

as before and computes

$$\hat{\sigma}^2 + L\hat{\sigma}_{TA}^2 + JL\hat{\sigma}_{PA}^2 + KL\hat{\sigma}_T^2$$

for testing this source of variation. Thus instances in which exact tests exist for certain model terms can be considered a special case of the algorithm.

If no exact  $F$  test exists for a certain model term, the degrees of freedom corresponding to the denominator used in calculating the  $F$  statistic for this term can be obtained as

$$\hat{\nu} = \frac{\hat{\gamma}^2}{\sum(\hat{\gamma}_i^2/\nu_i)}$$

$\hat{\gamma}$  is the value used as the denominator in calculating the  $F$  value, while the  $\hat{\gamma}_i$  and  $\nu_i$  are the calculated mean squares and degrees of freedom, respectively of the model terms whose EMS's when combined equal the expectation of  $\gamma$ . See [3] for a derivation of this formula. All values for calculating  $\hat{\nu}$  are known besides the  $\hat{\gamma}_i$  and these are obtained by constructing from the algorithm a table denoting the structure of the EMS's of the model terms. From the table a triangular set of equations solvable by a backward solution is easily obtained. The solution vector gives the calculated mean squares which are to be used in computing  $\hat{\nu}$ .

All concepts and methodology presented in this paper have been fully implemented on the IBM 7074 while the authors were members of the Statistical Laboratory at Iowa State University. An attempt was made to avoid machine dependencies. As a consequence, the program has been readily converted to the IBM 360, Model 50 now in use at Iowa State University. It is intended that the algorithms developed will form part of a more extensive statistical computing system oriented toward algebraic problem specification.

*Acknowledgment.* The authors wish to thank E. J. Carney, Assistant Professor of Statistics, Iowa State University, for constructing a preliminary version of the variance component algorithm to handle factorial models.

RECEIVED DECEMBER 1965; REVISED JULY 1966

#### REFERENCES

- HEMMERLE, W. J. Algebraic specifications of statistical models for analysis of variance computations. *J. ACM* 11 (1964), 234-239.
- SATTERTHWAITE, F. E. An approximate distribution of estimates of variance components. *Biometrics* 2 (1946), 110-114.
- SCHIEFFÉ, HENRY. *The Analysis of Variance*. John Wiley, New York, 1959.
- SCHLATER, J. E. Analysis of variance and covariance computations on a digital computer for balanced complete structures based on algebraic model specifications. M.S. Thesis, Iowa State U. Library, 1965.

COLLECTED ALGORITHMS FROM CACM  
1961-1966

An ACM Looseleaf Service

Subscriptions: ACM Members, \$15; Nonmembers \$25.

# Algorithms

J. G. HERRIOT, Editor

## ALGORITHM 293

### TRANSPORTATION PROBLEM [H]

G. BAYER (Recd. 9 July 1965 and 22 Aug. 1966)

Technische Hochschule, Braunschweig, Germany

**procedure** *transpl* (*m*, *n*, *inf*, *c*, *a*, *b*, *x*, *kw*); **value** *m*, *n*, *inf*;  
**integer** *m*, *n*, *inf*, *kw*; **integer array** *c*, *a*, *b*, *x*;

**comment** *transpl* is derived from Algorithm 258, *transport*, [*Comm. ACM* 8 (June 1965), 381] in order to reduce running time by about 50 percent. The following notation is used.

*c* *m*, *n*-matrix of unit costs,

*a* array of quantities available,

*b* array of quantities required, following the usual description of the transportation problem,

*inf* greatest positive integer within machine capacity,

*x* *m*, *n*-matrix of flows,

*kw* optimal total costs (computed by procedure).

*c*, *a*, *b* are disturbed by the procedure. Sum of  $a[i] = \text{sum of } b[i]$ . Multiple solutions are left out of account. [Ref.: G. Hadley, *Linear Programming*, Reading, London, 1962, p. 351];

**begin integer** *i*, *j*, *u*, *v*, *k*, *l*, *s*, *t*, *gd*, *h*, *p*, *cij*, *xij*, *ai*, *bj*, *lsvj*, *nlv*;  
**Boolean** *zg*;

**integer array** *g*, *listu*, *nlv*[1:*m*], *r*, *listv*[1:*n*], *ls*[0:*m*+*n*-1], *nl*[1:*m*×*n*], *lsv*[0:*n*];

**comment** in the for-statement  $u := \dots$  after *s33*, operate on all pairs *i*, *j* with  $c[i,j] = 0$ . To win time the array *nl* supervises those zeros; the *j*-indices of zeros in row *i* are kept in  $nl[(i-1) \times n + 1] \dots nl[nlv[i]]$ . In the for-statement  $v := \dots$  after *s33*, operate on all pairs *i*, *j* with  $x[i,j] \neq 0$  (and  $c[i,j] = 0$ ). *ls* supervises those essential zeros, the *i*-indices of essential zeros in column *j* are kept in  $ls[lsv[j-1]+1] \dots ls[lsv[j]$ . Procedure *in* adds to list *ls*, procedure *out* takes out from list *ls* an essential zero in position *i*, *j*;

**procedure** *in*;

**begin**

*lsvj* := *lsv*[*j*];

**for** *t* := *lsv*[*n*] **step** -1 **until** *lsvj* **do** *ls*[*t*+1] := *ls*[*t*];

**for** *t* := *j* **step** 1 **until** *n* **do** *lsv*[*t*] := *lsv*[*t*] + 1;

*ls*[*lsvj*+1] := *i*

**end** ;

**procedure** *out* ;

**begin**

*lsvj* := *lsv*[*j*];

**for** *t* := *lsv*[*j*-1]+1 **step** 1 **until** *lsvj* **do**

**begin**

**if** *ls*[*t*]  $\neq i$  **then go to** *next*;

*s* := *t*; **go to** *ex*;

*next*:

**end** ;

*ex*:

**for** *t* := *j* **step** 1 **until** *n* **do** *lsv*[*t*] := *lsv*[*t*]-1;

*lsvj* := *lsv*[*n*];

**for** *t* := *s* **step** 1 **until** *lsvj* **do** *ls*[*t*] := *ls*[*t*+1]

**end** ;

**for** *i* := 1 **step** 1 **until** *m* **do**

**for** *j* := 1 **step** 1 **until** *n* **do** *x*[*i*,*j*] := 0;

**for** *i* := 1 **step** 1 **until** *m* **do** *nlv*[*i*] := (*i*-1)×*n*;

```

lsv[0] := 0;
for j := 1 step 1 until n do
begin
  listw[j] := 1;
  lsv[j] := 0
end ;
s1:
kw := gd := 0;
comment gd is the defect, i.e., the sum of quantities not yet
transported;
for i := 1 step 1 until m do
begin
  h := inf;
  for j := 1 step 1 until n do
  if c[i, j] < h then h := c[i, j];
  for j := 1 step 1 until n do
  begin
    cij := c[i, j] := c[i, j] - h;
    if cij = 0 then
    begin
      listw[j] := 0;
      nlvi := nlv[i] := nlv[i] + 1;
      nl[nlvi] := j
    end
  end;
  kw := h × a[i] + kw
end see next comment;
for j := 1 step 1 until n do
begin
  if listw[j] = 0 then go to nextj1;
  h := inf;
  for i := 1 step 1 until m do
  if c[i, j] < h then h := c[i, j];
  for i := 1 step 1 until m do
  begin
    cij := c[i, j] := c[i, j] - h;
    if cij = 0 then
    begin
      nlvi := nlv[i] := nlv[i] + 1;
      nl[nlvi] := j
    end
  end;
  kw := h × b[j] + kw;
nextj1:
end;
comment in step 1 the usual reduction of the matrix of costs
is achieved (dual problem), zeros are listed in nl;
s2:
for i := 1 step 1 until m do
begin
  ai := a[i]; nlvi := nlv[i];
  for u := (i-1) × n + 1 step 1 until nlvi do
  begin
    if ai = 0 then go to nexti2;
    j := nl[u];
    bj := b[j];
    if bj = 0 then go to nextj4;
    h := x[i, j] := if ai < bj then ai else bj;
    ai := ai - h; b[j] := bj - h; in;
nextj4:
end;
nexti2:
a[i] := ai; gd := gd + ai
end;
comment applying a usual rule to all zeros we get an initial
flow (restricted primal problem) in step 2;

```

```

s31:
  if gd = 0 then go to s6;
  comment problem is solved if defect has become zero;
s32:
  for j := 1 step 1 until n do r[j] := 0;
  k := 0;
  for i := 1 step 1 until m do
  begin
    if a[i] ≠ 0 then
    begin
      k := k + 1; listu[k] := i; g[i] := inf
    end
    else g[i] := 0
  end;
  comment r[j] = 0 if column j is unlabeled, = i if labeled
  from row i. g[i] = 0 if row i is unlabeled, = inf if a[i] ≠ 0,
  i.e., a[i] is a possible source of flow. The indices i of labeled
  rows are kept in listu[1] ··· listu[k]. In step 3, consisting of
  step 32 and step 33, the maximal flow is found by the la-
  beling process. Labeling ends in only two ways: (a) a column j
  with b[j] > 0 has been labeled: go to step 4, (b) all labeling is
  done, but a positive flow has not been found: go to s5;
s33:
  l := 0;
  for u := 1 step 1 until k do
  begin
    i := listu[u]; nlvi := nlv[i];
    begin
      j := nl[s];
      if r[j] ≠ 0 then go to nextj5;
      r[j] := i; l := l + 1; listu[l] := j;
      if b[j] > 0 then go to s4;
nextj5:
    end
  end in each newly labeled row, see listu, look for zeros in
  unlabeled columns, list them in listv;
  if l = 0 then go to s5;
  k := 0;
  for v := 1 step 1 until l do
  begin
    j := listv[v]; lsvj := lsv[j];
    for s := lsv[j-1]+1 step 1 until lsvj do
    begin
      i := ls[s];
      if g[i] = 0 then
      begin
        g[i] := j; k := k + 1;
        listu[k] := i
      end
    end
  end in each newly labeled column, see listv, look for essential
  zeros in unlabeled rows, label these rows, list them in listu;
  if k = 0 then go to s5;
  go to s33;
  comment step 4. A column j with b[j] has been labeled, b[j]
  is the sink of a possible positive flow, the path of which is
  indicated by labels. Find the minimum flow h along the path;
  h := b[j]; p := j;
mark:
  i := r[j]; j := g[i];
  if j = inf then
  begin
    if a[i] < h then h := a[i]; go to re
  end;
  if x[i, j] < h then h := x[i, j];
  go to mark;

```

```

re: ;
  comment flow h along the labeled path thus reduces defect
  without changing total costs. Correct list of essential zeros
  if necessary. Start labeling anew, optimizing the restricted
  primal problem;
  j := p; b[j] := b[j] - h; a[i] := a[i] - h;
  gd := gd - h;
rel:
  i := r[j]; xij := x[i, j]; x[i, j] := xij + h;
  if xij = 0 then in;
  j := g[i];
  if j = inf then go to s31;
  xij := x[i, j] := x[i, j] - h;
  if xij = 0 then out;
  go to rel;
s5: ;
  comment step 5. Flow is maximal. To find a new solution to
  the dual, take the part of matrix c which is the intersection
  of labeled rows and unlabeled columns, reduce matrix in a
  certain way;
  k := 0; l := n + 1;
  for j := 1 step 1 until n do
  begin
    if r[j] = 0 then
      begin
        k := k + 1; listv[k] := j
      end
    else
      begin
        l := l - 1; listv[l] := j
      end
  end
  end list all labeled resp. unlabeled columns in listv;
  h := inf;
  for i := 1 step 1 until m do
  begin
    if g[i] = 0 then go to nexti6;
    for s := 1 step 1 until k do
    begin
      j := listv[s];
      if c[i, j] < h then h := c[i, j]
    end;
  nexti6:
  end find minimum h in partial matrix;
  for i := 1 step 1 until m do
  begin
    zg := g[i] ≠ 0; nlvi := (i-1) × n;
    for s := 1 step 1 until n do
    begin
      j := listv[s];
      if zg then cij := c[i, j] := c[i, j] - h
      else cij := c[i, j] := c[i, j] + h;
      if cij = 0 then
      begin
        nlvi := nlvi + 1;
        nl[nlvi] := j
      end
    end;
  end;
  for s := 1 step 1 until k do
  begin
    j := listv[s];
    if zg then cij := c[i, j] := c[i, j] - h
    else cij := c[i, j];
    if cij = 0 then
    begin
      nlvi := nlvi + 1;
      nl[nlvi] := j
    end
  end
end

```

```

end;
nlv[i] := nlvi
end reduction, add h to labeled columns, subtract h from
labeled rows. Construct new list of zeros;
kw := h × gd + kw;
comment total costs for new solution of dual;
go to s32;
s6: ;
comment solution, defect has become zero;
end

```

CERTIFICATION OF ALGORITHM 257 [D1]  
 HAVIE INTEGRATOR [Robert N. Kubik, *Comm.*  
*ACM* 8 (June 1965), 381]  
 I. FARKAS (Recd. 29 Apr. 1966 and 18 Aug. 1966)  
 Institute of Computer Science, University of Toronto,  
 Toronto 5, Ont., Canada

*Havieintegrator* was translated with some modifications into FORTRAN IV and was run on the IBM 7094 II at the Institute of Computer Science, University of Toronto. To reduce the effect of roundoff, the calculations were carried through in double precision internally and the result was rounded to single precision. The main change made was that the parameters *x* and *integrand* in *havieintegrator* were replaced by a single parameter of type FUNCTION in FORTRAN IV. The other change was that *mask* was removed. The maximum order of approximation was kept less than or equal to 25, and convergence was obtained in every case.

The results obtained for the two test cases were in agreement with the author's result. Besides, 14 other successful tests were made and those shown in Table I are typical.

TABLE I

Integrand	A	B	True value	eps	Error × 10 <sup>3</sup>	Order required
e <sup>x</sup>	0.0	1.0	1.7182818	10 <sup>-6</sup>	0	3
				10 <sup>-4</sup>	240	2
				10 <sup>-2</sup>	3700	2
x <sup>12</sup>	0.01	1.1	.26555932	10 <sup>-6</sup>	-2	4
				10 <sup>-4</sup>	59	3
				10 <sup>-2</sup>	36041	2
√x	0.0	1.0	.66666667	10 <sup>-6</sup>	-27	3
				10 <sup>-4</sup>	-1982	2
				10 <sup>-2</sup>	-126848	2
1/√x	0.01	1.0	1.8000000	10 <sup>-6</sup>	0	3
				10 <sup>-4</sup>	140	2
				10 <sup>-2</sup>	790	2

Like other integration algorithms that determine sample points in the interval in a deterministic manner, *havieintegrator* may fail in certain instances. For example, any integrand with the property that  $f(a) = f(b) = f[(a + b)/2]$  will lead to the value  $(b - a)f(a)$  which will in general not be an acceptable approximation to  $\int_a^b f(x) dx$ . Thus  $\int_0^{\pi} \sin^2 x dx$  leads to 0. Moreover,  $\int_0^{\infty} xe^{-x} dx$  leads to "almost zero" (in fact,  $5.7966 \times 10^{-17}$ ).

Please turn the page to the 1966 Algorithms Index.

## Index By Subject TC Algorithms, 1966

C1	<u>OPERATIONS ON POLYNOMIALS AND POWER SERIES</u>			F4	<u>SIMULTANEOUS LINEAR EQUATIONS</u>	
C1	273 SOLN.OF EQNS.BY REVERSION	1-66(11)		F4	288 LINEAR DIOPHANTINE EQUATIONS	7-66(514)
				F4	290 EXACT SOLUTION OF LINEAR EQNS.	9-66(683)
				F4	ITER,REFIN.=SOLN.OF POS,DEF.MTX NUM,MATH,V8(206)	
				F4	REAL AND COMPLEX LINEAR SYSTEM	NUM,MATH,V8(222)
C2	<u>ZEROS OF POLYNOMIALS</u>			F5	<u>ORTHOGONALIZATION</u>	
C2	256 MODIFIED GRAEFFE METHOD	6-65(379),9-66(687)		F5	SCHMIDT ORTHONORMALIZATION	COMPUTING V1(159)
C2	283 REAL SIMPLE ROOTS	4-66(273)				
C6	<u>SUMMATION OF SERIES, CONVERGENCE ACCELERATION</u>			G1	<u>SIMPLE CALCULATIONS ON STATISTICAL DATA</u>	
C6	277 CHEBYSHEV SERIES COEFFICIENTS	2-66(86)		G1	289 CONFIDENCE INTERVAL FOR A RATIO	7-66(514)
D1	<u>QUADRATURE</u>			G5	<u>RANDOM NUMBER GENERATORS</u>	
D1	257 HAVIE INTEGRATOR	6-65(381),11-66(795),		G5	266 PSEUDO-RANDOM NUMBERS	10-65(605),9-66(687)
D1	257 12-66(871)			G5	RANDOM UNIFORM	COMP,BULL,V9(105)
D1	279 CHEBYSHEV QUADRATURE	4-66(270),6-66(434)		G6	<u>PERMUTATIONS AND COMBINATIONS</u>	
D1	280 GREGORY QUADRATURE COEFFICIENTS	4-66(271)		G6	ALL PERMUTATIONS OF N OBJECTS	COMP,BULL,V9(104)
D1	281 ROMBERG QUADRATURE COEFFICIENTS	4-66(271)		H	<u>OPERATIONS RESEARCH, GRAPH STRUCTURES</u>	
D2	<u>ORDINARY DIFFERENTIAL EQUATIONS</u>			H	285 MUTUAL PRIMAL-DUAL METHOD	5-66(326)
D2	9 RUNGE-KUTTA	5-60(312),4-66(273)		H	286 EXAMINATION SCHEDULING	6-66(433),11-66(795)
D2	218 KUTTA-MERSON	12-63(737),10-64(585),		H	293 TRANSPORTATION PROBLEM	12-66(869)
D2	218 4-66(273)			J6	<u>PLOTTING</u>	
D2	EXTRAPOLATION METHOD	NUM,MATH,V8(10)		J6	278 GRAPH PLOTTER	2-66(88)
D5	<u>INTEGRAL EQUATIONS</u>			K2	<u>RELOCATION</u>	
D5	SYSTEM OF VOLTERRA EQNS.	ZH,VYCH,MAT,MAT,FIZ.-		K2	284 INTERCHANGE 2 BLOCKS OF DATA	5-66(326)
D5	1965(933)			M1	<u>SORTING</u>	
E1	<u>INTERPOLATION</u>			M1	271 QUICKSORT	11-65(669),5-66(354)
E1	AITKEN INTERPOLATION	COMP,J,V9(211)		R2	<u>SYMBOL MANIPULATION</u>	
E1	NEVILLE INTERPOLATION	COMP,J,V9(212)		R2	BASIC LIST PROCESSING	BIT 1966(166)
E2	<u>CURVE AND SURFACE FITTING</u>			S	<u>APPROXIMATION OF SPECIAL FUNCTIONS...</u>	
E2	275 EXPONENTIAL CURVE FIT	2-66(85)		S	FUNCTIONS ARE CLASSIFIED S01 TO S22, FOLLOWING	
E2	276 CONSTRAINED EXPONENTIAL FIT	2-66(85)		S	FLETCHER-MILLER-ROSENHEAD, INDEX OF MATH. TABLES	
E2	L1 APPROX. ON A DISCRETE SET	NUM,MATH,V8(299)		S14	34 GAMMA FUNCTION	2-61(106),7-62(391),
E2	CHEBYSHEV APPROX.=DISCRETE SET	NUM,MATH,V8(303)		S14	34 9-66(685)	
E4	<u>MINIMIZING OR MAXIMIZING A FUNCTION</u>			S14	54 GAMMA FUNCTION	4-61(180),9-66(685)
E4	178 MINIMIZE FUNCT. OF N VARIABLES	6-63(313),9-66(684)		S14	80 GAMMA FUNCTION	3-62(166),9-66(685)
E4	251 FUNCTION MINIMIZATION	3-65(169),9-66(686)		S14	221 GAMMA FUNCTION	3-64(143),10-64(586),
E4	MIN.OF UNIMODAL FCN.OF 1 VAR.	COMP,BULL,V9(104)		S14	221 9-66(685)	
F1	<u>MATRIX OPERATIONS, INCLUDING INVERSION</u>			S14	291 LOGARITHM OF GAMMA FCN.	9-66(684),9-66(685)
F1	274 HILBERT DERIVED TEST MATRIX	1-66(11)		S15	DERIV.OF BUYS ERROR FCN.	COMP,BULL,V9(105)
F1	287 INTEGER MATRIX TRIANGULATION	7-66(513)		S15	COMPL.ERROR INT.=COMPLEX ARG.	BIT 1965(290)
F1	SYMM,DECOMP.OF POS,DEF.BAND MTX	NUM,MATH,V7(357)		S21	56 ELLIPTIC INTEGRAL-SECOND KIND	4-61(180),1-66(12)
F1	SYMM,DECOMP.OF POS,DEF.MTX.	NUM,MATH,V7(368)		S22	282 DERIVATIVES OF EXP(X OR IX)/X	4-66(272)
F1	SYMM,DECOMP.OF POS,DEF.BAND MTX	COMPUTING V1(77)		S22	292 REGULAR COULOMB WAVE FCNS.	11-66(793)
F2	<u>EIGENVALUES AND EIGENVECTORS OF MATRICES</u>			Z	<u>ALL OTHERS</u>	
F2	HOUSEHOLDER RED.=COMPLEX MAT.	NUM,MATH,V8(79)		Z	MANY-ELECTRON WAVEFUNCTIONS	CACM 4-66(278)
F2	SYMM,MAT.=LLT AND STURM SEQ.	COMP,J,V9(103)				
F3	<u>DETERMINANTS</u>					
F3	41 DETERMINANT EVALUATION	4-61(176),9-63(520),				
F3	41 3-64(144),9-66(686)					
F3	269 DETERMINANT BY GAUSSIAN ELIM.	11-65(668),9-66(686)				

*Key*—1st column: A1, B1, B3, etc. is the key to the underlined Modified Share Classification heading each group of algorithms; 2d column: number of the algorithm in *CACM*; 3d column: title of algorithm; 4th column: month, year and page (in parens) in *CACM*, or reference elsewhere. This index supplements the previously published indexes: Index by Subject to Algorithms: 1960-1963 [*CACM* 7 (Mar. 1964), 146-149]; 1964 [*CACM* 7 (Dec. 1964), 703]; and 1965 [*CACM* 8 (Dec. 1965), 791].