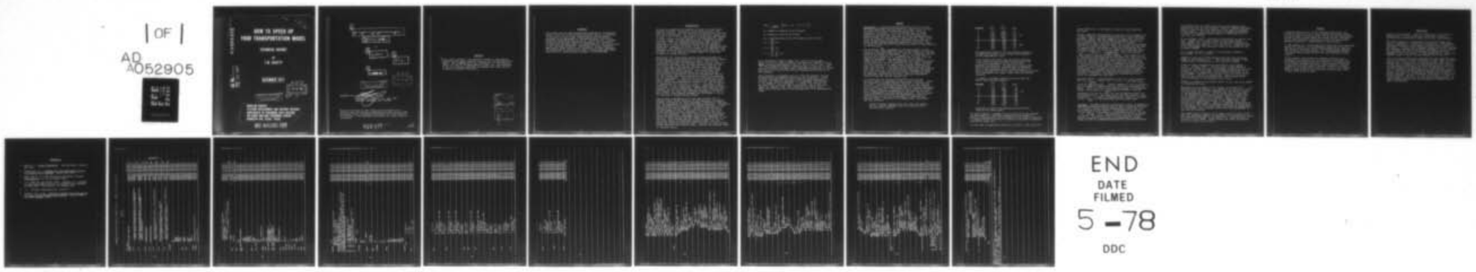
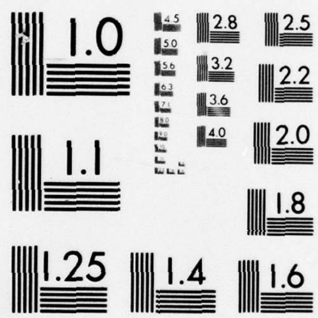
AD-A052 905 AIR FORCE MILITARY PERSONNEL CENTER RANDOLPH AFB TEX F/G 12/2
HOW TO SPEED UP YOUR TRANSPORTATION MODEL.(U)
DEC 77 T M BEATTY

UNCLASSIFIED NL

| OF |
AD
A052905



END
DATE
FILMED
5-78
DDC


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

HOW TO SPEED UP YOUR TRANSPORTATION MODEL

TECHNICAL REPORT

BY

T.M. BEATTY

DECEMBER 1977

MODELING BRANCH
SYSTEMS DEVELOPMENT AND SUPPORT DIVISION
DIRECTORATE OF PERSONNEL DATA SYSTEMS
AIR FORCE MILITARY PERSONNEL CENTER
RANDOLPH AFB, TEXAS 78148

HOW TO SPEED UP YOUR TRANSPORTATION MODEL.

TECHNICAL REP[redacted]T

BY

T. M. BEATTY

22 p.

DEC[redacted]77

APPROVED BY:

FREDERICK S. PHILLIPS, SR., COL, USAF
DIRECTOR, PERSONNEL DATA SYSTEMS

While the contents of this report are considered to be correct, they are subject to modification upon further study. This report does not promulgate official Air Force policies or positions. The technical conclusions are solely those of the author.

408 077

alt

ABSTRACT

This report presents a new method to compute a more nearly optimal initial basic feasible solution for the Transportation Model. The intergration of two techniques; (1) The Decision Index (DI) and (2) An Admissability Index (AI), have resulted in 50 to 75 percent reductions in computer run time required to derive an optimal solution.

FOREWORD

This report and the BEST program were prepared by the Modeling Branch of the Air Force Military Personnel Center in response to the need to solve large transportation and assignment problems in the management of the approximately one million personnel employed by the Air Force. The work of Dr. Joe Ward of the Air Force Human Resources Laboratory, who originally authored The Decision Index, is acknowledged for its contribution to the technique presented. Additionally, Gloria Jeaneatte McWilliams, whose thesis at the University of Texas indicated the efficiency of the Decision Index in developing approximate solutions for transportation problems, is acknowledged.

INTRODUCTION

To solve a transportation problem you need an initial basic feasible solution. The better (more nearly optimal) the starting solution, the more rapid the transportation algorithm will go to convergence. As so succintly stated by Hadley (#1) in 1962, "----- , It is worth while to spend some time finding a 'good' initial solution because it can considerably reduce the total number of iterations required to reach an optimal solution." For any transportation problem there exists a continuum of initial solutions ranging from the Northwest Corner Rule (probably the worst) to one of the numerous optimal solutions. The objective of this report is to describe a new technique which derives initial solutions which are very near optimum, i.e. will require a reduced number of iterations to reach an optimal solution.

McWilliams (#2) investigated several promising methods for developing approximate solutions to the transportation problem in her thesis at the University of Texas. In brief her conclusions dismissed the Northwest Corner Rule as a serious alternative and found that the Ward Decision Index and the C^q, in both static and dynamic versions, were superior to other algorithms currently in use, e.g. row/column minima, cost minima, and Vogel's method. She further noted that Ward's method and C^q method differ only by additive and multiplicative constants in the static version when M=N and for this reason the initial feasible solutions are identical for the two methods. (But this is only true when M=N in the static versions). It should be noted that McWilliams'thesis used a slight misinterpretation of Ward's DI. The original DI computation (#3) involves C.j and Ci., the sums of the j^{th} column and i^{th} row of the cost tableau. These sums consider the quotas in their computation. McWilliams'version of the DI computes the row and column sums without quota consideration. Ward suggest (#4) that this incorrect computation of the DI may not have significantly degraded the usefulness of the DI for starting a transportation problem.

The original promulgation of DI was made by Ward (#3) in seeking a near optimal solution to the sequential assignment problem confronting Air Force assignment counselors in dealing with job assignments for non-prior service personnel. Counselors were required to make sequential assignment decisions in the absence of perfect information. As each airman arrived at the assignment station, he was given his initial Air Force career assignment without knowledge of the vast labor pool yet to be considered. The result was intuitively sub-optimal where each sequential assignment subjectively maximized the payoff criteria against whatever happen to be remaining in the job pool. Ward conjectured that given estimates of the row and column means, of admittedly imprecise data, that the sequential assignment problem could be solved in a near optimum manner employing the Decision Index. Wherein the Decision Index is computed thusly:

$$DI_{ij} = \frac{1}{N(M-1)} \left(MC_{ij} - C_{i.} - C_{.j} + C_{..} \right)$$

M = Number of persons to be assigned,

N = Number of jobs to be filled,

C_{ij} = Productivity of the i^{th} person on the j^{th} job,

$$C_{i.} = \sum_{j=1}^{N} C_{ij}$$

$$C_{.j} = \sum_{i=1}^{M} C_{ij}$$

$$C_{..} = \sum_{i=1}^{M} \sum_{j=1}^{N} C_{ij}$$

As is discussed in Ward's paper (#3), this DI is the mean value of the cost used to compute the objective function for all $\frac{(M-1)!}{(M-N)!}$ possible assignments of the i^{th} man to the j^{th} job. (Note: This seemingly powerful technique has, to the best of this authors knowledge, not been widely implemented either as an assignment tool or in conjunction with the Transportation Model as is now proposed).

Early attempts to implement DIs as a starting solution for the transportation/assignment problem were not successful. These failures resulted from the type of problem being examined in which M, the number of personnel to be assigned, far exceeded N, the number of jobs available. In these first efforts, it was found that a DI start was little better, and in fact more costly in computer time, than a modified row minima start. These early failures led to the consideration of an admissability index.

METHOD

Background: A straight-forward application of DI's to the personnel assignment problem where $M > N$ resulted in solutions not much better than the modified row minima. This, in hindsight, is obvious. If all contenders were at least marginally qualified for all jobs in the case where 2000 persons sought 300 jobs, then the first 300 contenders would enter the solution and remaining 1700 would not. This result could be marginally better than a modified row minima but would consume greater computer resources in deriving. Thus the need for some method of selecting the order of admissability to the solution.

The statistic, $\frac{\mu_{c_i}}{\sigma_{c_i}}$, possessed an intuitive appeal and has proven a useful admissability index in application to personnel assignment problems investigated. Here μ_{c_i} represents the mean value of the costs for the i^{th} person for all real jobs (in application a shadow job is used which has a quota of M, the number of personnel to be assigned and is not used in computing μ_{c_i} and σ_{c_i}). The σ_{c_i} represents the standard deviation of the costs for the i^{th} person for all real jobs. The mean cost for a person is an overt index of which personnel will enter the solution. In general those persons with smallest mean costs will be in the solution and those with the greatest mean cost will not. Inclusion of σ_{c_i} as a moderator provides for earlier introduction to the solution of those who might be otherwise unattractive but have a few good assignment possibilities and limited alternatives.

In calculating the statistics μ_{c_i} and σ_{c_i} as well as the Decision Index, an important deviation was introduced. BIG M was observed to possess dramatic influence in the starting solution. (BIG M originally generated in the cost tableau is two orders of magnitude greater than the largest admissable real cost.) Early on, it was observed that using the original BIG M caused marginal assignments in the starting solution which subsequently left the optimal solution. This of course results from the use of σ_{c_i} in the admissability index and the follow-on use of the column means in computing the Decision Index. Reference to an example best illustrates why this occurred.

> Given 10 person competing for five jobs with admissability indices computed using the original BIG M, the following results:

Person#	μ_{c_i}	σ_{c_i}	AI	
1	200	1000	.20	
2	210	1050	.20	
3	1100	4400	.25	IN
4	350	1166	.30	
5	300	750	.40	
6	400	800	.50	
7	450	818	.55	
8	460	767	.60	OUT
9	500	769	.65	
10	550	786	.70	

The moderating effect of σ_{c_i} has effected exactly as intended and persons #3 and 4 were driven upward on the admissability list.

Given further that there was one highly difficult job in the quota bank on which most personnel had a BIG M cost and person #3 had a high cost, marginally acceptable. The resulting large column mean for the difficult job when used in computing the DI (see discussion below) caused person #3 to have an attractive DI and to enter the starting solution on an assignment that later must leave the optimal solution.

If however a suitable substitution were made for BIG M, the following results:

Person#	μ_{c_i}	σ_{c_i}	AI	
1	200	1000	.20	
2	210	1050	.20	
5	300	750	.40	IN
6	400	800	.50	
4	340	654	.52	
7	450	818	.55	
8	460	767	.60	
9	500	769	.65	
10	550	786	.70	
3	800	1111	.72	

Note that #3 leaves the starting solution and #4 remains but enters later.

The above example, intended only as illustrative and probably not derivable with real data, demonstrates an anomaly of Decision Indexing noted by and currently under study by Ward(#5) A rigorous derivation of how to handle this problem is expected from Ward's study (#5).

In this work, an empirical derivation of BIG M = 1025 with real costs $\leq$ 998 and $>$ 0 was found to result in good starting solutions.

Another important consideration is the cost of the shadow job. In early usage of transportation modeling to solve personnel problems, shadow jobs were costed at a fixed cost, less than BIG M but greater than the greatest admissable actual cost. The rationale was, of course, to assign all eligible contenders to actual jobs and assure that no BIG M assignments were in the optimal solution. When this shadow cost was used in computing DIs, the shadow DIs did not discriminate over contenders since all DIs in the shadow job column were equal to zero, i.e. $\left[C_{shadow} - \frac{(C_{shadow})*M}{M}\right]$. Again the statistic $\frac{\mu_{ci}}{\sigma_{ci}}$ (the AI) has an intuitive appeal. Since there will be no contention concerning people with small AIs, they will be in the solution and likewise, the large AIs will not be in the solution, the cost index for shadow jobs must discriminate about the threshold just prior to exhausting the quota bank. At this point there are perhaps contenders who should not be included in the initial start but should be assigned to the shadow job while the search continues for more cost-effective assignments. In several test runs, the AI used as shadow cost has demonstrated the required *discrimination about the* threshold. Therefore, from empirical considerations only, the AI is proposed for this role.

Decision Indices, as promulgated by Ward (#3), were computed with $DI_{ij} = \frac{1}{N(M-1)}$ $(MC_{ij} - C_{i.} - C_{.j} + C_{..})$ (see page 5 above for amplification). Several elements of this calculation are not needed in computing DI for starting a Transportation Model. Since the initial assignments are made in a sequential manner, the division by $\frac{1}{N(M-1)}$ may be deleted. Also the subtraction of the row total, $C_{i\cdot}$, may be removed as well as the addition of $C_{..}$, the total sum of the cost tableau. This leaves $MC_{ij} - C_{.j}$ which is computationally easier to implement as $C_{ij} - \frac{C_{.j}}{M}$, or the deviation from the column mean.

Software: The BEST program is presented in ALGOL in Appendix A. The various computations executed in BEST to produce the basis for Langley's Primal Simplex Transportation Model (#6) will be documented in this section. As appropriate, reference will be made to line numbers in the program listing. In line 300 through 4300, declarations of files and variables are found. Only three of the files are pertinent to the general application of starting a transportation problem. These are INP1 or CAREERS/DATA/TRANS which provides the number of destinations, the number of sources, the demand at each destination, and the cost tableau. ADVAN or CAREERS/DATA/BASIS which is used to write the basis for later input to Langley's Primal Simplex. The third file DIJS or CAREERS/DATA/REVERSE is used as a transient storage medium in computing the DI's from the cost tableau.

At line number 4400, the size of the problem is read in from INP1. These data are then printed in the output report. Line numbers 5000 through 10200 provide more declarations. Three defines are shown at 10300 to 10600. These are used to bit pack/unpack the variable A which is tag sorted in the procedure SORTER.

An in place tag sort is shown in the procedure SORTER at line number 10700 to 23900.

Demands at each of the destinations are read into the array NJOBS at line number 24100. Then the total number of jobs to be allocated is accumulated in the variable, TOTJOBS.

Costs are read in sequentially from INP1 at line numbers 26500 through 30800. In this block of code the BIG M adjustment is made and then the necessary computations for computing μ_{c_i} and σ_{c_i} are made. From these calculations the AI is computed, bit packed with the row index, and stored in the array ROWTOT. The AI is also stored in cost (Ø) to be used later as the cost for the shadow job. The adjusted costs are then written to the transient file, DIJS.

SORTER is invoked at line number 30900 to sort the array ROWTOT on ascending AI value.

Beginning at line number 32300 through line number 38700, the adjusted costs are randomly read (based on the sorted AI) from the file DIJS, the DI's are computed, and the basis is stored in the appropriate arrays. Of particular note in this area is the decrementing and testing on the variable TOTJOBS which is the number of jobs to be allocated. When all jobs have been allocated, this area is left and a wrapup area is used to assign all remaining personnel to the Shadow Job. Also note at line number 34000, the DI is computed dynamically as a function of the number of personnel remaining to be assigned. To accomplish this, the column totals are adjusted appropriately at line number 38400 through 38500 after each assignment in the starting solution.

The wrapup area is shown at line number 40200 through 45200. Here the necessary elements of the basis are calculated to assign all remaining personnel to the Shadow Job. Then the basis is written to file, ADVAN. Other reports and timing statistics are also written at this point.

RESULTS

A typical application of AI/DI starting technique is its production usage in the Air Force CAREERS JOB FINDER System. In this system, first-term airman who do not have reenlistment quotas in their current AFSCs are costed against jobs which do not have sufficient applicants to meet Air Force career manpower objectives. This produces a cost tableau.

In the problem run, 1908 prospective reenlistees were optimized against 939 jobs in 79 job categories producing an implicit cost array of 1,791,612 cells. The explicit array contained 150,732 cells with 100% density, i.e. inadmissable person-job-matches were costed at BIG M but were defined to Primal Simplex.

Primal Simplex using Langley's internal artifical start converged to a solution in 1,643 seconds of processor time. The problem was then solved using the AI/DI to generate the initial basic feasible solution. This required 87 seconds to generate the basis and 306 seconds in Langley's Primal Simplex to go to an optimal solution. That is, the problem takes 4 times longer to run with the artifical start than it does with the AI/DI start.

CONCLUSION

Hadley's truism stands. Ward's contribution to the arena of optimization is again recognized. McWilliam's initial investigation has served to stimulate further investigation.

The AI/DI technique is implemented with the prototype code shown at Appendix A. Results are satisfactory. Large person-job-match problems are now taken in stride without regard to ADP impact. In fact, these problems consume so little time that the stringent limitations on memory utilization can be relaxed. This relieves the user of the onerous problems of memory management which may prevail in some installations. There are, however, areas in which further research may be fruitful.

Given that many optimization problems deal with imprecise data, even to the extent of being subjective in some cases, how near optimal must a solution be, to be good enough? Does the AI coupled with a dynamic DI get close enough for most, some, or any optimization problems? Can tests be applied to the AI/DI objective function to determine if it is good enough, i.e. 1, 2,3, or perhaps 22 standard deviations from the mean objective value? Can the AI/DI be applied to sparse matrix problems? Network problems? Is the documented computation of the AI the correct method? The best? or only one of many? Is there a better index than the AI to use for costing Shadow Jobs? Is there adequate payoff from the introduction of Shadow Jobs to make it worthwhile?

REFERENCES

1. Hadley, G., Linear Programming. Addison-Wesley, Reading, MA, 1962.

2. McWilliams, G.J., A Method for the Approximate Solution of Transportation Problems. Austin, TX, 1970.

3. Ward, Joe H. Jr., "The Counseling Assignment Problem," Psychometrika, Vol 23, #1, March 1958.

4. ---. Memo for the Record, Subj: Comments on " A Method for the Approximate Solution to Transportation Problems" by McWilliams. San Antonio, TX, October 1974.

5. ---. Informal Communications, 1974-1977.

6. Langley, R.W. et al: Efficient Computational Devices for the Capacitated Transportation Problem. United States Air Force Academy, 1975.

APPENDIX A

```
BURROUGHS B6700 ALGOL COMPILER, VERSION 2.8.060,     MONDAY, 12/05/77, 09:28 AM.

                                  B E S T
                                  = = = =

$SET MERGE                                                                      000:0000:0
  $ SET LIST FORMAT LINEINFO                                          00000100  000:0000:0
BEGIN                                                                 00000200  000:0000:0
                                                                  B.0000 IS SEGMENT 00003
FILE                                                               1  00000300  003:0000:1
          DIJS(KIND=DISKPACK,PACKNAME="DADATA.",TITLE="CAREERS/DATA/R"  00000400  003:0000:1
           "EVERSE.",MYUSE=IO,UNITS=WORDS,AREASIZE=100,AREAS=78);      00000500  003:0000:1
                                                                     DATA IS 000C LONG
FILE                                                                  00000600  003:0000:1
          OTP(KIND=DISKPACK,PACKNAME="DADATA.",TITLE="CAREERS/DATA/DI"  00000700  003:0000:1
            "OFFR.", MAXRECSIZE=2,BLOCKSIZE=600,UNITS=WORDS,           00000800  003:0000:1
          AREASIZE=1200,AREAS=10);                                    00000900  003:0000:1
                                                                     DATA IS 000D LONG
FILE                                                                  00001000  003:0000:1
        CARD(KIND=READER);                                            00001100  003:0000:1
                                                                     DATA IS 0005 LONG
FILE                                                                  00001200  003:0000:1
        PRIN(KIND=PRINTER);                                           00001300  003:0000:1
                                                                     DATA IS 0005 LONG
FILE                                                                  00001400  003:0000:1
          INP1(KIND=DISKPACK,PACKNAME="DADATA.",TITLE="CAREERS/DATA/T"  00001500  003:0000:1
           "RANS.",FILETYPE=7);                                       00001600  003:0000:1
                                                                     DATA IS 000A LONG
FILE                                                                  00001700  003:0000:1
        INP2(KIND=DISKPACK,PACKNAME="DADATA.",MYUSE=IO,               00001800  003:0000:1
         TITLE="CAREERS/DATA/INVERSE.",FILETYPE=7);                   00001900  003:0000:1
                                                                     DATA IS 000B LONG
FILE                                                                  00002000  003:0000:1
          ADVAN(KIND=DISKPACK,PACKNAME="DADATA.",TITLE="CAREERS/DATA/"  00002100  003:0000:1
           "BASIS.",AREASIZE=1,AREAS=1,FILETYPE=7);                   00002200  003:0000:1
                                                                     DATA IS 000B LONG
INTEGER                                                               00002300  003:0000:1
        I,                                                            00002400  003:0000:1
        J,                                                            00002500  003:0000:1
        K,                                                            00002600  003:0000:1
        NROWS,                                                        00002700  003:0000:1
        NCOLS,                                                        00002800  003:0000:1
        COUNT,                                                        00002900  003:0000:1
        INIT,                                                         00003000  003:0000:1
        ISAV;                                                         00003100  003:0000:1
REAL                                                                  00003200  003:0000:1
        NASSN,                                                        00003300  003:0000:1
        DEC;                                                          00003400  003:0000:1
REAL                                                                  00003500  003:0000:1
        BIGM,                                                         00003600  003:0000:1
        SUM,                                                          00003700  003:0000:1
        BUF,                                                          00003800  003:0000:1
        OBJVAL;                                                       00003900  003:0000:1
FORMAT                                                                00004000  003:0000:1
        F5(X5,I6,X4,I6,X4,I10,X4,I10);                                00004100  003:0000:1
FORMAT                                                                00004200  003:0000:1
        F6(X6,I3,X8,I3);                                              00004300  003:0000:1
```

```
                READ(INP1[0],<2I6>,NCOLS,NROWS);                     00004400  003:0000:1
                DIJS.MAXRECSIZE := NCOLS;                            00004500  003:000A:2
                NASSN := NROWS;                                      00004600  003:000C:2
                WRITE(PRIN,<"     NR. AFSC,S(COLS)= ",I6,            00004700  003:000D:1
                  "     NR. AIRMEN(ROWS )= ",I6>,NCOLS,NROWS);       00004800  003:000F:0
                                                                        DATA IS 0013 LONG
                BEGIN                                                00004900  003:0015:2
      REAL                                                        2  00005000  003:0015:2
              TOTJOBS;                                               00005100  003:0015:2
                                                                   B.0001 IS SEGMENT 00008
      REAL                                                           00005200  008:0000:1
              LL,                                                    00005300  008:0000:1
              MM;                                                    00005400  008:0000:1
      REAL ARRAY POINT[O:NCOLS];                                     00005500  008:0000:1
      REAL                                                           00005600  008:0003:1
              TIME1,                                                 00005700  008:0003:1
              TIME2,                                                 00005800  008:0003:1
              TIME3,                                                 00005900  008:0003:1
              TIME4,                                                 00006000  008:0003:1
              ROWSUM,                                                00006100  008:0003:1
              COLTEM;                                                00006200  008:0003:1
      REAL                                                           00006300  008:0003:1
              TIMER,                                                 00006400  008:0003:1
              JOBS;                                                  00006500  008:0003:1
      REAL ARRAY ID,                                                 00006600  008:0003:1
              IU,                                                    00006700  008:0003:1
              IR,                                                    00006800  008:0003:1
              IP,                                                    00006900  008:0003:1
              IDA[C:NROWS+NCOLS];                                    00007000  008:0003:1
      LABEL                                                          00007100  008:0008:4
              START;                                                 00007200  008:0008:4
      LABEL                                                          00007300  008:0008:4
              WRAP;                                                  00007400  008:0008:4
      REAL ARRAY NJOBS[O:NROWS-1],                                   00007500  008:0008:4
              QUANT[O:NROWS],                                        00007600  008:000C:0
              OFFERS[O:NROWS, 1:3];                                  00007700  008:000F:0
      REAL ARRAY EXCESS[O:NCOLS];                                    00007800  008:0012:3
      REAL ARRAY ROWTOT[O:NROWS-1],                                  00007900  008:0015:3
              COST,                                                  00008000  008:0018:5
              COLSIG,                                                00008100  008:0018:5
              COL,                                                   00008200  008:0018:5
              COLSQ[O:NCOLS-1];                                      00008300  008:0018:5
      REAL                                                           00008400  008:001D:4
              DI;                                                    00008500  008:001D:4
      REAL                                                           00008600  008:001D:4
              KOSTJ;                                                 00008700  008:001D:4
      INTEGER                                                        00008800  008:001D:4
              UP;                                                    00008900  008:001D:4
      INTEGER                                                        00009000  008:001D:4
              Z;                                                     00009100  008:001D:4
      LABEL                                                          00009200  008:001D:4
              OUT;                                                   00009300  008:001D:4
      INTEGER                                                        00009400  008:001D:4
              KUP,                                                   00009500  008:001D:4
              KLO;                                                   00009600  008:001D:4
      REAL                                                           00009700  008:001D:4
              ROWSQ;                                                 00009800  008:001D:4
      REAL                                                           00009900  008:001D:4
              ROWTEM;                                                00010000  008:001D:4
      REAL                                                           00010100  008:001D:4
```

```
            DIK;                                                              00010200  008:001D:4
DEFINE                                                                        00010300  008:001D:4
         PACK(A,B)      = A := (A)&(B)[47:13:14] #,                           00010400  008:001D:4
         UNPACKDI(A)    = (A).[33:34] #,                                      00010500  008:001D:4
         UNPACKJ(A)     = (A).[47:14] #;                                      00010600  008:001D:4
PROCEDURE SORTER(A,IB,IC,FB,BC);                                              00010700  008:001D:4
% IN PLACE TAG SORT CACM ALGORITIHM #347                                      00010800  008:001D:4
% THIS SORT ROUTINE HANDLES A ONE DIMENSIONAL ARRAY.  IT IS MODIFIED          00010900  008:001D:4
% TO USE ONLY A SUBSET OF THE BITS OF THE ELEMENTS AS THE KEY.  IN            00011000  008:001D:4
% THIS MANNER THE ONE WORD MAY CARRY BOTH THE KEY AND 'TAG'.  THE             00011100  008:001D:4
% FORMAL PARAMETERS OF THE PROCEDURE ARE:                                     00011200  008:001D:4
%                                                                             00011300  008:001D:4
%     A ::= ONE DIMENSIONAL DATA ARRAY                                        00011400  008:001D:4
%    IB ::= LOW INDEX OF DATA                                                 00011500  008:001D:4
%    IC ::= HIGH INDEX OF DATA                                                00011600  008:001D:4
%    FB ::= FIRST BIT OF PARTIAL WORD OF SORT KEY                             00011700  008:001D:4
%    BC ::= NUMBER OF BITS IN THE KEY                                         00011800  008:001D:4
%                                                                             00011900  008:001D:4
            ARRAY                                                             00012000  008:001D:4
                        A[*];                                                 00012100  008:001D:4
INTEGER                                                                       00012200  008:001D:4
            IB,                                                               00012300  008:001D:4
            IC,                                                               00012400  008:001D:4
            FB,                                                               00012500  008:001D:4
            BC;                                                               00012600  008:001D:4
                    BEGIN                                                     00012700  008:001D:4
ARRAY                                                                         00012800  008:001D:4
                                                                    SORTER IS SEGMENT 00009
            IU[0:16],                                               3         00012900  009:0000:1
            IL[0:16];                                                         00013000  009:0000:1
REAL                                                                          00013100  009:0000:1
            T,                                                                00013200  009:0000:1
            TT;                                                               00013300  009:0000:1
INTEGER                                                                       00013400  009:0000:1
            I,                                                                00013500  009:0000:1
            J,                                                                00013600  009:0000:1
            K,                                                                00013700  009:0000:1
            L,                                                                00013800  009:0000:1
            M,                                                                00013900  009:0000:1
            IJ;                                                               00014000  009:0000:1
LABEL                                                                         00014100  009:0000:1
            L5,                                                               00014200  009:0000:1
            L10,                                                              00014300  009:0000:1
            L20,                                                              00014400  009:0000:1
            L30,                                                              00014500  009:0000:1
            L40,                                                              00014600  009:0000:1
            L50,                                                              00014700  009:0000:1
            L60,                                                              00014800  009:0000:1
            L70,                                                              00014900  009:0000:1
            L80,                                                              00015000  009:0000:1
            L90,                                                              00015100  009:0000:1
            L100,                                                             00015200  009:0000:1
            EXIT;                                                             00015300  009:0000:1
DEFINE                                                                        00015400  009:0000:1
         KEY            = [FB:BC] #;                                          00015500  009:0000:1
                       M := 1;                                                00015600  009:0000:1
                       I := IB;                                               00015700  009:0000:5
                       J := IC;                                               00015800  009:0001:5
L5:                                                                           00015900  009:0002:5
                       IF I GEQ J THEN                                        00016000  009:0002:5
```

```
          GO TO L70;                                  00016100   009:0003:4
L10:                                                  00016200   009:0004:1
       K := I;                                        00016300   009:0004:1
       IJ := (J+I)/2;                                 00016400   009:0005:1
       T := A[IJ];                                    00016500   009:0007:1
       IF A[I].KEY LEQ T.KEY THEN                     00016600   009:0008:3
          GO TO L20;                                  00016700   009:000C:3
       A[IJ] := A[I];                                 00016800   009:000D:0
       A[I] := T;                                     00016900   009:000F:4
       T := A[IJ];                                    00017000   009:0011:3
L20:                                                  00017100   009:0013:1
       L := J;                                        00017200   009:0013:1
       IF A[J].KEY GEQ T.KEY THEN                     00017300   009:0014:1
          GO TO L40;                                  00017400   009:0018:1
       A[IJ] := A[J];                                 00017500   009:0018:4
       A[J] := T;                                     00017600   009:001B:2
       T := A[IJ];                                    00017700   009:001D:1
       IF A[I].KEY LEQ T.KEY THEN                     00017800   009:001E:5
          GO TO L40;                                  00017900   009:0022:5
       A[IJ] := A[I];                                 00018000   009:0023:2
       A[I] := T;                                     00018100   009:0026:0
       T := A[IJ];                                    00018200   009:0027:5
       GO TO L40;                                     00018300   009:0029:3
L30:                                                  00018400   009:002A:0
       A[L] := A[K];                                  00018500   009:002A:0
       A[K] := TT;                                    00018600   009:002C:4
L40:                                                  00018700   009:002E:3
       L := L-1;                                      00018800   009:002E:3
       IF A[L].KEY GTR T.KEY THEN                     00018900   009:002F:5
          GO TO L40;                                  00019000   009:0033:3
       TT := A[L];                                    00019100   009:0034:0
L50:                                                  00019200   009:0035:4
       K := K+1;                                      00019300   009:0035:4
       IF A[K].KEY LSS T.KEY THEN                     00019400   009:0037:0
          GO TO L50;                                  00019500   009:003A:4
       IF K LEQ L THEN                                00019600   009:003B:1
          GO TO L30;                                  00019700   009:003C:0
       IF (L-I) LEQ (J-K) THEN                        00019800   009:003C:3
          GO TO L60;                                  00019900   009:003E:2
       IL[M] := I;                                    00020000   009:003E:5
       IU[M] := L;                                    00020100   009:0040:1
       I := K;                                        00020200   009:0041:3
       M := M+1;                                      00020300   009:0042:3
       GO TO L80;                                     00020400   009:0043:5
L60:                                                  00020500   009:0044:2
       IL[M] := K;                                    00020600   009:0044:2
       IU[M] := J;                                    00020700   009:0045:4
       J := L;                                        00020800   009:0047:0
       M := M+1;                                      00020900   009:0048:0
       GO TO L80;                                     00021000   009:0049:2
L70:                                                  00021100   009:0049:5
       M := M-1;                                      00021200   009:0049:5
       IF M=0 THEN                                    00021300   009:004B:1
          GO TO EXIT;                                 00021400   009:004B:3
       I := IL[M];                                    00021500   009:004C:0
       J := IU[M];                                    00021600   009:004D:2
L80:                                                  00021700   009:004E:4
       IF (J-I) GEQ 1 THEN                            00021800   009:004E:4
          GO TO L10;                                  00021900   009:004F:5
       IF I EQL IB THEN                               00022000   009:0050:2
```

```
                GO TO L5;                                   00022100  009:0051:1
             I := I-1;                                      00022200  009:0051:4
L90:                                                        00022300  009:0053:0
             I := I+1;                                      00022400  009:0053:0
             IF I EQL J THEN                                00022500  009:0054:2
                GO TO L70;                                  00022600  009:0054:5
             T := A[I+1];                                   00022700  009:0055:2
             IF A[I].KEY LEQ T.KEY THEN                     00022800  009:0057:2
                GO TO L90;                                  00022900  009:005B:2
             K := I;                                        00023000  009:005B:5
L100:                                                       00023100  009:005C:5
             A[K+1] := A[K];                                00023200  009:005C:5
             K := K-1;                                      00023300  009:005F:5
             IF T.KEY LSS A[K].KEY THEN                     00023400  009:0061:1
                GO TO L100;                                 00023500  009:0065:1
             A[K+1] := T;                                   00023600  009:0065:4
             GO TO L90;                                     00023700  009:0067:5
EXIT:                                                       00023800  009:0068:2
          END OF SORT ROUTINE;                              00023900  009:0068:2
                                                  SORTER(009) IS 0070 LONG
```

```
           BIGM := 99999;                                                 3   00024000  008:001D:4
           READ(INP1[1],NCOLS,NJOBS);                                         00024100  008:001E:3
           FOR I:=1 STEP 1 UNTIL NCOLS-1 DO                                   00024200  008:0026:2
              TOTJOBS := *+NJOBS[I];                                          00024300  008:002B:1
           KUP := NROWS-TOTJOBS;                                              00024400  008:002D:2
           NJOBS[0] := NROWS;                                                 00024600  008:002E:5
           WRITE(PRIN,/,"     AFSC     NR. JOBS");                            00024900  008:0030:0
           FOR I:=0 STEP 1 UNTIL NCOLS-1 DO                                   00024900  008:0039:2
              WRITE(PRIN,<X4,I6,X6,I6>,(I+1),NJOBS[I]);                       00025000  008:003E:1
           FOR I:=0 STEP 1 UNTIL NROWS+NCOLS DO                               00025100  008:0048:5
              ID[I] := IU[I] := IR[I] := IP[I] := 0;                          00025200  008:004D:5
           FOR I:=NROWS+1 STEP 1 UNTIL NROWS+NCOLS DO                         00025300  008:0052:3
              IP[I] := -9999;                                                 00025400  008:0057:3
           FOR I:=1 STEP 1 UNTIL NCOLS DO                                     00025500  008:0059:4
              IDA[I+NROWS] := NJOBS[I-1];                                     00025600  008:005E:1
           IDA[NCOLS+1] := NROWS;                                             00025800  008:0061:1
           FOR I:=0 STEP 1 UNTIL NROWS-1 DO                                   00026000  008:0062:5
              QUANT[I] := 1;                                                  00026100  008:0067:4
           INIT := 0;                                                         00026200  008:0069:2
           UP := NROWS-1;                                                     00026400  008:006A:0
START:                                                                        00026500  008:006B:2
           COL[0] := 0;                                                       00026600  008:006B:2
           FOR I:=INIT STEP 1 UNTIL NROWS-1 DO                                00026700  008:006C:2
              IF QUANT[I]>0 THEN                                              00026800  008:0070:5
              BEGIN                                                           00026900  008:0071:5
                 ROWSUM := 0;                                             3   00027000  008:0072:2
                 ROWSQ := 0;                                                  00027100  008:0073:0
                 READ(INP1[I+2],NCOLS,COST);                                  00027200  008:0073:4
                 TIME3 := TIME(2);                                            00027300  008:007B:2
                 FOR J:=0 STEP 1 UNTIL NCOLS-1 DO                             00027400  008:007C:5
                 BEGIN                                                        00027500  008:0081:4
                    JOBS := NJOBS[J];                                     4   00027600  008:0081:4
                    IF J=0 THEN                                               00027700  008:0082:5
                       KOSTJ := 0                                             00027800  008:0083:3
                    ELSE                                                      00027900  008:0084:0
                       IF COST[J]>0999 THEN                                   00028000  008:0084:4
                          KOSTJ := 1025                                       00028100  008:0086:3
                       ELSE                                                   00028200  008:0087:0
                          KOSTJ := COST[J];                                   00028500  008:0088:0
                    COLSQ[J] := *+(KOSTJ*KOSTJ);                              00028600  008:0089:4
                    ROWTEM := KOSTJ*JOBS;                                     00028700  008:008B:5
                    ROWSUM := *+ROWTEM;                                       00028800  008:008D:1
                    COL[J] := *+KOSTJ*QUANT[I];                               00028900  008:008E:3
                    ROWSQ := *+ROWTEM*KOSTJ;                                  00029000  008:0091:1
                    COST[J] := KOSTJ;                                         00029100  008:0093:0
                 END;                                                         00029200  008:0094:2
                 COST[0] := (ROWSQ-((ROWSUM**2)/(TOTJOBS+0)))/            4   00029300  008:0094:5
                   (TOTJOBS);                                                 00029400  008:0097:3
                 COST[0] := SQRT(COST[0]);                                    00029600  008:0098:1
                 COST[0] := ROWSUM/(TOTJOBS*(COST[0]+1.000D));                00029700  008:009A:1
                 ROWTOT[I] := COST[0]*100;                                    00029800  008:009D:4
                 ROWTOT[I] := INTEGER(ROWTOT[I]);                             00029900  008:009F:4
                 PACK(ROWTOT[I],I);                                           00030000  008:00A1:3
                 COLSQ[0] := *+(COST[0]*COST[0]);                             00030100  008:00A4:0
                 COL[0] := *+COST[0];                                         00030200  008:00A6:3
                 IF I LEQ UP THEN                                             00030400  008:00A8:2
                    WRITE(DIJS[I],NCOLS,COST);                                00030500  008:00A9:1
                 TIME3 := TIME(2)-TIME3;                                      00030600  008:00B1:2
                 TIME4 := *+TIME3;                                            00030700  008:00B3:2
              END;                                                            00030800  008:00B4:4
```

```
SORTER(ROWTOT,0,NROWS-1,33,34);                        3   008:00B5:1
FOR I:=0 STEP 1 UNTIL NCOLS-1 DO                           008:00BC:4
   COLSIG[I] :=                                            008:00C1:3
      SQRT((COLSQ[I]-(COL[I]*COL[I])/NROWS)/NROWS);        008:00C2:1
FOR I:=0 STEP 1 UNTIL NCOLS-1 DO                           008:00C7:0
   COL[I] := *;                                            008:00C8:5
TIME4 := TIME4/60;                                         008:00CD:4
WRITE(PRIN,<F7 .2>,TIME4);                                 008:00CF:0
TIME4 := 0;                                                008:00D6:2
FOR J:=INIT STEP 1 UNTIL UP DO                             008:00D7:0
BEGIN                                                      008:00DB:0
   IF TOTJOBS=0 THEN                                   3   008:00DB:0
      GO TO WRAP;                                          008:00DB:4
   COUNT := J;                                             008:00DC:1
   COUNT := UNPACKJ(ROWTOT[COUNT]);                        008:00DD:1
   READ(DIJS[COUNT],NCOLS,COST);                           008:00DE:4
   MM := 0;                                                008:00E6:2
   K := 0;                                                 008:00E7:0
   LL := 0;                                                008:00E7:4
   BUF := 10**63;                                          008:00E8:2
   FOR I:=0 STEP 1 UNTIL NCOLS-1 DO                        008:00EA:3
   BEGIN                                                   008:00EF:2
      IF NJOBS[I]>0 THEN                               4   008:00EF:2
      BEGIN                                                008:00F0:2
         LL := *+1;                                    5   008:00F0:5
         DI := COST[I]-COL[I]/(NASSN-J+0.001);             008:00F2:0
         IF DI<BUF THEN                                    008:00F7:0
         BEGIN                                             008:00F7:3
            BUF := DI;                                 6   008:00F8:0
            MM := LL;                                      008:00F8:5
            K := I;                                        008:00F9:4
         END;                                              008:00FA:4
      END;                                             6   008:00FA:4
   END;                                                5   008:00FA:4
   I := K;                                             4   008:00FB:1
   DI := BUF;                                              008:00FC:1
   IF COST[I]<999 OR I=0 THEN                              008:00FD:0
   BEGIN                                                   008:00FE:1
      NJOBS[I] := *-1;                                 4   008:00FF:4
      IF I NEQ 0 THEN                                      008:0101:2
         TOTJOBS := *-1;                                   008:0102:0
      COUNT := *+1;                                        008:0103:4
      IF IU[NROWS+I+1] NEQ 0 THEN                          008:0105:0
         IR[COUNT] := IU[NROWS+I+1];                       008:0106:5
      IU[NROWS+I+1] := COUNT;                              008:0109:5
      IF I=0 THEN                                          008:010C:0
         IP[COUNT] := IP[NROWS+I+1]+999                    008:010C:4
      ELSE                                                 008:010F:3
         IP[COUNT] := IP[NROWS+I+1]+COST[I];               008:0110:2
      ID[COUNT] := NROWS+I+1;                              008:0114:1
      IDA[NROWS+I+1] := *-1;                               008:0116:2
      IDA[COUNT] := 1;                                     008:0118:5
      OFFERS[COUNT,1] := I;                                008:011A:0
      QUANT[COUNT] := 0;                                   008:011B:4
      Z := COUNT;                                          008:011C:5
      IF I=0 THEN                                          008:011D:5
         OBJVAL := *+999                                   008:011E:3
      ELSE                                                 008:011F:2
         OBJVAL := *+COST[I];                              008:0120:3
      WRITE(PRIN,<7I7,6F13.2>,I,IDA[Z],ID[Z],IU[Z],        008:0122:4
```

```
             IR[Z],IP[Z],COUNT,DI,COST[I],OBJVAL,COST[0],                  008:012D:5
             COL[I],COLSIG[I]);                                            008:013A:2
         COUNT := *-1;                                                     008:0141:2
         DEC := *+1;                                                       008:0142:4
      END                                                                  008:0143:5
      ELSE                                                        4        008:0143:5
         EXCESS[I] := *+1;                                                 008:0143:5
      FOR KUP:=0 STEP 1 UNTIL NCOLS-1 DO                                   008:0146:0
         COL[KUP] := *-COST[KUP];                                          008:014A:5
   END;                                                                    008:014D:3
   TIME1 := TIME(2)-TIME1;                                        3        008:014E:0
   TIME2 := *+TIME1;                                                       008:0150:0
   TIME1 := TIME1/60;                                                      008:0151:2
   WRITE(PRIN,<F7 .2>,TIME1);                                              008:0152:4
   NASSN := *-DEC;                                                         008:015A:2
   FOR I:=0 STEP 1 UNTIL NROWS-1 DO                                        008:015B:4
      ROWTOT[I] := 0;                                                      008:0160:3
   FOR I:=0 STEP 1 UNTIL NCOLS-1 DO                                        008:0162:1
      COLSIG[I] := COLSQ[I] := COL[I] := 0;                                008:0167:0
   INIT := *+UP+1;                                                         008:016A:4
   UP := NROWS-1;                                                          008:016C:3
   IF INIT LEQ NROWS-1 THEN                                                008:016D:5
      GO TO START;                                                         008:016F:0
%*************************                                                 008:016F:3
WRAP:                                                                      008:016F:3
   I := 0;                                                                 008:016F:3
   FOR COUNT:=1 STEP 1 UNTIL NROWS DO                                      008:0170:1
   BEGIN                                                                   008:0174:4
      IF QUANT[COUNT]=1 THEN                                      3        008:0174:4
      BEGIN                                                                008:0175:4
         IF IU[NROWS+I+1] NEQ 0 THEN                              4        008:0176:1
            IR[COUNT] := IU[NROWS+I+1];                                    008:0178:0
         IU[NROWS+I+1] := COUNT;                                           008:017B:0
         IF I=0 THEN                                                       008:017D:1
            IP[COUNT] := IP[NROWS+I+1]+999                                 008:017D:5
         ELSE                                                              008:0180:4
            IP[COUNT] := IP[NROWS+I+1]+COST[COUNT];                        008:0181:3
         ID[COUNT] := NROWS+I+1;                                           008:0185:2
         IDA[NROWS+I+1] := *-1;                                            008:0187:3
         IDA[COUNT] := 1;                                                  008:018A:0
         OFFERS[COUNT,1] := I;                                             008:018B:1
         QUANT[COUNT] := 0;                                                008:018C:5
         OBJVAL := *+999;                                                  008:018E:0
      END;                                                                 008:018F:3
   END;                                                           4        008:018F:3
   WRITE(PRIN,/,"     AFSC     EXCESS JOBS");                     3        008:0190:0
%*************************                                                 008:0199:2
   FOR I:=0 STEP 1 UNTIL NCOLS-1 DO                                        008:0199:2
      WRITE(PRIN,F6,(I+1),NJOBS[I]);                                       008:019E:1
   WRITE(PRIN,<"     OBJECTIVE FUNCTION VALUE= ",F10 .0>,                  008:01A8:5
                                                           DATA IS 0027 LONG
     OBJVAL);                                                              008:01AB:3
%  FOR I:=1 STEP 1 UNTIL NCOLS DO WRITE(PRIN,F5,I,(OFFERS[I,1]+1),         008:01B0:2
%   OFFERS[I,2],OFFERS[I,3]);                                              008:01B0:2
%  WRITE(PRIN,/,"     AFSC     EXCESS JOBS");                              008:01B0:2
%  FOR I:=1 STEP 1 UNTIL NROWS-1 DO WRITE(PRIN,F8,(I+1),EXCESS[I]);        008:01B0:2
%FOR I:=1 STEP 1 UNTIL NCOLS DO                                            008:01B0:2
   FOR I:=1 STEP 1 UNTIL NCOLS DO                                          008:01B0:2
      WRITE(OTP,<2I6>,I,(OFFERS[I,1]+1));                                  008:01B4:5
%WRITE(OTP,<2I6>,I,(OFFERS[I,1]+1));                                       008:01BE:5
```

```
% WRITE(PRIN   ,<I4,4I6,I10>,FOR I:=1 STEP 1 UNTIL NROWS+NCOLS DO              00043700  008:01BE:5
%           [I,IDA[I],ID[I],IU[I],IR[I],IP[I]]);                                00043800  008:01BE:5
                    NROWS := NROWS+NCOLS;                                       00043900  008:01BE:5
                                          % ADVAN.MAXRECSIZE:=5*NROWS;          00044000  008:01C0:2
                                           %ADVAN.BLOCKSIZE:=5*NROWS;           00044100  008:01C0:2
                    ADVAN.MAXRECSIZE := 4*NROWS;                                00044200  008:01C0:2
                    ADVAN.BLOCKSIZE := 4*NROWS;                                 00044300  008:01C2:5
                    WRITE(ADVAN,*,FOR I := 1 STEP 1 UNTIL NROWS                 00044400  008:01C5:2
                      DO[ID[I],IU[I],IR[I],IDA[I]]);                            00044500  008:01C8:2
%       [ID[I],IU[I],IR[I],IDA[I],IP[I]]);                                      00044600  008:01DA:2
                    LOCK(OTP);                                                  00044700  008:01DA:2
                    LOCK(ADVAN);                                                00044800  008:01DB:5
                    TIMER := TIME2 DIV 60;                                      00044900  008:01DD:2
                    WRITE(PRIN,<I10>,TIMER);                                    00045000  008:01DE:4
                 END;                                                           00045100  008:01E6:2
                                                                              B.0001(008) IS 0210 LONG
            END.                                                            2   00045200  003:0016:0
                                                                              B.0000(003) IS 0032 LONG
                                                                                     DATA IS 0010 LONG
==========================================================================================================
UMBER OF ERRORS DETECTED = 0.
UMBER OF SEGMENTS = 11.  TOTAL SEGMENT SIZE = 831 WORDS.  CORE ESTIMATE = 3670 WORDS.  STACK ESTIMATE = 99
ROGRAM SIZE = 434 CARDS, 2291 SYNTACTIC ITEMS, 77 DISK SEGMENTS.
ROGRAM FILE NAME:   (09DAA00)BEST.
OMPILATION TIME = 24.280 SECONDS ELAPSED;  5.363 SECONDS PROCESSING;  2.697 SECONDS I/O.
==========================================================================================================
```