BOEING SCIENTIFIC RESEARCH LABORATORIES DOCUMENT D1-82-0854

BOEING

SCIENTIFIC RESEARCH LABORATORIES

AD691056

CAVAP, a FORTRAN Subroutine for Solving an Assignment Problem with Concave Objective

D. W. Walkup

DDC
AUG 8 1969
A

This document has been approved for public release and sale; its distribution is unlimited

Reproduced by the
CLEARINGHOUSE
for Federal Scientific & Technical
Information Springfield Va. 22151

MATHEMATICS RESEARCH JUNE 1969

18

D1-82-0854

CAVAP, A FORTRAN SUBROUTINE FOR SOLVING AN ASSIGNMENT PROBLEM WITH CONCAVE OBJECTIVE

by

D. W. Walkup

Mathematical Note No. 529

Mathematics Research Laboratory

BOEING SCIENTIFIC RESEARCH LABORATORIES

June 1969

ABSTRACT

This document describes a FORTRAN IV subroutine named CAVAP (an acronym for conCAVe Assignment Problem) which uses a branch and bound method to solve a variant of the assignment problem in which the objective to be minimized is concave. It is assumed that m tasks are to be performed and n different kinds of machines have been defined for possible use in performing them. In general a task may be performed by several machines of one type, or a different number of machines of another type, or possibly only by certain combinations of machines. It is assumed that total cost, which is to be minimized, is the sum of costs for each kind of machine, each of which is in turn a concave function of the number of machines of that type required. A detailed mathematical treatment of the problem and the solution method has been given elsewhere. CAVAP has been used to solve a space fleet selection problem with 20 space missions (tasks), 40 booster and spacecraft components (machines), and a number of alternative launch vehicle configurations and combinations for each mission. Running time was under 20 minutes on an IBM 360 model 44 with one disk.

1.0 GENERAL INFORMATION

This document describes a Fortran IV subroutine named CAVAP which solves a variant of the of the so-called assignment problem in which the objective to be minimized is concave. CAVAP is an acronym for conCAVe Assignment Problem. The method of solution is by branch and bounds. A mathematical treatment of the problem and solution method is given in [2]. For an example of the use of this subroutine, see the description in [1] of the routine for computing a minimum cost mixed space fleet from given vehicle designs when the fixed costs and learning curves give a concave cost vs. quantity relationship. CAVAP has been run as a subroutine to the program in [1] on an IBM 360 model 44 computer equipped with one disk. All limitation and running time data presented are taken from this context.

1.1 Purpose

1.1.1 Mathematical statement of the problem

Suppose m tasks are to be performed and n different kinds of machines have been defined for possible use in performing them. In general a task may be performed by several machines of one type, or a different number of machines of another type, or possibly only by certain combinations of machines. For each i, $1 \leq i \leq m$, it must be possible to write out all the alternatives to be considered for performing task number i. Let $k(i)$ denote the number of such alternatives. For each value of h, $1 \leq h \leq k(i)$, the hth alternative for task i can be given as a vector

$$x^{ih} = (x_1^{ih}, x_2^{ih}, \ldots, x_n^{ih})$$

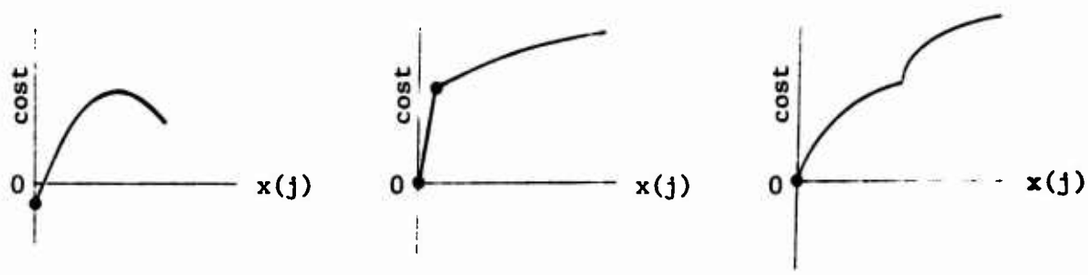
where each x_j^{ih} is the number (positive or zero) of machines of type j required in alternative h for performing task i. There is no requirement that the alternatives for different tasks are in any way related. For the application given in [1], the tasks are space missions, perhaps 20 in number; the machine types are booster and spacecraft components, perhaps 40 in number; and each alternative consists of the numbers of components of different types needed to build the number of launch vehicles of a particular configuration required to perform one mission.

A feasible (not necessarily optimal) solution of a problem consists in selecting exactly one alternative for each task, i.e. numbers $h(i)$. It is assumed that a machine to be used on one task cannot also be used on another, but different machines of the same type may be. Thus the number $x(j)$ of machines of type j required by a particular feasible solution is the sum of the numbers of machines of type j included in the selected alternatives, i.e.

$$x(j) = \sum_{i=1}^{m} x_j^{ih(i)}.$$

Next suppose the total cost of design, production, etc., of $x(j)$ machines of type j is given by a cost function $\phi(j,x(j))$. This implies that the cost of $x(j)$ machines of type j is completely independent of the numbers or kinds of other machines produced. Thus in the problem given in [1], a complete vehicle cannot be considered

a machine if for example it has a stage in common with another vehicle. It is also assumed that for each j the graph of $\phi(j,x(j))$ is concave, that is, the cost of an additional machine of type j decreases as the total number of machines of type j increases. Figure 1 illustrates acceptable and unacceptable functions ϕ. In most practical applications it is to be expected that ϕ will be zero for $x = 0$ and positive increasing for $x > 0$ as in Figure 1b, but this is not essential to the method of solution. Since no alternative can specify a negative number of machines of any type, it is immaterial what value, if any, ϕ gives when $x(j)$ is negative.



a. concave b. concave c. nonconcave

Figure 1

The total cost of performing all tasks using a particular feasible solution is

$$Z = \sum_{j=1}^{n} \phi(j,x(j)).$$

The problem solved by CAVAP is that of selecting the feasible solution (i.e. an alternative for each task or $h(i)$ for each i) which will minimize Z.

1.1.2 Need for a special algorithm

The method of complete enumeration is an obvious and very simple one for determining the least cost solution of any assignment problem. This is the method which computes the cost of each of the possible combinations of alternatives and retains the combination with the least cost. This method requires the examination of

$$s = k(1) \cdot k(2) \cdot \ldots \cdot k(m)$$

different feasible solutions. For one of the problems of the type described in [1], $m = 20$, $n = 38$, and

$$s = 5 \cdot 1 \cdot 5 \cdot 3 \cdot 12 \cdot 5 \cdot 11 \cdot 2 \cdot 6 \cdot 7 \cdot 5 \cdot 6 \cdot 15 \cdot 6 \cdot 15 \cdot 6 \cdot 6 \cdot 3 \cdot 7 \cdot 7 \doteq 10^{15}.$$

Even if the cost of each feasible solution could be computed in 1 microsecond, it would take more than 30 years to examine all 10^{15} of them. By comparison the same problem was solved using CAVAP in approximately 20 minutes on an IBM 360 model 44.

1.2 Limitations

The method of solution used by CAVAP belongs to a broad class called "branch and bound" methods. As a class these methods suffer two inherent limitations. First, as the size of the problem increases, the amount of computation time required increases exponentially. This means that there is a fairly well defined bound on problem size above which computation time is impractically large. The better the procedure, the higher this bound. Second, branch and bound methods make use of a list of partial solutions. In order to save recomputing time, the

amount of information saved with each item of the list is large. Thus auxiliary memory is usually used. Since the length of the list grows exponentially with the size of the problem, there is also a rough limit on problem size above which the list is too long even for auxiliary memory.

For the application discussed in [1] it was found that the limitation on problem size was the size of the scratch area on the only available disk, namely, 500 blocks of 180 words each. The largest problems which did not exceed this storage requirement ran about 20 minutes. For several scratch disks the limitation would become computation time.

Presumably with large problems round-off errors could become serious. In many computer programs the effect of round-off errors is to cause a fairly steady deterioration of accuracy as the amount of computation increases. The nature of the algorithm used by CAVAP is such that with increasing size of problem there will be little effect on the precision of answers until at some point round-off errors cause the process to go wild. It is *unlikely* that this would result in undetected errors. The most likely result would be an immediate error exit or an endless loop which would eventually fill up available disk storage and result in an error exit. Results with the problems of [1] suggest that the limit due to running time will be met before round-off becomes a problem.

As written, CAVAP will handle in theory up to 44 machines, up to 100 tasks, and up to 500 auxiliary memory items. These limits are

easily alterable. The only real limitations are the amount of running time and auxiliary memory available.

2.0 PROCEDURE

The naming of variables in the subroutine follows closely the notations found in [2] and in paragraph (1.1.1).

The method of solution is given in considerable detail in [2]. The deviations are minor and of a programming nature. The subroutine contains an initial section in which the input parameters and data are subjected to a complete check to insure that they meet the requirements set fourth in paragraph (3.0) below.

In addition to the normal return there are two abnormal returns from the subroutine. If the normal return occurs, a feasible solution has been found which has a cost which differs from the minimum cost in about the 5th decimal place. Abnormal return 1 occurs if the input data is improperly given. If the program must be abandoned for any reason before finding an optimum solution, a feasible solution and its cost are given. It may very well happen that the feasible solution in this case is actually optimum, but the subroutine has not yet succeeded in proving this fact. Three likely causes of the subroutine being abandoned are: exhausting the disk storage, finding that one of the cost functions given by PHI is not concave, or the accumulation of round-off errors.

3.0 INPUT AND OUTPUT DESCRIPTION

The calling sequence is

```
CALL CAVAP(M,N,PHI,JIN,XIN,ZBEST,HIBEST,XBEST,NITER,MAXITM,&a,&b)
```

where

Name	Type Integer or Real	Description
		(Input)
M	I	number of tasks. Must be between 1 and 100.
N	I	number of machines. Must be between 1 and 44.
PHI	R	transmitted name of a function subroutine PHI(J,X) which returns the cost of X machines of type J. This function must be concave over the range of values of X for the given problem. The function need not be defined for negative X.
JIN	I	input arrays described below.
XIN	R	
		(Output)
ZBEST	R	cost of the best feasible solution found. Set to infinity if the input data is improper.
HIBEST	I	an array whose i^{th} element is the number h of the alternative selected for task i in the best feasible solution found. This array should be of length at least 100 M.
XBEST	R	an array whose j^{th} element is the number of machines of type j required in the best

		feasible solution found. This array must be of length at least N.
NITER	I	number of iterations performed.
MAXITM	I	maximum number of items on disk at any time during solution of current problem.
		(Error returns)
	a	statement number of return if input data is improper.
	b	statement number of return if solution is abandoned.

The input arrays JIN and XIN contain the descriptions of the various alternatives for each of the tasks arranged in a compressed form. The first element of JIN must be -1. The second element of JIN must be 0. There must follow a string of from 1 to N distinct positive integers less than or equal to N. These are the numbers of the machines used in the first alternative for the first task. In the corresponding locations in XIN must be the "numbers" of these machines required. These numbers must be floating positive numbers; they need not be floating integers. The first location of JIN following this string must contain 0 or -1. If the entry is 0, another string describing a second alternative must follow. Any number of alternatives may follow, each one preceded by a 0 in JIN. If the entry following the description of an alternative is -1 the description of the alternatives for the next task must follow. Again the first alternative must be preceded by a 0 entry in JIN. The first entry in JIN following the last alternative of the last task must be -2. Each task must have at least one alternative and the number of

tasks described (number of -1 entries in JIN) must equal M. An example of an acceptable set of values for JIN and XIN involving 2 tasks with 1 and 2 alternatives respectively is shown below. The value of N is 4. The values marked * are immaterial.

JIN	XIN
-1	*
0	*
1	2.0
4	1.5
-1	*
0	*
2	3.0
0	*
3	1.2
2	.05
4	5.0
-2	*

FORTRAN IV MODEL 44 PS VERSION 3, LEVEL 2

```
      SUBROUTINE CAVAP (M,N,PHI,JIN,XIN,ZBEST,HIBEST,XBEST,
     1                  NITER,MAXITM,*,*)
C     FOR A DESCRIPTION OF THIS SUBROUTINE, SEE
C             'CAVAP, A FORTRAN SUBROUTINE FOR SOLVING AN ASSIGNMENT
C             PROBLEM WITH CONCAVE OBJECTIVE', BY D. WALKUP,
C             MATHEMATICAL NOTE NO. 529, BOEING SCIENTIFIC RESEARCH
C             LABORATORIES, SEPTEMBER 1967.
C     FOR A MATHEMATICAL TREATMENT OF THE PROBLEM AND THE THEORY OF THE
C             ALGORITHM USED, SEE
C             'ON A BRANCH-AND-BOUND METHOD FOR SEPARABLE CONCAVE
C             PROGRAMS', BY D. WALKUP, MATHEMATICAL NOTE NO. 527,
C             BOEING SCIENTIFIC RESEARCH LABORATORIES, SEPTEMBER 1967.
      IMPLICIT INTEGER (H-N)
      EXTERNAL PHI
      DIMENSION JIN(1),XIN(1),HIBEST(1),XBEST(1)
      DIMENSION HILIN(100),XLIN(44),Z(44),ZLINIT(500),LOCITM(500)
      DIMENSION INFO(180),P(44),Q(44),C(44),D(44)
      EQUIVALENCE (INFO(1),P(1)),(INFO(45),Q(1)),
     1            (INFO(89),C(1)),(INFO(133),D(1)),(INFO(177),DSUM),
     2            (INFO(178),JBK),(INFO(179),XBK),(INFO(180),ZBK)
      DIMENSION XIH(44),XIMAX(44)
      EQUIVALENCE (P,XIH),(XLIN,XIMAX)
      DATA FINF/Z7FFFFFFF/
      DEFINE FILE 1(1300,90,U,IX)
      XEPS = .00001
      OMEPS = .99999
      MMAX = 100
      NMAX =44
      ITEMAX = 500
      LINFO = 180
      NITER = 0
      NITEMS = 0
      MAXITM = 0
      ZBEST = FINF
      L = 0
      IF((M.LE.0).OR.(M.GT.MMAX).OR.(N.LE.0).OR.(N.GT.NMAX)) GO TO 900
C     CHECK INPUT LISTS JIN AND XIN AND FIND UPPER BOUNDS Q(J) ON
C             THE SOLUTION.
      DO 105 J = 1,N
  105 Q(J) = 0.
      I = 0
      L = 1
      IF(JIN(L).NE.-1) GO TO 900
  110 I = I + 1
      IF(I.GT.M) GO TO 900
      DO 115 J = 1,N
  115 XIMAX(J) = 0.
      H = 0
      L = L + 1
```

```
      IF(JIN(L).NE.0) GO TO 900
  120 H = H + 1
      DO 125 J = 1,N
  125 XIH(J) = 0.
      IF(JIN(L+1).LE.0) GO TO 900
  130 L = L + 1
      JJ = JIN(L)
      IF(JJ.LE.0) GO TO 140
      IF((JJ.GT.N).OR.(XIN(L).LE.0.).OR.(XIH(JJ).GT.0.)) GO TO 900
      XIH(JJ) = XIN(L)
      GO TO 130
  140 DO 141 J = 1,N
      IF(XIH(J).LE.XIMAX(J)) GO TO 141
      XIMAX(J) = XIH(J)
  141 CONTINUE
  145 IF(JJ.EQ.0) GO TO 120
      DO 150 J = 1,N
  150 Q(J) = Q(J) + XIMAX(J)
      IF(JJ.EQ.-1) GO TO 110
      IF(I.NE.M) GO TO 900
C     CHECKING OF INPUT LISTS JIN AND XIN COMPLETE.  USE OF XIH AND
C                 XIMAX FINISHED.
C     STEP 1.  FIND FIRST LINEARIZED SOLUTION.
C     COMPUTE INITIAL LINEARIZED OBJECTIVE FUNCTIONS.
      DSUM = 0.
      DO 160 J = 1,N
      P(J) = 0.
      ZP = PHI(J,0.)
      D(J) = ZP
      DSUM = DSUM + ZP
      C(J) = 0.
      IF(Q(J).EQ.0.) GO TO 160
      ZQ = PHI(J,Q(J))
      C(J) = (ZQ-ZP)/Q(J)
  160 CONTINUE
C     INITIALIZE DISK LOCATION LIST
      DO 170 L = 1,ITEMAX
  170 LOCITM(L) = 2*L-1
C     SOLVE INITIAL LINEARIZED PROGRAM AS A SPECIAL CASE OF STEP 4.
      NCASE = 2
      GO TO 240
C     STEP 2.  SELECT ITEM FROM DISK.
C     (REMOVAL OF ITEMS FROM LIST IS ACCOMPLISHED AS PART OF STEPS 3
C              AND 4 AT STATEMENT 248 BELOW.)
  200 IF(NITEMS.EQ.0) RETURN
      LR = LOCITM(1)
      READ (1'LR) (INFO(L),L=1,LINFO)
      NITEMS = NITEMS - 1
      IF(NITEMS.EQ.0) GO TO 225
```

```
C     PUSH DOWN LIST.
      DO 210 L = 1,NITEMS
      ZLINIT(L) = ZLINIT(L+1)
  210 LOCITM(L) = LOCITM(L+1)
      LOCITM(NITEMS+1) = LR
C     SAVE DATA FOR STEP 4.
  225 XBKS = XBK
      QS = Q(JBK)
      ZBKS = ZBK
      CS = C(JBK)
      ZPS = D(JBK)
      DSUMS = DSUM
      JBKS = JBK
C     STEP 3 INITIALIZATION.
      NCASE = 1
      Q(JBK) = XBK
      C(JBK) = (ZBK-ZPS)/(XBK-P(JBK))
      GO TO 240
C     STEP 4 INITIALIZATION.
  235 NCASE = 2
      P(JBKS) = XBKS
      Q(JBKS) = QS
      ZQ = PHI(JBKS,QS)
      C(JBKS) = (ZQ-ZBKS)/(QS-XBKS)
      D(JBKS) = ZBKS
      DSUM = DSUM - ZPS + ZBKS
  240 NITER = NITER + 1
C     FIND OPTIMUM SOLUTION ASSUMING LINEARIZED COSTS.
  300 DO 305 J = 1,N
  305 XLIN(J) = 0.
      I = 0
      L = 1
  310 I = I + 1
      ZI = FINF
      H = 0
      L = L + 1
  320 H = H + 1
      ZIH = 0.
      LIH1 = L + 1
  330 L = L + 1
      JJ = JIN(L)
      IF(JJ.LE.0) GO TO 340
      ZIH = ZIH + XIN(L)*C(JJ)
      GO TO 330
  340 IF(ZIH.GE.ZI) GO TO 345
      ZI = ZIH
      HI = H
      LI1 = LIH1
      LI2 = L - 1
```

```
  345 IF(JJ.EQ.0) GO TO 320
      HILIN(I) = HI
      DO 350 L350 = LI1,LI2
      J350 = JIN(L350)
  350 XLIN(J350) = XLIN(J350) + XIN(L350)
      IF(JJ.EQ.-1) GO TO 310
C     COMPUTE LINEAR COST ZLIN AND TRUE COST ZMIX OF THIS SOLUTION.
      ZLIN = DSUM
      ZMIX = 0.
      DO 241 J = 1,N
      ZLIN = ZLIN + (XLIN(J)-P(J))*C(J)
      Z(J) = PHI(J,XLIN(J))
  241 ZMIX = ZMIX + Z(J)
      IF(ZMIX.GE.ZBEST) GO TO 250
C     IF THIS IS A BETTER SOLUTION THAT THE BEST SO FAR, REPLACE IT.
      ZBEST = ZMIX
      DO 245 I = 1,M
  245 HIBEST(I) = HILIN(I)
      DO 247 J = 1,N
  247 XBEST(J) = XLIN(J)
C     CHECK LIST ZLINIT FOR REMOVABLE ITEMS.
  248 IF(NITEMS.EQ.0) GO TO 250
      IF(ZLINIT(NITEMS).LT.OMEPS*ZBEST) GO TO 250
      NITEMS = NITEMS - 1
      GO TO 248
C     DETERMINE IF THIS CASE SHOULD BE PUT ON THE LIST OR IGNORED.
  250 IF(ZLIN.GE.ZBEST*OMEPS) GO TO 260
C     DETERMINE INTERVAL TO BE BROKEN FOR THIS CASE.
      EMAX = 0.
      DO 255 J = 1,N
      E = Z(J) - D(J) - C(J)*(XLIN(J)-P(J))
      IF(E.LE.EMAX) GO TO 255
      EMAX = E
      JBK = J
      XBK = XLIN(J)
      ZBK = Z(J)
  255 CONTINUE
      IF(EMAX.EQ.0.) GO TO 1100
      IF(XBK.LE.P(JBK)*(1.+XEPS).OR.XBK.GE.Q(JBK)*(1.-XEPS)) GO TO 1100
C     SORT THE VALUE ZLIN INTO THE LIST ZLINIT, SAVE THIS CASE ON DISK.
      IF(NITEMS.GE.ITEMAX) GO TO 1000
      NITEMS = NITEMS + 1
      IF(NITEMS.GT.MAXITM) MAXITM = NITEMS
      LW = LOCITM(NITEMS)
      IT = NITEMS
  256 IF(IT.LE.1) GO TO 257
      IF(ZLINIT(IT-1).LE.ZLIN) GO TO 257
      ZLINIT(IT) = ZLINIT(IT-1)
      LOCITM(IT) = LOCITM(IT-1)
```

```
      IT = IT - 1
      GO TO 256
  257 ZLINIT(IT) = ZLIN
      LOCITM(IT) = LW
      WRITE (1'LW) (INFO(L),L=1,LINFO)
  260 IF(NCASE.EQ.2) GO TO 200
      GO TO 235
  900 WRITE (6,901) M,N,L
  901 FORMAT(' SUBROUTINE CAVAP ENTERED WITH IMPROPER DATA.'/10X'M ='I4
     1     ', N ='I4',LAST LOCATION IN INPUT LISTS JIN AND XIN CHECKED ='
     2     I5)
      RETURN 1
 1000 WRITE (6,1001) NITEMS,NITER
 1001 FORMAT(' SUBROUTINE CAVAP DISK STORAGE OF 'I5' ITEMS EXHAUSTED AFT
     1ER'I6' ITERATIONS.')
      RETURN 2
 1100 WRITE (6,1101) JBK,NITER
 1101 FORMAT(' SUBROUTINE CAVAP.  ERROR DETECTED IN COMPONENT'I3' AFTER'
     1I5' ITERATIONS.'/'     MAY BE CAUSED BY ROUND-OFF ERRORS OR NONCON
     2CAVITY OF FUNCTION PHI.')
      RETURN 2
      END
```

TOTAL MEMORY REQUIREMENTS 002704 BYTES

REFERENCES

[1] D. Walkup, A Fortran program for finding a minimum cost fleet of space vehicles, Mathematical Note No. 530, Boeing Scientific Research Laboratories, in preparation.

[2] D. Walkup, On a branch-and-bound method for separable concave programs, Mathematical Note No. 527, Boeing Scientific Research Laboratories, September 1967.