APPENDIX C

Contents:

1. **********
2. ********** INSTRUCTIONS FOR BTAP USER **********
3. **********
4.
5.
6.
7.
8.
9.
10.
11.
12. July 1979

13.

14. This program is based on "BTAP: A Computer Program
15. to Obtain Solutions to the Transportation-Allocation
16. Problem and Other Travelling Salesman Type Problems" by
17. Or and Pierskalla(1976)*. Several modifications are made
18. to the original program:
19. (a) Some machine dependent features are replaced by
20. standard FORTRAN IV statements.
21. (b) Plotting routines and program timers are not used
22. in this version.
23. (c) Five built-in options are available to the user
24. for selecting the allocation algorithms.
25. (d) The input instructions are simplified.
26. (f) Bounds on the capacities of each blood banks can be
27. specified.

28.
29.

30. ----------
31. * Or and Pierskalla, Dept. of Industrial Engineering and
32. Management Sciences, Northwestern University, Evanston,
33. Ill 60201, August 1976.

I. Program Options

The current version of this program has five options for selecting the allocation and allocation improvement algorithms. In addition, vehicle routing and dispatching can be performed in each option. These five options are:

Option 1: To find the optimal allocation and routing which minimize the expected emergency blood delivery cost.

Option 2: To find the optimal allocation and routing which minimize the vehicle routing cost.

Option 3: To find the optimal allocation and routing which minimize both emergency blood delivery and vehicle routine cost. An improvement algorithm is used to test pair-wise independent exchanges of hospitals between two banks for any possible cost saving.

Option 4: Same as Option 3 except that the exchanges of hospitals are performed among several banks in various combinations. This option will slow down the program excution considerably but may yield better results.

Option 5: To provide the allocation by user himself.

For each option, the user should decide whether the vehicle dispatching and bounds on the blood bank capacities are to be used or not. The bounding method in this program is a penality function technique. The penality for the excessive amount X over the bound is given by the following function

$$\text{Penality} = a * \lceil X / 1000 \rceil + b * X,$$

where [X / 1000] is the integer part of X/1000; a and b are penality parameters. The values of a and b can be modified in SUBROUTINE PNALTY. They are currently set at a=10, b=500.

II. Input Instructions

This program needs two to three input files depending on which program is selected. Files are numbered by the reading unit in the program. File #3 and #5 are necessary for all options; File #8 is needed only when Option 5 is used.

File #3 contains the informations for the geographical coordinates of hospitals and the corresponding emergency frequencies and expected blood usage. File #5 contains the program option control cards, number of blood banks, blood bank ID numbers, and bounds on the capacities for each bank. File #8 contains the allocation of hospitals assigned by the user. File #3 and #8 are assumed to be tape files or disk files. File #5 can be read in from keyboard or card reader. All of them are card image files.

The details in each file are listed below:

a) File #3

Card	Column	Format	Information
1	1-4	I4	Number of hospitals, n
2	1-5	F5.0	X-coordinate of hospital 1
	6-10	F5.0	Y-coordinate of hospital 1
	11-15	F5.0	Expected number of emergency deliveries per period for hospital 1
3	---- same as card #2 for hospital 2 ----------		
4	---- same as card #2 for hospital 3 ----------		
	- - - - - - - - - - - - - - - -		
	- - - - - - - - - - - - - - - -		
n	---- same as card #2 for hospital n ----------		
n+1	1-4	I4	Hospital index
	5-10	I6	Amount of blood used per year in this hospital
n+2	---- same as card #n+1 ----------		
n+3	---- same as card #n+1 ----------		
	- - - - - - - - - - - - - - - -		
	- - - - - - - - - - - - - - - -		
2n+1	---- same as card #n+1 ----------		

b) File #5

Card	column	Fromat	Information
1	1-2	I2	Number of blood banks, m
	3-7	I5	Index of hospital which is used as bank 1
	8-12	I5	- - - - same as above ------ bank 2
	13-17	I5	- - - - same as above ----- bank 3
	.....		
	.....		
2	1-2	I2	Index of the options to be used.
	4	L1	'T' if bounds are used for the capacities of each bank; 'F' if not used.
	5	L1	'T' if vehicle dispatching is to be performed; 'F' if not.
	(The following columns are needed if column 5 is T.)		
	6-10	I5	Maximum number of stops allowed for each vehicle
	11-15	I5	Carrying capacity for each vehicle
(The following card(s) is necessary only when column 4 of the second card is T.)			
3	1-10	I10	Lower bound for bank 1
	11-20	I10	Upper bound for bank 1
	21-30	I10	Lower bound for bank 2
	31-40	I10	Upper bound for bank 2
	41-50	I10	..... bank 3
	51-60	I10	..... bank 3
	.....		
	.....		
4	1	I1	SYSTEM COST OPTION. OPTIONS 1 THROUGH 4 COMPUTE THE COST FOR BOTH THE REGION AND THE CENTRAL BANKS. OPTION 5 COMPUTES THE COST ONLY FOR THE CENTRAL BANKS.

c) File #8

(This file is needed only when Option 5 is chosen.)

Card	Column	Format	Information	
1	6-10	I5	Index of the bank to which hospital 1 is assigned.	
	11-15	I5	----------	hospital 2 ---
	16-20	I5	----------	hospital 3 ---
	...	..	.....	
	...	..	.....	
	50-55	I5	----------	hospital 10 --
2	6-10	I5	----------	hospital 11 --
	11-15	I5	----------	12 --
	...	..	.....	
	...	..	.....	
	50-55	I5	----------	20 --
	.....			
	.....			
	and so on			
			----------	hospital n ----

**** Note 1: Format I indicates an integer without decimal point.
Format F indicates a real number with a decimal point.
**** Format L indicates a 'T' or 'F'.
**** Note 2: The maximum number of hospitals allowed is 150; the maximum number of banks allowed is 10.
**** Note 3: When bounds on the capacities are used, the SUBROUTINE TEST performs interchange testing even when the direction of change may increase the value of the penality function. It is very inefficient but the current program structure only allows such testing.

```
1.      //BTAP JOB (T595,005F,2,005,AA),'DEUERMEYER'
2.      /*LEVEL        0
3.      /*JOBPARM R=192
4.      // EXEC FORTX,REGION=192K
5.      //GO.FT07F001 DD SYSOUT=A
6.      //GO.FT03F001 DD DSN=WYL.AA.TQT.FILE3,DISP=SHR
7.      //GO.FT08F001 DD DSN=WYL.AA.TQT.FILE8,DISP=SHR
8.      //SOURCE DD *
9.            COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
10.           COMMON/BKSQ/NHOSP,NBANK
11.           LOGICAL VDP,TIMES,ALLOC1,ALLOC2,PLOTA1,PLOTA2
12.           LOGICAL IMPRV1,IMPRV2,OPTION(4,10),BOUND
13.           INTEGER VCAP
14.           COMMON/OPT/OPTION,PLOTA1,PLOTA2,TIMES
15.           COMMON/BDD/BOUND,LVOL(5),MVOL(5)
16.           COMMON/CO/JD
17.       311 FORMAT(/2X,'COMPUTATIONS ARE NOW STARTING, TIME IS ',F8.3/)
18.       312 FORMAT(/2X,'ROUTINGS BASED ON ALLOC1 ARE COMPUTED, TIME IS ',F8.3
19.       313 FORMAT(/2X,'ROUTINGS BASED ON ALLOC2 ARE COMPUTED, TIME IS ',F8.3
20.       314 FORMAT(/2X,'ALL INDEPENDENT EXCHANGES ARE TESTED, TIME IS ',F8.3)
21.       315 FORMAT(/2X,'DEPENDENT EXCHANGES ARE TESTED, TIME IS ',F8.3)
22.       316 FORMAT(/2X,'MULTIPLE VEHICLE SOLUTION IS COMPUTED, TIME IS ',F8.3
23.           READ(5,300) NBANK,(IBANK(J),J=1,NBANK)
24.           WRITE(6,901) NBANK,(IBANK(J),J=1,NBANK)
25.           READ(5,301)KOPT,BOUND,VDP,MSTOP,VCAP
26.       301 FORMAT(I2,1X,2L1,2I5)
27.           WRITE(6,701)KOPT
28.       701 FORMAT(' PROGRAM OPTION #',I2,' IS USED')
29.       901 FORMAT(' NUMBER OF BANKS=',I2,', BANK ID#=',10I4)
30.       300 FORMAT(I2,10(1X,I4))
31.           ALLOC1=OPTION(1,KOPT)
32.           ALLOC2=OPTION(2,KOPT)
33.           IMPRV1=OPTION(3,KOPT)
34.           IMPRV2=OPTION(4,KOPT)
35.           WRITE(6,702) (OPTION(J,KOPT),J=1,4),BOUND
36.       702 FORMAT(1X///' ALLOCATION PARAMETERS:',4X,'ALLOC1',2X,'ALLOC2',
37.          &2X,'IMPRV1',2X,'IMPRV2',3X,'BOUND'/25X,5(7X,L1))
38.           IF(.NOT.BOUND)GO TO 88
39.           READ(5,801) (LVOL(J),MVOL(J),J=1,NBANK)
40.       801 FORMAT(8I10)
41.           WRITE(6,802)
42.       802 FORMAT(' BOUNDS ON THE VOLUMES FOR EACH BANK:'//6X,
43.          &'BANK',4X,'LOWER BOUND',2X,'UPPER BOUND')
44.           DO 89 J=1,NBANK
45.        89 WRITE(6,803)J,LVOL(J),MVOL(J)
46.       803 FORMAT(8X,I2,5X,I10,3X,I10)
47.        88 CONTINUE
48.           WRITE(6,304)VDP,MSTOP,VCAP
49.       304 FORMAT(1X///' DISPATCHING PARAMETERS:',7X,'VDP',3X,'MSTOP',4X,
50.          &'VCAP'/32X,L1,2(3X,I5))
51.           READ(5,305) JD
52.       305 FORMAT (I1)
53.           WRITE(6,308)
```

```
      308 FORMAT (//)
          WRITE(6,303) JD
      303 FORMAT (/2X,'SYSTEM COST OPTION:',2X,I1)
          CALL READ1
          CALL DISMAT(2.0)
          IF(.NOT.TIMES) GO TO 1
          X=SECOND(X)
          WRITE(6,311) X
        1 CONTINUE
          IF (.NOT. ALLOC1) GO TO 20
          CALL ALOC1(3.0)
          CALL TRAVEL(0)
          IF(PLOTA1) CALL PLOTNG(0,NBANK,NBANK)
          IF(.NOT.TIMES) GO TO 20
          X=SECOND(X)
          WRITE(6,312) X
       20 IF (.NOT. ALLOC2) GO TO 40
          CALL ALOC2(3.0)
          CALL TRAVEL(1)
          IF(PLOTA2) CALL PLOTNG(1,NBANK,NBANK)
          IF(.NOT.TIMES) GO TO 40
          X=SECOND(X)
          WRITE(6,313) X
       40 KX=0
          IF ((.NOT. ALLOC1) .AND. (.NOT. ALLOC2)) KX=-2
          IF ((.NOT. ALLOC1) .OR. (.NOT. ALLOC2)) GO TO 41
C
          IF (IMPRV1) CALL IMPROV(1)
          IF (IMPRV2) CALL IMPROV(2)
          IF(.NOT.TIMES) GO TO 41
          X=SECOND(X)
          IF (IMPRV1 .AND. TIMES) WRITE(6,314) X
          IF (IMPRV2 .AND. TIMES) WRITE(2,315) X
       41 CONTINUE
          IF (.NOT. VDP) STOP
          IF (ALLOC2) KX=1
          IF (IMPRV1 .OR. IMPRV2) KX=-1
          CALL DISPAC(KX,MSTOP,VCAP)
          IF(.NOT.TIMES) STOP
          X=SECOND(X)
          WRITE(6,316) X
          STOP
          END
          BLOCK DATA
          LOGICAL BOUND,OPTION(4,10),PLOTA1,PLOTA2,TIMES
          COMMON/OPT/OPTION,PLOTA1,PLOTA2,TIMES
          DATA PLOTA1,PLOTA2,TIMES/.FALSE.,.FALSE.,.FALSE./
          DATA OPTION/.TRUE.,.FALSE.,.FALSE.,.FALSE.,.FALSE.,.TRUE.,
         &.FALSE.,.FALSE.,3*.TRUE.,.FALSE.,2*.TRUE.,.FALSE.,.TRUE.,
         &4*.FALSE.,20*.FALSE./
          END
          SUBROUTINE READ1
          COMMON /E/X(150),Y(150),ALFA(150),IBLAD(150)
          COMMON/BKSQ/NHOSP,NBANK
```

```
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
  102 FORMAT(3F5.0)
  104 FORMAT(I4,I6)
      DO 2 I=1,150
    2 IBLAD(I)=0
      READ(3,700) NHOSP
       WRITE(6,701) NHOSP
  701 FORMAT(1X///' NUMBER OF HOSPITALS=',I4)
  700 FORMAT(I4)
      IF(NHOSP.GT.150) STOP
      DO 1 N=1,NHOSP
    1 READ (3,102) X(N),Y(N),ALFA(N)
      DO 10 J=1,NHOSP
      READ(3,104) I,IBLD
      IBLAD(I)=IBLD
   10 CONTINUE
      RETURN
      END
      SUBROUTINE DISMAT(Z)
      COMMON /E/X(150),Y(150),ALFA(150),IBLAD(150)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/CMM/DM(11325)
      N=NHOSP
      DO 16 I=1,N
      II1=I-1
      I1=I-1
      KK= II1*N-(I*I1)/2
      DO 16 J=I,N
      DUM=(Y(I)-Y(J))**2+(X(I)-X(J))**2
      K=KK+J
   16 DM(K)= SQRT(DUM)
      RETURN
      END
      SUBROUTINE ALOC1(Z)
      COMMON/A/IHOSP(150,5)/E/X(150),Y(150),ALFA(150),IBLAD(150)
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
      COMMON/BKSQ/NHOSP,NBANK
  100 FORMAT(5X,10I5)
      N=NHOSP
      DO 1 I=1,NBANK
      EMCOST(I)=0
      NBLAD(I)=0
      NUM(I)=0
    1 CONTINUE
      DO 5 I=1,N
      II=IBANK(1)
      DUM=DMAT(I,II)
      NBR=1
      IF(NBANK.EQ.1) GO TO 6
      DO 4 J=2,NBANK
      KK=IBANK(J)
      DMT=DMAT(I,KK)
      IF(DMT.GE.DUM) GO TO 4
```

```
162.              DUM=DMT
163.             NBR=J
164.           4 CONTINUE
165.           6 CONTINUE
166.             IHOSP(I,4)=NBR
167.             NUM(NBR)=NUM(NBR)+1
168.             EMCOST(NBR)=EMCOST(NBR)+ALFA(I)*DUM
169.             NBLAD(NBR)=NBLAD(NBR)+IBLAD(I)
170.           5 CONTINUE
171.             WRITE(7,100)  (IHOSP(I,4),I=1,N)
172.             RETURN
173.             END
174.             SUBROUTINE ALOC2(Z)
175.             COMMON/A/IHOSP(150,5)/D/CANDID(600,4),IADRES(600)
176.             COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
177.             COMMON /E/X(150),Y(150),ALFA(150),IBLAD(150)
178.             COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
179.              COMMON/BKSQ/NHOSP,NBANK
180.         100 FORMAT(5X,10I5)
181.             N=NHOSP
182.             NSIZE=0
183.             KPOINT=0
184.             DO 5 I=1,N
185.           5 IHOSP(I,3)=0
186.             DO 6 J=1,NBANK
187.             EMCOST(J)=0
188.             NBLAD(J)=0
189.             JJ=IBANK(J)
190.           6 IHOSP(JJ,3)=J
191.             DO 10 I=1,NBANK
192.             II=IBANK(I)
193.             ISTAR=II
194.             DO 7 MM=1,3
195.             DMIN=9999
196.             DO 9 J=1,N
197.             IF (IHOSP(J,3) .NE. 0) GO TO 9
198.             DJ=DMAT(II,J)
199.             IF(DJ.GE.DMIN) GO TO 9
200.             DMIN=DJ
201.             IEND=J
202.           9 CONTINUE
203.             IHOSP(IEND,3)=I
204.             CALL PFIND(NSIZE,ISTAR,IEND,0,4,4,4)
205.             ISTAR=IEND
206.           7 CONTINUE
207.             NUM(I)=4
208.             CALL PFIND(NSIZE,ISTAR,II,0,4,4,4)
209.          10 CONTINUE
210.             MM=N-4*NBANK
211.             DO 30 JJ=1,MM
212.          11 KPOINT=KPOINT+1
213.             ISTAR=IADRES(KPOINT)
214.             IBAR=CANDID(ISTAR,2)
215.             K1=CANDID(ISTAR,3)
```

```
      K2=CANDID(ISTAR,4)
      IF (IHOSP(IBAR,3) .EQ. 0) GO TO 25
      CALL PFIND(NSIZE,K1,K2,KPOINT,4,4,4)
      GO TO 11
   25 IGRUP=IHOSP(K1,3)
      IHOSP(IBAR,3)=IGRUP
      NUM(IGRUP)=NUM(IGRUP)+1
      CALL PFIND(NSIZE,K1,IBAR,KPOINT,4,4,4)
      CALL PFIND(NSIZE,IBAR,K2,KPOINT,4,4,4)
   30 CONTINUE
      DO 40 I=1,N
      IHOSP(I,5)=IHOSP(I,3)
      IB=IHOSP(I,3)
      IBK=IBANK(IB)
      EMCOST(IB)=EMCOST(IB)+ALFA(I)*DMAT(I,IBK)
   39 NBLAD(IB)=NBLAD(IB)+IBLAD(I)
   40 CONTINUE
      WRITE(7,100)  (IHOSP(I,5),I=1,N)
      RETURN
      END
      SUBROUTINE TRAVEL(KX)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)/B/IHULLA(150,2)
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
      COMMON/BKSQ/NHOSP,NBANK
      N=NHOSP
      ISTAR=0
      DO 5 II=1,NBANK
      MEM=NUM(II)
      CALL CHULL1(IC,KX,II)
      CALL BOUND1(IC,TSCOST,KX,II)
      IF (NUM(II) .GE. 5) CALL REFIN1(TSCOST,KX,II)
      DO 6 I=1,MEM
    6 IHULLA(ISTAR+I,KX+1)=IHULL(I,1)
      ISTAR=ISTAR+NUM(II)
    5 CONTINUE
      CALL PRINTS(KX,1,1,1)
      RETURN
      END
      SUBROUTINE DISPAC(KX,STOPS,UNITS)
      INTEGER BPRDAY,STOPS,UNITS
      COMMON/A/IHOSP(150,5)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)/B/IHULLA(150,2)
      COMMON/F/IBLDTR(50),IBTR(5),R(150),TETA(150),IPOL(100),NUMTR(50)
     1,COST(9),IBLADT(9),IDUM(352)
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
      COMMON/E/X(150),Y(150),ALFA(150),IBLAD(150)
  100 FORMAT(5X,10I5)
  200 FORMAT(1H1/9X,'THE FOLLOWING ALLOCATION IS SUPPLIED EXTERNALLY'//
  210 FORMAT(6X,10I5)
      KK=KX
      N=NHOSP
      PI=3.141593
      IF (KX .LT. 0) KK=0
```

```
270.            IF (KX .NE. -2) GO TO 3
271.            READ (8,100) (IHOSP(I,4+KK), I=1,N)
272.            WRITE(6,200)
273.            WRITE(6,210) (IHOSP(I,4+KK), I=1,N)
274.          3 DO 5 IB=1,NBANK
275.            INUM=IBANK(IB)
276.            EMCOST(IB)=0.0
277.            NBLAD(IB)=0
278.            DO 4 I=1,N
279.            IF (IHOSP(I,4+KK) .NE. IB) GO TO 4
280.            DMT=DMAT(I,INUM)
281.            EMCOST(IB)=EMCOST(IB)+DMT
282.            NBLAD(IB)=NBLAD(IB)+IBLAD(I)
283.            IF (INUM .EQ. I) GO TO 4
284.            R(I)=DMT
285.            IF (X(I) .EQ. X(INUM)) GO TO 1
286.            TETA(I)=ATAN((Y(I)-Y(INUM))/(X(I)-X(INUM)))
287.            IF((Y(I)-Y(INUM))/(X(I)-X(INUM)).LT.0.0) TETA(I)=TETA(I)+PI
288.            IF (Y(I) .GT. Y(INUM)) GO TO 4
289.            IF (Y(I) .EQ. Y(INUM)) GO TO 2
290.            TETA(I)=TETA(I)+PI
291.            GO TO 4
292.          1 IF (Y(I) .GE. Y(INUM)) TETA(I)=PI/2.0
293.            IF (Y(I) .LT. Y(INUM)) TETA(I)=(3.0*PI)/2.0
294.            GO TO 4
295.          2 IF (X(I) .LT. X(INUM)) TETA(I)=PI
296.          4 CONTINUE
297.          5 CONTINUE
298.            ISTAR=0
299.            JSTAR=0
300.            DO 50 IB=1,NBANK
301.            INUM=IBANK(IB)
302.            TSCOST(IB)=0.0
303.            NS=0
304.            IBTR(IB)=0
305.            DO 15 J=1,N
306.            IHOSP(J,5-KK)=0
307.            IF (IHOSP(J,4+KK) .NE. IB) GO TO 15
308.            IF (INUM .EQ. J) GO TO 15
309.            NS=NS+1
310.            IF (NS .EQ. 1) GO TO 10
311.            IEND=NS-1
312.            DO 7 I=1,IEND
313.            JPOINT=IPOL(I)
314.            IF (TETA(J) .LT. TETA(JPOINT)) GO TO 8
315.            IF (TETA(J) .GT. TETA(JPOINT)) GO TO 7
316.            IF (R(J) .LT. R(JPOINT)) GO TO 8
317.          7 CONTINUE
318.         10 IPOL(NS)=J
319.            GO TO 15
320.          8 JEND=NS-I
321.            DO 9 JJ=1,JEND
322.          9 IPOL(NS+1-JJ)=IPOL(NS-JJ)
323.            IPOL(I)=J
```

```
      15 CONTINUE
         WRITE(6,100) (IPOL(J),J=1,NS)
         IBLD=0
         NM=0
         NTR=1
         DO 30 J=1,NS
         I=IPOL(J)
          BPRDAY=IBLAD(I)/260.+0.999
         IF ((IBLD+BPRDAY .GT. UNITS) .OR. (NM+1 .GT. STOPS)) GO TO 25
         IBLD=IBLD+BPRDAY
         NM=NM+1
         IHOSP(I,5-KK)=NTR
         GO TO 30
      25 NUM(NTR)=NM
         IBLADT(NTR)=IBLD
         NTR=NTR+1
         NM=1
         IBLD=BPRDAY
         IHOSP(I,5-KK)=NTR
      30 CONTINUE
         NUM(NTR)=NM
         IBLADT(NTR)=IBLD
         IBTR(IB)=NTR
         WRITE(6,100) NTR,(NUM(J),J=1,NTR)
         DO 40 J=1,NTR
         IHOSP(INUM,5-KK)=J
         NUM(J)=NUM(J)+1
         CALL CHULL1(IC,1-KK,J)
         CALL BOUND1(IC,COST,1-KK,J)
         IF (NUM(J) .GE. 6) CALL REFIN1(COST,1-KK,J)
         NS=NUM(J)
         DO 41 I=1,NS
      41 IHULLA(ISTAR+I,KK+1)=IHULL(I,1)
         TSCOST(IB)=TSCOST(IB)+COST(J)
         NUMTR(JSTAR+J)=NUM(J)
         IBLDTR(JSTAR+J)=IBLADT(J)
         ISTAR=ISTAR+NUM(J)
      40 CONTINUE
         JSTAR=JSTAR+NTR
      50 CONTINUE
         DO 55 J=1,JSTAR
      55 NUM(J)=NUMTR(J)
         I=STOPS
         J=UNITS
         CALL PRINTS(KX,2,I,J)
         CALL PLOTNG(KX,NBANK,JSTAR)
         RETURN
         END
         SUBROUTINE CHULL1(IC,KX,IX)
         COMMON/A/IHOSP(150,5)/E/X(150),Y(150),ALFA(150),IBLAD(150)
         COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
         COMMON/BKSQ/NHOSP,NBANK
         N=NHOSP
         IC=0
```

```
      DO 1 KY=1,N
      I=KY
      IF (IHOSP(I,4+KX) .EQ. IX) GO TO 2
    1 CONTINUE
      STOP
    2 BEST=X(I)
      MEND=I
      XLAST=X(I)
      MSTAR=I
      DO 5 KY=1,N
    5 IHOSP(KY,3)=0
      ISTART=I+1
      DO 10 J=ISTART,N
      IF (IHOSP(J,4+KX) .NE. IX) GO TO 10
      IF (X(J) .GE. BEST) GO TO 8
      MSTAR=J
      BEST=X(J)
    8 IF (X(J) .LE. XLAST) GO TO 10
      MEND=J
      XLAST=X(J)
   10 CONTINUE
      IHOSP(MSTAR,3)=1
      M=MSTAR
   11 BEST=-10000
      DO 20 I=1,N
      IF (IHOSP(I,4+KX) .NE. IX) GO TO 20
      IF (X(I)-X(M)) 20,16,17
   16 IF (Y(I) .EQ. Y(M)) GO TO 20
      IF (Y(I) .GT. Y(M)) SLOPE=9999
      IF (Y(I) .LT. Y(M)) SLOPE=-9999
      GO TO 18
   17 SLOPE=(Y(I)-Y(M))/(X(I)-X(M))
   18 IF (SLOPE .LE. BEST) GO TO 20
      BEST=SLOPE
      MNEXT=I
   20 CONTINUE
      IHOSP(MNEXT,3)=IC+2
      CALL PFIND(IC,M,MNEXT,0,KX,IX,1)
      IHULL(IC,1)=M
      IHULL(IC,2)=MNEXT
      M=MNEXT
      IF (M .NE. MEND) GO TO 11
   21 BEST=-10000
      DO 30 I=1,N
      IF (IHOSP(I,4+KX) .NE. IX) GO TO 30
      IF (X(I)-X(M)) 24,23,30
   23 IF (Y(I) .EQ. Y(M)) GO TO 30
      IF (Y(I) .LT. Y(M)) SLOPE=9999
      IF (Y(I) .GT. Y(M)) SLOPE=-9999
      GO TO 25
   24 SLOPE=(Y(I)-Y(M))/(X(I)-X(M))
   25 IF (SLOPE .LE. BEST) GO TO 30
      BEST=SLOPE
      MNEXT=I
```

```
 30 CONTINUE
    IHOSP(MNEXT,3)=IC+2
    CALL PFIND(IC,M,MNEXT,0,KX,IX,1)
    IHULL(IC,1)=M
    IHULL(IC,2)=MNEXT
    M=MNEXT
    IF (M .NE. MSTAR) GO TO 21
    RETURN
    END
    SUBROUTINE PFIND(NSIZE,K1,K2,KPOINT,KX,IX,IENT)
    COMMON/A/IHOSP(150,5)/D/CANDID(600,4),IADRES(600)
    COMMON /E/X(150),Y(150),ALFA(150),IBLAD(150)
    COMMON/BKSQ/NHOSP,NBANK
    N=NHOSP
    BESS=100000
    IF(IENT.EQ.4) GO TO 100
 19 DO 20 I=1,N
    IF (IHOSP(I,4+KX) .NE. IX) GO TO 20
    IF (IHOSP(I,3) .GE. 1) GO TO 20
    D1=DMAT(I,K1)
    D2=DMAT(I,K2)
    D12=DMAT(K1,K2)
    DIF=D1+D2-D12
    ANG=(D1+D2)/D12
    DIS=DIF*ANG
    IF (DIS .GT. BESS) GO TO 20
    BESS=DIS
    IBEST=I
 20 CONTINUE
    GO TO 25
100 CONTINUE
    YMID=(Y(K1)+Y(K2))/2.0
    XMID=(X(K1)+X(K2))/2.0
    DO 10 I=1,N
    IF (IHOSP(I,3) .GE. 1) GO TO 10
    DMID=SQRT((XMID-X(I))**2+(YMID-Y(I))**2)
    D1=DMAT(I,K1)
    D2=DMAT(I,K2)
    D12=DMAT(K1,K2)
    DIS=AMIN1(D1,D2,DMID)
    COSI=(D1**2 + D2**2 - D12**2)/(2.0*D1*D2)
    DIS=DIS+COSI
    IF (DIS .GT. BESS) GO TO 10
    BESS=DIS
    IBEST=I
 10 CONTINUE
 25 NSIZE=NSIZE+1
    CANDID(NSIZE,1)=BESS
    CANDID(NSIZE,2)=IBEST
    CANDID(NSIZE,3)=K1
    CANDID(NSIZE,4)=K2
    CALL ADRES(NSIZE,KPOINT)
    RETURN
    END
```

```
      SUBROUTINE ADRES(NSIZE,KPOINT)
      COMMON/A/IHOSP(150,5)/D/CANDID(600,4),IADRES(600)
      IF (NSIZE .EQ. 1) GO TO 10
      ISTAR=KPOINT+1
      IEND=NSIZE-1
      DO 1 I=ISTAR,IEND
      JPOINT=IADRES(I)
      IF (CANDID(JPOINT,1) .GT. CANDID(NSIZE,1)) GO TO 8
    1 CONTINUE
   10 IADRES(NSIZE)=NSIZE
      RETURN
    8 JEND=NSIZE-I
      DO 9 J=1,JEND
    9 IADRES(NSIZE+1-J)=IADRES(NSIZE-J)
      IADRES(I)=NSIZE
      RETURN
      END
      SUBROUTINE BOUND1(IC,COST,KX,IX)
      COMMON/A/IHOSP(150,5)/D/CANDID(600,4),IADRES(600)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
      DIMENSION COST(9)
      N=NHOSP
      KPOINT=0
      NSIZE=IC
      ICONT=IC
      DO 1 I=1,300
    1 IHULL(I,3)=0
      MM=NUM(IX)-IC
      IF (MM .EQ. 0) GO TO 31
      DO 30 JJ=1,MM
   11 KPOINT=KPOINT+1
      ISTAR=IADRES(KPOINT)
      IBAR=CANDID(ISTAR,2)
      K1=CANDID(ISTAR,3)
      K2=CANDID(ISTAR,4)
      IF (IHOSP(IBAR,3) .EQ. 0) GO TO 25
      CALL PFIND(NSIZE,K1,K2,KPOINT,KX,IX,1)
      GO TO 11
   25 DO 26 I=1,IC
      IF (IHULL(I,1) .EQ. K1) IHULL(I,3)=1
   26 CONTINUE
      ICONT=ICONT+1
      IHOSP(IBAR,3)=ICONT
      IC =IC+1
      IHULL(IC,1)=K1
      IHULL(IC,2)=IBAR
      CALL PFIND(NSIZE,K1,IBAR,KPOINT,KX,IX,1)
      IC=IC+1
      IHULL(IC,1)=IBAR
      IHULL(IC,2)=K2
      CALL PFIND(NSIZE,IBAR,K2,KPOINT,KX,IX,1)
   30 CONTINUE
   31 COST(IX)=0
```

```
      DO 40 I=1,IC
      IF (IHULL(I,3) .EQ. 1) GO TO 40
      ISTAR=IHULL(I,1)
      IHOSP(ISTAR,2)=IHULL(I,2)
      IEND=IHULL(I,2)
      IHOSP(IEND,1)=IHULL(I,1)
      COST(IX)=COST(IX)+DMAT(ISTAR,IEND)
   40 CONTINUE
      DO 41 II=1,N
      IF (IHOSP(II,KX+4) .EQ. IX) GO TO 43
   41 CONTINUE
      STOP
   43 IPREV=II
      INEXT=IHOSP(II,2)
      IEND=NUM(IX)
      DO 45 I=1,IEND
      IHULL(I,1)=IPREV
      IHULL(I,2)=INEXT
      IPREV=INEXT
      INEXT=IHOSP(INEXT,2)
   45 CONTINUE
      RETURN
      END
      SUBROUTINE REFIN1(COST,KX,IX)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/A/IHOSP(150,5)
      DIMENSION COST(9)
      N=NHOSP
      DO 11 JJ=1,2
      KK=3-JJ
      DO 11 I=1,N
      IF (IHOSP(I,4+KX) .NE. IX) GO TO 12
      I1=I
      J1=IHOSP(I,1)
      I2=IHOSP(I,1)
      DO 6 IN=1,KK
    6 I2=IHOSP(I2,2)
      J2=IHOSP(I2,2)
      DIF1=DMAT(I1,J1) + DMAT(I2,J2) - DMAT(J1,J2)
      ISTAR=J2
      IEND=NUM(IX)-KK-1
      DO 10 J=1,IEND
      K1=ISTAR
      K2=IHOSP(ISTAR,2)
      DIF2=DMAT(I2,K1)+DMAT(I1,K2)-DMAT(K1,K2)
      IF (DIF1 .GT. DIF2) GO TO 7
      ISTAR=IHOSP(ISTAR,2)
   10 CONTINUE
      GO TO 12
    7 CALL CHANGE(I1,I2,J1,J2,K1,K2)
   12 CONTINUE
   11 CONTINUE
      COST(IX)=0
```

```
      DO 13 J=1,N
      IF (IHOSP(J,4+KX) .EQ. IX) GO TO 14
   13 CONTINUE
      STOP
   14 IPREV=J
      INEXT=IHOSP(J,2)
      IEND=NUM(IX)
      DO 15 I=1,IEND
      COST(IX)=COST(IX)+DMAT(IPREV,INEXT)
      IHULL(I,1)=IPREV
      IHULL(I,2)=INEXT
      IPREV=INEXT
      INEXT=IHOSP(INEXT,2)
   15 CONTINUE
      RETURN
      END
      SUBROUTINE CHANGE(I1,I2,J1,J2,K1,K2)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON /A/ IHOSP(150,5)
      N=NHOSP
      IHOSP(J1,2)=J2
      IHOSP(J2,1)=J1
      IHOSP(K1,2)=I2
      IHOSP(K2,1)=I1
      IHOSP(I1,1)=K2
      INEXT=I2
      IPREV=K1
   10 IHOSP(INEXT,2)=IHOSP(INEXT,1)
      IHOSP(INEXT,1)=IPREV
      IF (INEXT .EQ. I1) GO TO 15
      IPREV=INEXT
      INEXT=IHOSP(INEXT,2)
      GO TO 10
   15 CONTINUE
      RETURN
      END
      SUBROUTINE IMPROV(EXINDP)
      INTEGER EXINDP
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/F/IALTER(5,5,10),NALTER(5,5),IIHOSP(150,3),IDUM(150)
      COMMON/A/IHOSP(150,5)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)/B/IHULLA(150,2)
  110 FORMAT(1H1//5X,'LIST OF POSSIBLE EXCHANGES'/)
  120 FORMAT(///5X,'HOSPITALS IN GROUP',I3,' TO BE TRIED IN GROUP'
     1,I3/)
  121 FORMAT(2X,10I5)
      N=NHOSP
      DO 1 I=1,NBANK
      DO 1 J=1,NBANK
      NALTER(I,J)=0
      DO 1 K=1,10
      IALTER(I,J,K)=0
    1 CONTINUE
      DO 10 I=1,N
```

```
      IIHOSP(I,3)=-1
      IF (IHOSP(I,4) .EQ. IHOSP(I,5)) GO TO 10
      II=IHOSP(I,4)
      JJ=IHOSP(I,5)
      IBI=IBANK(II)
      IBJ= IBANK(JJ)
      DI=DMAT(IBI,I)-DMAT(IBJ,I)
      NALTER(II,JJ)=NALTER(II,JJ)+1
      NFIND=NALTER(II,JJ)
      IF (NALTER(II,JJ) .GT. 10) NALTER(II,JJ)=10
      NM=NALTER(II,JJ)
      JEND=NM-1
      IF (NM .EQ. 1) GO TO 7
       DO 6 J=1,JEND
      JX=IALTER(II,JJ,J)
      DJ=DMAT(IBI,JX)-DMAT(IBJ,JX)
      IF(DI.LE.DJ) GO TO 8
    6 CONTINUE
      IF (NFIND.LT. 11) IALTER(II,JJ,NM)=I
      JX=IALTER(II,JJ,NM)
      DJ=DMAT(IBI,JX)-DMAT(IBJ,JX)
      IF(DI.LE.DJ) IALTER(II,JJ,NM)=I
      GO TO 10
    8 JEND=NM-J
      DO 9 J=1,JEND
      IALTER(II,JJ,NM+1-J)=IALTER(II,JJ,NM-J)
    9 CONTINUE
    7 IALTER(II,JJ,NM-JEND)=I
   10 CONTINUE
      WRITE(6,110)
      DO 12 I=1,NBANK
      DO 11 J=1,NBANK
      IF (NALTER(I,J) .EQ. 0) GO TO 11
      WRITE(6,120) J,I
      NM=NALTER(I,J)
      WRITE(6,121) (IALTER(I,J,K),K=1,NM)
   11 CONTINUE
   12 CONTINUE
      DO 15 I=1,N
      IHOSP(I,4)=IHOSP(I,5)
      IIHOSP(I,1)=IHOSP(I,1)
      IIHOSP(I,2)=IHOSP(I,2)
      IHULLA(I,1)=IHULLA(I,2)
   15 CONTINUE
      IF (EXINDP.EQ.1) CALL ALTER1(3.0)
      IF (EXINDP.EQ.2) CALL ALTER2(3.0)
      CALL PLOTNG(-1,NBANK,NBANK)
      CALL PRINTS(-1,1,1,1)
      RETURN
      END
      SUBROUTINE ALTER1(Z)
      LOGICAL INDIC
      COMMON/F/IALTER(5,5,10),NALTER(5,5),IIHOSP(150,3),IDUM(150)
      COMMON/BKSQ/NHOSP,NBANK
```

```
      DIMENSION COST(9),LIST(5)
      N=NHOSP
      IEND=NBANK-1
      DO 30 I=1,IEND
      JSTAR=I+1
      DO 29 J=JSTAR,NBANK
      ITER=MAX0(NALTER(I,J),NALTER(J,I))
      IF (ITER .EQ. 0) GO TO 29
      DO 25 K1=1,ITER
      IF (K1 .GT. NALTER(I,J)) GO TO 20
      LIST(1)=IALTER(I,J,K1)
      CALL TEST(I,J,COST,LIST,1,1,INDIC)
      IF (INDIC) CALL UPDATE(I,J,COST,LIST,1,1)
C*** THE MODIFIED PLOTNG CALLS ONLY PLOT THE RESULTS
C*** THE INTERMEDIATE WILL NOT BE PLOTTED
C**   IF (INDIC) CALL PLOTNG(-1,NBANK,NBANK)
   20 IF (K1 .GT. NALTER(J,I)) GO TO 25
      LIST(1)=IALTER(J,I,K1)
      CALL TEST(J,I,COST,LIST,1,0,INDIC)
      IF (INDIC) CALL UPDATE(J,I,COST,LIST,1,0)
C**   IF (INDIC) CALL PLOTNG(-1,NBANK,NBANK)
   25 CONTINUE
   29 CONTINUE
   30 CONTINUE
      RETURN
      END
      SUBROUTINE ALTER2(Z)
      LOGICAL INDIC
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/F/IALTER(5,5,10),NALTER(5,5),IIHOSP(150,3),IDUM(150)
      DIMENSION COST(9),LIST(5)
      N=NHOSP
      IEND=NBANK-1
      DO 30 I=1,IEND
      JSTAR=I+1
      DO 29 J=JSTAR,NBANK
      ITER=MAX0(NALTER(I,J),NALTER(J,I))
      IF (ITER .EQ. 0) GO TO 29
      DO 25 K1=1,ITER
      IZ=I
      JZ=J
      KX=1
   15 IF (K1 .GT. NALTER(IZ,JZ)) GO TO 20
      J1=IALTER(IZ,JZ,K1)
      IF (IIHOSP(J1,3) .EQ. 0) GO TO 20
      NBR=1
      LIST(1)=J1
      CALL TEST(IZ,JZ,COST,LIST,NBR,KX,INDIC)
      IF (INDIC) GO TO 19
      J2=IIHOSP(J1,1)
      IF (IIHOSP(J1,3) .EQ. J2) GO TO 16
      IIHOSP(J1,3)=J2
      LIST(2)=J2
      NBR=2
```

```
      CALL TEST(IZ,JZ,COST,LIST,NBR,KX,INDIC)
      IF (INDIC) GO TO 19
   16 J3=IIHOSP(J1,2)
      IF (IIHOSP(J3,3) .EQ. J1) GO TO 17
      IIHOSP(J3,3)=J1
      NBR=2
      LIST(2)=J3
      CALL TEST(IZ,JZ,COST,LIST,NBR,KX,INDIC)
      IF (INDIC) GO TO 19
   17 LIST(2)=J2
      LIST(3)=J3
      NBR=3
      CALL TEST(IZ,JZ,COST,LIST,NBR,KX,INDIC)
      IF (INDIC) GO TO 19
      GO TO 20
   19 CALL UPDATE(IZ,JZ,COST,LIST,NBR,KX)
C*** NO INTERMEDIATE PLOTTING
C**   CALL PLOTNG(-1,NBANK,NBANK)
   20 IF (IZ .EQ. J) GO TO 25
      IZ=J
      JZ=I
      KX=0
      GO TO 15
   25 CONTINUE
   29 CONTINUE
   30 CONTINUE
      RETURN
      END
      SUBROUTINE TEST(I,J,COST,LIST,NBR,KX,INDIC)
      COMMON/BKSQ/NHOSP,NBANK
      LOGICAL INDIC,BOUND
      COMMON/BDD/BOUND,LVOL(5),MVOL(5)
      COMMON/F/IALTER(5,5,10),NALTER(5,5),IIHOSP(150,3),IDUM(150)
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
      COMMON /E/X(150),Y(150),ALFA(150),IBLAD(150)/A/IHOSP(150,5)
      COMMON/CO/JD
      DIMENSION COST(9),LIST(5)
  100 FORMAT(1H1/10X,'TEST DATA'//)
  110 FORMAT(///2X,'ORIGINAL ROUTING COST FOR GROUP',I3,'=',F14.2/
     12X,'REVISED ROUTING COST FOR GROUP',I3,'=',F9.3//)
  115 FORMAT(///2X,'ORIGINAL VOLUME FOR GROUP',I3,'IS',I8/2X,
     &'REVISED VOLUME FOR GROUP',I3,'IS',I8//)
  120 FORMAT(/2X,'MARGINAL DIFFERENCE IN EMERGENCY COST=',F9.3///)
  130 FORMAT(2X,'MARGINAL DIFFERENCE IN SYSTEM COST=',F14.2///)
  140 FORMAT(2X,'TOTAL MARGINAL DIFFERENCE=',F14.2///)
  153 FORMAT(2X,10I5)
  160 FORMAT(//5X,'HOSPITALS ASSIGNED TO GROUP',I3,
     1'  FROM GROUP',I3,' >',5I5//)
  200 FORMAT(//2X,'REVISED ROUTING FOR GROUP',I3/)
       N=NHOSP
      EMDIF=0
      SYDIF=0.
      INDIC=.FALSE.
```

```
      NUM(I)=NUM(I)+NBR
      NUM(J)=NUM(J)-NBR
      IVOL1=NBLAD(I)
      IVOL2=NBLAD(J)
      DO 1 II=1,NBR
      JJ=LIST(II)
      IVOL1=IVOL1+IBLAD(JJ)
      IVOL2=IVOL2-IBLAD(JJ)
    1 IHOSP(JJ,4+KX)=I
C**   IF(.NOT.BOUND)GO TO 9
C**   IF(IVOL1.GT.MVOL(I)) GO TO 14
C**   IF(IVOL2.LT.LVOL(J))GO TO 14
C** 9 CONTINUE
      CALL CHULL1(IC,KX,I)
      CALL BOUND1(IC,COST,KX,I)
      IF (NUM(I) .GE. 5) CALL REFIN1(COST,KX,I)
      NI=NUM(I)
      DO 5 II=1,NI
    5 IDUM(II)=IHULL(II,1)
      CALL CHULL1(IC,KX,J)
      CALL BOUND1(IC,COST,KX,J)
      IF (NUM(J) .GE. 5) CALL REFIN1(COST,KX,J)
      NJ=NUM(J)
      TSDIF=COST(I)+COST(J)-TSCOST(I)-TSCOST(J)
      SYDIF=CBC(JD,IVOL1)-CBC(JD,NBLAD(I))
      SYDIF=SYDIF+CBC(JD,IVOL2)-CBC(JD,NBLAD(J))
      IF (.NOT.BOUND)GO TO 99
      ADIF=PNALTY(IVOL1,I)-PNALTY(NBLAD(I),I)+PNALTY(IVOL2,J)-
     &PNALTY(NBLAD(J),J)
   99 CONTINUE
      IB=IBANK(I)
      JB=IBANK(J)
      DO 10 II=1,NBR
      I1=LIST(II)
      EMDIF=EMDIF+ALFA(I1)*(DMAT(IB,I1)-DMAT(JB,I1))
   10 CONTINUE
      DIF=SYDIF+TSDIF*260*.03/5.1+EMDIF*20.*.03*.25
      TADIF=DIF+ADIF
C ** THE QUANTITY TO BE TESTED: DIF   HAS BEEN CHANGED TO TADIF **
      IF (TADIF .LE. 0.) INDIC=.TRUE.
      WRITE(6,100)
      WRITE(6,160) I,J,(LIST(II),II=1,NBR)
      WRITE(6,200) I
      WRITE(6,153) (IDUM(II),II=1,NI)
      WRITE(6,200) J
      WRITE(6,153) (IHULL(II,1),II=1,NJ)
      WRITE(6,110) I,TSCOST(I),I,COST(I)
      WRITE(6,110) J,TSCOST(J),J,COST(J)
      WRITE(6,115)I,NBLAD(I),I,IVOL1
      WRITE(6,115)J,NBLAD(J),J,IVOL2
      WRITE(6,120) EMDIF
      WRITE(6,130) SYDIF
      WRITE(6,140)DIF
      IF(.NOT.BOUND)GO TO 14
```

```
      WRITE(6,310)ADIF
      WRITE(6,320)TADIF
  310 FORMAT(2X,'MARGINAL DIFFERENCE OF ARTIFITIAL PENALITIES=',E14.2)
  320 FORMAT(2X,'TOTAL COMBINED MARGINAL DIFFERENCE=',E14.2)
   14 CONTINUE
      IF (INDIC) RETURN
      NUM(I)=NUM(I)-NBR
      NUM(J)=NUM(J)+NBR
      DO 15 II=1,NBR
      JJ=LIST(II)
   15 IHOSP(JJ,4+KX)=J
      RETURN
      END
      SUBROUTINE UPDATE(I,J,COST,LIST,NBR,KX)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/F/IALTER(5,5,10),NALTER(5,5),IIHOSP(150,3),IDUM(150)
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
      COMMON/B/IHULLA(150,2)/E/X(150),Y(150),ALFA(150),IBLAD(150)
      COMMON/A/IHOSP(150,5)
      DIMENSION COST(9),LIST(5)
      N=NHOSP
      ISTARX=0
      ISTAR=0
      DO 40 II=1,NBANK
      IEND=NUM(II)
      IF (II .EQ .I) GO TO 20
      IF (II .EQ. J) GO TO 30
      DO 15 JJ=1,IEND
      IHULLA(ISTAR+JJ,1)=IHULLA(ISTARX+JJ,1)
   15 CONTINUE
      ISTARX=ISTARX+NUM(II)
      GO TO 39
   20 ISTARX=ISTARX+NUM(II)-NBR
      GO TO 39
   30 DO 35 JJ=1,IEND
      IHULLA(ISTAR+JJ,1)=IHULL(JJ,1)
   35 CONTINUE
      ISTARX=ISTARX+NUM(II)+NBR
   39 ISTAR=ISTAR+NUM(II)
   40 CONTINUE
      ISTAR=0
      DO 45 II=1,NBANK
      IF (II .NE. I) GO TO 44
      IEND=NUM(II)
      DO 41 JJ=1,IEND
      IHULLA(ISTAR+JJ,1)=IDUM(JJ)
   41 CONTINUE
      GO TO 46
   44 ISTAR=ISTAR+NUM(II)
   45 CONTINUE
   46 TSCOST(I)=COST(I)
      TSCOST(J)=COST(J)
      DO 50 II=1,NBR
```

```
      JJ=LIST(II)
      IIHOSP(JJ,3)=0
      EMCOST(I)=EMCOST(I)+ALFA(JJ)*DMAT(I,JJ)
      EMCOST(J)=EMCOST(J)-ALFA(JJ)*DMAT(J,JJ)
      NBLAD(I)=NBLAD(I)+IBLAD(JJ)
      NBLAD(J)=NBLAD(J)-IBLAD(JJ)
      IHOSP(JJ,5-KX)=I
   50 CONTINUE
      DO 51 II=1,N
      IIHOSP(II,1)=IHOSP(II,1)
      IIHOSP(II,2)=IHOSP(II,2)
   51 CONTINUE
      RETURN
      END
      SUBROUTINE PRINTS(KX,NX,ISTOP,IUNIT)
      COMMON/C/IBANK(10),NUM(50),IHULL(300,3)/B/IHULLA(150,2)
      COMMON/BKSQ/NHOSP,NBANK
      COMMON/K/TSCOST(9),EMCOST(5),NBLAD(5)
      COMMON/F/IBLDTR(50),IBTR(5),IDUM(820)
      COMMON/CO/JD
  110 FORMAT(//5X,'OPTIMAL ALLOCATION AND ROUTING '/6X,'ALLOCATION BASED
     1 ON EMERGENCY COSTS ONLY'//)
  111 FORMAT (///3X,'BANK ',I2,' , IDENTIFICATION-HOSPITAL',I4/3X,'ROUTI
     1NG'/)
C ** THE FOLLOWING FORMAT IS NOT REFERENCED **
C*109 FORMAT(2X,5F10.3)
  112 FORMAT(10X,10I5)
  113 FORMAT(///3X,'EMERGENCY COST',9X,F9.2//3X,'ROUTINE DELIVERY COST '
     1,F10.2//)
  120 FORMAT(//5X,'OPTIMAL ALLOCATION AND ROUTING '/6X,'ALLOCATION BASED
     1 ON ROUTING COSTS ONLY'//)
  125 FORMAT(//5X,'NUMBER OF HOSPITALS IN THE SYSTEM ',I7/)
  126 FORMAT(/5X,'AMOUNT OF BLOOD USED IN THE SYSTEM',2X,I9/)
  127 FORMAT(/5X,'SYSTEM COST FOR CBC',2X,F11.2/)
  130 FORMAT(// ,5X,'OPTIMAL ALLOCATION AND ROUTING '/6X,'ALLOCATION BAS
     1ED ON ROUTING,EMERGENCY AND SYSTEM COSTS'//)
  210 FORMAT(1H1/40X,'SINGLE VEHICLE SOLUTION'/)
  220 FORMAT(1H1/40X,'MULTI VEHICLE SOLUTION'//5X,'CONSTRAINTS>',4X,
     1'MAXIMUM NUMBER OF STOPS-',I4/21X,'MAXIMUM NUMBER OF UNITS-',I4/)
  230 FORMAT(/6X,'TRUCK NO',I3,6X,'NUMBER OF STOPS',I3,6X,'NUMBER OF UNI
     1TS',I5/)
      N=NHOSP
      KK=KX
      IF (KX .LT. 0) KK=0
      IF (NX .EQ. 1) WRITE(6,210)
      IF (NX .EQ. 2) WRITE(6,220) ISTOP,IUNIT
      IF (KX .EQ. 0) WRITE(6,110)
      IF (KX .EQ. 1) WRITE(6,120)
      IF (KX .EQ. -1) WRITE(6,130)
      ECOST=0
      RCOST=0
      SCOST=0
      ISTAR=0
      JSTAR=0
```

```
      IPREV=0
      DO 6 II=1,NBANK
      WRITE(6,111) II,IBANK(II)
      IF (NX .EQ. 1) GO TO 4
      NTR=IBTR(II)
      DO 3 J=1,NTR
      WRITE(6,230) J, NUM(JSTAR+J),IBLDTR(JSTAR+J)
      JEND=NUM(JSTAR+J)
      WRITE(6,112) (IHULLA(ISTAR+I,KK+1),I=1,JEND)
      ISTAR=ISTAR+JEND
    3 CONTINUE
      MEMBER=ISTAR-IPREV-NTR+1
      IPREV=ISTAR
      WRITE(6,125) MEMBER
      JSTAR=JSTAR+NTR
      GO TO 5
    4 MEMBER=NUM(II)
      WRITE(6,112) (IHULLA(ISTAR+I,1+KK),I=1,MEMBER)
      WRITE(6,125) NUM(II)
      ISTAR=ISTAR+NUM(II)
    5 WRITE(6,126) NBLAD(II)
      ECOST=ECOST+EMCOST(II)
      RCOST=RCOST+TSCOST(II)
      CBCOST=CBC(JD,NBLAD(II))
C     SCOST=SCOST+CBCOST
      WRITE(6,127)CBCOST
    6 CONTINUE
C     TOT=SCOST+(RCOST*.03/5.1)*260+(ECOST*.03*.25)*20
      WRITE(6,113) ECOST,RCOST
      ITOT=0
      CCC=0.
      COREG=0.
C     COST AT THE COMMUNITY CENTERS
      DO 7 I=1,NBANK
      ITOT=ITOT+NBLAD(I)
      CCC=CCC+CBC(JD,NBLAD(I))
    7 CONTINUE
      IF (JD.EQ.5) GO TO 11
C     COST AT THE REGIONAL CENTER
      COREG=REG(JD,ITOT)
   11 WRITE(6,8) COREG
    8 FORMAT (//3X,'TOTAL COST AT THE REGIONAL CENTER',9X,F11.2)
      WRITE(6,9)CCC
    9 FORMAT (//3X,'TOTAL SYSTEM COST AT THE COMMUNITY CENTERS',9X,
     * F11.2)
      TSC=COREG+CCC
      WRITE(6,10) TSC
   10 FORMAT (//3X,'TOTAL SYSTEM COST',9X,F11.2//)
      RETURN
      END
      SUBROUTINE PLOTNG(KX,NBANK,NTRUCK)
      RETURN
      END
      FUNCTION DMAT(I,J)
```

```
      COMMON/CMM/DM(11325)
      COMMON/BKSQ/NHOSP,NBANK
      MN=I
      MX=J
      IF(I.LT.J) GO TO 1
      MN=J
      MX=I
    1 MN1=MN-1
      IS=MN*MN1/2
      K=MN1*NHOSP + MX - IS
      DMAT=DM(K)
      RETURN
      END
C ******* THE FOLLOWING IS A DUMMY FUNCTION ********
      FUNCTION SECOND(X)
      SECOND=0.
      RETURN
      END
      FUNCTION REG(I,IVOL)
      DIMENSION COEFF(4,2)
      DATA COEFF/2.81,2.2,1.52,0.45,395.,340.,340.,0./
      XVOL=IVOL/1000.
      REG=IVOL*(COEFF(I,1)+COEFF(I,2)/XVOL/XVOL)
      RETURN
      END
      FUNCTION CBC(I,IVOL)
      DIMENSION COEFF(5,2)
      DATA COEFF/.44,1.19,1.31,2.9,3.29,98.8,178.8,187.,490.4,539./
      XVOL=IVOL/1000.
      CBC=IVOL*(COEFF(I,1)+COEFF(I,2)/XVOL/XVOL)
      RETURN
      END
      FUNCTION PNALTY(IVOL,K)
      COMMON/BDD/BOUND,LVOL(5),MVOL(5)
      PNALTY=0.
      CM1=500.
      CM2=10.
      IDIF=IVOL-MVOL(K)
      JDIF=LVOL(K)-IVOL
      IF(IDIF.LE.0)GO TO 10
      PNALTY=((IDIF/1000)**2)*CM2+IDIF*CM1
   10 IF(JDIF.LE.0)GO TO 20
      PNALTY=((JDIF/1000)**2)*CM2+JDIF*CM1
   20 RETURN
      END
//SYSIN DD *
 3    1    2    3  109
 3 TT 1000 1000
     50000    130000     50000    150000     50000    110000
2
```

```
 RU BTP

NUMBER OF BANKS= 3, BANK ID#=    1    2    3
PROGRAM OPTION # 1 IS USED


ALLOCATION PARAMETERS:     ALLOC1   ALLOC2   IMPRV1   IMPRV2    BOUND
                             T        F        F        F         T
BOUNDS ON THE VOLUMES FOR EACH BANK:

      BANK     LOWER BOUND  UPPER BOUND
         1           50000       130000
         2           50000       150000
         3           50000       110000


DISPATCHING PARAMETERS:         VDP    MSTOP     VCAP
                                 T     1000     1000


 SYSTEM COST OPTION:  2


NUMBER OF HOSPITALS= 117
         1     2     3     3     1     1     3     1     3     1
         2     3     3     2     3     1     2     3     1     1
         3     3     3     1     2     2     1     1     3     3
         3     1     1     3     3     2     2     1     3     2
         2     2     3     2     1     3     3     3     1     3
         3     2     3     3     3     1     3     3     3     3
         2     3     2     2     1     3     1     2     3     3
         1     2     2     2     3     3     2     3     1     2
         3     2     1     3     3     1     1     2     2     2
         3     3     1     3     3     3     3     3     2     3
         3     3     3     3     1     1     1     1     3     3
         3     1     1     1     1     3     3
```

SINGLE VEHICLE SOLUTION

OPTIMAL ALLOCATION AND ROUTING
ALLOCATION BASED ON EMERGENCY COSTS ONLY

BANK 1 , IDENTIFICATION-HOSPITAL 1
ROUTING

1	93	33	67	16	38	65	49	87	86
71	105	107	113	108	106	112	115	114	27
20	79	10	56	24	8	45	28	5	83
19	32	6							

NUMBER OF HOSPITALS IN THE SYSTEM 33

AMOUNT OF BLOOD USED IN THE SYSTEM 100073

SYSTEM COST FOR CBC 120873.57

BANK 2 , IDENTIFICATION-HOSPITAL 2
ROUTING

2	64	14	90	99	37	41	40	73	72
68	80	25	74	82	89	61	63	17	42
36	11	26	77	88	44	52			

NUMBER OF HOSPITALS IN THE SYSTEM 27

AMOUNT OF BLOOD USED IN THE SYSTEM 89967

SYSTEM COST FOR CBC 109048.13

BANK 3 , IDENTIFICATION-HOSPITAL 3
ROUTING

3	18	81	39	91	22	94	53	76	13
54	95	48	92	46	31	97	57	51	34
21	85	116	117	102	111	23	12	100	103
110	104	109	101	59	4	35	47	66	55
69	7	58	15	70	75	98	62	43	30
9	29	96	60	78	84	50			

NUMBER OF HOSPITALS IN THE SYSTEM 57

AMOUNT OF BLOOD USED IN THE SYSTEM 177816

SYSTEM COST FOR CBC 212606.57

EMERGENCY COST 43894.99

ROUTINE DELIVERY COST 12120.34

TOTAL COST AT THE REGIONAL CENTER 810207.47

TOTAL SYSTEM COST AT THE COMMUNITY CENTERS 442528.27

TOTAL SYSTEM COST 1252735.70

27	115	114	45	28	56	79	20	112	5
106	108	83	113	24	10	105	19	8	107
71	38	65	6	32	16	67	86	87	49
33	93								
1	32								
44	52	63	36	11	17	88	42	89	26
61	80	77	74	25	82	68	64	99	90
14	40	72	37	73	41				
1	26								
50	78	84	55	69	101	7	47	60	35
66	96	58	15	59	29	70	4	75	109
104	110	97	57	98	51	30	31	103	62
100	12	43	46	9	92	111	102	23	34
48	85	21	94	53	117	76	116	13	54
95	22	18	91	39	81				
1	56								

MULTI VEHICLE SOLUTION

CONSTRAINTS> MAXIMUM NUMBER OF STOPS-1000
MAXIMUM NUMBER OF UNITS-1000

OPTIMAL ALLOCATION AND ROUTING
ALLOCATION BASED ON EMERGENCY COSTS ONLY

BANK 1 , IDENTIFICATION-HOSPITAL 1
ROUTING

TRUCK NO 1 NUMBER OF STOPS 33 NUMBER OF UNITS 335

1	93	33	67	16	38	65	49	87	86
71	105	107	113	108	106	112	115	114	27
20	79	10	56	24	8	45	28	5	83
19	32	6							

NUMBER OF HOSPITALS IN THE SYSTEM 33

AMOUNT OF BLOOD USED IN THE SYSTEM 100073

SYSTEM COST FOR CBC 120873.57

BANK 2 , IDENTIFICATION-HOSPITAL 2
ROUTING

TRUCK NO 1 NUMBER OF STOPS 27 NUMBER OF UNITS 317

2	64	14	90	99	37	41	40	73	72
68	80	25	74	82	89	61	63	17	42
36	11	26	77	88	44	52			

NUMBER OF HOSPITALS IN THE SYSTEM 27

AMOUNT OF BLOOD USED IN THE SYSTEM 89967

SYSTEM COST FOR CBC 109048.13

BANK 3 , IDENTIFICATION-HOSPITAL 3
ROUTING

TRUCK NO 1 NUMBER OF STOPS 57 NUMBER OF UNITS 598

3	18	81	39	91	22	94	53	76	13
54	95	48	92	46	31	97	57	51	34
21	85	116	117	102	111	23	12	100	103
110	104	109	101	59	4	35	47	66	55
69	7	58	15	70	75	98	62	43	30
9	29	96	60	78	84	50			

NUMBER OF HOSPITALS IN THE SYSTEM 57

AMOUNT OF BLOOD USED IN THE SYSTEM 177816

SYSTEM COST FOR CBC 212606.57

EMERGENCY COST 43894.99

ROUTINE DELIVERY COST 12120.34

TOTAL COST AT THE REGIONAL CENTER 810207.47

TOTAL SYSTEM COST AT THE COMMUNITY CENTERS 442528.27

TOTAL SYSTEM COST 1252735.70

THE MATHEMATICAL MODEL AND ALGORITHIMS FOR BTAP

The Blood Transportation-Allocation Problem (BTAP), can be stated as follows: "Locations and expected blood requirements of a set of N hospitals are given. Each hospital is to be assigned to a community blood center (CBC) which will periodically supply the hospital's blood requirements as well as supply its emergency blood demands at the time of the emergency. The blood shipments are to be made by special delivery vehicles which have given capacities and given limits on the number of deliveries they can make per day. The problem is to decide how many CBCs to set up, where to locate them, how to allocate the hospitals to the banks, and how to route the periodic supply operation, so that the total of transportation costs (periodic and emergency supply costs) and the system costs are a minimum."

The following notation will be used in formulating the BTAP.

i) "N" is the number of demand points.

ii) "M" is the number of supply points.

iii) "n" is the maximum number of supply vehicles available.

iv) $\mathcal{D} = \{H_1, \ldots, H_N\}$ is a set of N demand points.

v) $\mathcal{S} = \{H_{N+1}, \ldots, H_{N+M}\}$ is a set of M supply points.

vi) $\mathcal{V} = \mathcal{D} \cup \mathcal{S}$ is the set of all points involved in the problem.

vii) d_{ij} is the "distance" from H_i to H_j. It should be noted that although Euclidean distances among locations of hospitals and central banks are used in the solution procedure, one could obtain a matrix of accurate travel times between all pairs of hospitals and banks, and one could use this matrix or any other "distance measure" instead of the Euclidean distance matrix.

viii) C_k, k = 1,..., n is the capacity of supply vehicle k.

ix) Q_i, i = 1,..., N is the requirement of demand point i.

x) D_k, k = 1,..., n is the maximum distance supply vehicle k may travel.

xi) γ_i, i = 1,..., N is the expected number of emergency deliveries to hospital H_i per period. γ_i is the probability that the demand at H_i exceeds the supply at H_i given the optimal inventory level at H_i is used.

xii) $s(\ell, k)$ is the systems cost function of a region, where ℓ is the number of hospitals in that region, and k is the amount of blood used per year in that region. $s(\ell, k)$ is defined and finite for all feasible combinations of ℓ and k.

xiii) y_{ij}, i = 1,..., N; j = N+1,..., N+M is a zero-one variable such that y_{ij} is 1 if hospital H_i is assigned to central bank H_j and is 0 otherwise.

xiv) x_{ijk}, i = 1,..., N+M; j = 1,..., N+m; k = 1,..., n is a zero-one variable such that x_{ijk} is 1 if vehicle k goes from hospital H_i to H_j and is 0 otherwise.

viii) C_k, k = 1,..., n is the capacity of supply vehicle k.

ix) Q_i, i = 1,..., N is the requirement of demand point i.

x) D_k, k = 1,..., n is the maximum distance supply vehicle k may travel.

xi) γ_i, i = 1,..., N is the expected number of emergency deliveries to hospital H_i per period. γ_i is the probability that the demand at H_i exceeds the supply at H_i given the optimal inventory level at H_i is used.

xii) $s(\ell, k)$ is the systems cost function of a region, where ℓ is the number of hospitals in that region, and k is the amount of blood used per year in that region. $s(\ell, k)$ is defined and finite for all feasible combinations of ℓ and k.

xiii) y_{ij}, i = 1,..., N; j = N+1,..., N+M is a zero-one variable such that y_{ij} is 1 if hospital H_i is assigned to central bank H_j and is 0 otherwise.

xiv) x_{ijk}, i = 1,..., N+M; j = 1,..., N+m; k = 1,..., n is a zero-one variable such that x_{ijk} is 1 if vehicle k goes from hospital H_i to H_j and is 0 otherwise.

The BTAP is:

Problem 1

$$\min z^1(x, y) = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{n} x_{ijk} d_{ij} + \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} \gamma_i y_{ij} d_{ij}$$

$$+ \sum_{j=N+1}^{N+M} s\left(\sum_{i=1}^{N} y_{ij}, \sum_{i=1}^{N} (300Q_i)(y_{ij}) \right) \qquad (1)$$

subject to

$$\sum_{k=1}^{n} \sum_{j=1}^{N+M} x_{ijk} = 1 \qquad i = 1, \ldots, N \qquad (2)$$

$$\sum_{j=1}^{N+M} \sum_{i=1}^{N} Q_i x_{ijk} \leq C_k \qquad k = 1, \ldots, n \qquad (3)$$

$$\sum_{j=1}^{N+M} \sum_{i=1}^{N+M} d_{ij} x_{ijk} \leq D_k \qquad k = 1, \ldots, n \qquad (4)$$

$$\sum_{\{i:H_i \in S\}} \sum_{\{j:H_j \in \bar{S}\}} \sum_{k=1}^{n} x_{ijk} \geq 1 \quad \text{for all } (S, \bar{S}) \qquad (5)$$

where S is any proper subset of $\mathcal{N}$ containing $\mathcal{S}$ and $\bar{S}$ is the complement of S.

$$\sum_{j=1}^{N+M} x_{hjk} = \sum_{i=1}^{N+M} x_{ihk} \qquad k = 1, \ldots, n; \quad h = 1, \ldots, N+M \qquad (6)$$

$$\sum_{j=N+1}^{N+M} y_{ij} = 1 \qquad i = 1,\ldots, N \tag{7}$$

$$y_{ij} \geq \sum_{h=1}^{N+M} x_{ihk} + \sum_{h=1}^{N+M} x_{jhk} - 1 \qquad i = 1,\ldots, N;\ j = N+1,\ldots, N+M;\ k = 1,\ldots, n \tag{8}$$

$$x_{ijk} = 0, 1 \qquad i = 1,\ldots, N+M;\ j = 1,\ldots, N+M;\ k = 1,\ldots, n \tag{9}$$

(note that $x_{iik} = 0$)

$$y_{ij} = 0, 1 \qquad i = 1,\ldots, N;\ j = N+1,\ldots, N+M\ . \tag{10}$$

The explanation of these constraint sets are as follows. Constraints (2) require that every hospital receive a shipment from some vehicle; (3) are the vehicle capacity constraints; (4) are the maximum travel distance constraints (note, it is implicitly assumed that $Q_i \leq C_k$ for $i = 1,\ldots, N$ and $k = 1,\ldots, n$); (5) require that graph $\mathcal{L}$ corresponding to x is connected; (6) imply that a vehicle departs from a point h if and only if it enters there (conservation of flow); (7) requires that each hospital is assigned to a central bank; (8) contains the coupling constraints between variables $x = \{x_{ijk}\}$ and $y = \{y_{ij}\}$. It means that if there is vehicle k passing from both hospital i $\left(\sum_{h=1}^{N+M} x_{ihk} = 1\right)$ and from bank j $\left(\sum_{h=1}^{N+M} x_{jhk} = 1\right)$ then hospital i is assigned to bank j ($y_{ij} \geq 1 + 1 - 1 = 1$).

Problem 1 has some redundant constraints, namely constraint set (7). In order to show this, assume (7) was removed and a feasible solution was found such that

$$\sum_{j=N+1}^{N+M} y_{i^\circ j} = 0$$

then, by (8) we have

$$0 \geq \sum_{j=N+1}^{N+M} \sum_{h=1}^{N+M} x_{i^\circ hk} + \sum_{j=N+1}^{N+M} \sum_{h=1}^{N+M} x_{jhk} - \sum_{j=N+1}^{N+M} 1$$

$$= (M) \sum_{h=1}^{N+M} x_{i^\circ hk} + \sum_{j=N+1}^{N+M} \sum_{h=1}^{N+M} x_{jhk} - M \qquad k = 1, \ldots, n .$$

Through constraint set (2) we know that there exists a $k^\circ \in [1, 2, \ldots, n]$ such that

$$\sum_{h=1}^{N+M} x_{i^\circ hk^\circ} = 1 \qquad \text{(vehicle } k^\circ \text{ passes from } i^\circ\text{)}$$

and through constraint sets (5) and (6) we know that

$$\sum_{j=N+1}^{N+M} \sum_{h=1}^{N+M} x_{jhk^\circ} = 1 \qquad \text{(vehicle } k^\circ \text{ passes from a bank)}$$

which contradicts the above inequality. Therefore $\sum_{j=N+1}^{N+M} y_{ij} = 1$ when constraint set (8) is satisfied, and thus constraint set (7) is redundant. We have included (7) in Problem 1 for the reader's intuition.

In Problem 1 the variables $x = \{x_{ijk}\}$ correspond to the routing of the periodic delivery vehicles and the variables $y = \{y_{ij}\}$ correspond to the allocations of the hospitals to the blood banks. For a given $x = \{x_{ijk}\}$, $y = \{y_{ij}\}$ is uniquely determined, but the converse is not true; if we are given the allocations, a series of M vehicle dispatch problems have to be solved, in order to obtain the routings. Problem 1 has a finite feasible solution set and a nonempty optimal solution set. However, the underlying Multiple Vehicle Dispatch Problem (MVDP) makes it a complex integer programming problem; for N of any significant size ($N \geq 20$), the BTAP is too large to be solved by conventional mathematical programming techniques in a reasonable amount of time. In the following

section we will introduce and discuss a good heuristic solution procedure for the BTAP. First, let us thoroughly examine the subproblems of the BTAP that we will be using in the heuristic approaches.

In Problem 1, if γ_i, $i = 1,\ldots, N$ are small or emergency costs negligible (actual γ_i's range from .0002 to .06 when optimal ordering policies are followed, see Pierskalla and Yen [1978]) and the function $s(\ell, k)$ is essentially constant, then

$$z^2(x) = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{n} x_{ijk} d_{ij}$$

would be the dominating term in the objective function (1). Then we could just solve the MVDP,

Problem 2

$$\min \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{n} x_{ijk} d_{ij} \tag{11}$$

subject to

$$\sum_{k=1}^{n} \sum_{j=1}^{N+M} x_{ijk} = 1 \qquad i = 1,\ldots, N \tag{12}$$

$$\sum_{j=1}^{N+M} \sum_{i=1}^{N} Q_i x_{ijk} \le C_k \qquad k = 1,\ldots, n \tag{13}$$

$$\sum_{j=1}^{N+M} \sum_{i=1}^{N+M} d_{ij} x_{ijk} \le D_k \qquad k = 1,\ldots, n \tag{14}$$

$$\sum_{\{i:H_i \in S\}} \sum_{\{j:H_j \in \bar{S}\}} \sum_{k=1}^{n} x_{ijk} \ge 1 \qquad \text{for all } (S, \bar{S}) \tag{15}$$

where S is any proper subset of $\mathcal{N}$ containing $\mathcal{S}$,

$$\sum_{j=1}^{N+M} x_{hjk} = \sum_{i=1}^{N+M} x_{ihk} \qquad k = 1,\ldots, n;\ h = 1,\ldots, N+M \tag{16}$$

$$x_{ijk} = 0, 1 \qquad i = 1,\ldots, N+M;\ j = 1,\ldots, N+M;\ k = 1,\ldots, n \tag{17}$$

in order to obtain the optimal x^* for Problem 1. The optimal allocations, y^*, would then be uniquely determined by x^*.

On the other hand, if γ_i, $i = 1,\ldots, N$ are relatively large (which might happen under nonoptimal ordering policies) or system costs and periodic delivery costs are negligible, then

$$z^3(y) = \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} \gamma_i y_{ij} d_{ij}$$

would be the dominating term in the objective function. Then we could just solve the allocation problem,

Problem 3

$$\min \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} \gamma_i y_{ij} d_{ij} \tag{18}$$

subject to

$$\sum_{j=N+1}^{N+M} y_{ij} = 1 \qquad i = 1,\ldots, N \tag{19}$$

$$y_{ij} = 0, 1 \qquad i = 1,\ldots, N;\ j = N+1,\ldots, N+M \tag{20}$$

in order to get the optimal y° for Problem 1. Then, optimal routings, x°, would be obtained by solving a vehicle dispatch problem for each one of the M regions determined by y°.

Let x^* be an optimal solution of Problem 2. Let y^* be the allocations determined by x^*. Let y° be an optimal solution of Problem 3. Define $P_h(y)$ to be the set consisting of central bank $N+h$ and the hospitals which it serves. Let $P(y) = \{P_1(y), P_2(y),\ldots, P_M(y)\}$ such that $H_{N+h} \in P_h(y)$, $h = 1,\ldots, M$; $H_i \in P_h(y)$ if and only if $y_{i,N+h} = 1$, $i = 1,\ldots, N$; $h = 1,\ldots, M$. (It should be noted that $\bigcup_{j=1}^{M} P_j(y) = \mathcal{X}$ and $P_{j_1}(y) \cap P_{j_2}(y) = \emptyset$, $j_1 \neq j_2$; $j_1, j_2 = 1,\ldots, M$; for all y feasible in Problem 1.)

Let x° be the routings obtained by solving the following vehicle dispatch problem for h = 1,..., M (and renumbering the vehicles so that each corresponds to a different circuit).

Problem 4

$$\min \sum_{\{i:H_i \in P_h(y^\circ)\}} \sum_{\{j:H_j \in P_h(y^\circ)\}} \sum_{k=1}^{n} x_{ijk} d_{ij} \tag{21}$$

subject to

$$\sum_{k=1}^{n} \sum_{\{j:H_j \in P_h(y^\circ)\}} x_{ijk} = 1 \quad \text{for all i such that } H_i \in P_h(y^\circ) \tag{22}$$

$$\sum_{\{j:H_j \in P_h(y^\circ)\}} \sum_{\{i:H_i \in P_h(y^\circ)\}} Q_i x_{ijk} \leq C_k \qquad k = 1,\ldots, n \tag{23}$$

$$\sum_{\{j:H_j \in P_h(y^\circ)\}} \sum_{\{i:H_i \in P_h(y^\circ)\}} d_{ij} x_{ijk} \leq D_k \qquad k = 1,\ldots, n \tag{24}$$

$$\sum_{\{i:H_i \in S\}} \sum_{\{j:H_j \in \bar{S}\}} \sum_{k=1}^{n} x_{ijk} \geq 1 \quad \text{for all } (S, \bar{S}) \tag{25}$$

where S is any proper subset of $P_h(y^\circ)$

$$\sum_{\{j:H_j \in P_n(y^\circ)\}} x_{\ell jk} = \sum_{\{i:H_i \in P_n(y^\circ)\}} x_{i\ell k} \quad \text{for all } \ell \text{ such that} \tag{26}$$

$H_\ell \in P_h(y^\circ)$ and k = 1,..., n

$$x_{ijk} = 0, 1 \quad \{i:H_i \in P_h(y^\circ)\}, \{j:H_j \in P_h(y^\circ)\}, \quad k = 1,\ldots, n \tag{27}$$

$(x_{iik} = 0)$.

It directly follows from the above definitions that

$$z^2(x^*) \leq z^2(x^\circ)$$

$$z^3(y^\circ) \leq z^3(y^*)$$

and if the systems costs are essentially constant

$$z^2(x^*) + z^3(y^\circ)$$

would be a good lower bound on the optimal value of Problem 1.

Heuristic Solution Procedures for the BTAP

Let $EX(j_1, j_2)$ be the set of all hospitals such that

$H_i \in EX(j_1, j_2)$ if and only if $H_i \notin \mathcal{S}$ and $H_i \in P_{j_1}(y^*)$, $H_i \in P_{j_2}(y^\circ)$.

In other words $H_i \in EX(j_1, j_2)$ means that hospital H_i would have been allocated to bank H_{N+j_1} if only periodic delivery costs were minimized, and hospital H_i would have been allocated to bank H_{N+j_2}, if only emergency delivery costs were minimized.

The basic idea behind the solution procedures that will be discussed in this section is fairly simple: First, the feasible solutions (x^*, y^*) and (x°, y°) of Problem 1, discussed in the previous section, are obtained. Then, they are compared and for each pair (j_1, j_2), $j_1 \neq j_2$; $j_1, j_2 = 1,\ldots, M$, hospitals $H_i \in EX(j_1, j_2)$ are removed temporarily from $P_{j_1}(y^*)$ and inserted into $P_{j_2}(y^*)$. (Let us call this operation an exchange between sets $P_{j_1}(y^*)$ and $P_{j_2}(y^*)$). After each exchange, Problem 4 is solved for the two sets $P_{j_1}(y^*)$ and $P_{j_2}(y^*)$, under consideration, in order to obtain the corresponding components of the variable (x, y) of Problem 1. The components of (x, y) corresponding to sets $P_j(y^*)$, $j \neq j_1$, $j \neq j_2$, are left unchanged. The resulting feasible solution, to Problem 1, is compared with the better of (x^*, y^*) and (x°, y°). If there is a decrease in the objective function value, $z^1(x, y)$, the exchange is made permanent. Then another exchange is considered and so on.

Two algorithms have been developed, both based on the idea described above, and differing only in the way the exchanges implied by the sets "$EX(j_1, j_2)$, $j_1 \neq j_2$, $j_1, j_2 = 1,\ldots, M$" are ordered and executed. The first algorithm is reasonably fast, but the set of exchanges it tests

is not very large; it tests the elements in $\bigcup_{(j_1,j_2)} EX(j_1, j_2)$ only one by one and independent of each other. The second algorithm tests a larger set, besides testing the elements in $\bigcup_{(j_1,j_2)} EX(j_1, j_2)$, it also tests many different combinations of them. Hence, the second algorithm produces a better solution. Unfortunately, these latter tests are time consuming, and they slow down the algorithm considerably. Both of these algorithms were originally developed assuming no vehicle capacity constraints and no maximum number of stops per vehicle type constraints. In other words, constraint sets (3) and (4) of Problem 1 were assumed to be nonbinding (in which case, of course, the underlying MVDP reduces to the Multiple Traveling Salesman Problem (MTSP)). Then both algorithms were extended to provide solutions to the BTAP also when the constraints on the vehicles were binding. It should be noted that, in these algorithms heuristic procedures are employed to solve Problems 2 and 4. So, (x^*, y^*) obtained is not the optimal solution of Problem 2, but a near optimal solution. Similarly, x° is a combination of near optimal solutions obtained by solving Problem 4 for h = 1,..., M. Therefore, the inequality

$$z^2(x^*) \leq z^2(x) \quad \text{for all x feasible for Problem 2}$$

which is always true when x^* is optimal, might not always hold in our heuristic approaches, since there x^* is only "near optimal."

Algorithm 1:

The following algorithm is a solution procedure for the BTAP, when the constraints on the vehicles (constraint sets (3) and (4) of Problem 1)

are not binding. In this algorithm the sets $EX(j_1, j_2)$ are ordered such that for all $H_{i_1}, H_{i_1+1} \in EX(j_1, j_2)$

$$d_{i_1 j_1} - d_{i_1 j_2} \leq d_{i_1+1, j_1} - d_{i_1+1, j_2} .$$

In other words the marginal decrease in the emergency delivery cost if H_{i_1} were to be placed in $P_{j_2}(y^*)$ is greater than the decrease if H_{i_1+1} were to be placed in $P_{j_2}(y^*)$. Then, each set $EX(j_1, j_2)$, $j_1 \neq j_2$, $j_1, j_2 = 1, \ldots, M$ is considered one at a time. Starting with the first element, all elements of the set $EX(j_1, j_2)$, under consideration, are, one by one, temporarily removed from $P_{j_1}(y^*)$ and inserted into $P_{j_2}(y^*)$. Then two traveling salesman problems are solved for these two sets. The resulting feasible solution to Problem 1 is compared with (x^*, y^*) and (x°, y°); if it is better, the change is made permanent.

Assume the following parameters are given: the coordinates of N hospitals and M banks; the function $s(\ell, k)$; the daily blood usage, Q_i, in each hospital H_i; the period for the periodic deliveries; and the expected number of emergency deliveries γ_i for each hospital H_i in one period. Then the steps of algorithm 1 are as follows:

1: Read in all the data.

2: Compute the distance matrix, $D = \{d_{ij}\}$, $i = 1, \ldots, N+M$; $j = 1, \ldots, N+M$.

3: For each hospital H_{i_1}, determine the closest bank H_{N+j_1}. Set $y^\circ_{i_1 j_1} = 1$ (This step will determine y°.)

4: For each set $P_j(y^\circ)$, $j = 1, \ldots, M$, solve the traveling salesman problem, using the convex hull algorithm or any traveling salesman algo-

rithm available. The convex hull algorithm is given in Appendix A. (This step will determine x°.)

5: Apply the multiple assignment algorithm to the given set of points. This algorithm is given in Appendix B. (This step will determine y^{*}.)

6: For each set $P_j(y^{*})$, $j = 1, \ldots, M$, solve the traveling salesman problem, using the convex hull algorithm. (This step will determine x^{*}.)

7: Let $z_{min} = \min(z(x^{*}, y^{*}), z(x^{\circ}, y^{\circ}))$, where

$$z(x, y) = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} x_{ij} d_{ij} + \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} \gamma_i y_{ij} d_{ij} + \sum_{j=N+1}^{N+M} s\left(\sum_{i=1}^{N} y_{ij},\ 300 \sum_{i=1}^{N} Q_i y_{ij}\right).$$

Note that $z(x, y)$ is just the objective function of Problem 1 with the subscript k of x and summation over k deleted. Since there is only a single vehicle serving each supply point, this deletion does not affect the optimal value of Problem 1.

Let $(\tilde{x}, \tilde{y}) = (x^{*}, y^{*})$

8: a) Execute the following operations for each pair (j_1, j_2), $j_1 \neq j_2$, $j_1, j_2 = 1, \ldots, M$. When done, go to 9.

b) Determine the set $EX(j_1, j_2)$, if it is empty go to 8a.

c) Order $EX(j_1, j_2)$ such that for all $H_{i_1}, H_{i_1+1} \in EX(j_1, j_2)$

$$d_{i_1 j_1} - d_{i_1 j_2} \leq d_{i_1+1, j_1} - d_{i_1+1, j_2}.$$

d) Go to 8a.

Case 15: obtains the solution (x, y) to the BTAP, using the assignments y° obtained in case 9. The vehicle constraints are 20 stops per vehicle and 150 units per vehicle maximum.

Based on these cases, we can make a few interesting observations.

1. Algorithm 2, which is far slower than algorithm 1, produces very little significant improvement in the solution.

2. The problem is insensitive to the parameter γ_i, i = 1,..., N. (Even though a large value was assumed for γ_i, i = 1,..., N, in the marginal analysis of (x^*, y^*) and (x°, y°), the periodic delivery cost considerations always dominated the emergency cost considerations.)

3. In the existence of binding capacity constraints, $\tilde{y}$ produces very little significant improvement over y°.

These observations, provided that they hold in other cases in future research, imply that complex heuristics do very little over simple heuristics to improve the solution of the BTAP. This is a very interesting result; it might enable us to use simple heuristics and decision rules in the complex blood transportation-allocation problem. For example, if the first observation is true in general, algorithm 2 may be abandoned with considerable savings in execution time. If the last observation is true in general, one might simply use the allocations given by y°, which are very easy to determine, in the solution procedure of the BTAP, and avoid all the complicated processes to obtain y^* or $\tilde{y}$. The second observation reduces the need for accurate estimates for γ_i, the expected number of emergency deliveries to hospital H_i in one period. These observations, of course, have to be further tested, and this is left for future research.

9: a) Execute the following operations for each pair (j_1, j_2), $j_1 \neq j_2$, $j_1, j_2 = 1,\ldots, M$. When done, go to 10.

b) If $EX(j_1, j_2)$ is empty, go to 9a.

c) Remove the first element of $EX(j_1, j_2)$; let it be H_{i_1}. Define $\bar{y} = \{\bar{y}_{ij}\}$ such that $\bar{y}_{i_1 j_1} = 0$, $\bar{y}_{i_1 j_2} = 1$, and $\bar{y}_{ij} = \tilde{y}_{ij}$ otherwise.

d) Solve the traveling salesman problem for the sets $P_{j_1}(\bar{y})$ and $P_{j_2}(\bar{y})$. Store the resulting tours in $\bar{x} = \{\bar{x}_{ij}\}$ such that

$\bar{x}_{ij} = 1$ if $H_i \in P_{j_1}(\bar{y})$, $H_j \in P_{j_1}(\bar{y})$ and the convex hull algorithm solution to $P_{j_1}(\bar{y})$ contains an edge between H_i and H_j.

$\bar{x}_{ij} = 0$ if $H_i \in P_{j_1}(\bar{y})$, $H_j \in P_{j_1}(\bar{y})$ and the convex hull solution to $P_{j_1}(\bar{y})$ does not contain an edge between H_i and H_j.

$\bar{x}_{ij} = 1$ if $H_i \in P_{j_2}(\bar{y})$, $H_j \in P_{j_2}(\bar{y})$ and the convex hull solution to $P_{j_2}(\bar{y})$ contains an edge between H_i and H_j.

$\bar{x}_{ij} = 0$ if $H_i \in P_{j_2}(\bar{y})$, $H_j \in P_{j_2}(\bar{y})$ and the convex hull solution to P_{j_2} does not contain an edge between H_i and H_j.

$\bar{x}_{ij} = \tilde{x}_{ij}$ otherwise.

e) Compare z_{min} with $z(\bar{x}, \bar{y})$. If z_{min} is smaller go to 9b; otherwise proceed.

f) Update $\tilde{x}$, $\tilde{y}$, and z_{min}. $\tilde{x} = \bar{x}$, $\tilde{y} = \bar{y}$, and $z_{min} = z(\bar{x}, \bar{y})$.

g) Go to 9b.

10: All of the exchanges are completed, terminate. $(\tilde{x}, \tilde{y})$ is the resulting near optimal solution; z_{min} is the resulting near optimal value.

In most applications, the above algorithm will produce a very good solution in a reasonable amount of time (see the next section for the actual execution times). However, in some extreme cases (i.e., cases in which most of the sets $EX(j_1, j_2)$ are very large), it could be very time consuming, considering that two traveling salesman problems are solved after each exchange and there are as many exchanges as the total number of elements in the sets $EX(j_1, j_2)$, $j_1 \neq j_2$; $j_1, j_2 = 1, \ldots, M$ (i.e., there are "$\sum_{j_1=1}^{M} \sum_{\substack{j_2=1 \\ j_2 \neq j_1}}^{M} |EX(j_1, j_2)|$" exchanges).

Algorithm 2

The following algorithm is a solution procedure for the BTAP, when the constraints on the vehicles are not binding. In this algorithm, the sets $EX(j_1, j_2)$ are ordered such that for all H_i, $H_{i+1} \in EX(j_1, j_2)$

$$d_{i_1 j_1} - d_{i_1 j_2} \leq d_{i_1+1, j_1} - d_{i_1+1, j_2} .$$

Then each pair of sets $EX(j_1, j_2)$, $EX(j_2, j_1)$, $1 \leq j_1 < j_2 \leq M$ are considered one at a time. The first elements H_{i_1}, H_{i_2} of the sets under consideration, $EX(j_1, j_2)$ and $EX(j_2, j_1)$, respectively, are removed, and the following temporary exchanges are done in the given order:

H_{i_1} is removed from $P_{j_1}(y^*)$ and inserted into $P_{j_2}(y^*)$.

H_{i_1} and one of the hospitals adjacent to it (adjacent in the graph defined by $(\tilde{x})$ are removed from $P_{j_1}(y^*)$ and inserted into $P_{j_2}(y^*)$.

H_{i_1} and the other hospital adjacent to it (adjacent in the graph defined by $(\tilde{x})$ are removed from $P_{j_1}(y^*)$ and inserted into $P_{j_2}(y^*)$.

H_{i_1} and both of the hospitals adjacent to it are removed from $P_{j_1}(y^*)$ and inserted into $P_{j_2}(y^*)$.

H_{i_2} is removed from $P_{j_2}(y^*)$ and inserted into $P_{j_1}(y^*)$.

H_{i_2} and one of the hospitals adjacent to it are removed from $P_{j_2}(y^*)$ and inserted into $P_{j_1}(y^*)$.

H_{i_2} and the other hospital adjacent to it are removed from $P_{j_2}(y^*)$ and inserted into $P_{j_1}(y^*)$.

H_{i_2} and both of the hospitals adjacent to it are removed from $P_{j_2}(y^*)$ and inserted into $P_{j_1}(y^*)$.

After each exchange the new feasible solution to Problem 1 is determined (as described in the previous section) and compared with (x°, y°), (x^*, y^*). If it is a better solution, the corresponding exchange is made permanent. This process is continued until both of the sets $EX(j_1, j_2)$, $EX(j_2, j_1)$, for j_1, j_2 under consideration, are empty. (Details of this algorithm are given in Or [1976].)

Extensions of Algorithm 1 and Algorithm 2

The algorithms presented in previous sections may be extended quite easily, to provide solutions also in the cases in which the constraints on the vehicles are binding (i.e., the underlying multidepot problem is the MVDP rather than the MTSP). One possible extension would be to use the Gillett and Miller sweep algorithm instead of the convex hull algorithm in steps 4, 6, 9d of algorithm 1 and in algorithm 2. Let us call this extension 1. Another extension would be to leave the first nine steps of the algorithm the same and change step 10 to:

10: Consider $\tilde{y}$ to be the near optimal allocations. For each set $P_j(\tilde{y})$, j = 1,..., M, solve the vehicle dispatch problem using the sweep algorithm.

Let us call this last extension, extension 2. Notice that in extension 2 the sweep algorithm is applied only M times (in step 10), whereas in extension 1, it is applied M times in step 4, M times in step 6 and then twice after each exchange (it replaces the convex hull algorithm). Hence, as the sweep algorithm is slower than the convex hull algorithm, extension 1 is slower than extension 2. Extension 2 is coded in fortran IV and tested with the data given in Appendix 1 (see the next section for the computational results).

Considering how large and complex the BTAP is, algorithms 1 and 2 and their extensions are at least a good start; they produce some very acceptable feasible solutions and give us some insights about the BTAP, as will be discussed in the next section. However, we should note that we do not really know how good (or bad) a solution we are getting from these algorithms. For example, we do not know whether the allocations, $\tilde{y}$ (which were obtained for the MTSP and used in step 10 of extension 2, in solving the MVDP) are near optimal allocation for the MVDP or not. Similarly, we do not know how good an assignment for the MVDP will be obtained in step 5 of extension 1 using the assignment algorithm of section 3.5, which was developed for the MTSP. Also, the sweep algorithm itself has an arbitrary nature, the solutions it produces are not guaranteed to be near optimal. These are all relevant shortcomings of our solution procedures and more research will be done to resolve them. (These algorithms and their extensions are available in Or and Pierskalla [1976].)

The Convex Hull Algorithm

(A Heuristic Algorithm for the Traveling Salesman Problem)

The Convex Hull Algorithm is a "tour building" algorithm. It starts with a subtour and builds that into a tour by including unassigned nodes into the graph one by one. The Karg-Thompson [1964] algorithm is also based on this principle. The two algorithms, however, differ in some very important aspects. The Karg-Thompson algorithm starts with a triangle and "expands" arbitrarily to include the other nodes, whereas the Convex Hull Algorithm starts with the circuit forming the boundary of the convex hull of the nodes and "contracts" in a definite order to include all the other nodes.

Let N be the set of n nodes. The convex hull on N, call G, is the smallest convex set that includes N. So, by definition, the boundary of G is made up of edges that connect nodes of N, and it is a circuit. Also, there exists an optimal tour of N, in which the relative order of nodes on the boundary of G is preserved [Bellmore and Nemhauser, 1968]. This means, if the optimal tour were followed, nodes on the boundary of G would be visited in the same order as if the boundary of G were followed. This observation is one important reason why the boundary of G is a good starting point. The basic idea of the Convex Hull Algorithm will work when nodes are in R^3 or for that matter R^n (for whatever physical meaning the Traveling Salesman Problem might have in R^n) and when the underlying distance matrix is not symmetric. However, for descriptive simplicity, computational speed and real world problems of interest to us, we will

confine ourselves to R^2 and to symmetric problems from now on. Also it should be noted that the distance matrix does not have to obey the triangle inequality for the algorithm to be successful. (This last point is shown by the performance of the algorithm on test problems 2 and 4). However, the Convex Hull Algorithm does depend on the topology of the given set of nodes and is not expected to perform well on randomly generated distance matrices. In this last class of problems there is the additional difficulty of defining the boundary of the convex hull.

The Algorithm

Assume the coordinates of n nodes and a distance norm (Euclidean, Rectilinear, etc.) or a distance matrix and nodes on the boundary of the convex hull of N (this is the starting circuit) are given. The algorithm is as follows:

1. If you are given the distance matrix and nodes on the boundary of G, go to step 2. Otherwise, compute the distance matrix and determine the circuit that forms the boundary of G. This is the starting circuit. If the starting circuit has n nodes, stop, this is the optimal solution [Bellmore and Nemhauser, 1968]. Otherwise continue.

2. a) For each edge on the starting circuit, find the "closest" (this term will be defined separately) node among nodes that are not in the circuit yet. These are the "candidate nodes" to be included in the circuit.

b) Store the two endpoints of each edge, the corresponding candidate node and the "distance" between that node and the edge as four attributes of the entity candidate.

c) Order the stored candidates according to how "close" they are to the circuit (from closest to the farthest).

3. Pick and remove the candidate at the top of the ordered list (closest to the existing circuit). Check if the candidate node is already in the circuit by having been chosen at an earlier iteration. If it is not, go to step 5. Otherwise continue.

4. Consider the edge defined by the two endpoint attributes of the chosen entity, candidate. For this edge, find the "closest" node among nodes that are not in the circuit yet. Include this node and the edge in the ordered candidate list the same way as in step 2. Go to step 3.

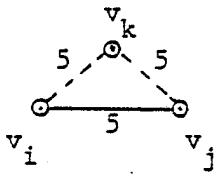
5. a) Consider the edge defined by the two attributes of the chosen entity, candidate. Modify the existing circuit by deleting this edge and adding the two edges that connect the endpoints of the deleted edge to the chosen candidate node.

b) For each of the two new edges find the "closest" node among nodes that are not in the circuit yet, and include the edge and that node in the ordered candidate list the same way as in step 2.

6. If all of the n nodes are in the circuit constructed, terminate. The existing circuit is the near-optimal Traveling Salesman Tour. Otherwise go to step 3.

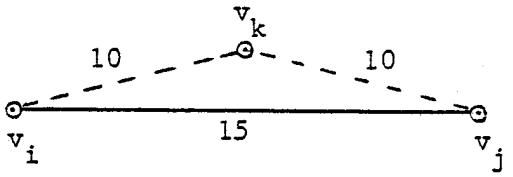
In the above algorithm "closeness" of a node v_k to an edge (v_i, v_j) can be measured by many different criteria. The length of the shortest straight line from v_k to (v_i, v_j) is one criterion. The difference given by [D: distance (v_i, v_k) + distance (v_k, v_j) - distance (v_i, v_j)] is

another criterion. The ratio [R: (distance (v_i, v_k) + distance (v_k, v_j))/ distance (v_i, v_j)] is still another. We have experimented with all of the above criteria and their combinations. The best results were obtained by using the product of difference and ratio [R x D: $(d(v_i, v_k) + d(v_k, v_j) - d(v_i, v_j)) \times (d(v_i, v_k) + d(v_k, v_j))/d(v_i, v_j)$]; however, we do not yet fully understand why this criterion yields better results. A hypothesis is that D alone has trouble distinguishing between cases of equal differences such as



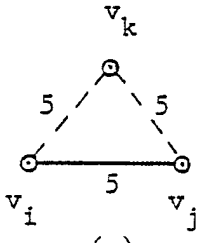
(a)

and



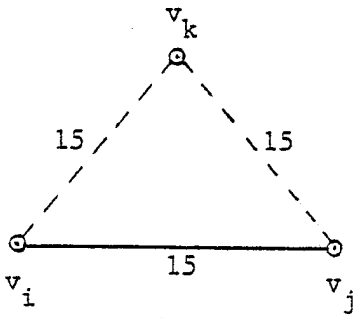
(b)

and R alone has trouble with the equal ratios of similar triangles such as



(c)

and



(d)

whereas topology and common sense tell us that case (b) should be considered before case (a) and that case (c) should be considered before case (d).

Extensions

The algorithm described above is very fast; however, its accuracy can be improved. In this case one might be willing to sacrifice some speed for additional accuracy. One source of inaccuracy is the algorithm's inflexibility in changing the order of nodes already in the circuit in later iterations. The extensions we have devised try to solve this problem.

In Extension 1 the search for the "closest" node (to a given edge) in steps 4 and 5 is done over all nodes (rather than being restricted to a subset of nodes, i.e., nodes which are not in the circuit yet). Then in step 4 the chosen candidate node (which is already in the circuit) is not immediately discarded, but it is checked to ascertain whether the solution would improve if that node is removed from its existing position and placed in a new position (as defined by the edge contained in the chosen candidate). If an improvement is found, the route is altered accordingly, and if not then it is discarded and a new candidate is chosen.

In Extension 2 the algorithm (steps 1 to 6) is not changed, but an extra step is added to "refine" the tour obtained. In this refinement process a check is made to ascertain whether the existing tour would improve if the position (in that tour) of any edge is changed. If an improvement is found, the tour is altered accordingly. Then a check is made to discover whether the existing tour would improve again if the position (in that tour) of any node is changed. If an improvement is found, the tour is altered accordingly. This concept of tour improvements is similar to the concept of λ - optimality in the Lin [1973] algorithm.

Results

The Convex Hull Algorithm and its extensions were coded in Fortran IV and run on the CDC 6400 computer of Northwestern University. The execution times obtained are quite impressive even on large (117 node) problems. The solutions obtained for some well-known problems in the literature are 0% to 1.5% above the known optimal solutions. Table A-1 below summarizes the performance of the Convex Hull Algorithm on these problems. Many of the test problems in the literature have a randomly generated distance matrix, this is the main reason why our comparative tests are limited to 5 problems. Table A-2 below gives the performance of the algorithm with and without its extensions and A-3 with different "closeness" criteria. Finally, our calculations and results from the experiments show that this algorithm is of the n^3 type. (The execution times increase in proportion to n^3 when number of nodes n increases.)

Test Problem	Reference	Number of Nodes	Convex Hull Algorithm Solution [3]	Optimal Solution	% Difference	Execution Time of C. Hull Alg. [2]
1[1]	Karp 1962	25	1723	1711	0.70%	0.301
2	Karg 1964	33	10861	10861	None	0.496
3	Dantzig 1959	42	707	699	1.14%	0.789
4	Karg 1964	57	13066	12955	0.85%	1.479
5	Krolak 1971	100	21282	21282	None	3.897

[1] The convex hull for this problem is obtained by plotting the nodes on a plane from the distance matrix, using a compass.

[2] All execution times are given in seconds.

[3] The algorithm used is the convex hull algorithm with both of its extensions.

Test Problem	Reference	Number of Nodes	Convex Hull Algorithm Without Extensions		Convex Hull Alg. With Extension 1		Convex Hull Alg. With Extension 2		Convex Hull Alg. With Extension 1,2	
			Solution	Exec. Time	Solution	Time	Solution	Time	Solution	Time
2	Karg 1964	33	10929	0.37	10929	0.41	10861	0.45	10861	0.49
3	Dantzig 1959	42	724	0.54	716	0.69	707	0.66	707	0.79
4	Karg 1964	57	13114	1.11	13170	1.35	13066	1.17	13066	1.47
5	Krolak 1971	100	22083	2.33	21310	3.24	21366	2.80	21282	3.89
6	Appendix 1	117	11343.1	2.84			11193.4	3.72		

Test Problem	Number of Nodes	Solution with [1] criterion "R x D"	Solution with [1] criterion "R"	Solution with [1] criterion "D"
2	33	10929	11586	10929
3	42	724	732	724
4	57	13114	14482	13957
5	100	22083	22516	23049

[1] The algorithm used is the original convex hull algorithm without its extensions.

Multiple Assignment Algorithm

(A Heuristic Algorithm for MTSP)

Given N demand points and M supply points and a distance relationship between all pairs of points, the algorithm below first assigns the demand points to the supply points and then uses the Convex Hull Algorithm, discussed in previous sections, to solve M separate Traveling Salesman Problems.

The algorithm is:

1. For each supply point create a group and find the p closest unassigned demand points (p can be 1, 2, 3, 4 or 5, the choice of p will be discussed after the presentation of the algorithm). Assign them to that supply point by putting them into the corresponding group.

2. For each group construct a circuit that includes all the points in that circuit. These will be the starting circuits and their union will be the starting graph.

3. a) For each edge on each starting circuit find the "closest" unassigned point. These are the "candidate" points to be assigned to a supply point by being included in the corresponding groups.

b) Store the two endpoints of each edge, the corresponding candidate point, and the "distance" between that point and the edge as the four attributes of the entity, "candidate".

c) Order the stored candidates according to how "close" they are to the existing graph (from closest to the farthest).

4. Pick and remove the candidate at the top of the candidate list (closest to the existing graph). Check if the candidate point is already in a group (assigned to a supply point). If it isn't, go to step 6; otherwise continue.

5. Consider the edge, defined by the two endpoint attributes of the chosen entity, candidate. For this edge, find the "closest" unassigned point. Include this point and the edge in the ordered candidate list, in the same way as in step 3. Go to step 4.

6. a) Consider the edge defined by the two endpoint attributes of the chosen entity, candidate. Modify the existing graph by deleting this edge from the circuit that contained it and adding to that circuit the two edges that connect the endpoints of the deleted edge to the chosen candidate point. Assign the candidate point by including it in the corresponding group.

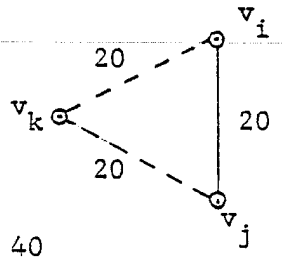
b) For each of the two new edges find the "closest" unassigned point and include the edge and that point to the candidate list the same way as in step 3.

7. If all of the N demand points are assigned, proceed to step 8. Otherwise go to step 4.

8. The assignments are completed. Each group contains one supply point and the demand points assigned to it. Apply the Convex Hull Algorithm to each group, to get the delivery routes, then terminate.

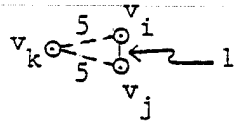
In the above assignment algorithm, a number of different "closeness" criteria have been tried. The best results were obtained by using the

criterion "sum of the shortest straight line from the node to the edge (SH) and the cosine of the angle between the two new edges at the node (ANG)". Again, we do not yet fully understand why this criterion yields better results. The "product of difference (D) and ratio (R)" criterion, which gave good results when used in the Convex Hull Algorithm, did not perform that well in the above assignment algorithm. One hypothesis against the use of the ratio (alone or as a multiplier) is that, in early stages of the algorithm, most of the edges in the existing circuits are relatively short, hence using their lengths as denominators in the ratios will produce large "distances" and will bias the candidate nodes towards the longer edges. For example, using the criterion "R x D", the algorithm would consider the case in figure A-1, which has an R x D value of 80, before the case in figure A-2, which has an R x D value of 90, even though the latter seems like a better first choice.



R = 2, D = 40

Fig. (1)



R = 10

D = 9

Fig. (2)

The criterion SH was also tested in the assignment algorithm. This criterion's major drawback was that it failed to distinguish between cases such as the ones in figures A-3 and A-4 (it is clear that the case in figure A-4 should be considered first). The criterion "SH + ANG" overcame that drawback. A criterion such as "ANG x SH" would also have overcome that drawback, but using ANG as a multiplier would have caused problems similar to the ones caused by using R as a multiplier.

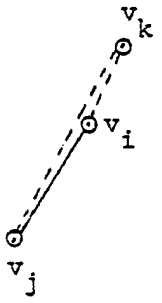


Fig. 3

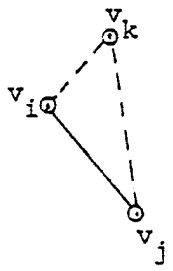


Fig. 4

Sometimes, because of the topology of the demand points, the solution we get from the above assignment algorithm is very sensitive to the choice of p, the number of demand points in the starting circuits. If it is too large, the arbitrary nature of the initial assignments will take their toll. For example, given the topology in figures A-5 and A-6, p = 4 would lead to the solution illustrated in figure A-5, and $p \leq 3$ would lead to the better solution illustrated in figure A-6. If p is too small, on the other hand, the probability of an unbalance growth

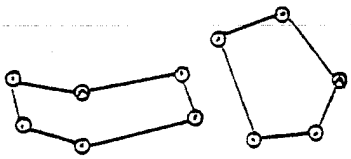


Fig. 5

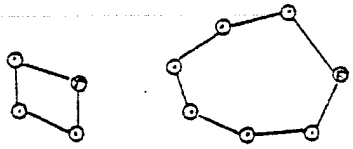


Fig. 6

⊙ represents a supply point

⊙ represents a demand point

in the groups will be larger. For an example of unbalanced growth, take the case p = 1. At the end of step 3 of the algorithm, there will be two entries in the candidate list for each group (note that the two entities corresponding to the same group will be equal in distance and candidate point attributes). Then, if the candidates corresponding to any one

group have comparatively large distance values, the other groups might grow very large before the algorithm ever reaches those candidates in the expanding ordered candidate list. On the average, lower values of p give better results in smaller problems and higher values of p in larger problems. In the application of the assignment algorithm to the blood transportation allocation problem, three different values of p (two, three and four) were tested. The best results were obtained using p = 3.

It should be noted that in the above assignment algorithm, actual construction and storage of the circuits are not really necessary, as long as one keeps track of the assignments and the candidate list accurately; those steps are included only to clarify the reasoning behind the algorithm and to complete the parallel between the Convex Hull Algorithm and this one.

This assignment algorithm is very similar, in structure, to the Convex Hull Algorithm. So, once the latter is coded, it is very easy to extend it to the assignment process as described above. In fact, this assignment algorithm is coded as a subroutine (subroutine ALLOC2, Or and Pierskalla [1976]) and applied to some MTSP in solving the blood transportation allocation problem. Also, it is very easy to adopt the assignment algorithm for solving the MVDP. The first seven steps would be the same and in step 8 an algorithm for the MVDP would be used instead of the Convex Hull Algorithm. This feature too is coded as a subroutine (subroutine DISPATCH).

There is a possible improvement of the assignment algorithm based on a concept first used by Gillett and Johnson [1974]. This is changing step 1 to:

1. a) For each supply point initialize a group.

b) For each demand point i find the closest supply point j_1 and the second closest supply point j_2 and compute the ratio, $r(i) = d_{i,j_1}/d_{i,j_2}$.

c) Assign all demand points i that have $r(i) > \delta$ (for a predetermined δ) to their closest supply point by including them in the corresponding group.

The original step 1 is very rigid in the way it forms the starting groups, since it does not take into consideration the topology of the problem. "The p closest unassigned demand points to a supply point" might be a very poor decision rule with which to assign points. On the other hand initially assigning only those points i with large r(i) is a more topologically oriented decision rule. If δ is large enough, there is a very good chance that assignments made through this decision rule are optimal assignments (optimal in the sense that there exists an optimal solution in which the assignments of those points are the same). The following example will clarify this argument.

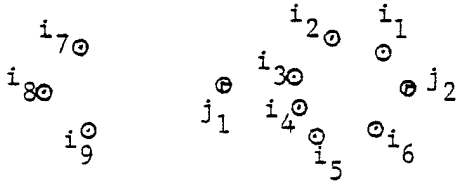


Fig. 7

Given the configuration of figure A-7, original step 1 would assign i_3, i_4, i_5 to j_1 and i_1, i_2, i_6 to j_2 and the application of the whole algorithm would result in the circuits given by figure A-8.

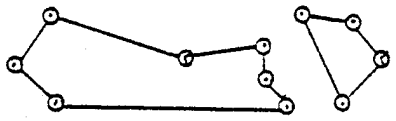


Fig. 8

Whereas the revised step 1 would assign i_7, i_8, i_9 to j_1 and i_1, i_6 to j_2 and the application of the whole algorithm would result in the circuits given by figure A-9.

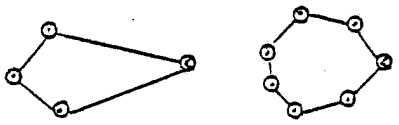


Fig. 9

So, through the above arguments, this change is expected to improve the performance of the assignment algorithm; however, it has not yet been coded and tested.