ALGORITHM 548
Solution of the Assignment Problem [H]

GIORGIO CARPANETO and PAOLO TOTH
University of Bologna, Italy

Key Words and Phrases assignment problem, Hungarian algorithm
CR Categories 5.39, 8.3
Language. Fortran

DESCRIPTION

Problem

The algorithm presented in this paper solves the assignment problem of the following form: given an $n \times n$ cost matrix $(a_{i,j})$, find a permutation (C_i) of integers $1, \ldots, n$ that minimizes

$$T = \sum_{i=1}^{n} a_{i,C_i}.$$

It is supposed, without loss of generality, that the elements of the cost matrix are nonnegative integers.

Algorithm

To give the reader a better understanding of the algorithm proposed below, we define:

C_j	as the row assigned to column j ($j = 1, \ldots, n$);
LC_j	as the label of column j; if $LC_j = 0$, column j is unlabeled ($j = 1, \ldots, n$);
LR_i	as the label of row i; if $LR_i = 0$, row i is unlabeled ($i = 1, \ldots, n$);
T	as the assignment cost;
P_i	as the set containing the columns corresponding to the unassigned zero elements of row i of the cost matrix ($i = 1, \ldots, n$);
RH	as the set containing the current not-completely-explored rows;
U	as the set containing the unassigned rows;
first (s)	as the first element of set s;
next (s)	as the element following the last considered element of set s;
last (s)	as the last element of set s.

Received 5 August 1976 and 23 October 1978.

Author's address: Istituto di Automatica, Facoltà di Ingegneria, Universita Di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy.

Step 1. [Initialization]
Set $C_j = 0$, for $j = 1, \ldots, n$;
set $U = \emptyset$, $N = \{i \mid 1 \le i \le n\}$.

Step 2. [Reduction of the initial cost matrix]
Set $S_j = \min_{i \in N}[a_{i,j}]$, for all $j \in N$;
set $Q_i = \min_{j \in N}[a_{i,j} - S_j]$, for all $i \in N$;
set $a'_{i,j} = a_{i,j} - S_j - Q_i$, for all $i \in N, j \in N$;
set $P_i = \{j \in N \mid a'_{i,j} = 0\}$, for all $i \in N$; set $T = \sum_{k \in N}(S_k + Q_k)$.

Step 3. [Choice of the initial solution]
Set $i = 1$.
a. Set $j = \mathit{first}\,\{k \in P_i \mid C_k = 0\}$; if j exists, go to Step 3d. Otherwise, set $j = \mathit{first}\,\{P_i\}$.
b. Set $m = \mathit{first}\,\{k \in P_{C_j} \mid C_k = 0\}$; if m exists, go to Step 3c. Otherwise, set $j = \mathit{next}\,\{P_i\}$; if j exists, repeat Step 3b; if not, set $U = U \cup \{i\}$, go to Step 3e.
c. Set $C_m = C_j$, $P_{C_j} = P_{C_j} \cup \{j\} - \{m\}$.
d. Set $C_j = i$, $P_i = P_i - \{j\}$.
e. If $i < n$, set $i = i + 1$, go to Step 3a.

Step 4. [Search for a new assignment]
If set U is empty, stop (the optimal assignment is given by vector (C_j), the minimum cost is given by T). Otherwise, set $RH = \emptyset$; set $LR_k = LC_k = 0$, for $k = 1, 2, \ldots, n$; set $r = \mathit{first}\,\{U\}$, $LR_r = -1$.
a. If set P_r is empty, go to Step 4c. Otherwise, if set P_r has at least two elements, set $RH = RH \cup \{r\}$; in any case, set $l = \mathit{first}\,\{P_r\}$.
b. If $LC_l = 0$, set $LC_l = r$, if $C_l = 0$, go to Step 6; if not, set $r = C_l$, $LR_r = l$, go to Step 4a. Otherwise ($LC_l \ne 0$), if $r \in RH$, go to Step 4d.
c. If set RH is empty, go to Step 5. Otherwise, set $r = \mathit{first}\,\{RH\}$.
d. Set $l = \mathit{next}\,\{P_r\}$, if $l = \mathit{last}\,\{P_r\}$, set $RH = RH - \{r\}$; in any case, go to Step 4b.

Step 5. [Reduction of the current cost matrix]
Set $SLR = \{i \in N \mid LR_i \ne 0\}$, $SUC = \{j \in N \mid LC_j = 0\}$.
Set $H = \min_{i \in SLR, j \in SUC}[a'_{i,j}]$.
For all $i \in SLR, j \in SUC$: set $a'_{i,j} = a'_{i,j} - H$, if $a'_{i,j} = 0$, set $P_i = P_i \cup \{j\}$, $RH = RH \cup \{i\}$;
For all $i \in N - SLR, j \in N - SUC$: if $j \in P_i$, set $P_i = P_i - \{j\}$; in any case set $a'_{i,j} = a'_{i,j} + H$.
Set $T = T + H$, $r = \mathit{first}\,\{RH\}$, go to Step 4d.

Step 6. [Assignment of a new row]
Set $C_l = r$, $P_r = P_r - \{l\}$; if $LR_r < 0$, set $U = U - \{r\}$, go to Step 4.
Otherwise, set $l = LR_r$, $P_r = P_r \cup \{l\}$, $r = LC_l$, repeat Step 6.

The efficiency of the algorithm is mainly due to the pointer technique utilized to locate the unexplored rows and the zero elements of the current cost matrix. It is worthwhile to note that, as far as storage requirement is concerned, for each set P_i ($i = 1, \ldots, n$) only the pointer to the first element needs to be stored; in fact, if column j is contained in set P_i, the corresponding element of the current cost matrix (i.e., $a'_{i,j}$) is zero and it is thus possible to replace this element by the pointer to the column following j in set P_i.

We obtained a further improvement on the original Hungarian algorithm by modifying the choice of the initial solution, as described in Step 3 of the proposed algorithm.

Program

Fortran IV subroutine ASSCT, based on the algorithm previously presented, is completely self-contained and communication to it is made solely through the

Table I. Cost Range 1-100

	A		B		C	
n	Average	Maximum	Average	Maximum	Average	Maximum
50	0.16	0.20	0.21	0.31	0.23	0.33
100	0.54	0.66	0.93	1.15	1.10	1.27
150	0.96	1.10	2.10	2.81	2.79	3.42
200	1.40	1.68	4.37	5.27	5.38	7.24

Table II. Cost Range 1-1000

	A		B		C	
n	Average	Maximum	Average	Maximum	Average	Maximum
50	0.41	0.51	0.56	0.78	0.60	0.89
100	2.15	2.61	2.97	3.92	3.34	4.24
150	4.55	6.28	6.60	8.37	7.24	8.42
200	6.70	8.01	10.36	12.26	11.83	13.15

Table III. Cost Range 1-10,000

	A		B		C	
n	Average	Maximum	Average	Maximum	Average	Maximum
50	0.48	0.61	0.66	0.81	0.72	0.96
100	3.71	4.60	5.78	7.49	6.80	9.55
150	11.74	15.57	18.72	21.96	21.35	23.94
200	19.85	25.40	32.69	43.52	42.63	48.16

parameter list. The subroutine is called by means of the statement:

CALL ASSCT (N, A, C, T).

All the parameters are integer and their meanings are the following:

Input: N = number of rows and columns of the cost matrix;
A = cost matrix.
Output: C = assignment vector;
T = minimum assignment cost.

After execution of subroutine ASSCT, the values of the elements of the cost matrix are changed. Vector C must be dimensioned at least at N; matrix A at least at (N, N + 1). As presently dimensioned, the size limitation for ASSCT is N $\leq$ 200.

Computational Results

Subroutine ASSCT was tested in a CDC 6600 with over 1500 random problems of varying sizes. No breakdown in the method occurred.

To evaluate the efficiency of the proposed algorithm, the computing times of subroutine ASSCT were compared with those of the most efficient algorithms for solution of the assignment problem for dense matrices [1, 3].

Table IV. Sparse Matrices, Cost Range 1–100, $n = 200$

Number of coefficients	1500	2250	3000	3750	4500
AP-AB	0.97	1.12	1.48	1.61	1.68
PD-AAL	1.63	1.14	1.89	1.29	1.80
SUPERT-2	1.26	1.57	1.98	2.17	2.53
ASSCT	15.29	5.80	6.49	2.62	2.17

Tables I, II, and III show the computing times corresponding to the algorithms:

A: Subroutine ASSCT.

B: Hungarian algorithm as presented in [5] and coded in Fortran IV by the authors.

C: Algorithm presented in [3] (the Fortran IV program, coded by Bourgeois and Lassalle, is taken from the CDC library of CERN (Geneva, Switzerland).

In Tables I, II, and III the values of the cost matrix were generated as uniformly random integers in the ranges, respectively, 1–100, 1–1000, and 1–10,000. All codes were run on a CDC 6600. For each cost range, each algorithm, and each value of n, 20 different problems were solved, and the average and maximum computing times, expressed in seconds, are given.

The tables show that subroutine ASSCT is always superior to the other codes, mainly for large values of n and for small ranges of costs. In addition, for all codes the computing times get worse with the increase in size of coefficients; in fact, when the size increases, the number of zero elements of the current cost matrix decreases.

In order to evaluate the effect of sparseness on the performance of the proposed algorithm, the same test problems considered by Barr, Glover, and Klingman in [2] were solved on the same machine (a CDC-6600). Subroutine ASSCT was compared with the most efficient codes for the solution of sparse assignment problems, viz., codes AP-AB, PD-AAL, and SUPERT-2 presented, respectively, in [2], [6], and [1]; the corresponding computing times, expressed in seconds, are given in Table IV.[1]

Table IV shows that the codes designed to solve sparse assignment problems are superior to subroutine ASSCT for very sparse matrices because this subroutine does not include any mechanism for taking advantage of sparsity. However, the trends of the computing times indicate that for fairly dense matrices the performance of the proposed algorithm greatly increases with respect to those of the other codes.

Further details of the algorithm and extensive computational results are given in [4].

REFERENCES

1. Barr, R.S. Streamling primal simplex transportation codes. Res. Rep., Center for Cybernetic Studies, U. of Texas, Austin, Tex. To appear.
2. Barr, R.S., Glover, F., and Klingman, D. The alternating basis algorithm for assignment problems. *Math. Programming 13* (1977), 1–13.

[1] The times in Table IV were communicated to the authors by an anonymous referee who has given us permission to publish them.

3. Bourgeois, F., and Lassalle, J.C. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Comm. ACM 14*, 12 (Dec. 1971), 802-804.
4. Carpaneto, G., and Toth, P. An efficient algorithm for the assignment problem. Tech. Rep. 39, Istituto di Automatica, U. of Bologna, Bologna, Italy, 1976.
5. Christofides, N. *Graph Theory. An Algorithmic Approach.* Academic Press, London, 1975.
6. Hatch, R.S. Optimization strategies for large scale assignment and transportation type problems. ORSA/TIMS Conf., San Juan, Puerto Rico, 1974.

ALGORITHM

```
      SUBROUTINE ASSCT ( N, A, C, T )
      INTEGER A(2ØØ,2Ø1), C(2ØØ), CH(2ØØ), LC(2ØØ), LR(2ØØ),
     *        LZ(2ØØ), NZ(2ØØ), RH(2Ø1), SLC(2ØØ), SLR(2ØØ),
     *        U(2Ø1)
      INTEGER H, Q, R, S, T
      EQUIVALENCE (LZ,RH), (NZ,CH)
C
C THIS SUBROUTINE SOLVES THE SQUARE ASSIGNMENT PROBLEM
C THE MEANING OF THE INPUT PARAMETERS IS
C N = NUMBER OF ROWS AND COLUMNS OF THE COST MATRIX, WITH
C     THE CURRENT DIMENSIONS THE MAXIMUM VALUE OF N IS 2ØØ
C A(I,J) = ELEMENT IN ROW I AND COLUMN J OF THE COST MATRIX
C ( AT THE END OF COMPUTATION THE ELEMENTS OF A ARE CHANGED)
C THE MEANING OF THE OUTPUT PARAMETERS IS
C C(J) = ROW ASSIGNED TO COLUMN J (J=1,N)
C T = COST OF THE OPTIMAL ASSIGNMENT
C ALL PARAMETERS ARE INTEGER
C THE MEANING OF THE LOCAL VARIABLES IS
C A(I,J) = ELEMENT OF THE COST MATRIX IF A(I,J) IS POSITIVE,
C          COLUMN OF THE UNASSIGNED ZERO FOLLOWING IN ROW I
C          (I=1,N) THE UNASSIGNED ZERO OF COLUMN J (J=1,N)
C          IF A(I,J) IS NOT POSITIVE
C A(I,N+1) = COLUMN OF THE FIRST UNASSIGNED ZERO OF ROW I
C            (I=1,N)
C CH(I) = COLUMN OF THE NEXT UNEXPLORED AND UNASSIGNED ZERO
C         OF ROW I (I=1,N)
C LC(J) = LABEL OF COLUMN J (J=1,N)
C LR(I) = LABEL OF ROW I (I=1,N)
C LZ(I) = COLUMN OF THE LAST UNASSIGNED ZERO OF ROW I(I=1,N)
C NZ(I) = COLUMN OF THE NEXT UNASSIGNED ZERO OF ROW I(I=1,N)
C RH(I) = UNEXPLORED ROW FOLLOWING THE UNEXPLORED ROW I
C         (I=1,N)
C RH(N+1) = FIRST UNEXPLORED ROW
C SLC(K) = K-TH ELEMENT CONTAINED IN THE SET OF THE LABELLED
C          COLUMNS
C SLR(K) = K-TH ELEMENT CONTAINED IN THE SET OF THE LABELLED
C          ROWS
C U(I) = UNASSIGNED ROW FOLLOWING THE UNASSIGNED ROW I
C        (I=1,N)
C U(N+1) = FIRST UNASSIGNED ROW
C
C THE VECTORS C,CH,LC,LR,LZ,NZ,SLC,SLR MUST BE DIMENSIONED
C AT LEAST AT (N), THE VECTORS RH,U AT  LEAST AT (N+1),
C THE MATRIX A AT LEAST AT (N,N+1)
C
C INITIALIZATION
      MAXNUM = 1Ø**14
      NP1 = N+1
      DO 1Ø J=1,N
        C(J) = Ø
        LZ(J) = Ø
        NZ(J) = Ø
```

```
      U(J) = 0
   10 CONTINUE
      U(NP1) = 0
      T = 0
C REDUCTION OF THE INITIAL COST MATRIX
      DO 40 J=1,N
        S = A(1,J)
        DO 20 L=2,N
          IF ( A(L,J) .LT. S ) S = A(L,J)
   20   CONTINUE
        T = T+S
        DO 30 I=1,N
          A(I,J) = A(I,J)-S
   30   CONTINUE
   40 CONTINUE
      DO 70 I=1,N
        Q = A(I,1)
        DO 50 L=2,N
          IF ( A(I,L) .LT. Q ) Q = A(I,L)
   50   CONTINUE
        T = T+Q
        L = NP1
        DO 60 J=1,N
          A(I,J) = A(I,J)-Q
          IF ( A(I,J) .NE. 0 ) GO TO 60
          A(I,L) = -J
          L = J
   60   CONTINUE
   70 CONTINUE
C CHOICE OF THE INITIAL SOLUTION
      K = NP1
      DO 140 I=1,N
        LJ = NP1
        J = -A(I,NP1)
   80   IF ( C(J) .EQ. 0 ) GO TO 130
        LJ = J
        J = -A(I,J)
        IF ( J .NE. 0 ) GO TO 80
        LJ = NP1
        J = -A(I,NP1)
   90   R = C(J)
        LM = LZ(R)
        M = NZ(R)
  100   IF ( M .EQ. 0 ) GO TO 110
        IF ( C(M) .EQ. 0 ) GO TO 120
        LM = M
        M = -A(R,M)
      GO TO 100
  110   LJ = J
        J = -A(I,J)
        IF ( J .NE. 0 ) GO TO 90
        U(K) = I
        K = I
        GO TO 140
  120   NZ(R) = -A(R,M)
        LZ(R) = J
        A(R,LM) = -J
        A(R,J) = A(R,M)
        A(R,M) = 0
        C(M) = R
  130   C(J) = I
        A(I,LJ) = A(I,J)
        NZ(I) = -A(I,J)
```

```
        LZ(I) = LJ
        A(I,J) = 0
  140 CONTINUE
C RESEARCH OF A NEW ASSIGNMENT
  150 IF ( U(NP1) .EQ. 0 ) RETURN
      DO 160 I=1,N
        CH(I) = 0
        LC(I) = 0
        LR(I) = 0
        RH(I) = 0
  160 CONTINUE
      RH(NP1) = -1
      KSLC = 0
      KSLR = 1
      R = U(NP1)
      LR(R) = -1
      SLR(1) = R
      IF ( A(R,NP1) .EQ. 0 ) GO TO 220
  170 L = -A(R,NP1)
      IF ( A(R,L) .EQ. 0 ) GO TO 180
      IF ( RH(R) .NE. 0 ) GO TO 180
      RH(R) = RH(NP1)
      CH(R) = -A(R,L)
      RH(NP1) = R
  180 IF ( LC(L) .EQ. 0 ) GO TO 200
      IF ( RH(R) .EQ. 0 ) GO TO 210
  190 L = CH(R)
      CH(R) = -A(R,L)
      IF ( A(R,L) .NE. 0 ) GO TO 180
      RH(NP1) = RH(R)
      RH(R) = 0
      GO TO 180
  200 LC(L) = R
      IF ( C(L) .EQ. 0 ) GO TO 360
      KSLC = KSLC+1
      SLC(KSLC) = L
      R = C(L)
      LR(R) = L
      KSLR = KSLR+1
      SLR(KSLR) = R
      IF ( A(R,NP1) .NE. 0 ) GO TO 170
  210 CONTINUE
      IF ( RH(NP1) .GT. 0 ) GO TO 350
C REDUCTION OF THE CURRENT COST MATRIX
  220 H = MAXNUM
      DO 240 J=1,N
        IF ( LC(J) .NE. 0 ) GO TO 240
        DO 230 K=1,KSLR
          I = SLR(K)
          IF ( A(I,J) .LT. H ) H = A(I,J)
  230   CONTINUE
  240 CONTINUE
      T = T+H
      DO 290 J=1,N
        IF ( LC(J) .NE. 0 ) GO TO 290
        DO 280 K=1,KSLR
          I = SLR(K)
          A(I,J) = A(I,J)-H
          IF ( A(I,J) .NE. 0 ) GO TO 280
          IF ( RH(I) .NE. 0 ) GO TO 250
          RH(I) = RH(NP1)
          CH(I) = J
          RH(NP1) = I
  250     L = NP1
```

```
  26Ø     NL = -A(I,L)
          IF ( NL .EQ. Ø ) GO TO 27Ø
          L = NL
          GO TO 26Ø
  27Ø     A(I,L) = -J
  28Ø   CONTINUE
  29Ø CONTINUE
      IF ( KSLC .EQ. Ø ) GO TO 35Ø
      DO 34Ø I=1,N
        IF ( LR(I) .NE. Ø ) GO TO 34Ø
        DO 33Ø K=1,KSLC
          J = SLC(K)
          IF ( A(I,J) .GT. Ø ) GO TO 32Ø
          L = NP1
  3ØØ     NL = - A(I,L)
          IF ( NL .EQ. J ) GO TO 31Ø
          L = NL
          GO TO 3ØØ
  31Ø     A(I,L) = A(I,J)
          A(I,J) = H
          GO TO 33Ø
  32Ø     A(I,J) = A(I,J)+H
  33Ø   CONTINUE
  34Ø CONTINUE
  35Ø R = RH(NP1)
      GO TO 19Ø
C ASSIGNMENT OF A NEW ROW
  36Ø C(L) = R
      M = NP1
  37Ø NM = -A(R,M)
      IF ( NM .EQ. L ) GO TO 38Ø
      M = NM
      GO TO 37Ø
  38Ø A(R,M) = A(R,L)
      A(R,L) = Ø
      IF ( LR(R) .LT. Ø ) GO TO 39Ø
      L = LR(R)
      A(R,L) = A(R,NP1)
      A(R,NP1) = -L
      R = LC(L)
      GO TO 36Ø
  39Ø U(NP1) = U(R)
      U(R) = Ø
      GO TO 15Ø
      END
```