AUCTION CODES FOR THE ASSIGNMENT PROBLEM
SOME OF THE CODES ARE FROM THE APPENDIXES OF THE TEXTBOOK
"LINEAR NETWORK OPTIMIZATION: ALGORITHMS AND CODES",
MIT PRESS,1991
by Dimitri P. Bertsekas

The versions given are the most
recent as of

May 7, 1992

All codes can be
compiled with the popular Absoft compiler on all Macintosh computers (subject to memory limitations). By changing a few input and
output statements, the codes can be easily adapted for other computers and FORTRAN compilers. The codes should not be modified except for the minimum modifications necessary to make them run on other compilers. Please
state clearly your modifications when reporting research.

The following codes are included:

1) AUCTION: Forward auction algorithm for symmetric assignment problems.

2) AUCTION_FLP: Same as AUCTION but uses floating point arithmetic to deal with problems with large cost range and/or dimension. For such problems the prices may overflow the integer range in the course of the algorithm.

3) AUCTION_AS: Auction algorithm for asymmetric assignment problems.

4) AUCTION_FR: Forward/reverse auction algorithm for symmetric assignment problems.

5) NAUCTION_SP: Combined naive auction and sequential shortest path method for assignment.

The codes
include a built-in random problem generator. This generator can be replaced by other code that reads an assignment problem from a file.

\subsect{AUCTION}This code implements the forward auction
algorithm with $\e$-scaling for the symmetric assignment problem; cf.\
Section 4.1 of the author's network optimization book.

```
C ***************************************************************
C
C  SAMPLE CALLING PROGRAM FOR THE AUCTION ALGORITHM
C
C   THIS DRIVER CREATES A SYMMETRIC ASSIGNMENT PROBLEM
C   WITH EQUAL NUMBER OF ROWS AND COLUMNS,
C   AND CALLS THE AUCTION SUBROUTINE TO FIND AN
C   ASSIGNMENT OF MAXIMAL VALUE.
C
C ***************************************************************

 PARAMETER(MAXNODES=10000, MAXARCS=100000)
 
       IMPLICIT NONE
 INTEGER N,NA,A,ISMALL,BEGEPS,ENDEPS,CYCLES
```

```
      INTEGER NUMPHASES,STARTINCR,TEST,PROFIT
      INTEGER I,J,IA,ARC,NOASS,ICOST,ABSCOST,CURARC
      INTEGER CURCOL,FSTARC,LSTARC,MINCOST,MAXCOST,MCOST
      INTEGER FOUT(MAXNODES),COLASS(MAXNODES)
      INTEGER ASSIGN(MAXNODES),PCOL(MAXNODES)
      INTEGER COST(MAXARCS),END(MAXARCS)
      REAL*8 FACTOR,TT1,TT2,TCOST,AVERAGE
      COMMON/ARRAYC/COST/ARRAYS/END/ARRAYF/FOUT/BK1/N,A,ISMALL
     $/BK2/CYCLES,AVERAGE,NUMPHASES/ARRAYA/ASSIGN/PCOLA/PCOL

C ***************************************************************
C
C     PROBLEM GENERATION CODE STARTS HERE
C     THE USER MAY REPLACE THIS CODE WITH A CODE THAT READS
C     HIS/HER PROBLEM FROM A FILE
C
C ***************************************************************

C     THIS CODE INCLUDES A UNIFORM RANDOM NUMBER GENERATOR
C     WHICH RETURNS A VALUE IN (0,1)

C     INITIALIZE RANDOM GENERATOR

      INTEGER MULT,MODUL,I15,I16,JRAN,ISEED
      REAL RAN
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN

      ISEED=13502460
      CALL SETRAN(ISEED)

      PRINT*,'GENERATING A SYMMETRIC ASSIGNMENT PROBLEM'
      PRINT*,'*****************************************'

C **** READ THE NUMBER OF ROWS N & THE NUMBER OF ARCS A ****

      PRINT*,'ENTER THE NUMBER OF ROWS (AND COLUMNS)'
      READ*,N
5     PRINT*,'ENTER THE NUMBER OF ARCS PER ROW (>1)'
      READ*,NA
      IF (NA.LT.2) GOTO 5
      PRINT*,'ENTER THE MINIMUM AND THE MAXIMUM COST'
      READ*,MINCOST,MAXCOST

C     THE NUMBER OF ARCS IS N*NA

      A=N*NA

C     THE ARCS INCIDENT TO ROW I ARE FOUT(I) TO FOUT(I+1)-1
C     ALSO, FOR FEASIBILITY EACH ROW IS DIRECTLY CONNECTED
C     WITH THE CORRESPONDING COLUMN

      DO 20 I=1,N
       FOUT(I)=1+(I-1)*NA
20    CONTINUE
      FOUT(N+1)=A+1

C     GENERATE THE END(ARC) AND COST(ARC) WHICH ARE THE COLUMN
C     AND THE COST COEFFICIENT ASSOCIATED WITH ARC

      DO 25 ARC=1,A
```

```
      END(ARC)=1+RAN()*N
IF ((END(ARC).GT.N).OR.(END(ARC).LT.1)) THEN
  PRINT*,'ERROR IN PROBLEM GENERATION'
  PAUSE
  STOP
END IF
COST(ARC)=MINCOST+RAN()*(MAXCOST-MINCOST)
25    CONTINUE

C   MODIFY THE END OF THE LAST ARC OUT OF EACH ROW FOR FEASIBILITY
C   AND SET ITS COST TO -MAXCOST

     DO 30 I=1,N
      END(FOUT(I+1)-1)=I
COST(FOUT(I+1)-1)=-MAXCOST
30    CONTINUE


C
C ***************************************************************
C
C         PROBLEM GENERATION CODE ENDS HERE
C
C ***************************************************************

C      SCALE THE COST TO WORK WITH INTEGER EPSILON

      MAXCOST=0
      DO 35 IA=1,A
        ABSCOST=IABS(COST(IA))
        IF (ABSCOST.GT.MAXCOST) MAXCOST=ABSCOST
        COST(IA)=COST(IA)*(N+1)
35 CONTINUE

C *** ISMALL IS A VERY SMALL INTEGER FOR YOUR MACHINE ***

      ISMALL=-2000000000

IF (MAXCOST.GT.INT(ABS(ISMALL)/(N+1))) THEN
  PRINT*,'THE COST RANGE IS TOO LARGE FOR INTEGER ARITHMETIC'
  PAUSE
  STOP
END IF

MAXCOST=MAXCOST*(N+1)


C    THE FOLLOWING PARAMETERS BEGEPS, FACTOR, ENDEPS, AND STARTINCR
C      ARE PASSED TO THE AUCTION ALGORITHM. VALUES BETWEEN
C       (A) MAXCOST/5 AND MAXCOST/2 FOR BEGEPS
C       (B) 4 AND 6 FOR FACTOR
C (C) N/10 AND 1 FOR ENDEPS
C       (D) 1 AND BEGEPS FOR STARTINCR
C      HAVE WORKED WELL FOR LARGE SPARSE PROBLEMS.
C      FOR DENSE PROBLEMS IT IS RECOMMENDED THAT
CBEGEPS BE SET TO A SMALLER VALUE,
CENDEPS BE SET TO 1,
C      STARTINCR BE SET TO 1.

      PRINT*,'*****************************************'
```

```
      PRINT*,'MAXIMUM COST IS ',MAXCOST
      PRINT*,'ENTER THE STARTING EPSILON'
      READ*,BEGEPS
      IF (BEGEPS.LT.1) BEGEPS=1
      PRINT*,'ENTER THE EPSILON REDUCTION FACTOR'
      READ*,FACTOR
□ENDEPS=N/10
□IF (ENDEPS.LT.1) ENDEPS=1
□IF (ENDEPS.GT.BEGEPS) ENDEPS=BEGEPS
□STARTINCR=BEGEPS/10
□IF (STARTINCR.LT.1) STARTINCR=1

      PRINT*,'*******************************************'
□PRINT*,'STARTING EPSILON = ',BEGEPS
□PRINT*,'EPSILON REDUCTION FACTOR = ',FACTOR
□PRINT*,'THRESHOLD EPSILON BEFORE IT IS SET TO 1 = ',ENDEPS
□PRINT*,'STARTING MIN BIDDING INCREMENT = ',STARTINCR
      PRINT*,'CALLING AUCTION TO SOLVE THE PROBLEM'
      PRINT*,'*******************************************'

C     GET STARTING TIME FOR THE MAC II

     TT1 = LONG(362)/60.0

     CALL AUCTION(BEGEPS,FACTOR,ENDEPS,STARTINCR)

C     GET ENDING TIME FOR THE MAC II

     TT2 = LONG(362)/60.0 - TT1
     PRINT *,'FINISHED --- TOTAL CPU TIME', TT2,' SECS'
     PRINT*,'*******************************************'

C     *** DISPLAY RESULTS ***

X       WRITE(9,2010) CYCLES
X2010   FORMAT(' NO OF AUCTION CYCLES',I7)
X       WRITE(9,2020) AVERAGE
X2020   FORMAT(' AVERAGE NUMBER OF BIDS PER CYCLE',F9.3)
      WRITE(9,2030) NUMPHASES
2030   FORMAT('NO OF EPSILON SUBPROBLEMS SOLVED =',I7)

C      CHECK OPTIMALITY OF SOLUTION & CALCULATE COST

      DO 40 I=1,N
        COLASS(I)=0
40□CONTINUE
      NOASS=0
      DO 50 J=1,N
        IF (ASSIGN(J).GT.0) THEN
          NOASS=NOASS+1
          COLASS(ASSIGN(J))=J
        END IF
50□CONTINUE
      IF (NOASS.NE.N) THEN
        PRINT*,'# OF ASSIGNED ROWS NOT EQUAL TO # OF ROWS'
      END IF
      TCOST=0
      DO 60 I=1,N
□  J=COLASS(I)
        IF (J.EQ.0) THEN
```

```
          PRINT*,'ROW ',I,' IS UNASSIGNED'
        END IF
        FSTARC=FOUT(I)
        LSTARC=FOUT(I+1)-1
      MCOST=ISMALL
        DO 70 CURARC=FSTARC,LSTARC
          CURCOL=END(CURARC)
          IF (CURCOL.EQ.J) THEN
            IF (MCOST.LT.COST(CURARC)) THEN
              MCOST=COST(CURARC)
         END IF
       END IF
70     CONTINUE
        PROFIT = MCOST-PCOL(J)
        DO 72 CURARC=FSTARC,LSTARC
         J = END(CURARC)
         TEST = COST(CURARC)-PCOL(J)-PROFIT
         IF (TEST.GT.1) THEN
           PRINT *, '1-CS VIOLATED AT ARC',CURARC
         END IF
72       CONTINUE
      TCOST=TCOST+MCOST/(N+1)
60    CONTINUE
      WRITE(9,2100) TCOST
2100    FORMAT(' ASSIGNMENT COST=',F14.2)
      PRINT *, ' PROGRAM ENDED; <CR> TO EXIT '
      PAUSE
      END



C ***************************************************************
C
C     AUCTION CODE FOR N BY N ASSIGNMENT PROBLEMS
C
C        WRITTEN BY DIMITRI P. BERTSEKAS
C         (MODIFICATIONS BY PAUL TSENG)
C
C             VERSION 1.1, SEPT. 1990
C
C THIS CODE IMPLEMENTS THE AUCTION ALGORITHM WITH E-SCALING.
C IT SOLVES A SEQUENCE OF SUBPROBLEMS AND DECREASES
C EPSILON BY A CONSTANT FACTOR BETWEEN SUBPROBLEMS.
C THIS VERSION CORRESPONDS TO A GAUSS-SEIDEL MODE
C AND SOLVES EPSILON SUBPROBLEMS INEXACTLY.
C
C THE CODE IS AN IMPROVED VERSION OF AN EARLIER (SEPT. 1985)
C AUCTION CODE WITH E-SCALING WRITTEN BY DIMITRI P. BERTSEKAS
C
C THE CODE TREATS THE PROBLEM AS A MAXIMIZATION PROBLEM.
C TO SOLVE A MINIMIZATION PROBLEM, REVERSE THE SIGN OF THE
C ARC COSTS PRIOR TO CALLING AUCTION, AND REVERSE AGAIN
C THE SIGN OF THE OPTIMAL COST UPON RETURN FROM AUCTION.
C THIS CODE ALLOWS MULTIPLE ARCS BETWEEN A ROW AND A COLUMN.
C
C THIS VERSION OF THE AUCTION ALGORITHM IS ADAPTIVE. HERE THE MINIMAL
C BIDDING INCREMENT (STORED IN THE VARIABLE INCR) MAY BE SMALLER THAN
C EPSILON. FOR EVERY SUBPROBLEM, INCR STARTS AT THE PARAMETER VALUE
C STARTINCR (WHICH IS PASSED TO THE AUCTION ROUTINE) AND IS INCREASED
C BY A FACTOR OF 2 AT THE END OF EACH CYCLE, UP TO A MAXIMUM VALUE OF
```

```
C EPSILON. THIS ADAPTIVE FEATURE IS PARTICULARLY EFFECTIVE FOR
C DENSE PROBLEMS. IT CAN BE DEFEATED BY SELECTING STARTINCR=BEGEPS
C
C ****************************************************************
C
C THE USER MUST SUPPLY THE FOLLOWING PROBLEM DATA IN FORWARD STAR FORMAT
C (THAT IS, ALL ARCS OF THE SAME ROW ARE NUMBERED CONSECUTIVELY):
C  N=NUMBER OF ROWS (EQUALS NUMBER OF COLUMNS)
C  A=NUMBER OF ARCS
C  FOUT(ROW)=FIRST ARC COMING OUT OF ROW
C  COST(ARC)=COST OF ARC
C  END(ARC)=COLUMN CORRESPONDING TO ARC
C
C AND THE FOLLOWING PARAMETERS FOR THE AUCTION ALGORITHM:
C  BEGEPS=STARTING VALUE OF EPSILON (MUST BE NO LESS THAN 1)
C  ENDEPS=FINAL VALUE OF EPSILON BEFORE IT IS SET TO 1
C  FACTOR=FACTOR BY WHICH EPSILON IS DECREASED BETWEEN SUBPROBLEMS
C       STARTINCR=THE STARTING VALUE OF THE BIDDING INCREMENT
C       ENDEPS SHOULD NOT EXCEED BEGEPS.
C       FACTOR MUST BE GREATER THAN 1.
C
C FOUT(.) IS AN ARRAY OF LENGTH N.
C COST(.),END(.) ARE ARRAYS OF LENGTH A.
C
C THE SOLUTION IS CONTAINED IN THE ARRAY ASSIGN(.) WHERE
C    ASSIGN(COL) GIVES THE ROW ASSIGNED TO COL.
C
C THIS ALGORITHM DOES NOT CHECK FOR INFEASIBILITY OF THE PROBLEM.
C TO MAKE SURE THE PROBLEM IS FEASIBLE THE USER MAY ADD
C    ADDITIONAL VERY SMALL COST ARCS.
C
C THIS CODE MAY FAIL DUE TO INTEGER OVERFLOW IF THE NUMBER OF NODES
C OR THE COST RANGE (OR BOTH) ARE LARGE. TO CORRECT THIS SITUATION,
C THE PRICES AND OTHER RELATED VARIABLES (MAX1,MAX2,TMAX ETC) SHOULD
C BE DECLARED AS DOUBLE PRECISION REALS.
C ****************************************************************
C ALL PROBLEM DATA ARE INTEGER
C ****************************************************************


 SUBROUTINE AUCTION(BEGEPS,FACTOR,ENDEPS,STARTINCR)

 PARAMETER(MAXNODES=10000, MAXARCS=100000)
 
       IMPLICIT NONE
       INTEGER NONEWLIST,THRESH,INCR,STARTINCR,INCRFACTOR
       INTEGER A,K,N,I,J,CURARC,CURCOL
       INTEGER ROW,FSTARC,FSTCOL,SNDARC,SNDCOL,TMAX,TMIN,BSTCOL
       INTEGER MAX1,MAX2,TRDARC,EPSILON,BEGEPS,ENDEPS
       INTEGER M,ISMALL,ILARGE,LARGEINCR,CYCLES
       INTEGER NUMPHASES,LSTARC,NOLIST,OLDROW
       INTEGER FOUT(MAXNODES),PCOL(MAXNODES),LIST(MAXNODES)
 INTEGER ASSIGN(MAXNODES)
       INTEGER COST(MAXARCS),END(MAXARCS)
       REAL*8 AVERAGE,FACTOR
 COMMON/ARRAYC/COST/ARRAYS/END/ARRAYF/FOUT/BK1/N,A,ISMALL
     $ /BK2/CYCLES,AVERAGE,NUMPHASES/ARRAYA/ASSIGN/PCOLA/PCOL

C ****************************************************************
```

```
C   ******* CHECK VALIDITY OF PARAMETERS PASSED *******

      IF (BEGEPS.LT.1) THEN
  PRINT*,'STARTING VALUE OF EPSILON IS LESS THAN 1'
  PRINT*,'EXECUTION ABORTED'
  STOP
END IF
IF (ENDEPS.GT.BEGEPS) THEN
  PRINT*,'PARAMETER ENDEPS IS GREATER THAN PARAMETER BEGEPS'
  PRINT*,'ENDEPS IS SET AT THE DEFAULT VALUE OF 1'
  ENDEPS=1
END IF
      IF ((FACTOR.LE.1).AND.(BEGEPS.GT.1)) THEN
  PRINT*,'EPSILON REDUCTION FACTOR IS NOT GREATER THAN 1'
  PRINT*,'EXECUTION ABORTED'
  STOP
END IF
      IF (STARTINCR.LT.1) THEN
  PRINT*,'MIN BIDDING INCREMENT IS LESS THAN 1'
  PRINT*,'STARTINCR IS SET AT THE DEFAULT VALUE OF 1'
  STARTINCR=1
END IF

C   ******* INITIALIZATION *******
 
EPSILON=BEGEPS
      ILARGE=-ISMALL
      LARGEINCR=INT(ILARGE/10)
      THRESH=INT(0.2*N)
INCRFACTOR=2
      IF (THRESH.GT.100) THRESH=100
X       CYCLES=1
X       AVERAGE=N
       NUMPHASES=1
  DO 10 J=1,N
         ASSIGN(J)=0
         PCOL(J)=ISMALL
10 CONTINUE

      FOUT(N+1)=A+1
      NOLIST=N
      DO 20 I=1,N
         LIST(I)=I
20 CONTINUE

C ***************************************************************
C
C  THIS IMPLEMENTATION OF THE AUCTION ALGORITHM OPERATES IN CYCLES.
C  EACH CYCLE CONSISTS OF ONE BID BY EACH OF THE ROWS THAT ARE
C  UNASSIGNED AT THE START OF THE CYCLE (THESE ROWS ARE STORED IN
C  THE ARRAY LIST(.)). AS THE CYCLE PROGRESSES NEW
C  ROWS BECOME UNASSIGNED; THESE ARE STORED IN LIST(.)
C  AND WILL SUBMIT A BID AT THE NEXT CYCLE.
C  NOTE THAT ONLY ONE ROW SUBMITS A BID AT A TIME; THIS IS KNOWN AS
C  THE GAUSS-SEIDEL VERSION OF THE ALGORITHM. THE VERSION WHERE ALL
C  UNASSIGNED ROWS SUBMIT A BID AT THE SAME TIME IS KNOWN AS THE JACOBI
C  VERSION OF THE ALGORITHM, AND HAS NOT BEEN IMPLEMENTED. IT GENERALLY
C  TENDS TO RUN SOMEWHAT SLOWER THAN THE GAUSS-SEIDEL VERSION, BUT
C  IT ADMITS A HIGHER DEGREE OF PARALLELIZATION. A JACOBI VERSION
C  MAY BE PREFERABLE ON A PARALLEL MACHINE, BUT IS USUALLY INFERIOR
```

```
C  TO THE GAUSS-SEIDEL VERSION ON A SERIAL MACHINE.
C
C ***************************************************************
C
C      START SUBPROBLEM (SCALING PHASE) W/ NEW EPSILON
C
C ***************************************************************

12     CONTINUE

      IF (EPSILON.EQ.1) THRESH=0
       INCR=STARTINCR
      IF (INCR.GT.EPSILON) INCR=EPSILON

C ***************************************************************
C
C      START AUCTION CYCLE WITH NEW LIST
C
C ***************************************************************

15     CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED ROWS

       NONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED ROWS

       DO 100 I=1,NOLIST
       ROW=LIST(I)
       FSTARC=FOUT(ROW)
       LSTARC=FOUT(ROW+1)-1
       FSTCOL=END(FSTARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC

       IF (FSTARC.EQ.LSTARC) THEN
         PCOL(FSTCOL)=PCOL(FSTCOL)+LARGEINCR
           OLDROW=ASSIGN(FSTCOL)
           ASSIGN(FSTCOL)=ROW
           IF (OLDROW.GT.0) THEN
             NONEWLIST=NONEWLIST+1
             LIST(NONEWLIST)=OLDROW
           END IF
         GO TO 100
       END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS

       SNDARC=FSTARC+1
       SNDCOL=END(SNDARC)
       MAX1=COST(FSTARC)-PCOL(FSTCOL)
       MAX2=COST(SNDARC)-PCOL(SNDCOL)
      IF (MAX1.GE.MAX2) THEN
        BSTCOL=FSTCOL
      ELSE
        TMAX=MAX1
        MAX1=MAX2
        MAX2=TMAX
        BSTCOL=SNDCOL
```

```
      END IF
        IF (SNDARC.LT.LSTARC) THEN
        TRDARC=SNDARC+1
        DO 40 CURARC=TRDARC,LSTARC
          CURCOL=END(CURARC)
          TMAX=COST(CURARC)-PCOL(CURCOL)
          IF (TMAX.GT.MAX2) THEN
      IF (TMAX.GT.MAX1) THEN
        MAX2=MAX1
        MAX1=TMAX
        BSTCOL=CURCOL
      ELSE
        MAX2=TMAX
      END IF
          END IF
   40 CONTINUE
        END IF


C  ROW BIDS FOR BSTCOL INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTCOL, WHILE ANY ROW ASSIGNED TO BSTCOL BECOMES UNASSIGNED

         PCOL(BSTCOL)=PCOL(BSTCOL)+MAX1-MAX2+INCR
      OLDROW=ASSIGN(BSTCOL)
      ASSIGN(BSTCOL)=ROW
      IF (OLDROW.GT.0) THEN
        NONEWLIST=NONEWLIST+1
        LIST(NONEWLIST)=OLDROW
      END IF

  100   CONTINUE

C     ************* END OF AN AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS

X       AVERAGE=(CYCLES*AVERAGE+NOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF THERE ARE STILL 'MANY' UNASSIGNED ROWS, THAT IS, IF THE
C  NUMBER OF UNASSIGNED ROWS IS GREATER THAN
C  THE PARAMETER THRESH. IF NOT, REPLACE CURRENT LIST WITH THE NEW LIST,
C  AND GO FOR ANOTHER CYCLE. OTHERWISE, IF EPSILON > 1, REDUCE EPSILON,
C  RESET THE ASSIGNMENT TO EMPTY AND RESTART AUCTION;
C  IF EPSILON = 1 TERMINATE.
C  ALSO INCREASE THE MINIMAL BIDDING INCREMENT UP TO A MAXIMUM
C  VALUE OF EPSILON (THIS IS THE ADAPTIVE FEATURE)

      INCR=INCR*INCRFACTOR
      IF (INCR.GT.EPSILON) INCR=EPSILON
        IF (NONEWLIST.GT.THRESH) THEN
          NOLIST=NONEWLIST
          GO TO 15
        END IF

C ***************************************************************
C
C             END OF SUBPROBLEM (SCALING PHASE)
C
C ***************************************************************
```

```
C ****** IF EPSILON IS 1 TERMINATE ******

      IF (EPSILON.EQ.1) THEN
      RETURN
      ELSE

C   ELSE REDUCE EPSILON AND RESET THE ASSIGNMENT TO EMPTY

      NUMPHASES=NUMPHASES+1
        EPSILON=INT(EPSILON/FACTOR)
      IF (EPSILON.GT.INCR) EPSILON=INT(EPSILON/FACTOR)
        IF ((EPSILON.LT.1).OR.(EPSILON.LT.ENDEPS)) EPSILON=1
        THRESH=INT(THRESH/FACTOR)

X     PRINT*,'*** END OF A SCALING PHASE; NEW EPSILON=',EPSILON

        TMIN=ILARGE
        DO 200 J=1,N
      IF (PCOL(J).LT.TMIN) TMIN=PCOL(J)
         IF (ASSIGN(J).GT.0) THEN
          NONEWLIST=NONEWLIST+1
          LIST(NONEWLIST)=ASSIGN(J)
          ASSIGN(J)=0
         END IF
200     CONTINUE

C     RESET MINIMUM PRICE TO ISMALL

        INCR=TMIN-ISMALL
      DO 210 J=1,N
      PCOL(J)=PCOL(J)-INCR
210       CONTINUE


C     FINAL PARAMETER UPDATES BEFORE RETURNING FOR ANOTHER SCALING PHASE

        NOLIST=NONEWLIST
      IF (STARTINCR.LT.EPSILON) STARTINCR=FACTOR*STARTINCR
        GO TO 12
    END IF


     END



     SUBROUTINE SETRAN(ISEED)
     IMPLICIT REAL*8 (A-H,O-Z) , INTEGER*4 (I-N)
C*********************************************************************
C PORTABLE CONGRUENTIAL (UNIFORM) RANDOM NUMBER GENERATOR:
C     NEXT_VALUE = [(7**5) * PREVIOUS_VALUE] MODULO[(2**31)-1]
C
C THIS GENERATOR CONSISTS OF TWO ROUTINES:
C   (1) SETRAN - INITIALIZES CONSTANTS AND SEED
C   (2) RRAN   - GENERATES A REAL RANDOM NUMBER
C
C THE GENERATOR REQUIRES A MACHINE WITH AT LEAST 32 BITS OF PRECISION.
C THE SEED (ISEED) MUST BE IN THE RANGE (1,(2**31)-1).
C*********************************************************************
```

```
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IF(ISEED.LT.1) STOP 77
      MULT=16807
      MODUL=2147483647
      I15=2**15
      I16=2**16
      JRAN=ISEED
      RETURN
      END

      REAL FUNCTION RAN()
      IMPLICIT REAL*4 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C RAN GENERATES A REAL RANDOM NUMBER BETWEEN 0 AND 1
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IXHI=JRAN/I16
      IXLO=JRAN-IXHI*I16
      IXALO=IXLO*MULT
      LEFTLO=IXALO/I16
      IXAHI=IXHI*MULT
      IFULHI=IXAHI+LEFTLO
      IRTLO=IXALO-LEFTLO*I16
      IOVER=IFULHI/I15
      IRTHI=IFULHI-IOVER*I15
      JRAN=((IRTLO-MODUL)+IRTHI*I16)+IOVER
      IF(JRAN.LT.0) JRAN=JRAN+MODUL
      RAN = FLOAT(JRAN)/FLOAT(MODUL)
      RETURN
      END
```

\subsect{AUCTION_FLP}This code is the same as the preceding one except that it uses floating point arithmetic when updating prices. This is useful when the cost range and the dimension of the problem are large, in which case the prices may overflow the integer range in the course of the algorithm. However, the floating point arithmetic slows down the code somewhat.

```
C ***************************************************************
C
C  SAMPLE CALLING PROGRAM FOR THE AUCTION ALGORITHM
C
C   THIS DRIVER CREATES A SYMMETRIC ASSIGNMENT PROBLEM
C   WITH EQUAL NUMBER OF ROWS AND COLUMNS,
C   AND CALLS THE AUCTION SUBROUTINE TO FIND AN
C   ASSIGNMENT OF MAXIMAL VALUE.
C
C   MODIFIED BY DAVID CASTANON (OCT. 1991) TO WORK WITH ARBITRARILY
C   LARGE COST RANGE (UP TO THE INTEGER RANGE OF THE MACHINE);
C   TO ACCOMPLISH THIS, THE OBJECT PRICES AND EPSILON ARE
C   DOUBLE PRECISION VARIABLES, SO THE PRICES CANNOT OVERFLOW THE
C   MACHINE'S INTEGER RANGE.
C
C ***************************************************************

 PARAMETER(MAXNODES=10000, MAXARCS=100000)
 
       IMPLICIT NONE
```

```
      INTEGER N,NA,A,ISMALL,CYCLES
      INTEGER NUMPHASES
      INTEGER I,J,IA,ARC,NOASS,ICOST,ABSCOST,CURARC
      INTEGER CURCOL,FSTARC,LSTARC,MINCOST,MAXCOST,MCOST
     1    INTEGER FOUT(MAXNODES),COLASS(MAXNODES)
      INTEGER ASSIGN(MAXNODES)
      INTEGER COST(MAXARCS),END(MAXARCS)
     1    REAL*8 TT1,TT2,TCOST
      REAL*8 PCOL(MAXNODES)
      REAL*8 AVERAGE
      REAL*8 BEGEPS,ENDEPS,FACTOR,STARTINCR,TEST,PROFIT
      COMMON/ARRAYC/COST/ARRAYS/END/ARRAYF/FOUT/BK1/N,A,ISMALL
     $ /BK2/CYCLES,AVERAGE,NUMPHASES/ARRAYA/ASSIGN/PCOLA/PCOL

C ***************************************************************
C
C     PROBLEM GENERATION CODE STARTS HERE
C     THE USER MAY REPLACE THIS CODE WITH A CODE THAT READS
C     HIS/HER PROBLEM FROM A FILE
C
C ***************************************************************

C     THIS CODE INCLUDES A UNIFORM RANDOM NUMBER GENERATOR
C     WHICH RETURNS A VALUE IN (0,1)

C     INITIALIZE RANDOM GENERATOR

      INTEGER MULT,MODUL,I15,I16,JRAN,ISEED
      REAL RAN
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN

      ISEED=13502460
      CALL SETRAN(ISEED)

      PRINT*,'GENERATING A SYMMETRIC ASSIGNMENT PROBLEM'
      PRINT*,'*****************************************'

C **** READ THE NUMBER OF ROWS N & THE NUMBER OF ARCS A ****

      PRINT*,'ENTER THE NUMBER OF ROWS (AND COLUMNS)'
      READ*,N
5     PRINT*,'ENTER THE NUMBER OF ARCS PER ROW (>1)'
      READ*,NA
      IF (NA.LT.2) GOTO 5
      PRINT*,'ENTER THE MINIMUM AND THE MAXIMUM COST'
      READ*,MINCOST,MAXCOST

C     THE NUMBER OF ARCS IS N*NA

      A=N*NA

C     THE ARCS INCIDENT TO ROW I ARE FOUT(I) TO FOUT(I+1)-1
C     ALSO, FOR FEASIBILITY EACH ROW IS DIRECTLY CONNECTED
C     WITH THE CORRESPONDING COLUMN

      DO 20 I=1,N
       FOUT(I)=1+(I-1)*NA
20    CONTINUE
      FOUT(N+1)=A+1
```

```
C     GENERATE THE END(ARC) AND COST(ARC) WHICH ARE THE COLUMN
C     AND THE COST COEFFICIENT ASSOCIATED WITH ARC

      DO 25 ARC=1,A
        END(ARC)=1+RAN()*N
      IF ((END(ARC).GT.N).OR.(END(ARC).LT.1)) THEN
        PRINT*,'ERROR IN PROBLEM GENERATION'
        PAUSE
        STOP
      END IF
      COST(ARC)=MINCOST+RAN()*(MAXCOST-MINCOST)
25    CONTINUE

C   MODIFY THE END OF THE LAST ARC OUT OF EACH ROW FOR FEASIBILITY
C   AND SET ITS COST TO -MAXCOST

      DO 30 I=1,N
        END(FOUT(I+1)-1)=I
      COST(FOUT(I+1)-1)=-MAXCOST
30    CONTINUE


C
C ****************************************************************
C
C         PROBLEM GENERATION CODE ENDS HERE
C
C ****************************************************************

C      SCALE THE COST TO WORK WITH INTEGER EPSILON

       MAXCOST=0
       DO 35 IA=1,A
         ABSCOST=IABS(COST(IA))
         IF (ABSCOST.GT.MAXCOST) MAXCOST=ABSCOST
35    CONTINUE

C *** ISMALL IS A VERY SMALL INTEGER FOR YOUR MACHINE ***

       ISMALL=-2000000000
      
      IF (MAXCOST.GT.ABS(ISMALL)) THEN
        PRINT*,'THE COST RANGE IS TOO LARGE FOR INTEGER ARITHMETIC'
        PAUSE
        STOP
      END IF
      

C     THE FOLLOWING PARAMETERS BEGEPS, FACTOR, ENDEPS, AND STARTINCR
C       ARE PASSED TO THE AUCTION ALGORITHM. VALUES BETWEEN
C        (A) MAXCOST/5 AND MAXCOST/2 FOR BEGEPS
C        (B) 4 AND 6 FOR FACTOR
C       (C) 1/10 AND 1/(N+1) FOR ENDEPS
C        (D) 1 AND BEGEPS FOR STARTINCR
C       HAVE WORKED WELL FOR LARGE SPARSE PROBLEMS.
C       FOR DENSE PROBLEMS IT IS RECOMMENDED THAT
C      BEGEPS BE SET TO A SMALLER VALUE,
C      ENDEPS BE SET TO 1/(N+1),
C       STARTINCR BE SET TO 1/(N+1).
```

```
      PRINT*,'*******************************************'
      PRINT*,'MAXIMUM COST IS ',MAXCOST
      PRINT*,'ENTER THE STARTING EPSILON'
      READ*,BEGEPS
      IF (BEGEPS.LT.1.0/(N+1)) BEGEPS=1.0/(N+1)
      PRINT*,'ENTER THE EPSILON REDUCTION FACTOR'
      READ*,FACTOR
      ENDEPS=1.0/10.0
      IF (ENDEPS.LT.1.0/(N+1)) ENDEPS=1.0/(N+1)
      IF (ENDEPS.GT.BEGEPS) ENDEPS=BEGEPS
      STARTINCR=BEGEPS/10.0
      IF (STARTINCR.LT.1.0/(N+1)) STARTINCR=1.0/(N+1)

      PRINT*,'*******************************************'
      PRINT*,'STARTING EPSILON = ',BEGEPS
      PRINT*,'EPSILON REDUCTION FACTOR = ',FACTOR
      PRINT*,'THRESHOLD EPSILON BEFORE IT IS SET TO 1/(N+1) = ',ENDEPS
      PRINT*,'STARTING MIN BIDDING INCREMENT = ',STARTINCR
      PRINT*,'CALLING AUCTION TO SOLVE THE PROBLEM'
      PRINT*,'*******************************************'

C     GET STARTING TIME FOR THE MAC II

      TT1 = LONG(362)/60.0

      CALL AUCTION(BEGEPS,FACTOR,ENDEPS,STARTINCR)

C     GET ENDING TIME FOR THE MAC II

      TT2 = LONG(362)/60.0 - TT1
      PRINT *,'FINISHED --- TOTAL CPU TIME', TT2,' SECS'
      PRINT*,'*******************************************'

C     *** DISPLAY RESULTS ***

X       WRITE(9,2010) CYCLES
X2010   FORMAT(' NO OF AUCTION CYCLES',I7)
X       WRITE(9,2020) AVERAGE
X2020   FORMAT(' AVERAGE NUMBER OF BIDS PER CYCLE',F9.3)
      WRITE(9,2030) NUMPHASES
2030   FORMAT('NO OF EPSILON SUBPROBLEMS SOLVED =',I7)

C      CHECK FEASIBILITY OF SOLUTION & CALCULATE COST

      DO 40 I=1,N
        COLASS(I)=0
40    CONTINUE
      NOASS=0
      DO 50 J=1,N
        IF (ASSIGN(J).GT.0) THEN
          NOASS=NOASS+1
          COLASS(ASSIGN(J))=J
        END IF
50    CONTINUE
      IF (NOASS.NE.N) THEN
        PRINT*,'# OF ASSIGNED ROWS NOT EQUAL TO # OF ROWS'
      END IF
      TCOST=0
      DO 60 I=1,N
       J=COLASS(I)
```

```
        IF (J.EQ.0) THEN
          PRINT*,'ROW ',I,' IS UNASSIGNED'
        END IF
        FSTARC=FOUT(I)
        LSTARC=FOUT(I+1)-1
  MCOST=ISMALL
        DO 70 CURARC=FSTARC,LSTARC
          CURCOL=END(CURARC)
          IF (CURCOL.EQ.J) THEN
            IF (MCOST.LT.COST(CURARC)) THEN
         MCOST=COST(CURARC)
      END IF
    END IF
70   CONTINUE
        PROFIT=MCOST-PCOL(J)
        DO 72 CURARC=FSTARC,LSTARC
     J=END(CURARC)
     TEST=COST(CURARC)-PCOL(J)-PROFIT
     IF (TEST.GT.1.0/N) THEN
        PRINT *,'1/(N+1)-CS VIOLATED AT ARC',CURARC
     END IF
72       CONTINUE
  TCOST=TCOST+MCOST
60 CONTINUE
       WRITE(9,2100) TCOST
2100    FORMAT(' ASSIGNMENT COST=',F22.2)
       PRINT *, ' PROGRAM ENDED; <CR> TO EXIT '
 PAUSE
 END



C ***************************************************************
C
C    FLOATING POINT AUCTION CODE FOR N BY N ASSIGNMENT PROBLEMS
C
C      WRITTEN BY DIMITRI P. BERTSEKAS
C  (MODIFICATIONS BY DAVID CASTANON AND PAUL TSENG)
C
C           VERSION 1.2, OCT. 1991
C
C
C MODIFIED BY DAVID CASTANON (OCT. 1991) TO WORK WITH ARBITRARILY
C LARGE COST RANGE (UP TO THE INTEGER RANGE OF THE MACHINE);
C TO ACCOMPLISH THIS, THE OBJECT PRICES AND EPSILON ARE
C DOUBLE PRECISION VARIABLES, SO THE PRICES CANNOT OVERFLOW THE
C MACHINE'S INTEGER RANGE.
C
C THIS CODE IMPLEMENTS THE AUCTION ALGORITHM WITH E-SCALING.
C IT SOLVES A SEQUENCE OF SUBPROBLEMS AND DECREASES
C EPSILON BY A CONSTANT FACTOR BETWEEN SUBPROBLEMS.
C THIS VERSION CORRESPONDS TO A GAUSS-SEIDEL MODE
C AND SOLVES EPSILON SUBPROBLEMS INEXACTLY.
C
C THE CODE IS AN IMPROVED VERSION OF AN EARLIER (SEPT. 1985)
C AUCTION CODE WITH E-SCALING WRITTEN BY DIMITRI P. BERTSEKAS
C
C THE CODE TREATS THE PROBLEM AS A MAXIMIZATION PROBLEM.
C TO SOLVE A MINIMIZATION PROBLEM, REVERSE THE SIGN OF THE
C ARC COSTS PRIOR TO CALLING AUCTION, AND REVERSE AGAIN
```

```
C THE SIGN OF THE OPTIMAL COST UPON RETURN FROM AUCTION.
C THIS CODE ALLOWS MULTIPLE ARCS BETWEEN A ROW AND A COLUMN.
C
C THIS VERSION OF THE AUCTION ALGORITHM IS ADAPTIVE. HERE THE MINIMAL
C BIDDING INCREMENT (STORED IN THE VARIABLE INCR) MAY BE SMALLER THAN
C EPSILON. FOR EVERY SUBPROBLEM, INCR STARTS AT THE PARAMETER VALUE
C STARTINCR (WHICH IS PASSED TO THE AUCTION ROUTINE) AND IS INCREASED
C BY A FACTOR OF 2 AT THE END OF EACH CYCLE, UP TO A MAXIMUM VALUE OF
C EPSILON. THIS ADAPTIVE FEATURE IS PARTICULARLY EFFECTIVE FOR
C DENSE PROBLEMS. IT CAN BE DEFEATED BY SELECTING STARTINCR=BEGEPS
C
C ***************************************************************
C
C THE USER MUST SUPPLY THE FOLLOWING PROBLEM DATA IN FORWARD STAR FORMAT
C (THAT IS, ALL ARCS OF THE SAME ROW ARE NUMBERED CONSECUTIVELY):
C  N=NUMBER OF ROWS (EQUALS NUMBER OF COLUMNS)
C  A=NUMBER OF ARCS
C  FOUT(ROW)=FIRST ARC COMING OUT OF ROW
C  COST(ARC)=COST OF ARC
C  END(ARC)=COLUMN CORRESPONDING TO ARC
C
C AND THE FOLLOWING PARAMETERS FOR THE AUCTION ALGORITHM:
C  BEGEPS=STARTING VALUE OF EPSILON (MUST BE NO LESS THAN 1/(N+1))
C  ENDEPS=FINAL VALUE OF EPSILON BEFORE IT IS SET TO 1/(N+1)
C  FACTOR=FACTOR BY WHICH EPSILON IS DECREASED BETWEEN SUBPROBLEMS
C      STARTINCR=THE STARTING VALUE OF THE BIDDING INCREMENT
C      ENDEPS SHOULD NOT EXCEED BEGEPS.
C      FACTOR MUST BE GREATER THAN 1.
C
C FOUT(.) IS AN ARRAY OF LENGTH N.
C COST(.),END(.) ARE ARRAYS OF LENGTH A.
C
C THE SOLUTION IS CONTAINED IN THE ARRAY ASSIGN(.) WHERE
C      ASSIGN(COL) GIVES THE ROW ASSIGNED TO COL.
C
C THIS ALGORITHM DOES NOT CHECK FOR INFEASIBILITY OF THE PROBLEM.
C TO MAKE SURE THE PROBLEM IS FEASIBLE THE USER MAY ADD
C ADDITIONAL VERY SMALL COST ARCS.
C
C ***************************************************************
C ALL PROBLEM DATA ARE INTEGER
C ***************************************************************


 SUBROUTINE AUCTION(BEGEPS,FACTOR,ENDEPS,STARTINCR)

 PARAMETER(MAXNODES=10000, MAXARCS=100000)
 
      IMPLICIT NONE
      INTEGER NONEWLIST,THRESH
      INTEGER A,K,N,I,J,CURARC,CURCOL
      INTEGER ROW,FSTARC,FSTCOL,SNDARC,SNDCOL,BSTCOL
      INTEGER TRDARC
      INTEGER M,ISMALL,ILARGE,LARGEINCR,CYCLES
      INTEGER NUMPHASES,LSTARC,NOLIST,OLDROW
      INTEGER FOUT(MAXNODES),LIST(MAXNODES)
 INTEGER ASSIGN(MAXNODES)
      INTEGER COST(MAXARCS),END(MAXARCS)
 REAL*8 BEGEPS,ENDEPS,FACTOR,STARTINCR,INCRFACTOR
 REAL*8 PCOL(MAXNODES)
```

```
□REAL*8 MAX1,MAX2,TMAX,EPSILON,INCR
      REAL*8 AVERAGE
□COMMON/ARRAYC/COST/ARRAYS/END/ARRAYF/FOUT/BK1/N,A,ISMALL
   $□/BK2/CYCLES,AVERAGE,NUMPHASES/ARRAYA/ASSIGN/PCOLA/PCOL

C ***************************************************************

C   ******* CHECK VALIDITY OF PARAMETERS PASSED *******

      IF (BEGEPS.LE.1.0/(N+2)) THEN
□  PRINT*,'STARTING VALUE OF EPSILON IS LESS THAN 1/(N+1)'
□  PRINT*,'EXECUTION ABORTED'
□  STOP
□END IF
□IF (ENDEPS.GT.BEGEPS) THEN
□  PRINT*,'PARAMETER ENDEPS IS GREATER THAN PARAMETER BEGEPS'
□  PRINT*,'ENDEPS IS SET AT THE DEFAULT VALUE OF 1/(N+1)'
□  ENDEPS=1.0/(N+1)
□END IF
      IF ((FACTOR.LE.1).AND.(BEGEPS.GE.1.0/N)) THEN
□  PRINT*,'EPSILON REDUCTION FACTOR IS NOT GREATER THAN 1'
□  PRINT*,'EXECUTION ABORTED'
□  STOP
□END IF□
      IF (STARTINCR.LE.1.0/(N+2)) THEN
□  PRINT*,'MIN BIDDING INCREMENT IS LESS THAN 1/(N+1)'
□  PRINT*,'STARTINCR IS SET AT THE DEFAULT VALUE OF 1/(N+1)'
□  STARTINCR=1.0/(N+1)
□END IF

C   ******* INITIALIZATION *******
 □
□EPSILON=BEGEPS
      ILARGE=-ISMALL
      LARGEINCR=INT(ILARGE/10)
      THRESH=INT(0.2*N)
□INCRFACTOR=2.0
      IF (THRESH.GT.100) THRESH=100
X       CYCLES=1
X       AVERAGE=N
       NUMPHASES=1
□ DO 10 J=1,N
         ASSIGN(J)=0
         PCOL(J)=ISMALL
10□CONTINUE

      FOUT(N+1)=A+1
      NOLIST=N
      DO 20 I=1,N
         LIST(I)=I
20□CONTINUE

C ***************************************************************
C
C  THIS IMPLEMENTATION OF THE AUCTION ALGORITHM OPERATES IN CYCLES.
C  EACH CYCLE CONSISTS OF ONE BID BY EACH OF THE ROWS THAT ARE
C  UNASSIGNED AT THE START OF THE CYCLE (THESE ROWS ARE STORED IN
C  THE ARRAY LIST(.)). AS THE CYCLE PROGRESSES NEW
C  ROWS BECOME UNASSIGNED; THESE ARE STORED IN LIST(.)
C  AND WILL SUBMIT A BID AT THE NEXT CYCLE.
```

```
C  NOTE THAT ONLY ONE ROW SUBMITS A BID AT A TIME; THIS IS KNOWN AS
C  THE GAUSS-SEIDEL VERSION OF THE ALGORITHM. THE VERSION WHERE ALL
C  UNASSIGNED ROWS SUBMIT A BID AT THE SAME TIME IS KNOWN AS THE JACOBI
C  VERSION OF THE ALGORITHM, AND HAS NOT BEEN IMPLEMENTED. IT GENERALLY
C  TENDS TO RUN SOMEWHAT SLOWER THAN THE GAUSS-SEIDEL VERSION, BUT
C  IT ADMITS A HIGHER DEGREE OF PARALLELIZATION. A JACOBI VERSION
C  MAY BE PREFERABLE ON A PARALLEL MACHINE, BUT IS USUALLY INFERIOR
C  TO THE GAUSS-SEIDEL VERSION ON A SERIAL MACHINE.
C
C ****************************************************************
C
C      START SUBPROBLEM (SCALING PHASE) W/ NEW EPSILON
C
C ****************************************************************

12     CONTINUE

      IF (EPSILON.LE.1.0/(N+1)) THRESH=0
       INCR=STARTINCR
      IF (INCR.GT.EPSILON) INCR=EPSILON

C ****************************************************************
C
C      START AUCTION CYCLE WITH NEW LIST
C
C ****************************************************************

15     CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED ROWS

       NONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED ROWS

       DO 100 I=1,NOLIST
       ROW=LIST(I)
       FSTARC=FOUT(ROW)
       LSTARC=FOUT(ROW+1)-1
       FSTCOL=END(FSTARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC

       IF (FSTARC.EQ.LSTARC) THEN
        PCOL(FSTCOL)=PCOL(FSTCOL)+LARGEINCR
        OLDROW=ASSIGN(FSTCOL)
        ASSIGN(FSTCOL)=ROW
        IF (OLDROW.GT.0) THEN
          NONEWLIST=NONEWLIST+1
          LIST(NONEWLIST)=OLDROW
        END IF
         GO TO 100
       END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS

       SNDARC=FSTARC+1
       SNDCOL=END(SNDARC)
       MAX1=COST(FSTARC)-PCOL(FSTCOL)
       MAX2=COST(SNDARC)-PCOL(SNDCOL)
```

```
IF (MAX1.GE.MAX2) THEN
   BSTCOL=FSTCOL
ELSE
   TMAX=MAX1
   MAX1=MAX2
   MAX2=TMAX
   BSTCOL=SNDCOL
END IF
       IF (SNDARC.LT.LSTARC) THEN
       TRDARC=SNDARC+1
       DO 40 CURARC=TRDARC,LSTARC
         CURCOL=END(CURARC)
         TMAX=COST(CURARC)-PCOL(CURCOL)
         IF (TMAX.GT.MAX2) THEN
     IF (TMAX.GT.MAX1) THEN
       MAX2=MAX1
       MAX1=TMAX
       BSTCOL=CURCOL
     ELSE
       MAX2=TMAX
     END IF
         END IF
40 CONTINUE
       END IF


C  ROW BIDS FOR BSTCOL INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTCOL, WHILE ANY ROW ASSIGNED TO BSTCOL BECOMES UNASSIGNED

        PCOL(BSTCOL)=PCOL(BSTCOL)+MAX1-MAX2+INCR
  OLDROW=ASSIGN(BSTCOL)
  ASSIGN(BSTCOL)=ROW
  IF (OLDROW.GT.0) THEN
    NONEWLIST=NONEWLIST+1
    LIST(NONEWLIST)=OLDROW
  END IF

100    CONTINUE

C      ************* END OF AN AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS

X       AVERAGE=(CYCLES*AVERAGE+NOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF THERE ARE STILL 'MANY' UNASSIGNED ROWS, THAT IS, IF THE
C  NUMBER OF UNASSIGNED ROWS IS GREATER THAN
C  THE PARAMETER THRESH. IF NOT, REPLACE CURRENT LIST WITH THE NEW LIST,
C  AND GO FOR ANOTHER CYCLE. OTHERWISE, IF EPSILON > 1, REDUCE EPSILON,
C  RESET THE ASSIGNMENT TO EMPTY AND RESTART AUCTION;
C  IF EPSILON = 1 TERMINATE.
C  ALSO INCREASE THE MINIMAL BIDDING INCREMENT UP TO A MAXIMUM
C  VALUE OF EPSILON (THIS IS THE ADAPTIVE FEATURE)

INCR=INCR*INCRFACTOR
IF (INCR.GT.EPSILON) INCR=EPSILON
       IF (NONEWLIST.GT.THRESH) THEN
          NOLIST=NONEWLIST
          GO TO 15
```

```
      END IF

C ************************************************************
C
C             END OF SUBPROBLEM (SCALING PHASE)
C
C ************************************************************

C ****** IF EPSILON IS 1 TERMINATE ******

     IF (EPSILON.LE.1.0/N) THEN
      RETURN
     ELSE

C   ELSE REDUCE EPSILON AND RESET THE ASSIGNMENT TO EMPTY

      NUMPHASES=NUMPHASES+1
        EPSILON=EPSILON/FACTOR
      IF (EPSILON.GT.INCR) EPSILON=EPSILON/FACTOR
        IF ((EPSILON.LT.1.0/N).OR.(EPSILON.LT.ENDEPS)) THEN
       EPSILON=1.0/(N+1)
      END IF
        THRESH=INT(THRESH/FACTOR)

X     PRINT*,'*** END OF A SCALING PHASE; NEW EPSILON=',EPSILON

        DO 200 J=1,N
         IF (ASSIGN(J).GT.0) THEN
          NONEWLIST=NONEWLIST+1
          LIST(NONEWLIST)=ASSIGN(J)
          ASSIGN(J)=0
         END IF
200     CONTINUE



C     FINAL PARAMETER UPDATES BEFORE RETURNING FOR ANOTHER SCALING PHASE

        NOLIST=NONEWLIST
      IF (STARTINCR.LT.EPSILON) STARTINCR=FACTOR*STARTINCR
        GO TO 12
    END IF


     END



     SUBROUTINE SETRAN(ISEED)
     IMPLICIT REAL*8 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C PORTABLE CONGRUENTIAL (UNIFORM) RANDOM NUMBER GENERATOR:
C     NEXT_VALUE = [(7**5) * PREVIOUS_VALUE] MODULO[(2**31)-1]
C
C THIS GENERATOR CONSISTS OF TWO ROUTINES:
C   (1) SETRAN - INITIALIZES CONSTANTS AND SEED
C   (2) RRAN   - GENERATES A REAL RANDOM NUMBER
C
C THE GENERATOR REQUIRES A MACHINE WITH AT LEAST 32 BITS OF PRECISION.
C THE SEED (ISEED) MUST BE IN THE RANGE (1,(2**31)-1).
```

```
C************************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IF(ISEED.LT.1) STOP 77
      MULT=16807
      MODUL=2147483647
      I15=2**15
      I16=2**16
      JRAN=ISEED
      RETURN
      END

      REAL FUNCTION RAN()
      IMPLICIT REAL*4 (A-H,O-Z) , INTEGER*4 (I-N)
C************************************************************************
C RAN GENERATES A REAL RANDOM NUMBER BETWEEN 0 AND 1
C************************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IXHI=JRAN/I16
      IXLO=JRAN-IXHI*I16
      IXALO=IXLO*MULT
      LEFTLO=IXALO/I16
      IXAHI=IXHI*MULT
      IFULHI=IXAHI+LEFTLO
      IRTLO=IXALO-LEFTLO*I16
      IOVER=IFULHI/I15
      IRTHI=IFULHI-IOVER*I15
      JRAN=((IRTLO-MODUL)+IRTHI*I16)+IOVER
      IF(JRAN.LT.0) JRAN=JRAN+MODUL
      RAN = FLOAT(JRAN)/FLOAT(MODUL)
      RETURN
      END
```

\subsect{AUCTION-AS}This code implements the forward/reverse auction
algorithm with $\e$-scaling for the asymmetric assignment problem; cf.\
Section 4.2.1 of the author's network optimization book.
It can also solve as a special case the symmetric assignment problem.
In this case as well as in the case where $\e$-scaling is bypassed ($\e=1$ throughout the
algorithm), only the forward part of the algorithm is used. Note also that this code uses the
``third best'' object implementation described in Exercise 1.7 of Section 4.1 of the network
optimization book.

```
C ***************************************************************
C
C  SAMPLE CALLING PROGRAM FOR AUCTION ALGORITHM
C         FOR THE ASYMMETRIC ASSIGNMENT PROBLEM
C
C   THIS DRIVER CREATES AN ASYMMETRIC ASSIGNMENT PROBLEM
C   WITH LESS OR EQUAL NUMBER OF ROWS THAN COLUMNS,
C   AND CALLS THE AUCTION_AS SUBROUTINE TO FIND AN
C   ASSIGNMENT OF MAXIMAL VALUE.
C
C   THIS VERSION USES A THIRD BEST VALUE
C
C ***************************************************************

 PARAMETER(MAXNODES=10000, MAXARCS=100000)
```

```
      IMPLICIT NONE
      INTEGER M,N,NA,A,IA,K,ILARGE,BEGEPS,ENDEPS,CYCLES
      INTEGER NUMPHASES,STARTINCR,LAMBDA
      INTEGER I,J,ARC,NOASS,ICOST,ABSCOST,CURARC
      INTEGER CURCOL,FSTARC,LSTARC,MINCOST,MAXCOST,MCOST
      INTEGER EXTRA,REMAINDER,INDEX,COUNT
      INTEGER COL_ASSIGNED_TO(MAXNODES),ROW_ASSIGNED_TO(MAXNODES)
      INTEGER FOUT(MAXNODES),FIN(MAXNODES),NXTIN(MAXARCS)
      INTEGER COST(MAXARCS),START(MAXARCS),END(MAXARCS)
      INTEGER PCOL(MAXNODES),PROW(MAXNODES)
      INTEGER PRDARC(MAXNODES)
      REAL*8 FACTOR,TT1,TT2,TCOST,AVERAGE
      COMMON/ARRAYC/COST/ARRAYS/START/ARRAYE/END
      COMMON/ARRAYFO/FOUT/ARRAYFI/FIN/ARRAYNI/NXTIN
      COMMON/ARRAYRA/ROW_ASSIGNED_TO/ARRAYCA/COL_ASSIGNED_TO
      COMMON/ARRAYPC/PCOL/ARRAYPR/PROW
      COMMON/BK1/M,N,A,ILARGE
      COMMON/BK2/CYCLES,AVERAGE,NUMPHASES,LAMBDA

C ***************************************************************
C
C     PROBLEM GENERATION CODE STARTS HERE
C     THE USER MAY REPLACE THIS CODE WITH A CODE THAT READS
C     HIS/HER PROBLEM FROM A FILE
C
C ***************************************************************

C     THIS CODE INCLUDES A UNIFORM RANDOM NUMBER GENERATOR
C     WHICH RETURNS A VALUE IN (0,1)

C     INITIALIZE RANDOM GENERATOR

      INTEGER MULT,MODUL,I15,I16,JRAN,ISEED
      REAL RAN
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN

      ISEED=13502460
      CALL SETRAN(ISEED)

      PRINT*,'GENERATING AN ASYMMETRIC ASSIGNMENT PROBLEM'
      PRINT*,'******************************************'

C **** READ THE NUMBER OF ROWS M, COLUMNS N, AND ARCS A ****

      PRINT*,'ENTER THE NUMBER OF ROWS'
      READ*,M

2     PRINT*,'ENTER THE NUMBER OF COLUMNS (>= # OF ROWS)'
      READ*,N
      IF (N.LT.M) GO TO 2

5     PRINT*,'ENTER THE NUMBER OF ARCS PER ROW (>1)'
      READ*,NA
      IF (NA.LT.2) GOTO 5

      PRINT*,'ENTER THE MINIMUM AND THE MAXIMUM COST'
      READ*,MINCOST,MAXCOST

C     THE NUMBER OF ARCS IS M*NA+N-M
```

```
      A=M*NA+N-M
      PRINT*,'THE NUMBER OF ARCS IS ',A

C     THE ARCS INCIDENT TO ROW I ARE FOUT(I) TO FOUT(I+1)-1
C     ALSO, FOR FEASIBILITY EACH ROW IS DIRECTLY CONNECTED
C     WITH THE CORRESPONDING COLUMN;
C     ALSO EACH COLUMN HAS AT LEAST ONE ARC

      EXTRA=INT((N-M)/M)

      DO 20 I=1,M
        FOUT(I)=1+(I-1)*(NA+EXTRA)
20    CONTINUE
      FOUT(M+1)=A+1

      DO 22 I=1,M
        DO 23 ARC=FOUT(I),FOUT(I+1)-1
         START(ARC)=I
23      CONTINUE
22    CONTINUE

C     GENERATE THE END(ARC) AND COST(ARC) WHICH ARE THE COLUMN
C     AND THE COST COEFFICIENT ASSOCIATED WITH ARC

      DO 25 ARC=1,A
        END(ARC)=1+RAN()*N
IF ((END(ARC).GT.N).OR.(END(ARC).LT.1)) THEN
         PRINT*,'ERROR IN PROBLEM GENERATION'
         PAUSE
         STOP
END IF
COST(ARC)=MINCOST+RAN()*(MAXCOST-MINCOST)
25    CONTINUE

C   MODIFY THE END OF THE LAST ARC OUT OF EACH ROW FOR FEASIBILITY
C   AND SET ITS COST TO -MAXCOST

      DO 30 I=1,M
        END(FOUT(I+1)-1)=I
COST(FOUT(I+1)-1)=-MAXCOST
30    CONTINUE

C   MODIFY THE END OF SOME ARCS SO THAT EACH COLUMN HAS AT LEAST ONE ARC

      IF (EXTRA.GE.1) THEN
        DO 32 I=1,M
         DO 33 K=1,EXTRA
          END(FOUT(I)+K-1)=K*M+I
33       CONTINUE
32      CONTINUE
       END IF

       REMAINDER=N-M*(1+EXTRA)
       IF (REMAINDER.GE.1) THEN
         INDEX=FOUT(M)+EXTRA-1
         DO 35 J=1,REMAINDER
          END(INDEX+J)=(1+EXTRA)*M+J
35       CONTINUE
       END IF
```

```
C    CONSTRUCT THE FIN() AND NXTIN() ARRAYS

     DO 42 J=1,N
FIN(J)=0
      PRDARC(J)=0
42    CONTINUE

     DO 43 ARC=1,A
      NXTIN(ARC)=0
      J=END(ARC)
      IF (FIN(J).NE.0) THEN
        NXTIN(PRDARC(J))=ARC
      ELSE
        FIN(J)=ARC
      END IF
      PRDARC(J)=ARC
43    CONTINUE


C ***************************************************************
C
C         PROBLEM GENERATION CODE ENDS HERE
C
C ***************************************************************



C      SCALE THE COST TO WORK WITH INTEGER EPSILON

      MAXCOST=0
      DO 45 IA=1,A
        ABSCOST=IABS(COST(IA))
        IF (ABSCOST.GT.MAXCOST) MAXCOST=ABSCOST
        COST(IA)=COST(IA)*(N+1)
45 CONTINUE

C *** ILARGE IS A VERY LARGE INTEGER FOR YOUR MACHINE ***

      ILARGE=2000000000

IF (MAXCOST.GT.INT(ILARGE/(N+1))) THEN
  PRINT*,'THE COST RANGE IS TOO LARGE FOR INTEGER ARITHMETIC'
  PAUSE
  STOP
END IF

MAXCOST=MAXCOST*(N+1)

LAMBDA=-ILARGE

C  THE FOLLOWING PARAMETERS BEGEPS, FACTOR, ENDEPS, AND STARTINCR
C      ARE PASSED TO THE AUCTION ALGORITHM. VALUES BETWEEN
C       (A) MAXCOST/5 AND MAXCOST/2 FOR BEGEPS
C       (B) 4 AND 6 FOR FACTOR
C (C) M/10 AND 1 FOR ENDEPS
C       (D) 1 AND BEGEPS FOR STARTINCR
C      HAVE WORKED WELL FOR LARGE SPARSE PROBLEMS.
C      FOR DENSE PROBLEMS AND FOR VERY ASYMMETRIC PROBLEMS
C      IT IS RECOMMENDED THAT
```

```
C□BEGEPS BE SET TO A SMALLER VALUE (POSSIBLY 1),
C□ENDEPS BE SET TO 1,
C       STARTINCR BE SET TO 1.

C       FOR VERY ASYMMETRIC PROBLEMS, BEGEPS=1, CORRESPONDING TO
C       NO E-SCALING, MAY WORK BEST AND SHOULD BE AT LEAST TRIED.

        PRINT*,'MAXIMUM COST IS ',MAXCOST
        PRINT*,'ENTER THE STARTING EPSILON'
        READ*,BEGEPS
        IF (BEGEPS.LT.1) BEGEPS=1
        PRINT*,'ENTER THE EPSILON REDUCTION FACTOR'
        READ*,FACTOR
□ENDEPS=M/10
□IF (ENDEPS.LT.1) ENDEPS=1
□IF (ENDEPS.GT.BEGEPS) ENDEPS=BEGEPS
□STARTINCR=BEGEPS/10
□IF (STARTINCR.LT.1) STARTINCR=1

        PRINT*,'*****************************************'
□PRINT*,'STARTING EPSILON = ',BEGEPS
□PRINT*,'EPSILON REDUCTION FACTOR = ',FACTOR
□PRINT*,'THRESHOLD EPSILON BEFORE IT IS SET TO 1 = ',ENDEPS
□PRINT*,'STARTING MIN BIDDING INCREMENT = ',STARTINCR
        PRINT*,'CALLING ASYMMETRIC ASSIGNMENT AUCTION'
        PRINT*,'*****************************************'

C     GET STARTING TIME FOR THE MAC II

      TT1 = LONG(362)/60.0

      CALL AUCTION_AS(BEGEPS,FACTOR,ENDEPS,STARTINCR)

C     GET ENDING TIME FOR THE MAC II

      TT2 = LONG(362)/60.0 - TT1
      PRINT *,'FINISHED --- TOTAL CPU TIME', TT2,' SECS'
      PRINT*,'*****************************************'

C     *** DISPLAY RESULTS ***

X       WRITE(9,2010) CYCLES
X2010   FORMAT(' NO OF AUCTION CYCLES',I7)
X       WRITE(9,2020) AVERAGE
X2020   FORMAT(' AVERAGE NUMBER OF BIDS PER CYCLE',F9.3)
        WRITE(9,2030) NUMPHASES
2030    FORMAT('NO OF EPSILON SUBPROBLEMS SOLVED =',I7)

C       CHECK OPTIMALITY & CALCULATE COST

□DO 48 J=1,N
          I=ROW_ASSIGNED_TO(J)
          IF (N.GT.M) THEN
□    ROW_ASSIGNED_TO(J)=0
□   ELSE
□     IF (I.GT.0) THEN
□       COL_ASSIGNED_TO(I)=J
□     END IF
□   END IF
48□CONTINUE
```

```
      TCOST=0
      DO 50 I=1,M
  J=COL_ASSIGNED_TO(I)
  ROW_ASSIGNED_TO(J)=I
        IF (J.EQ.0) THEN
          PRINT*,'ROW ',I,' IS UNASSIGNED'
        END IF
  IF (PCOL(J).LT.LAMBDA) THEN
    PRINT*,'CS VIOLATION AT ASSIGNED COLUMN ',J
  END IF
        FSTARC=FOUT(I)
        LSTARC=FOUT(I+1)-1
  MCOST=-ILARGE
        DO 55 ARC=FSTARC,LSTARC
         CURCOL=END(ARC)
    IF (PROW(I)+PCOL(CURCOL).LT.COST(ARC)-1) THEN
     PRINT*,'1-CS VIOLATED AT ARC ',ARC
    END IF
          IF (CURCOL.EQ.J) THEN
           IF (MCOST.LT.COST(ARC)) THEN
        MCOST=COST(ARC)
     END IF
    END IF
55  CONTINUE
  TCOST=TCOST+MCOST/(M+1)
  IF (PROW(I)+PCOL(J).NE.MCOST) THEN
    PRINT*,'1-CS VIOLATED AT ROW ',I
  END IF
50CONTINUE

      COUNT=0
DO 60 J=1,N
        IF (ROW_ASSIGNED_TO(J).EQ.0) THEN
    IF (PCOL(J).GT.LAMBDA) THEN
           PRINT*,'CS VIOLATION AT UNASSIGNED COLUMN ',J
    END IF
   ELSE
    COUNT=COUNT+1
       END IF
60CONTINUE

      IF (COUNT.LT.M) THEN
  PRINT*,'THE NUMBER OF ASSIGNED COLUMNS IS WRONG'
END IF


      WRITE(9,2100) TCOST
2100    FORMAT(' ASSIGNMENT COST=',F18.2)
      PRINT *, ' PROGRAM ENDED; <CR> TO EXIT '
PAUSE
END




C ***************************************************************
C
C    AUCTION CODE FOR ASYMMETRIC M BY N ASSIGNMENT PROBLEMS
```

```
C
C        WRITTEN BY DIMITRI P. BERTSEKAS
C
C                DEC. 1990
C
C THIS CODE IMPLEMENTS A FORWARD/REVERSE AUCTION ALGORITHM
C WITH E-SCALING FOR THE ASYMMETRIC ASSIGNMENT PROBLEM.
C IT SOLVES A SEQUENCE OF SUBPROBLEMS AND DECREASES
C EPSILON BY A CONSTANT FACTOR BETWEEN SUBPROBLEMS.
C THIS VERSION CORRESPONDS TO A GAUSS-SEIDEL MODE.
C
C THE CODE TREATS THE PROBLEM AS A MAXIMIZATION PROBLEM.
C TO SOLVE A MINIMIZATION PROBLEM, REVERSE THE SIGN OF THE
C ARC COSTS PRIOR TO CALLING AUCTION, AND REVERSE AGAIN
C THE SIGN OF THE OPTIMAL COST UPON RETURN FROM AUCTION.
C THIS CODE ALLOWS MULTIPLE ARCS BETWEEN A ROW AND A COLUMN.
C
C THIS VERSION OF THE AUCTION ALGORITHM IS ADAPTIVE. HERE THE MINIMAL
C BIDDING INCREMENT (STORED IN THE VARIABLE INCR) MAY BE SMALLER THAN
C EPSILON. FOR EVERY SUBPROBLEM, INCR STARTS AT THE PARAMETER VALUE
C STARTINCR (WHICH IS PASSED TO THE AUCTION ROUTINE) AND IS INCREASED
C BY A FACTOR OF 2 AT THE END OF EACH CYCLE, UP TO A MAXIMUM VALUE OF
C EPSILON. THIS ADAPTIVE FEATURE IS PARTICULARLY EFFECTIVE FOR
C DENSE PROBLEMS. IT CAN BE DEFEATED BY SELECTING STARTINCR=BEGEPS
C
C ***************************************************************
C
C THE USER MUST SUPPLY THE FOLLOWING PROBLEM DATA IN FORWARD STAR FORMAT
C (THAT IS, ALL ARCS OF THE SAME ROW ARE NUMBERED CONSECUTIVELY):
C  M=NUMBER OF ROWS
C  N=NUMBER OF COLUMNS
C  A=NUMBER OF ARCS
C  FOUT(ROW)=FIRST ARC COMING OUT OF ROW
C  FIN(COL)=FIRST ARC COMING INTO COL
C       NXTIN(ARC)=NEXT ARC INCIDENT TO THE SAME COLUMN AS ARC
C  COST(ARC)=COST OF ARC
C       START(ARC)=ROW CORRESPONDING TO ARC
C  END(ARC)=COLUMN CORRESPONDING TO ARC
C
C AND THE FOLLOWING PARAMETERS FOR THE AUCTION ALGORITHM:
C  BEGEPS=STARTING VALUE OF EPSILON (MUST BE NO LESS THAN 1)
C  ENDEPS=FINAL VALUE OF EPSILON BEFORE IT IS SET TO 1
C  FACTOR=FACTOR BY WHICH EPSILON IS DECREASED BETWEEN SUBPROBLEMS
C       STARTINCR=THE STARTING VALUE OF THE BIDDING INCREMENT
C       ENDEPS SHOULD NOT EXCEED BEGEPS.
C       FACTOR MUST BE GREATER THAN 1.
C
C FOUT(.) IS AN ARRAY OF LENGTH N.
C COST(.),END(.) ARE ARRAYS OF LENGTH A.
C
C THE SOLUTION IS CONTAINED IN THE ARRAY ROW_ASSIGNED_TO(.) WHERE
C    ROW_ASSIGNED_TO(COL) GIVES THE ROW ASSIGNED TO COL.
C ALSO COL_ASSIGNED_TO(ROW) GIVES THE COLUMN ASSIGNED TO ROW.
C
C THIS ALGORITHM DOES NOT CHECK FOR INFEASIBILITY OF THE PROBLEM.
C TO MAKE SURE THE PROBLEM IS FEASIBLE THE USER MAY ADD
C    ADDITIONAL VERY SMALL COST ARCS.
C
C THIS CODE MAY FAIL DUE TO INTEGER OVERFLOW IF THE NUMBER OF NODES
C OR THE COST RANGE (OR BOTH) ARE LARGE. TO CORRECT THIS SITUATION,
```

```
C THE PRICES AND OTHER RELATED VARIABLES (MAX1,MAX2,TMAX ETC) SHOULD
C BE DECLARED AS DOUBLE PRECISION REALS.
C ****************************************************************
C ALL PROBLEM DATA ARE INTEGER
C ****************************************************************



      SUBROUTINE AUCTION_AS(BEGEPS,FACTOR,ENDEPS,STARTINCR)

      PARAMETER(MAXNODES=10000, MAXARCS=100000)
      
            IMPLICIT NONE
            INTEGER A,K,N,I,J,M,CURARC,CURCOL,CURROW,LAMBDA,DELTA
            INTEGER THRESH,INCR,STARTINCR,INCRFACTOR,EPS_THRESH
            INTEGER NOLIST,RNOLIST,NONEWLIST,RNONEWLIST
            INTEGER ROW,COLUMN, BSTROW,BSTCOL
            INTEGER FSTARC,FSTCOL,NXTARC,NXTROW,LSTARC,SNDARC,SNDCOL
            INTEGER MAX1,MAX2,TMAX,TMIN,TRDARC,EPSILON,BEGEPS,ENDEPS
            INTEGER ISMALL,ILARGE,LARGEINCR,CYCLES
            INTEGER NUMPHASES,OLDROW,OLDCOL
      INTEGER MAX3,BSTARC,SBSTARC
      INTEGER CUR_BARC,TRDVAL(MAXNODES)
      LOGICAL INIT
      INTEGER BEST_ARC(MAXNODES),SECD_ARC(MAXNODES)
            INTEGER COL_ASSIGNED_TO(MAXNODES),ROW_ASSIGNED_TO(MAXNODES)
            INTEGER FOUT(MAXNODES),FIN(MAXNODES),NXTIN(MAXARCS)
      INTEGER COST(MAXARCS),START(MAXARCS),END(MAXARCS)
      INTEGER LIST(MAXNODES),REV_LIST(MAXNODES)
      INTEGER PCOL(MAXNODES),PROW(MAXNODES)
            REAL*8 AVERAGE,FACTOR
      COMMON/ARRAYC/COST/ARRAYS/START/ARRAYE/END
            COMMON/ARRAYFO/FOUT/ARRAYFI/FIN/ARRAYNI/NXTIN
            COMMON/ARRAYRA/ROW_ASSIGNED_TO/ARRAYCA/COL_ASSIGNED_TO
            COMMON/ARRAYPC/PCOL/ARRAYPR/PROW
            COMMON/BK1/M,N,A,ILARGE
      COMMON/BK2/CYCLES,AVERAGE,NUMPHASES,LAMBDA

C ****************************************************************

C   ******* CHECK VALIDITY OF PARAMETERS PASSED *******

            IF (BEGEPS.LT.1) THEN
         PRINT*,'STARTING VALUE OF EPSILON IS LESS THAN 1'
         PRINT*,'EXECUTION ABORTED'
         STOP
      END IF
      IF (ENDEPS.GT.BEGEPS) THEN
         PRINT*,'PARAMETER ENDEPS IS GREATER THAN PARAMETER BEGEPS'
         PRINT*,'ENDEPS IS SET AT THE DEFAULT VALUE OF 1'
         ENDEPS=1
      END IF
            IF ((FACTOR.LE.1).AND.(BEGEPS.GT.1)) THEN
         PRINT*,'EPSILON REDUCTION FACTOR IS NOT GREATER THAN 1'
         PRINT*,'EXECUTION ABORTED'
         STOP
      END IF
            IF (STARTINCR.LT.1) THEN
         PRINT*,'MIN BIDDING INCREMENT IS LESS THAN 1'
         PRINT*,'STARTINCR IS SET AT THE DEFAULT VALUE OF 1'
```

```
      STARTINCR=1
     END IF

C   ******* INITIALIZATION *******
      
      EPSILON=BEGEPS
      ISMALL=-ILARGE
      LARGEINCR=INT(ILARGE/10)
      THRESH=INT(0.2*M)
      INCRFACTOR=2
      EPS_THRESH=BEGEPS/(FACTOR**3)
      IF (EPS_THRESH.LT.2)  EPS_THRESH=2
      INIT=.FALSE.
      IF (THRESH.GT.100) THRESH=100
X       CYCLES=1
X       AVERAGE=N
        NUMPHASES=1

      DO 10 I=1,N
         PCOL(I)=ISMALL
10    CONTINUE

        FOUT(M+1)=A+1

C ***************************************************************
C
C  THIS IMPLEMENTATION OF THE AUCTION ALGORITHM OPERATES IN CYCLES.
C  EACH CYCLE CONSISTS OF ONE BID BY EACH OF THE ROWS THAT ARE
C  UNASSIGNED AT THE START OF THE CYCLE (THESE ROWS ARE STORED IN
C  THE ARRAY LIST(.)). AS THE CYCLE PROGRESSES NEW
C  ROWS BECOME UNASSIGNED; THESE ARE STORED IN LIST(.)
C  AND WILL SUBMIT A BID AT THE NEXT CYCLE.
C
C  THE ALGORITHM FIRST FINDS A FEASIBLE ASSIGNMENT, WHERE ALL PERSONS
C  ARE ASSIGNED, USING A COMBINED FORWARD/REVERSE AUCTION.
C  THEN THE ALGORITHM USES A MODIFIED REVERSE AUCTION TO TO SATISFY
C  THE REMAINING E-CS CONDITIONS.
C
C ***************************************************************
C
C      START SUBPROBLEM (SCALING PHASE) W/ NEW EPSILON
C
C ***************************************************************

12      CONTINUE

C     INITIALIZE ROW ASSIGNMENT LISTS

        NOLIST=M
        DO 20 I=1,M
          LIST(I)=I
20     CONTINUE

        DO 22 J=1,N
          ROW_ASSIGNED_TO(J)=0
22     CONTINUE

        INCR=STARTINCR
     IF (INCR.GT.EPSILON) INCR=EPSILON
     IF (EPSILON.EQ.1) THRESH=0
```

```
C ***************************************************************
C
C      START FORWARD AUCTION CYCLE
C
C ***************************************************************

     IF (EPSILON.LT.EPS_THRESH)  THEN

17     CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED ROWS

       NONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED ROWS

       DO 103 I=1,NOLIST
        ROW=LIST(I)
        FSTARC=FOUT(ROW)
        LSTARC=FOUT(ROW+1)-1
        FSTCOL=END(FSTARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC

        IF (FSTARC.EQ.LSTARC) THEN
          PCOL(FSTCOL)=PCOL(FSTCOL)+LARGEINCR
         PROW(ROW)=COST(FSTARC)-PCOL(FSTCOL)
          OLDROW=ROW_ASSIGNED_TO(FSTCOL)
          ROW_ASSIGNED_TO(FSTCOL)=ROW
          IF (OLDROW.GT.0) THEN
            NONEWLIST=NONEWLIST+1
            LIST(NONEWLIST)=OLDROW
          END IF
          GO TO 103
        END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS

        IF (((FSTARC+2).LE.LSTARC).AND.(INIT)) THEN
        
         BSTARC=BEST_ARC(ROW)
         SBSTARC=SECD_ARC(ROW)
         BSTCOL=END(BSTARC)
         SNDCOL=END(SECD_ARC(ROW))
        
          MAX1=COST(BSTARC)-PCOL(BSTCOL)
         MAX2=COST(SBSTARC)-PCOL(SNDCOL)
         IF((MAX1.GE.TRDVAL(ROW)).AND.(MAX2.GE.TRDVAL(ROW))) THEN
        
        IF (MAX1.LT.MAX2) THEN
          TMAX=MAX1
          MAX1=MAX2
          MAX2=TMAX
          BSTCOL=SNDCOL
        END IF
        GO TO 70

         END IF
```

```
      END IF
C
      SNDARC=FSTARC+1
      SNDCOL=END(SNDARC)
      MAX1=COST(FSTARC)-PCOL(FSTCOL)
      MAX2=COST(SNDARC)-PCOL(SNDCOL)
      IF (MAX1.GE.MAX2) THEN
         BSTARC=FSTARC
         SBSTARC=SNDARC
      ELSE
        TMAX=MAX1
        MAX1=MAX2
        MAX2=TMAX
        BSTARC=SNDARC
        SBSTARC=FSTARC
      END IF
      IF (SNDARC.LT.LSTARC) THEN
       TRDARC=SNDARC+1
       MAX3=ISMALL
       DO 43 CURARC=TRDARC,LSTARC
          CURCOL=END(CURARC)
          TMAX=COST(CURARC)-PCOL(CURCOL)
          IF (TMAX.GT.MAX2) THEN
C           IF (TMAX.GT.MAX1) THEN
C             SBSTARC=BSTARC
C             BSTARC=CURARC
C             MAX3=MAX2
C             MAX2=MAX1
C             MAX1=TMAX
C           ELSE
C             SBSTARC=CURARC
C             MAX3=MAX2
C             MAX2=TMAX
C           END IF
C           ELSE
C             IF (MAX3.LT.TMAX) MAX3=TMAX
            END IF
43         CONTINUE
           END IF

         BEST_ARC(ROW)=BSTARC
         BSTCOL=END(BSTARC)
          SECD_ARC(ROW)=SBSTARC
        TRDVAL(ROW)=MAX3

70      CONTINUE
C

C  ROW BIDS FOR BSTCOL INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTCOL, WHILE ANY ROW ASSIGNED TO BSTCOL BECOMES UNASSIGNED

      PCOL(BSTCOL)=PCOL(BSTCOL)+MAX1-MAX2+INCR
      PROW(ROW)=MAX2-INCR
      OLDROW=ROW_ASSIGNED_TO(BSTCOL)
      ROW_ASSIGNED_TO(BSTCOL)=ROW
      IF (OLDROW.GT.0) THEN
        NONEWLIST=NONEWLIST+1
        LIST(NONEWLIST)=OLDROW
      END IF
```

```
103    CONTINUE

C     ************* END OF A FORWARD AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS

X       AVERAGE=(CYCLES*AVERAGE+NOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF A SWITCH TO MODIFIED REVERSE AUCTION
C  SHOULD BE MADE (IF NONEWLIST EQUALS ZERO).
C  ALSO INCREASE THE MINIMAL BIDDING INCREMENT UP TO A MAXIMUM
C  VALUE OF EPSILON (THIS IS THE ADAPTIVE FEATURE)

 INCR=INCR*INCRFACTOR
 IF (INCR.GT.EPSILON) INCR=EPSILON
 IF (NONEWLIST.GT.THRESH) THEN
     NOLIST=NONEWLIST
     INIT=.TRUE.
     GO TO 17
 END IF
 
      ELSE

C ****************************************************************
C
C      START FORWARD AUCTION CYCLE
C
C ****************************************************************

15      CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED ROWS

       NONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED ROWS

       DO 100 I=1,NOLIST
  ROW=LIST(I)
  FSTARC=FOUT(ROW)
  LSTARC=FOUT(ROW+1)-1
  FSTCOL=END(FSTARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC

  IF (FSTARC.EQ.LSTARC) THEN
          PCOL(FSTCOL)=PCOL(FSTCOL)+LARGEINCR
     PROW(ROW)=COST(FSTARC)-PCOL(FSTCOL)
           OLDROW=ROW_ASSIGNED_TO(FSTCOL)
           ROW_ASSIGNED_TO(FSTCOL)=ROW
           IF (OLDROW.GT.0) THEN
             NONEWLIST=NONEWLIST+1
             LIST(NONEWLIST)=OLDROW
           END IF
           GO TO 100
         END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS
```

```
      SNDARC=FSTARC+1
      SNDCOL=END(SNDARC)
      MAX1=COST(FSTARC)-PCOL(FSTCOL)
      MAX2=COST(SNDARC)-PCOL(SNDCOL)
      IF (MAX1.GE.MAX2) THEN
        BSTCOL=FSTCOL
      ELSE
        TMAX=MAX1
        MAX1=MAX2
        MAX2=TMAX
        BSTCOL=SNDCOL
      END IF
      IF (SNDARC.LT.LSTARC) THEN
       TRDARC=SNDARC+1
       DO 40 CURARC=TRDARC,LSTARC
          CURCOL=END(CURARC)
          TMAX=COST(CURARC)-PCOL(CURCOL)
          IF (TMAX.GT.MAX2) THEN
            IF (TMAX.GT.MAX1) THEN
              MAX2=MAX1
              MAX1=TMAX
              BSTCOL=CURCOL
            ELSE
              MAX2=TMAX
            END IF
          END IF
40      CONTINUE
        END IF


C  ROW BIDS FOR BSTCOL INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTCOL, WHILE ANY ROW ASSIGNED TO BSTCOL BECOMES UNASSIGNED

      PCOL(BSTCOL)=PCOL(BSTCOL)+MAX1-MAX2+INCR
      PROW(ROW)=MAX2-INCR
      OLDROW=ROW_ASSIGNED_TO(BSTCOL)
      ROW_ASSIGNED_TO(BSTCOL)=ROW
      IF (OLDROW.GT.0) THEN
        NONEWLIST=NONEWLIST+1
        LIST(NONEWLIST)=OLDROW
      END IF

100   CONTINUE

C     ************* END OF A FORWARD AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS

X       AVERAGE=(CYCLES*AVERAGE+NOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF A SWITCH TO MODIFIED REVERSE AUCTION
C  SHOULD BE MADE (IF NONEWLIST IS NO MORE THAN THE THRESHOLD).
C  ALSO INCREASE THE MINIMAL BIDDING INCREMENT UP TO A MAXIMUM
C  VALUE OF EPSILON (THIS IS THE ADAPTIVE FEATURE)

      INCR=INCR*INCRFACTOR
      IF (INCR.GT.EPSILON) INCR=EPSILON
      IF (NONEWLIST.GT.THRESH) THEN
        NOLIST=NONEWLIST
```

```
     GO TO 15
  END IF
 
     END IF

C ***************************************************************
C
C           START OF MODIFIED REVERSE AUCTION
C
C ***************************************************************

      IF (EPSILON.GT.1) GOTO 400
      IF (N.EQ.M) GOTO 400

      INCR=EPSILON


C  COMPUTE LAMBDA WHICH IS THE MINIMUM COLUMN PRICE OVER ALL
C  ASSIGNED COLUMNS, AND COMPILE THE LIST OF UNASSIGNED COLUMNS

      LAMBDA=ILARGE
      RNOLIST=0
      DO 310 J=1,N
       ROW=ROW_ASSIGNED_TO(J)
  IF (ROW.GT.0) THEN
    COL_ASSIGNED_TO(ROW)=J
    IF (LAMBDA.GT.PCOL(J)) LAMBDA=PCOL(J)
  ELSE
    RNOLIST=RNOLIST+1
    REV_LIST(RNOLIST)=J
  END IF
310    CONTINUE


      IF (RNOLIST.NE.N-M) THEN
        PRINT*,'NUMBER OF UNASSIGNED ROWS IS WRONG'
   PAUSE
   STOP
      END IF


C  START OF A NEW MODIFIED REVERSE AUCTION CYCLE


315      CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED COLUMNS

       RNONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED COLUMNS

       DO 340 J=1,RNOLIST
   COLUMN=REV_LIST(J)
   IF (PCOL(COLUMN).LE.LAMBDA) GO TO 340
   CURARC=FIN(COLUMN)
   NXTARC=NXTIN(CURARC)
   CURROW=START(CURARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC
```

```
      IF (NXTARC.EQ.0) THEN
        MAX1=COST(CURARC)-PROW(CURROW)
            IF (LAMBDA.GE.MAX1-INCR) THEN
          PCOL(COLUMN)=LAMBDA
          GO TO 340
        END IF
        PCOL(COLUMN)=LAMBDA
            PROW(CURROW)=COST(CURARC)-LAMBDA
            OLDCOL=COL_ASSIGNED_TO(CURROW)
            COL_ASSIGNED_TO(CURROW)=COLUMN
        IF (PCOL(OLDCOL).GT.LAMBDA) THEN
          RNONEWLIST=RNONEWLIST+1
          REV_LIST(RNONEWLIST)=OLDCOL
        END IF
            GO TO 340
           END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS

      NXTROW=START(NXTARC)
      MAX1=COST(CURARC)-PROW(CURROW)
      MAX2=COST(NXTARC)-PROW(NXTROW)
      IF (MAX1.GE.MAX2) THEN
        BSTROW=CURROW
      ELSE
        TMAX=MAX1
        MAX1=MAX2
        MAX2=TMAX
        BSTROW=NXTROW
      END IF

      CURARC=NXTIN(NXTARC)

330   IF (CURARC.GT.0) THEN
        CURROW=START(CURARC)
        TMAX=COST(CURARC)-PROW(CURROW)
        IF (TMAX.GT.MAX2) THEN
          IF (TMAX.GT.MAX1) THEN
            MAX2=MAX1
            MAX1=TMAX
            BSTROW=CURROW
          ELSE
            MAX2=TMAX
          END IF
        END IF
        CURARC=NXTIN(CURARC)
        GO TO 330
      END IF


C  COLUMN BIDS FOR BSTROW INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTROW, WHILE ANY COLUMN ASSIGNED TO BSTROW BECOMES UNASSIGNED


      IF (LAMBDA.GE.MAX1-INCR) THEN
        PCOL(COLUMN)=LAMBDA
        GO TO 340
      END IF
      DELTA=MAX1-MAX2+INCR
```

```
      IF (DELTA.GT.MAX1-LAMBDA) DELTA=MAX1-LAMBDA
      PROW(BSTROW)=PROW(BSTROW)+DELTA
      PCOL(COLUMN)=MAX1-DELTA
      OLDCOL=COL_ASSIGNED_TO(BSTROW)
      COL_ASSIGNED_TO(BSTROW)=COLUMN
      IF (PCOL(OLDCOL).GT.LAMBDA) THEN
        RNONEWLIST=RNONEWLIST+1
        REV_LIST(RNONEWLIST)=OLDCOL
      END IF

340   CONTINUE

C     ************* END OF A MODIFIED REVERSE AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS

X       AVERAGE=(CYCLES*AVERAGE+RNOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF THERE ARE STILL 'MANY' UNASSIGNED ELIGIBLE COLUMNS, THAT IS,
C  IF THE NUMBER OF ELIGIBLE UNASSIGNED COLUMNS IS GREATER THAN
C  THE PARAMETER THRESH. IF NOT, REPLACE CURRENT LIST WITH THE NEW LIST,
C  AND GO FOR ANOTHER CYCLE. OTHERWISE, IF EPSILON > 1, REDUCE EPSILON,
C  RESET THE ASSIGNMENT TO EMPTY AND RESTART AUCTION;
C  IF EPSILON = 1 TERMINATE.

      IF (RNONEWLIST.GT.THRESH) THEN
        RNOLIST=RNONEWLIST
        GO TO 315
      END IF

C ***************************************************************
C
C               END OF SUBPROBLEM (SCALING PHASE)
C
C ***************************************************************

400   CONTINUE

C ****** IF EPSILON IS 1 TERMINATE ******

      IF (EPSILON.EQ.1) THEN
        RETURN
      ELSE

C   ELSE REDUCE EPSILON AND UPDATE PARAMETERS FOR THE NEXT SCALING PHASE

        NUMPHASES=NUMPHASES+1
        EPSILON=INT(EPSILON/FACTOR)
        IF (EPSILON.GT.INCR) EPSILON=INT(EPSILON/FACTOR)
        IF ((EPSILON.LT.1).OR.(EPSILON.LT.ENDEPS)) EPSILON=1
        IF (STARTINCR.LT.EPSILON) STARTINCR=FACTOR*STARTINCR
        THRESH=INT(THRESH/FACTOR)

X       PRINT*,'*** END OF A SCALING PHASE; NEW EPSILON=',EPSILON

        GO TO 12
```

```
      END IF

      END


      SUBROUTINE SETRAN(ISEED)
      IMPLICIT REAL*8 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C PORTABLE CONGRUENTIAL (UNIFORM) RANDOM NUMBER GENERATOR:
C     NEXT_VALUE = [(7**5) * PREVIOUS_VALUE] MODULO[(2**31)-1]
C
C THIS GENERATOR CONSISTS OF TWO ROUTINES:
C   (1) SETRAN - INITIALIZES CONSTANTS AND SEED
C   (2) RRAN   - GENERATES A REAL RANDOM NUMBER
C
C THE GENERATOR REQUIRES A MACHINE WITH AT LEAST 32 BITS OF PRECISION.
C THE SEED (ISEED) MUST BE IN THE RANGE (1,(2**31)-1).
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IF(ISEED.LT.1) STOP 77
      MULT=16807
      MODUL=2147483647
      I15=2**15
      I16=2**16
      JRAN=ISEED
      RETURN
      END
C
      REAL FUNCTION RAN()
      IMPLICIT REAL*4 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C RAN GENERATES A REAL RANDOM NUMBER BETWEEN 0 AND 1
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IXHI=JRAN/I16
      IXLO=JRAN-IXHI*I16
      IXALO=IXLO*MULT
      LEFTLO=IXALO/I16
      IXAHI=IXHI*MULT
      IFULHI=IXAHI+LEFTLO
      IRTLO=IXALO-LEFTLO*I16
      IOVER=IFULHI/I15
      IRTHI=IFULHI-IOVER*I15
      JRAN=((IRTLO-MODUL)+IRTHI*I16)+IOVER
      IF(JRAN.LT.0) JRAN=JRAN+MODUL
      RAN = FLOAT(JRAN)/FLOAT(MODUL)
      RETURN
      END
```

\subsect{AUCTION-FR}This code implements the combined forward/reverse auction algorithm with $\e$-scaling for the symmetric assignment problem; cf.\ Section 4.2 of the author's network optimization book.
For good performance, it is frequently not essential to use $\e$-scaling in this code. It is therefore recommended that the code be tried without $\e$-scaling by setting the starting $\e$ to 1.

```
C ****************************************************************
C
C  SAMPLE CALLING PROGRAM FOR COMBINED FORWARD/REVERSE
C                AUCTION ALGORITHM
C
C   THIS DRIVER CREATES A SYMMETRIC ASSIGNMENT PROBLEM
C   WITH EQUAL NUMBER OF ROWS AND COLUMNS,
C   AND CALLS THE AUCTION_FR SUBROUTINE TO FIND AN
C   ASSIGNMENT OF MAXIMAL VALUE.
C
C ****************************************************************


 PARAMETER(MAXNODES=10000, MAXARCS=100000)
 
      IMPLICIT NONE
 INTEGER N,NA,A,IA,ILARGE,BEGEPS,ENDEPS,CYCLES
 INTEGER NUMPHASES,STARTINCR
 INTEGER I,J,ARC,NOASS,ICOST,ABSCOST,CURARC
 INTEGER CURCOL,FSTARC,LSTARC,MINCOST,MAXCOST,MCOST
      INTEGER COL_ASSIGNED_TO(MAXNODES),ROW_ASSIGNED_TO(MAXNODES)
      INTEGER FOUT(MAXNODES),FIN(MAXNODES),NXTIN(MAXARCS)
 INTEGER COST(MAXARCS),START(MAXARCS),END(MAXARCS)
 INTEGER PCOL(MAXNODES),PROW(MAXNODES)
 INTEGER PRDARC(MAXNODES)
      REAL*8 FACTOR,TT1,TT2,TCOST,AVERAGE
 COMMON/ARRAYC/COST/ARRAYS/START/ARRAYE/END
      COMMON/ARRAYFO/FOUT/ARRAYFI/FIN/ARRAYNI/NXTIN
      COMMON/ARRAYRA/ROW_ASSIGNED_TO/ARRAYCA/COL_ASSIGNED_TO
      COMMON/ARRAYPC/PCOL/ARRAYPR/PROW
      COMMON/BK1/N,A,ILARGE/BK2/CYCLES,AVERAGE,NUMPHASES

C ****************************************************************
C
C    PROBLEM GENERATION CODE STARTS HERE
C    THE USER MAY REPLACE THIS CODE WITH A CODE THAT READS
C    HIS/HER PROBLEM FROM A FILE
C
C ****************************************************************

C    THIS CODE INCLUDES A UNIFORM RANDOM NUMBER GENERATOR
C    WHICH RETURNS A VALUE IN (0,1)

C    INITIALIZE RANDOM GENERATOR

    INTEGER MULT,MODUL,I15,I16,JRAN,ISEED
    REAL RAN
    COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN

    ISEED=13502460
    CALL SETRAN(ISEED)

    PRINT*,'GENERATING A SYMMETRIC ASSIGNMENT PROBLEM'
    PRINT*,'****************************************'

C **** READ THE NUMBER OF ROWS N & THE NUMBER OF ARCS A ****

    PRINT*,'ENTER THE NUMBER OF ROWS (AND COLUMNS)'
    READ*,N
5    PRINT*,'ENTER THE NUMBER OF ARCS PER ROW (>1)'
```

```
      READ*,NA
      IF (NA.LT.2) GOTO 5
      PRINT*,'ENTER THE MINIMUM AND THE MAXIMUM COST'
      READ*,MINCOST,MAXCOST

C     THE NUMBER OF ARCS IS N*NA

      A=N*NA

C     THE ARCS INCIDENT TO ROW I ARE FOUT(I) TO FOUT(I+1)-1
C     ALSO, FOR FEASIBILITY EACH ROW IS DIRECTLY CONNECTED
C     WITH THE CORRESPONDING COLUMN

      DO 20 I=1,N
        FOUT(I)=1+(I-1)*NA
20    CONTINUE
      FOUT(N+1)=A+1

      DO 22 I=1,N
        DO 23 ARC=FOUT(I),FOUT(I+1)-1
          START(ARC)=I
23      CONTINUE
22    CONTINUE

C     GENERATE THE END(ARC) AND COST(ARC) WHICH ARE THE COLUMN
C     AND THE COST COEFFICIENT ASSOCIATED WITH ARC

      DO 25 ARC=1,A
        END(ARC)=1+RAN()*N
        IF ((END(ARC).GT.N).OR.(END(ARC).LT.1)) THEN
          PRINT*,'ERROR IN PROBLEM GENERATION'
          PAUSE
          STOP
        END IF
        COST(ARC)=MINCOST+RAN()*(MAXCOST-MINCOST)
25    CONTINUE

C   MODIFY THE END OF THE LAST ARC OUT OF EACH ROW FOR FEASIBILITY
C   AND SET ITS COST TO -MAXCOST

      DO 30 I=1,N
        END(FOUT(I+1)-1)=I
        COST(FOUT(I+1)-1)=-MAXCOST
30    CONTINUE

C     CONSTRUCT THE FIN() AND NXTIN() ARRAYS

      DO 32 J=1,N
        FIN(J)=0
        PRDARC(J)=0
32    CONTINUE

      DO 33 ARC=1,A
        NXTIN(ARC)=0
        J=END(ARC)
        IF (FIN(J).NE.0) THEN
          NXTIN(PRDARC(J))=ARC
        ELSE
          FIN(J)=ARC
        END IF
```

```
      PRDARC(J)=ARC
33    CONTINUE


C ***************************************************************
C
C         PROBLEM GENERATION CODE ENDS HERE
C
C ***************************************************************


C      SCALE THE COST TO WORK WITH INTEGER EPSILON

       MAXCOST=0
       DO 35 IA=1,A
         ABSCOST=IABS(COST(IA))
         IF (ABSCOST.GT.MAXCOST) MAXCOST=ABSCOST
         COST(IA)=COST(IA)*(N+1)
35□CONTINUE

C *** ILARGE IS A VERY LARGE INTEGER FOR YOUR MACHINE ***
C     THE SCALED COSTS SHOULD BE SIGNIFICANTLY SMALLER THAN
C     ILARGE AND SIGNIFICANTLY LARGER THAN -ILARGE


       ILARGE=2000000000

□
□IF (MAXCOST.GT.INT(ILARGE/(N+1))) THEN
□  PRINT*,'THE COST RANGE IS TOO LARGE FOR INTEGER ARITHMETIC'
□  PAUSE
□  STOP
□END IF
□
□MAXCOST=MAXCOST*(N+1)


C  □THE FOLLOWING PARAMETERS BEGEPS, FACTOR, ENDEPS, AND STARTINCR
C       ARE PASSED TO THE AUCTION ALGORITHM. VALUES BETWEEN
C        (A) MAXCOST/2 AND 1 FOR BEGEPS
C        (B) 4 AND 10 FOR FACTOR
C□ (C) N AND 1 FOR ENDEPS
C        (D) 1 AND BEGEPS FOR STARTINCR
C       HAVE WORKED WELL FOR LARGE SPARSE PROBLEMS.
C       FOR DENSE PROBLEMS IT IS RECOMMENDED THAT
C□BEGEPS BE SET TO A SMALLER VALUE,
C□ENDEPS BE SET TO 1,
C       STARTINCR BE SET TO 1.

C       GENERALLY, THE COMBINED FORWARD/REVERSE ALGORITHM
C       WORKS BEST WITH A SMALLER VALUE OF BEGEPS THAN THE
C       FORWARD ALGORITHM.
C       IT IS WORTH TRYING BEGEPS=1, IN WHICH CASE THERE IS
C       ONLY ONE SCALING PHASE (THAT IS, NO E-SCALING IS USED).

       PRINT*,'*****************************************'
       PRINT*,'MAXIMUM COST IS ',MAXCOST
       PRINT*,'ENTER THE STARTING EPSILON'
       READ*,BEGEPS
       IF (BEGEPS.LT.1) BEGEPS=1
```

```
      PRINT*,'ENTER THE EPSILON REDUCTION FACTOR'
      READ*,FACTOR
ENDEPS=N/10
IF (ENDEPS.LT.1) ENDEPS=1
IF (ENDEPS.GT.BEGEPS) ENDEPS=BEGEPS
STARTINCR=1
IF (STARTINCR.LT.1) STARTINCR=1

      PRINT*,'*******************************************'
PRINT*,'STARTING EPSILON = ',BEGEPS
PRINT*,'EPSILON REDUCTION FACTOR = ',FACTOR
PRINT*,'THRESHOLD EPSILON BEFORE IT IS SET TO 1 = ',ENDEPS
PRINT*,'STARTING MIN BIDDING INCREMENT = ',STARTINCR
      PRINT*,'CALLING FORWARD/REVERSE AUCTION'
      PRINT*,'*******************************************'

C     GET STARTING TIME FOR THE MAC II

     TT1 = LONG(362)/60.0

     CALL AUCTION_FR(BEGEPS,FACTOR,ENDEPS,STARTINCR)

C     GET ENDING TIME FOR THE MAC II

     TT2 = LONG(362)/60.0 - TT1
     PRINT *,'FINISHED --- TOTAL CPU TIME', TT2,' SECS'
     PRINT*,'*******************************************'

C     *** DISPLAY RESULTS ***

X       WRITE(9,2010) CYCLES
X2010    FORMAT(' NO OF AUCTION CYCLES',I7)
X       WRITE(9,2020) AVERAGE
X2020    FORMAT(' AVERAGE NUMBER OF BIDS PER CYCLE',F9.3)
      WRITE(9,2030) NUMPHASES
2030    FORMAT('NO OF EPSILON SUBPROBLEMS SOLVED =',I7)

C      CHECK OPTIMALITY & CALCULATE COST

      TCOST=0
      DO 40 I=1,N
  J=COL_ASSIGNED_TO(I)
        IF (J.EQ.0) THEN
          PRINT*,'ROW ',I,' IS UNASSIGNED'
        END IF
  IF (ROW_ASSIGNED_TO(J).NE.I) THEN
     PRINT*,'ASSIGNMENT MIXUP: ROW ',I,' COLUMN ',J
  END IF
        FSTARC=FOUT(I)
        LSTARC=FOUT(I+1)-1
  MCOST=-ILARGE
        DO 45 ARC=FSTARC,LSTARC
         CURCOL=END(ARC)
   IF (PROW(I)+PCOL(CURCOL).LT.COST(ARC)-1) THEN
     PRINT*,'1-CS VIOLATED AT ARC ',ARC
   END IF
          IF (CURCOL.EQ.J) THEN
           IF (MCOST.LT.COST(ARC)) THEN
       MCOST=COST(ARC)
     END IF
```

```
      END IF
45    CONTINUE
      TCOST=TCOST+MCOST/(N+1)
      IF (PROW(I)+PCOL(J).NE.MCOST) THEN
        PRINT*,'1-CS VIOLATED AT ROW ',I
      END IF
40    CONTINUE

      DO 50 J=1,N
        IF (ROW_ASSIGNED_TO(J).EQ.0) THEN
          PRINT*,'COLUMN ',J,' IS UNASSIGNED'
        END IF
50    CONTINUE

      WRITE(9,2100) TCOST
2100  FORMAT(' ASSIGNMENT COST=',F18.2)
      PRINT *, ' PROGRAM ENDED; <CR> TO EXIT '
      PAUSE
      END




C ***************************************************************
C
C     FORWARD/REVERSE AUCTION CODE FOR N BY N ASSIGNMENT PROBLEMS
C
C       WRITTEN BY DIMITRI P. BERTSEKAS
C
C              DEC. 1990
C
C THIS CODE IMPLEMENTS THE FORWARD/REVERSE AUCTION ALGORITHM
C WITH E-SCALING FOR SYMMETRIC N BY N ASSIGNMENT PROBLEMS.
C IT SOLVES A SEQUENCE OF SUBPROBLEMS AND DECREASES
C EPSILON BY A CONSTANT FACTOR BETWEEN SUBPROBLEMS.
C THIS VERSION CORRESPONDS TO A GAUSS-SEIDEL MODE
C AND SOLVES EPSILON SUBPROBLEMS INEXACTLY.
C
C THE CODE IS AN IMPROVED VERSION OF AN EARLIER (SEPT. 1985)
C AUCTION CODE WITH E-SCALING WRITTEN BY DIMITRI P. BERTSEKAS
C
C THE CODE TREATS THE PROBLEM AS A MAXIMIZATION PROBLEM.
C TO SOLVE A MINIMIZATION PROBLEM, REVERSE THE SIGN OF THE
C ARC COSTS PRIOR TO CALLING AUCTION, AND REVERSE AGAIN
C THE SIGN OF THE OPTIMAL COST UPON RETURN FROM AUCTION.
C THIS CODE ALLOWS MULTIPLE ARCS BETWEEN A ROW AND A COLUMN.
C
C THIS VERSION OF THE AUCTION ALGORITHM IS ADAPTIVE. HERE THE MINIMAL
C BIDDING INCREMENT (STORED IN THE VARIABLE INCR) MAY BE SMALLER THAN
C EPSILON. FOR EVERY SUBPROBLEM, INCR STARTS AT THE PARAMETER VALUE
C STARTINCR (WHICH IS PASSED TO THE AUCTION ROUTINE) AND IS INCREASED
C BY A FACTOR OF 2 AT THE END OF EACH CYCLE, UP TO A MAXIMUM VALUE OF
C EPSILON. THIS ADAPTIVE FEATURE IS PARTICULARLY EFFECTIVE FOR
C DENSE PROBLEMS. IT CAN BE DEFEATED BY SELECTING STARTINCR=BEGEPS
C
C THIS CODE MAY FAIL DUE TO INTEGER OVERFLOW IF THE NUMBER OF NODES
C OR THE COST RANGE (OR BOTH) ARE LARGE. TO CORRECT THIS SITUATION,
C THE PRICES AND OTHER RELATED VARIABLES (MAX1,MAX2,TMAX ETC) SHOULD
C BE DECLARED AS DOUBLE PRECISION REALS.
C ***************************************************************
```

```
C
C THE USER MUST SUPPLY THE FOLLOWING PROBLEM DATA IN FORWARD STAR FORMAT
C (THAT IS, ALL ARCS OF THE SAME ROW ARE NUMBERED CONSECUTIVELY):
C  N=NUMBER OF ROWS (EQUALS NUMBER OF COLUMNS)
C  A=NUMBER OF ARCS
C  FOUT(ROW)=FIRST ARC COMING OUT OF ROW
C  FIN(COL)=FIRST ARC COMING INTO COL
C       NXTIN(ARC)=NEXT ARC INCIDENT TO THE SAME COLUMN AS ARC
C  COST(ARC)=COST OF ARC
C       START(ARC)=ROW CORRESPONDING TO ARC
C  END(ARC)=COLUMN CORRESPONDING TO ARC
C
C AND THE FOLLOWING PARAMETERS FOR THE AUCTION ALGORITHM:
C  BEGEPS=STARTING VALUE OF EPSILON (MUST BE NO LESS THAN 1)
C  ENDEPS=FINAL VALUE OF EPSILON BEFORE IT IS SET TO 1
C  FACTOR=FACTOR BY WHICH EPSILON IS DECREASED BETWEEN SUBPROBLEMS
C       STARTINCR=THE STARTING VALUE OF THE BIDDING INCREMENT
C       ENDEPS SHOULD NOT EXCEED BEGEPS.
C       FACTOR MUST BE GREATER THAN 1 (UNLESS BEGEPS=1).
C
C FOUT(.) IS AN ARRAY OF LENGTH N.
C COST(.),END(.) ARE ARRAYS OF LENGTH A.
C
C THE SOLUTION IS CONTAINED IN THE ARRAY ROW_ASSIGNED_TO(.) WHERE
C    ROW_ASSIGNED_TO(COL) GIVES THE ROW ASSIGNED TO COL.
C ALSO COL_ASSIGNED_TO(ROW) GIVES THE COLUMN ASSIGNED TO ROW.
C
C THIS ALGORITHM DOES NOT CHECK FOR INFEASIBILITY OF THE PROBLEM.
C TO MAKE SURE THE PROBLEM IS FEASIBLE THE USER MAY ADD
C    ADDITIONAL VERY SMALL COST ARCS.
C
C ********************************************************************
C ALL PROBLEM DATA ARE INTEGER
C ********************************************************************


      SUBROUTINE AUCTION_FR(BEGEPS,FACTOR,ENDEPS,STARTINCR)

 PARAMETER(MAXNODES=10000, MAXARCS=100000)
 
      IMPLICIT NONE
      INTEGER A,K,N,I,J,M,CURARC,CURCOL,CURROW
      INTEGER THRESH,INCR,STARTINCR,INCRFACTOR
      INTEGER NOLIST,RNOLIST,NONEWLIST,RNONEWLIST
      INTEGER ROW,COLUMN, BSTROW,BSTCOL
      INTEGER FSTARC,FSTCOL,NXTARC,NXTROW,LSTARC,SNDARC,SNDCOL
      INTEGER MAX1,MAX2,TMAX,TMIN,TRDARC,EPSILON,BEGEPS,ENDEPS
      INTEGER ISMALL,ILARGE,LARGEINCR,CYCLES
      INTEGER NUMPHASES,OLDROW,OLDCOL
      INTEGER COL_ASSIGNED_TO(MAXNODES),ROW_ASSIGNED_TO(MAXNODES)
      INTEGER FOUT(MAXNODES),FIN(MAXNODES),NXTIN(MAXARCS)
 INTEGER COST(MAXARCS),START(MAXARCS),END(MAXARCS)
 INTEGER LIST(MAXNODES),REV_LIST(MAXNODES)
 INTEGER PCOL(MAXNODES),PROW(MAXNODES)
      REAL*8 AVERAGE,FACTOR
 LOGICAL READY_SWITCH,SWITCH
 COMMON/ARRAYC/COST/ARRAYS/START/ARRAYE/END
      COMMON/ARRAYFO/FOUT/ARRAYFI/FIN/ARRAYNI/NXTIN
      COMMON/ARRAYRA/ROW_ASSIGNED_TO/ARRAYCA/COL_ASSIGNED_TO
      COMMON/ARRAYPC/PCOL/ARRAYPR/PROW
```

```
      COMMON/BK1/N,A,ILARGE/BK2/CYCLES,AVERAGE,NUMPHASES

C ************************************************************

C   ******* CHECK VALIDITY OF PARAMETERS PASSED *******

      IF (BEGEPS.LT.1) THEN
   PRINT*,'STARTING VALUE OF EPSILON IS LESS THAN 1'
   PRINT*,'EXECUTION ABORTED'
   STOP
 END IF
 IF (ENDEPS.GT.BEGEPS) THEN
   PRINT*,'PARAMETER ENDEPS IS GREATER THAN PARAMETER BEGEPS'
   PRINT*,'ENDEPS IS SET AT THE DEFAULT VALUE OF 1'
   ENDEPS=1
 END IF
      IF ((FACTOR.LE.1).AND.(BEGEPS.GT.1)) THEN
   PRINT*,'EPSILON REDUCTION FACTOR IS NOT GREATER THAN 1'
   PRINT*,'EXECUTION ABORTED'
   STOP
 END IF
      IF (STARTINCR.LT.1) THEN
   PRINT*,'MIN BIDDING INCREMENT IS LESS THAN 1'
   PRINT*,'STARTINCR IS SET AT THE DEFAULT VALUE OF 1'
   STARTINCR=1
 END IF

C   ******* INITIALIZATION *******
 
 EPSILON=BEGEPS
      ISMALL=-ILARGE
      LARGEINCR=INT(ILARGE/10)
      THRESH=INT(0.2*N)
 INCRFACTOR=2
      IF (THRESH.GT.100) THRESH=100
X       CYCLES=1
X       AVERAGE=N
       NUMPHASES=1

C    INITIALIZE FORWARD/REVERSE SWITCH PARAMETERS

       READY_SWITCH=.FALSE.
  SWITCH=.TRUE.


  DO 10 J=1,N
        PCOL(J)=0
10  CONTINUE

       FOUT(N+1)=A+1

C ************************************************************
C
C  THIS IMPLEMENTATION OF THE AUCTION ALGORITHM OPERATES IN CYCLES.
C  EACH FORWARD AUCTION CYCLE CONSISTS OF ONE BID BY EACH OF THE ROWS
C  THAT ARE UNASSIGNED AT THE START OF THE CYCLE (THESE ROWS ARE
C  STORED IN THE ARRAY LIST(.)). AS THE CYCLE PROGRESSES NEW
C  ROWS BECOME UNASSIGNED; THESE ARE STORED IN LIST(.)
C  AND WILL SUBMIT A BID AT THE NEXT CYCLE.
C  REVERSE AUCTION CYCLES ARE STRUCTURED SIMILARLY.
```

```
C
C ***************************************************************
C
C      START SUBPROBLEM (SCALING PHASE) W/ NEW EPSILON
C
C ***************************************************************
C
12     CONTINUE

C    INITIALIZE ROW AND COLUMN ASSIGNMENT LISTS

       NOLIST=N
       DO 20 I=1,N
         COL_ASSIGNED_TO(I)=0
         LIST(I)=I
20     CONTINUE

       RNOLIST=N
       DO 22 J=1,N
         ROW_ASSIGNED_TO(J)=0
         REV_LIST(J)=J
22     CONTINUE

       INCR=STARTINCR
       IF (INCR.GT.EPSILON) INCR=EPSILON
       IF (EPSILON.EQ.1) THRESH=0

C ***************************************************************
C
C      START FORWARD AUCTION CYCLE
C
C ***************************************************************

15     CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED ROWS

       NONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED ROWS

       DO 100 I=1,NOLIST
         ROW=LIST(I)
         IF (COL_ASSIGNED_TO(ROW).GT.0) GO TO 100
         FSTARC=FOUT(ROW)
         LSTARC=FOUT(ROW+1)-1
         FSTCOL=END(FSTARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC

         IF (FSTARC.EQ.LSTARC) THEN
           PCOL(FSTCOL)=PCOL(FSTCOL)+LARGEINCR
           PROW(ROW)=COST(FSTARC)-PCOL(FSTCOL)
           OLDROW=ROW_ASSIGNED_TO(FSTCOL)
           ROW_ASSIGNED_TO(FSTCOL)=ROW
           COL_ASSIGNED_TO(ROW)=FSTCOL
           IF (OLDROW.GT.0) THEN
             COL_ASSIGNED_TO(OLDROW)=0
             NONEWLIST=NONEWLIST+1
             LIST(NONEWLIST)=OLDROW
```

```
          END IF
          GO TO 100
        END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS

      SNDARC=FSTARC+1
      SNDCOL=END(SNDARC)
      MAX1=COST(FSTARC)-PCOL(FSTCOL)
      MAX2=COST(SNDARC)-PCOL(SNDCOL)
      IF (MAX1.GE.MAX2) THEN
        BSTCOL=FSTCOL
      ELSE
        TMAX=MAX1
        MAX1=MAX2
        MAX2=TMAX
        BSTCOL=SNDCOL
      END IF
      IF (SNDARC.LT.LSTARC) THEN
       TRDARC=SNDARC+1
       DO 40 CURARC=TRDARC,LSTARC
         CURCOL=END(CURARC)
         TMAX=COST(CURARC)-PCOL(CURCOL)
         IF (TMAX.GT.MAX2) THEN
           IF (TMAX.GT.MAX1) THEN
             MAX2=MAX1
             MAX1=TMAX
             BSTCOL=CURCOL
           ELSE
             MAX2=TMAX
           END IF
         END IF
40       CONTINUE
        END IF


C  ROW BIDS FOR BSTCOL INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTCOL, WHILE ANY ROW ASSIGNED TO BSTCOL BECOMES UNASSIGNED

      PCOL(BSTCOL)=PCOL(BSTCOL)+MAX1-MAX2+INCR
      PROW(ROW)=MAX2-INCR
      COL_ASSIGNED_TO(ROW)=BSTCOL
      OLDROW=ROW_ASSIGNED_TO(BSTCOL)
      ROW_ASSIGNED_TO(BSTCOL)=ROW
      IF (OLDROW.GT.0) THEN
        COL_ASSIGNED_TO(OLDROW)=0
        NONEWLIST=NONEWLIST+1
        LIST(NONEWLIST)=OLDROW
      END IF

100    CONTINUE

C     ************* END OF A FORWARD AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS

X       AVERAGE=(CYCLES*AVERAGE+NOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF THERE ARE STILL 'MANY' UNASSIGNED ROWS, THAT IS, IF THE
```

```
C  NUMBER OF UNASSIGNED ROWS IS GREATER THAN
C  THE PARAMETER THRESH. IF NOT, REPLACE CURRENT LIST WITH THE NEW LIST,
C  AND GO FOR ANOTHER CYCLE. OTHERWISE, IF EPSILON > 1, REDUCE EPSILON,
C  RESET THE ASSIGNMENT TO EMPTY AND RESTART AUCTION;
C  IF EPSILON = 1 TERMINATE.
C  ALSO INCREASE THE MINIMAL BIDDING INCREMENT UP TO A MAXIMUM
C  VALUE OF EPSILON (THIS IS THE ADAPTIVE FEATURE)


       INCR=INCR*INCRFACTOR
  IF (INCR.GT.EPSILON) INCR=EPSILON
  IF (NONEWLIST.GT.THRESH) THEN
     IF (SWITCH) THEN
       NOLIST=NONEWLIST
       GO TO 115
     END IF
     IF (NONEWLIST.LT.RNOLIST) READY_SWITCH=.TRUE.
     IF ((NONEWLIST.EQ.NOLIST).AND.(READY_SWITCH)) THEN
       READY_SWITCH=.FALSE.
       GO TO 115
     ELSE
       NOLIST=NONEWLIST
       GO TO 15
     END IF
  ELSE
     GO TO 300
  END IF


C ****************************************************************
C
C       START REVERSE AUCTION CYCLE
C
C ****************************************************************

115      CONTINUE

C  INITIALIZE COUNT OF NEXT LIST OF UNASSIGNED COLUMNS

       RNONEWLIST=0

C  CYCLE THROUGH THE CURRENT LIST OF UNASSIGNED COLUMNS

       DO 200 J=1,RNOLIST
   COLUMN=REV_LIST(J)
   IF (ROW_ASSIGNED_TO(COLUMN).GT.0) GO TO 200
   CURARC=FIN(COLUMN)
   NXTARC=NXTIN(CURARC)
   CURROW=START(CURARC)

C  FIRST TAKE CARE OF THE EXCEPTIONAL CASE WHERE ROW HAS ONLY ONE ARC

   IF (NXTARC.EQ.0) THEN
          PROW(CURROW)=PROW(CURROW)+LARGEINCR
     PCOL(COLUMN)=COST(CURARC)-PROW(CURROW)
          OLDCOL=COL_ASSIGNED_TO(CURROW)
          COL_ASSIGNED_TO(CURROW)=COLUMN
          ROW_ASSIGNED_TO(COLUMN)=CURROW
          IF (OLDCOL.GT.0) THEN
      ROW_ASSIGNED_TO(OLDCOL)=0
```

```
            RNONEWLIST=RNONEWLIST+1
            REV_LIST(RNONEWLIST)=OLDCOL
          END IF
          GO TO 200
        END IF

C  NEXT TAKE CARE OF THE REGULAR CASE WHERE ROW HAS MULTIPLE ARCS

      NXTROW=START(NXTARC)
      MAX1=COST(CURARC)-PROW(CURROW)
      MAX2=COST(NXTARC)-PROW(NXTROW)
      IF (MAX1.GE.MAX2) THEN
        BSTROW=CURROW
      ELSE
        TMAX=MAX1
        MAX1=MAX2
        MAX2=TMAX
        BSTROW=NXTROW
      END IF

      CURARC=NXTIN(NXTARC)

130   IF (CURARC.GT.0) THEN
        CURROW=START(CURARC)
        TMAX=COST(CURARC)-PROW(CURROW)
        IF (TMAX.GT.MAX2) THEN
          IF (TMAX.GT.MAX1) THEN
            MAX2=MAX1
            MAX1=TMAX
            BSTROW=CURROW
          ELSE
            MAX2=TMAX
          END IF
        END IF
        CURARC=NXTIN(CURARC)
        GO TO 130
      END IF


C  COLUMN BIDS FOR BSTROW INCREASING ITS PRICE, AND GETS ASSIGNED
C  TO BSTROW, WHILE ANY COLUMN ASSIGNED TO BSTROW BECOMES UNASSIGNED


      PROW(BSTROW)=PROW(BSTROW)+MAX1-MAX2+INCR
      PCOL(COLUMN)=MAX2-INCR
      ROW_ASSIGNED_TO(COLUMN)=BSTROW
      OLDCOL=COL_ASSIGNED_TO(BSTROW)
      COL_ASSIGNED_TO(BSTROW)=COLUMN
      IF (OLDCOL.GT.0) THEN
        ROW_ASSIGNED_TO(OLDCOL)=0
        RNONEWLIST=RNONEWLIST+1
        REV_LIST(RNONEWLIST)=OLDCOL
      END IF

200   CONTINUE

C     ************* END OF A REVERSE AUCTION CYCLE ***************

C  OPTIONALLY COLLECT STATISTICS
```

```
X       AVERAGE=(CYCLES*AVERAGE+RNOLIST)/(CYCLES+1)
X       CYCLES=CYCLES+1

C  CHECK IF THERE ARE STILL 'MANY' UNASSIGNED COLUMNS, THAT IS, IF THE
C  NUMBER OF UNASSIGNED COLUMNS IS GREATER THAN
C  THE PARAMETER THRESH. IF NOT, REPLACE CURRENT LIST WITH THE NEW LIST,
C  AND GO FOR ANOTHER CYCLE. OTHERWISE, IF EPSILON > 1, REDUCE EPSILON,
C  RESET THE ASSIGNMENT TO EMPTY AND RESTART AUCTION;
C  IF EPSILON = 1 TERMINATE.
C  ALSO INCREASE THE MINIMAL BIDDING INCREMENT UP TO A MAXIMUM
C  VALUE OF EPSILON (THIS IS THE ADAPTIVE FEATURE)


INCR=INCR*INCRFACTOR
IF (INCR.GT.EPSILON) INCR=EPSILON
       IF (RNONEWLIST.GT.THRESH) THEN
     IF (SWITCH) THEN
       SWITCH=.FALSE.
       RNOLIST=RNONEWLIST
       GO TO 15
     END IF
     IF (RNONEWLIST.LT.NOLIST) READY_SWITCH=.TRUE.
     IF ((RNONEWLIST.EQ.RNOLIST).AND.(READY_SWITCH)) THEN
       READY_SWITCH=.FALSE.
       GO TO 15
     ELSE
       RNOLIST=RNONEWLIST
       GO TO 115
     END IF
ELSE
    GO TO 300
       END IF


C ***************************************************************
C
C           END OF SUBPROBLEM (SCALING PHASE)
C
C ***************************************************************


300    CONTINUE

C ****** IF EPSILON IS 1 TERMINATE ******

      IF (EPSILON.EQ.1) THEN
   RETURN
      ELSE

C   ELSE REDUCE EPSILON AND UPDATE PARAMETERS FOR THE NEXT SCALING PHASE

    NUMPHASES=NUMPHASES+1
        EPSILON=INT(EPSILON/FACTOR)
   IF (EPSILON.GT.INCR) EPSILON=INT(EPSILON/FACTOR)
        IF ((EPSILON.LT.1).OR.(EPSILON.LT.ENDEPS)) EPSILON=1
   IF (STARTINCR.LT.EPSILON) STARTINCR=FACTOR*STARTINCR
        THRESH=INT(THRESH/FACTOR)

X   PRINT*,'*** END OF A SCALING PHASE; NEW EPSILON=',EPSILON
```

```
        GO TO 12
 END IF

 END



      SUBROUTINE SETRAN(ISEED)
      IMPLICIT REAL*8 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C PORTABLE CONGRUENTIAL (UNIFORM) RANDOM NUMBER GENERATOR:
C     NEXT_VALUE = [(7**5) * PREVIOUS_VALUE] MODULO[(2**31)-1]
C
C THIS GENERATOR CONSISTS OF TWO ROUTINES:
C   (1) SETRAN - INITIALIZES CONSTANTS AND SEED
C   (2) RRAN   - GENERATES A REAL RANDOM NUMBER
C
C THE GENERATOR REQUIRES A MACHINE WITH AT LEAST 32 BITS OF PRECISION.
C THE SEED (ISEED) MUST BE IN THE RANGE (1,(2**31)-1).
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IF(ISEED.LT.1) STOP 77
      MULT=16807
      MODUL=2147483647
      I15=2**15
      I16=2**16
      JRAN=ISEED
      RETURN
      END
C
      REAL FUNCTION RAN()
      IMPLICIT REAL*4 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C RAN GENERATES A REAL RANDOM NUMBER BETWEEN 0 AND 1
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IXHI=JRAN/I16
      IXLO=JRAN-IXHI*I16
      IXALO=IXLO*MULT
      LEFTLO=IXALO/I16
      IXAHI=IXHI*MULT
      IFULHI=IXAHI+LEFTLO
      IRTLO=IXALO-LEFTLO*I16
      IOVER=IFULHI/I15
      IRTHI=IFULHI-IOVER*I15
      JRAN=((IRTLO-MODUL)+IRTHI*I16)+IOVER
      IF(JRAN.LT.0) JRAN=JRAN+MODUL
      RAN = FLOAT(JRAN)/FLOAT(MODUL)
      RETURN
      END
```

\sect{NAUCTION_SP: Combined
Naive Auction and Sequential Shortest Path Code}This code implements the sequential shortest path method for the assignment problem, preceded by an extensive initialization using the naive auction algorithm (cf.\ Section 1.2.4 of the author's network optimization book).

```
C ***************************************************************
C
C COMBINED NAIVE AUCTION AND SEQUENTIAL SHORTEST PATH METHOD
C          FOR SYMMETRIC N X N ASSIGNMENT PROBLEMS
C             FINDS AN OPTIMAL ASSIGNMENT
C
C            WRITTEN BY DIMITRI P. BERTSEKAS
C
C ***************************************************************
C
C USER MUST SUPPLY THE FOLLOWING PROBLEM DATA
C N=NUMBER OF ROWS AND NUMBER OF COLUMNS
C A=NUMBER OF ARCS
C FOUT(ROW)=1ST ARC COMING OUT OF ROW
C COST(ARC)=COST OF ARC
C END(ARC)=COLUMN CORRESPONDING TO ARC
C
C FOUT(.) IS AN N - LENGTH ARRAY
C COST(.),END(.),NXTEND(.) ARE ARRAYS OF LENGTH A
C THE OPTIMAL ASSIGNMENT IS CONTAINED IN THE ARRAY ASSIGN(.) WHERE
C   ASSIGN(COL) IS THE ROW ASSIGNED TO COL
C THIS ALGORITHM DOES NOT CHECK FOR INFEASIBILITY OF THE PROBLEM
C TO MAKE SURE THE PROBLEM IS FEASIBLE THE USER MAY ADD
C   ADDITIONAL VERY HIGH COST ARCS
C
C THIS CODE ALLOWS MULTIPLE ARCS BETWEEN A ROW AND A COLUMN.
C
C ***************************************************************
C ALL PROBLEM DATA ARE INTEGER
C ***************************************************************
C
      IMPLICIT INTEGER (A-Z)
      INTEGER AUCTNUM,A,STARC,ENDARC,CURROW,ARC,COUNT
      INTEGER HCOUNT,ROW,FSTARC,FSTCOL,SNDARC,SNDCOL,TMAX,BSTCOL
      INTEGER TMARG,TA,ROWPR,OLMARG,PRICE,TPRICE,CURCOL,DUMMY
      INTEGER FOUT(6000),PCOL(6000),PROW(6000),MARG(6000)
      INTEGER ASSIGN(6000),COLLAB(6000),ROWLAB(6000)
      INTEGER LIST(6000),SCAN(6000)
      INTEGER COST(70000),END(70000)
□LOGICAL MAXSET
      REAL THRESH,RAND,TT,TIMER,TCOST

C
C ***************************************************************
C
C    PROBLEM GENERATION CODE STARTS HERE
C    THE USER MAY REPLACE THIS CODE WITH A CODE THAT READS
C    HIS/HER PROBLEM FROM A FILE
C
C ***************************************************************
C
C   RANDOM GENERATOR STUFF
C
    INTEGER MULT,MODUL,I15,I16,JRAN,ISEED
    REAL RAN
    COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
C    UNIFORM RANDOM NUMBER GENERATOR WHICH RETURNS A VALUE IN (0,1)
C
C    INITIALIZE RANDOM GENERATOR
```

```
C
      ISEED=13502460
      CALL SETRAN(ISEED)

      PRINT*,'GENERATING A SYMMETRIC ASSIGNMENT PROBLEM'
      PRINT*,'****************************************'

C **** READ THE NUMBER OF ROWS N & THE NUMBER OF ARCS A ****

      PRINT*,'ENTER THE NUMBER OF ROWS (AND COLUMNS)'
      READ*,N
5     PRINT*,'ENTER THE NUMBER OF ARCS PER ROW (>1)'
      READ*,NA
      IF (NA.LT.2) GOTO 5
      PRINT*,'ENTER THE MINIMUM AND THE MAXIMUM COST'
      READ*,MINCOST,MAXCOST

C     THE NUMBER OF ARCS IS N*NA

      A=N*NA

C     THE ARCS INCIDENT TO ROW I ARE FOUT(I) TO FOUT(I+1)-1
C     FOR FEASIBILITY EACH ROW IS DIRECTLY CONNECTED
C     WITH THE CORRESPONDING COLUMN

      DO 20 I=1,N
        FOUT(I)=1+(I-1)*NA
20    CONTINUE

C **** GENERATE THE END(IA) AND COST(IA) WHICH ARE THE COLUMN
C     AND THE COST COEFFICIENT ASSOCIATED WITH ARC IA ****

      DO 25 IA=1,A
        END(IA)=1+RAN()*N
IF ((END(IA).GT.N).OR.(END(IA).LT.1)) THEN
  PRINT*,'ERROR IN PROBLEM GENERATION'
  PAUSE
  STOP
END IF
COST(IA)=MINCOST+RAN()*(MAXCOST-MINCOST)
25    CONTINUE

C   MODIFY THE END OF THE LAST ARC OUT OF EACH ROW FOR FEASIBILITY
C   AND SET ITS COST TO -MAXCOST

      DO 30 I=1,N
        END(FOUT(I+1)-1)=I
COST(FOUT(I+1)-1)=-MAXCOST
30    CONTINUE

C
C ***************************************************************
C
C         PROBLEM GENERATION CODE ENDS HERE
C
C ***************************************************************
C

C      MAIN ALGORITHM BEGINS HERE
```

```
C
C      THE ALGORITHM INTERNALLY TREATS THE PROBLEM AS A
C      MINIMIZATION PROBLEM.
C      TO SOLVE AN ASSIGNMENT MAXIMIZATION PROBLEM, SET
C      THE FOLLOWING PARAMETER MAXSET TO TRUE, ELSE SET IT TO FALSE

       MAXSET=.TRUE.
 
 IF (MAXSET) THEN
         DO 35 IA=1,A
    COST(IA)=-COST(IA)
35        CONTINUE
       END IF


 PRINT*,'NO OF ROWS (AND COLUMNS):',N
 PRINT*,'NO OF ARCS:',A
       PRINT*,'COMBINED NAIVE AUCTION AND SEQ. SH. PATH METHOD'
 PRINT *,'*********************************************'
 TIMER = LONG(362)

       ISIMPL=0
       IPRC=0
       ISMALL=-20000000
       ILARGE=-ISMALL
       COUNT=0
       HCOUNT=0
       FOUT(N+1)=A+1

C      THE FOLLOWING INITIALIZATION UP TO THE START OF THE
C      NAIVE AUCTION CYCLES WAS SUGGESTED BY JONKER AND VOLGENANT

C      INITIALIZE COLUMN PRICES

       DO 105 J=1,N
         PCOL(J)=ILARGE
105     CONTINUE

C      FIND BEST ROW FOR EACH COLUMN

         DO 120 I=1,N
    PROW(I)=0
    ROWLAB(I)=0
    FSTARC=FOUT(I)
    LSTARC=FOUT(I+1)-1
    DO 110 ARC=FSTARC,LSTARC
      J=END(ARC)
      IF (COST(ARC).LT.PCOL(J)) THEN
        PCOL(J)=COST(ARC)
  ASSIGN(J)=I
      END IF
110         CONTINUE
120       CONTINUE
 
C      CONSTRUCT ROW ASSIGNMENT AND DEASSIGN MULTIPLY ASSIGNED ROWS

       DO 130 J=1,N
         J0=N-J+1
    I=ASSIGN(J0)
    IF (ROWLAB(I).NE.0) THEN
```

```
      ROWLAB(I)=-ABS(ROWLAB(I))
      ASSIGN(J0)=0
    ELSE
      ROWLAB(I)=J0
    END IF
130     CONTINUE

C     CONSTRUCT CANDIDATE LIST AND CORRECT COLUMN PRICES


      NEWNOL=0
      DO 150 I=1,N
    CURCOL=ROWLAB(I)
        IF (CURCOL.EQ.0) THEN
          NEWNOL=NEWNOL+1
          LIST(NEWNOL)=I
      GOTO 150
        END IF
        IF (CURCOL.LT.0) THEN
      ROWLAB(I)=-CURCOL
    ELSE
      MAX2=ISMALL
      FSTARC=FOUT(I)
      LSTARC=FOUT(I+1)-1
      DO 140 ARC=FSTARC,LSTARC
        J=END(ARC)
        IF (J.NE.CURCOL) THEN
      IF (PCOL(J)-COST(ARC).GT.MAX2) THEN
        MAX2=PCOL(J)-COST(ARC)
      END IF
        END IF
140         CONTINUE
          PROW(I)=MAX2
      PCOL(CURCOL)=PCOL(CURCOL)+MAX2
    END IF
150     CONTINUE


      IF (NEWNOL.EQ.0) GO TO 2000

C     ************************************************************

C      END OF INITIALIZATION; DO A NUMBER OF AUCTION CYCLES
C      WHICH IS SET BELOW IN THE PARAMETER AUCTNUM BASED ON THE
C      SPARSITY OF THE PROBLEM

      IF (N*N.LT.10*A) AUCTNUM=2
      IF ((N*N.GE.10*A).AND.(N*N.LT.25*A)) AUCTNUM=3
      IF ((N*N.GE.25*A).AND.(N*N.LT.50*A)) AUCTNUM=4
      IF ((N*N.GE.50*A).AND.(N*N.LT.100*A)) AUCTNUM=5
      IF (N*N.GE.100*A) AUCTNUM=6

C      TO RUN A PURE SEQUENTIAL SHORTEST PATH METHOD SET
C      AUCTNUM=0

      IF (AUCTNUM.EQ.0) GOTO 1000

C     START OF A NEW CYCLE
C
80      NOLIST=NEWNOL
```

```
      I=1
      NEWNOL=0

C     TAKE UP A ROW FOR ITERATION

100   ROW=LIST(I)
      I=I+1

      FSTARC=FOUT(ROW)
      LSTARC=FOUT(ROW+1)-1
      BSTCOL=END(FSTARC)
      IF (FSTARC.EQ.LSTARC) THEN
        OLDROW=ASSIGN(BSTCOL)
        ASSIGN(BSTCOL)=ROW
        ROWLAB(ROW)=BSTCOL
        PROW(ROW)=ISMALL
        PCOL(BSTCOL)=ISMALL+COST(FSTARC)
        IF (OLDROW.GT.0) THEN
          I=I-1
          LIST(I)=OLDROW
          ROWLAB(OLDROW)=0
        END IF
        ISIMPL=ISIMPL+1
        GO TO 100
      END IF
      SNDARC=FSTARC+1
      SNDCOL=END(SNDARC)
      MAX1=PCOL(BSTCOL)-COST(FSTARC)
      MAX2=PCOL(SNDCOL)-COST(SNDARC)
IF (MAX1.LT.MAX2) THEN
  TMAX=MAX1
  MAX1=MAX2
  MAX2=TMAX
  FSTCOL=BSTCOL
  BSTCOL=SNDCOL
  SNDCOL=FSTCOL
END IF
      IF (SNDARC.LT.LSTARC) THEN
       TRDARC=SNDARC+1
       DO 108 ARC=TRDARC,LSTARC
         CURCOL=END(ARC)
         TMAX=PCOL(CURCOL)-COST(ARC)
         IF (TMAX.GT.MAX2) THEN
    IF (TMAX.GT.MAX1) THEN
MAX2=MAX1
MAX1=TMAX
SNDCOL=BSTCOL
BSTCOL=CURCOL
    ELSE
MAX2=TMAX
SNDCOL=CURCOL
    END IF
         END IF
108     CONTINUE
      END IF
      PROW(ROW)=MAX2
      OLDROW=ASSIGN(BSTCOL)
      INCR=MAX1-MAX2
IF (INCR.GT.0) THEN
  PCOL(BSTCOL)=PCOL(BSTCOL)-INCR
```

```
      ASSIGN(BSTCOL)=ROW
      ROWLAB(ROW)=BSTCOL
      IF (OLDROW.GT.0) THEN
              I=I-1
              LIST(I)=OLDROW
              ROWLAB(OLDROW)=0
              ISIMPL=ISIMPL+1
        GOTO 100
      END IF
    ELSE
      IF (OLDROW.GT.0) THEN
        BSTCOL=SNDCOL
        OLDROW=ASSIGN(BSTCOL)
      END IF
      IF (OLDROW.EQ.0) THEN
        ASSIGN(BSTCOL)=ROW
        ROWLAB(ROW)=BSTCOL
      ELSE
        NEWNOL=NEWNOL+1
        LIST(NEWNOL)=ROW
      END IF
    END IF
          ISIMPL=ISIMPL+1
          IF (I.LE.NOLIST) GOTO 100

C     END OF A NAIVE AUCTION CYCLE

          COUNT=COUNT+1

          IF (NEWNOL.EQ.0) THEN
      NASSIH=NEWNOL
      GOTO 2000
    END IF
          IF (COUNT.LT.AUCTNUM) GOTO 80



C ***************************************************************
C
C       ******* END OF NAIVE AUCTION PART *******
C
C
C       ******* START OF SEQ. SH. PATH METHOD *******
C
C ***************************************************************



1000     NASSIH=NEWNOL
X        IHPRC=0
1020     DO 180 I=1,NEWNOL
           SCAN(I)=LIST(I)
180      CONTINUE
         DO 190 J=1,N
           MARG(J)=1
190      CONTINUE
         NOSCAN=NEWNOL
         IL1=1
         IL2=NOSCAN
1030     DO 1060 I=IL1,IL2
```

```
        CURROW=SCAN(I)
        ROWPR=PROW(CURROW)
      FSTARC=FOUT(CURROW)
      LSTARC=FOUT(CURROW+1)-1
      DO 1050 ARC=FSTARC,LSTARC
            CURCOL=END(ARC)
            OLMARG=MARG(CURCOL)
            IF (OLMARG.EQ.0) GO TO 1050
            TMARG=PCOL(CURCOL)-ROWPR-COST(ARC)
        IF (TMARG.EQ.0) THEN
          IF (ASSIGN(CURCOL).EQ.0) THEN
            GO TO 1500
C     PERFORM AUGMENTATION

          ELSE
            MARG(CURCOL)=0
            NOSCAN=NOSCAN+1
            SCAN(NOSCAN)=ASSIGN(CURCOL)
            COLLAB(CURCOL)=CURROW
          END IF
        ELSE
          IF ((OLMARG.GT.0).OR.(TMARG.GT.OLMARG)) THEN
            MARG(CURCOL)=TMARG
            COLLAB(CURCOL)=CURROW
          END IF
        END IF
1050    CONTINUE
1060    CONTINUE

C       CURRENT SCAN PHASE COMPLETE; CHECK FOR MORE ROWS TO SCAN

      IF (IL2.LT.NOSCAN) THEN
        IL1=IL2+1
        IL2=NOSCAN
        GO TO 1030
      END IF


C       PRICE CHANGE


      IPRC=IPRC+1
X       IHPRC=IHPRC+1
      INCR=ISMALL
      DO 200 J=1,N
        TMARG=MARG(J)
        IF ((TMARG.LT.0).AND.(TMARG.GT.INCR)) INCR=TMARG
200     CONTINUE
      DO 210 I=1,NOSCAN
        PROW(SCAN(I))=PROW(SCAN(I))+INCR
210     CONTINUE
      DO 220 J=1,N
        IF (MARG(J).EQ.0) PCOL(J)=PCOL(J)+INCR
220     CONTINUE
      DO 1400 J=1,N
        IF (MARG(J).GE.0) GO TO 1400
         MARG(J)=MARG(J)-INCR
          IF (MARG(J).EQ.0) THEN
            IF (ASSIGN(J).EQ.0) THEN
```

```
              CURCOL=J
              CURROW=COLLAB(CURCOL)

C      PERFORM AUGMENTATION

              GO TO 1500
             ELSE
              NOSCAN=NOSCAN+1
              SCAN(NOSCAN)=ASSIGN(J)
             END IF
           END IF
1400    CONTINUE
        IL1=IL2+1
        IL2=NOSCAN
        GO TO 1030
 
C       END PRICE CHANGE


C       AUGMENTATION STARTS HERE

1500    IF (ROWLAB(CURROW).EQ.0) THEN
          ASSIGN(CURCOL)=CURROW
          ROWLAB(CURROW)=CURCOL
          GO TO 1700
        END IF
        TA=ROWLAB(CURROW)
        ASSIGN(CURCOL)=CURROW
        ROWLAB(CURROW)=CURCOL
        CURCOL=TA
        CURROW=COLLAB(CURCOL)
        GO TO 1500
1700    IF (NEWNOL.EQ.1) GO TO 2000

        DO 240 I=1,NEWNOL-1
          IF (LIST(I).EQ.CURROW) THEN
            DO 250 K=I+1,NEWNOL
              LIST(K-I)=LIST(K)
250         CONTINUE
            DO 260 K=1,I-1
              LIST(NEWNOL-I+K)=SCAN(K)
260         CONTINUE
            NEWNOL=NEWNOL-1
            GO TO 1020
          END IF

240     CONTINUE

        NEWNOL=NEWNOL-1
        GO TO 1020
 
C       END AUGMENTATION


C ******************************************************
C
C       EXIT ROUTINE. CHECKS FOR OPTIMALITY OF THE SOLUTION
C       CALCULATES THE OPTIMAL COST, AND COMPILES SOLUTION
C       STATISTICS. THE USER MAY REPLACE THIS BY CODE THAT
C       WRITES AT THE APPROPRIATE DEVICE THE OPTIMAL SOLUTION
```

```
C       CONTAINED IN THE ARRAY ASSIGN(.).
C
C ******************************************************

2000    CONTINUE

        TT =(LONG(362) - TIMER)/60

        PRINT*, 'TOTAL TIME = ',TT, ' secs.'
        PRINT*,'NO OF NAIVE AUCTION ITERATIONS:',ISIMPL
        PRINT*,'NO OF SH. PATH ITERATIONS:',NASSIH
X         PRINT*,'NO OF PRICE CHANGES:',IHPRC

C       CHECK FEASIBILITY OF SOLUTION & CALCULATE COST

        DO 265 I=1,N
          ROWLAB(I)=0
265     CONTINUE

        NOASS=0
        DO 280 J=1,N
          IF (ASSIGN(J).GT.0) THEN
            ROWLAB(ASSIGN(J))=J
            NOASS=NOASS+1
          END IF
280     CONTINUE

        IF (NOASS.LT.N) THEN
          PRINT*,'THE NUMBER OF ASSIGNED COLUMNS IS',NOASS,'(TOO SMALL)'
        END IF

        TCOST=0
        DO 300 I=1,N
          J=ROWLAB(I)
          IF (J.EQ.0) THEN
            PRINT*,'ROW ',I,' IS UNASSIGNED'
          END IF
          FSTARC=FOUT(I)
          LSTARC=FOUT(I+1)-1
          MINCOST=ILARGE
          DO 310 ARC=FSTARC,LSTARC
            CURCOL=END(ARC)
            TMARG=PROW(I)-PCOL(CURCOL)+COST(ARC)
            IF (TMARG.LT.0) THEN
              PRINT*,'COMPL. SLACKNESS VIOLATION: ROW',I,' COLUMN',CURCOL
            END IF
            IF (CURCOL.EQ.J) THEN
              IF (MINCOST.GT.COST(ARC)) THEN
                MINCOST=COST(ARC)
                ROWMARG=TMARG
              END IF
            END IF
310       CONTINUE
          IF (ROWMARG.NE.0) THEN
            PRINT*,'COMPL. SLACK. VIOLATION AT ASSIGNED ARC OF ROW',I
          END IF
          IF (MAXSET) THEN
            TCOST=TCOST-MINCOST
          ELSE
            TCOST=TCOST+MINCOST
```

```
  END IF
300    CONTINUE

 WRITE(9,2100) TCOST
2100    FORMAT(' ASSIGNMENT COST=',F14.2)

 PRINT *,'*********************************************'
 PRINT *, 'PROGRAM ENDED; PRESS <CR>'

 PAUSE
 STOP

 END
 
 
 
      SUBROUTINE SETRAN(ISEED)
      IMPLICIT REAL*8 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C PORTABLE CONGRUENTIAL (UNIFORM) RANDOM NUMBER GENERATOR:
C     NEXT_VALUE = [(7**5) * PREVIOUS_VALUE] MODULO[(2**31)-1]
C
C THIS GENERATOR CONSISTS OF TWO ROUTINES:
C   (1) SETRAN - INITIALIZES CONSTANTS AND SEED
C   (2) RRAN   - GENERATES A REAL RANDOM NUMBER
C
C THE GENERATOR REQUIRES A MACHINE WITH AT LEAST 32 BITS OF PRECISION.
C THE SEED (ISEED) MUST BE IN THE RANGE (1,(2**31)-1).
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IF(ISEED.LT.1) STOP 77
      MULT=16807
      MODUL=2147483647
      I15=2**15
      I16=2**16
      JRAN=ISEED
      RETURN
      END
C
      REAL FUNCTION RAN()
      IMPLICIT REAL*4 (A-H,O-Z) , INTEGER*4 (I-N)
C**********************************************************************
C RAN GENERATES A REAL RANDOM NUMBER BETWEEN 0 AND 1
C**********************************************************************
      COMMON /RANDM/ MULT,MODUL,I15,I16,JRAN
      IXHI=JRAN/I16
      IXLO=JRAN-IXHI*I16
      IXALO=IXLO*MULT
      LEFTLO=IXALO/I16
      IXAHI=IXHI*MULT
      IFULHI=IXAHI+LEFTLO
      IRTLO=IXALO-LEFTLO*I16
      IOVER=IFULHI/I15
      IRTHI=IFULHI-IOVER*I15
      JRAN=((IRTLO-MODUL)+IRTHI*I16)+IOVER
      IF(JRAN.LT.0) JRAN=JRAN+MODUL
      RAN = FLOAT(JRAN)/FLOAT(MODUL)
      RETURN
      END
```