^{n} E_\alpha)$.

2. Procedure

If we define T_r to be the rth term in eq. (3), then

$$P\left(\bigcup_{\alpha=1}^{n} E_\alpha\right) = \sum_{r=1}^{n} T_r . \tag{4}$$

Each T_r is the sum of ${}_nC_r$ products which may be subdivided into $(n - r + 1)$ partial sums S_{ri}. Hence

$$T_r = (-1)^{r-1} \sum_{i=1}^{n-r+1} S_{ri} . \tag{5}$$

Each set of partial sums S_{ri} ($i = 1, 2, \cdots, n - r + 1$) may be determined from the previous set $\{S_{r-1}, i\}$ as follows:

$$S_{ri} = \sum_{j=i+1}^{n-r+1} P_i S_{r-1,j} . \tag{6}$$

Taking the first set of S_{1i} ($i = 1, 2, \cdots, n$) to be $S_{11} = p_1$, $S_{12} = p_2, \cdots, S_{1n} = p_n$, the probability $P(\bigcup_{\alpha=1}^{n} E_\alpha)$ may

* School of Engineering

```
      FUNCTION PRUN(N,PROB)
      DIMENSION PROB(50), PSUM(50), TERM(50)
C     PROGRAM TO DETERMINE THE PROBABILITY OF THE
                                   UNION OF
C     N(MAXIMUM 50) INDEPENDENT, BUT NOT MUTUALLY
                                   EXCLUSIVE EVENTS
C
C     PRUN=PROBABILITY OF THE UNION
C     N=NUMBER OF EVENTS INCLUDED
C     PROB(I)=PROBABILITY OF THE I-TH EVENT
C     TERM(R)=R-TH TERM IN THE N-TERM EXPRESSION
                                   FOR PRUN
C     PSUM(I)=I-TH OF (N-R+1) PARTIAL SUMS IN TERM(R)
C
C     INITIALIZATION OF VARIABLES
      DO1J=1,N
      PSUM(J)=0.0
    1 TERM(J)=0.0
      TERM(N)=1.0
      PRUN=0.0
C
C     EVALUATION OF PARTIAL SUMS AND TERMS OF PRUN
      DO2J=1,N
      TERM(1)=TERM(1)+PROB(J)
      TERM(N)=TERM(N)*PROB(J)
    2 PSUM(J)=PROB(J)
      IF(N-2)7,7,6
C     EVALUATION OF MIDDLE (N-2) TERMS OF PRUN
    6 I2=N-1
      J2=N
      DO4I=2,I2
      J2=J2-1
      K2=J2+1
      DO4J=1,J2
      PSUM(J)=0.0
      K1=J+1
      DO3K=K1,K2
    3 PSUM(J)=PSUM(J)+PROB(J)*PSUM(K)
    4 TERM(I)=TERM(I)+PSUM(J)
C
C     SUMMATION OF N TERMS OF PRUN
    7 SIGN=-1.0
      DO5J=1,N
      SIGN=-SIGN
    5 PRUN=PRUN+SIGN*TERM(J)
      RETURN
      END
```

Fig. 1

be determined; i.e.

$$P\left(\bigcup_{\alpha=1}^{n} E_\alpha\right) = \sum_{i=1}^{n} P_i + \sum_{r=2}^{n} (-1)^{r-1} \sum_{i=1}^{n-r+1} \sum_{j=i+1}^{n-r+1} P_i S_{r-1,j}\,. \tag{7}$$

The number of arithmetic operations is reduced in (7) to

$$N = 2(n-1) + \sum_{r=2}^{n} \left\{ \left[\sum_{i=1}^{n-r+1} 2(n-r+1) + 1 \right] + (n-r) \right\}; \tag{8}$$

i.e.

$$N = 2(n-1) + \sum_{r=2}^{n} [(n-r+1)^2 + (n-r)]. \tag{9}$$

For n as small as 10 the savings in time and storage space are considerable. For the minimum storage condition 6898 arithmetic operations are required at $n = 10$. By using the algorithm presented here, this can be reduced to 339 operations. At the same time the required storage space is reduced from 252 locations under the minimum time condition to 30 here.

Figure 1 shows a listing of the algorithm coded as a Fortran FUNCTION SUBPROGRAM.

Received November, 1967; revised March, 1968

REFERENCE

1. Wilks, S. S. *Mathematical Statistics.* Wiley, New York, 1962, p. 12.

Algorithms

J. G. HERRIOT, Editor

ALGORITHM 336
NETFLOW [H]

T. A. Bray and C. Witzgall
(Recd. 2 Oct. 1967 and 20 May 1968)
Boeing Scientific Research Laboratories, Seattle, WA 98124

KEY WORDS AND PHRASES: capacitated network, linear programming, minimum-cost flow, network flow, out-of-kilter
CR CATEGORIES: 5.32, 5.41

procedure *NETFLOW* (*nodes*, *arcs*, *I*, *J*, *cost*, *hi*, *lo*, *flow*, *pi*, *INFEAS*);
value *nodes*, *arcs*; **integer** *nodes*, *arcs*;
integer array *I*, *J*, *cost*, *hi*, *lo*, *flow*, *pi*; **label** *INFEAS*;
comment This procedure determines the least-cost flow over an upper and lower bound capacitated flow network.

Each directed network arc a is defined by nodes $I[a]$ and $J[a]$, has upper and lower flow bounds $hi[a]$ and $lo[a]$, and cost per unit of flow $cost[a]$. Costs and flow bounds may be any positive or negative integers. An upper flow bound must be greater than or equal to its corresponding lower flow bound for a feasible solution to exist. There may be any number of parallel arcs connecting any two nodes.

The procedure returns vectors *flow* and *pi*. $flow[a]$ is the computed optimal flow over network arc a. $pi[n]$ is a number—the dual variable—which represents the relative value of injecting one unit of flow into the network of node n. *NETFLOW* may be entered with any values in vectors *flow* and *pi* (such as those from a previous or a guessed solution) feasible or not. If the initial contents of *flow* do not conserve flow at any node, the solution values will also not conserve flow at that node, by the same amount.

This procedure is a revision (see remark by T. A. Bray and C. Witzgall [1]) of Algorithm 248 [2]. Like the original, it follows the out-of-kilter algorithm described by D. R. Fulkerson [3] and elsewhere. It follows the RAND code by R. J. Clasen (Fortran) in three instances, using a single set of labels *na*, which correspond to the *nb* of Algorithm 248, avoiding superfluous tests in the part following *BACK* (for instance, $c > 0 \wedge flow[a] < lo[a]$ is equivalent to $c > 0$ at this point of the program), and taking advantage of the fact that arcs remain in kilter and need not be rechecked again. In addition, the convention $inf = -1$ is adopted in order to permit costs and bounds of value around 99999999 without their interfering with the initiation of minimum search.

References:

1. Bray, T. A., and Witzgall, C. Remark on Algorithm 248, NETFLOW. *Comm. ACM 11* (Sept. 1968), 633.
2. Briggs, William A. Algorithm 248, NETFLOW. *Comm. ACM 8* (Feb. 1965), 103.
3. Fulkerson, D. R. An out-of-kilte method for minimal-cost flow problems. *J. Soc. Ind. Appl. Math. 9* (Mar. 1961), 18–27;