BTAP: A COMPUTER PROGRAM TO OBTAIN SOLUTIONS TO
THE BLOOD TRANSPORTATION-ALLOCATION PROBLEM
AND OTHER TRAVELING SALESMAN TYPE PROBLEMS

ILHAN OR
and
WILLIAM P. PIERSKALLA

AUGUST, 1976

Department of Industrial Engineering
and Management Sciences
Northwestern University
Evanston, Illinois 60201
BTAP: A Computer Program to Obtain Solutions to the
Blood Transportation-Allocation Problem and
Other Traveling Salesman Type Problems

by

Ilhan Or

and

William P. Pierskalla

August, 1976

This research was partially supported by PHS-HS00786-01 from
the National Center for Health Services Research, and
by GK-40034 from the National Science Foundation,
William P. Pierskalla, Principal Investigator
This is a descriptive paper to supplement the research in [OR, 1976]. All of the new algorithms discussed in [OR, 1976] are coded in Fortran IV, to form the alternate library BTAP. This paper contains the actual codings and descriptions of the subroutines and functions in the library BTAP. Section 1 gives some general information on how to use BTAP. Section 2 contains the descriptions of the subroutines and of the important parameters and variables. The appendix contains the codings of the subroutines and functions in BTAP and of the main program BTP.

Section 1.

The library BTAP is designed to be used with the main program BTP. The input data for the package (BTAP and BTP) is read from two sources: TAPE5, which is usually a permanent file, and TAPE1, which is the keyboard. This package is built to run on a CDC 6400 computer and needs 75000 octal core memory and 55000 octal extended core storage to execute. TAPE5 should be composed of two files. The first file should contain the coordinates and the parameters $\gamma_i$ of the hospitals involved in the given problem with the format "FORMAT (3F5.0)" (x-coordinate, y-coordinate, $\gamma_i$). Let there be N card images in this file. The second file should contain N card images with the format "FORMAT (I4, I6)" (each card image should contain a hospital and the amount of blood used per year in that hospital) and then values for SYCOST (1) to SYCOST (120) with the format "2X, 10I5".

Section 2.

The following is a short summary of all the important variables, parameters and other information in the program and how and where they are stored.
ALFA (150): For each hospital I, the expected number of emergency deliveries in one period, \( \gamma_i \), is stored in ALFA (I). \( \gamma_i, i = 1, \ldots, N \) are read and stored in subroutine READ1. ALFA (150) is in the common block \(|E|\).

CANDID (600, 4): This is the candidate list. CANDID (I, 3), and CANDID (I, 4) contain the endpoint nodes of an edge. CANDID (I, 2) contain the node (hospital) "closest" to that edge. CANDID (I, 1) contains the "distance" of the closest node from that edge. CANDID (600, 4) is in the common block \(|D|\).

DMT1 (150): This vector is used as temporary storage for information to be sent to or received from Extended Core Storage (ECS). If the ECS calling statement is "CALL READEC (DMT1, DATX (I, I), N)," DMT1 (J) will contain the Euclidean distance between nodes I and J. DMT 1 (150) is in common block \(|CM|\).

DMT2 (150): This vector is used in the same way as DMT1. It is also in common block \(|CM|\).

EMCOST (5): EMCOST (I) contains the total expected emergency delivery cost for region I. It is calculated in subroutine ALLOC1, ALLOC2, UPDATE and DISPACH. EMCOST (5) is in common block \(|K|\).

IADRES (600): IADRES (I) contains the address (row number) of the \( i^{th} \) best candidate so far (i.e., the distance given by CANDID (IADRES (I), 1) is the \( i^{th} \) smallest) in the candidate list CANDID. It is computed in subroutine ADRES and used in subroutines BOUND1 and ALLOC2. It is in common block \(|D|\).
IALTER (5, 5, 10): The vector IALTER (I, J, •) is equivalent to the set EX (I, J), which was defined and discussed in section 4.3 of [OR, 1976]. It is determined in subroutine IMPROVE and used in subroutines ALTER1 and ALTER2. IALTER (5, 5, 10) is in common block |F|.

IBANK (5): IBANK (I) contains the identification number of the central bank in region I. It is read from the keyboard in the main program BTP. IBANK (5) is in common block |C|.

IBLAD (150): IBLAD (I) contains the amount of blood used per year in hospital I. It is read from TAPES in subroutine READ1. IBLOOD (150) is in common block |E|.

IBLADTR (50): This array contains the number of units of blood carried by each periodic delivery vehicle. It is calculated and stored in subroutine DISPACH. IBLADTR (50) is in the common block |F|.

IBTR (5): IBTR (K) contains the number of active periodic delivery vehicles in region K. It is calculated and stored in subroutine DISPACH. IBTR (5) is in the common block |F|.

IHULL (300, 3): This array is used in the construction of tours in BOUND1 and ALLOC2. Each row contains information about an edge; the first two columns define the edge (they contain the two end nodes of the edge) and the third column indicates whether that edge is in the existing graph (value 0) or has been deleted (value 1). At the end of the subroutines BOUND1 and REFINE1, the first two columns of IHULL contain the resulting tours. IHULL (300, 3) is in common block |C|. 
IHOSP (150, 5): IHOSP (I, 3) is a 0, 1 indicator used in subroutines BOUND1 and PFIND. IHOSP (I, 3) is zero when hospital "I" is not yet in the existing graph and one otherwise. IHOSP (I, 4) indicates the region hospital I is assigned to under the assignment algorithms ALLOC1, ALTER1 and ALTER2. IHOSP (I, 5) indicates the region hospital I is assigned to under the assignment algorithm ALLOC2. IHOSP (I, 1) and IHOSP (I, 2) indicate the two hospitals adjacent to hospital I in the latest routing obtained. IHOSP (150, 5) is in common block |A|.

IHULLA (150, 2): This array contains routing information in a form that can be used by the output subroutines, PLOTING and PRINTS. The first column of IHULLA contains the routing obtained using the assignments of subroutines ALLOC1 or ALTER1 or ALTER2 (of course, only the latest routing obtained is there, the earlier ones are erased). The second column of IHULLA contains the routings obtained using the assignments of subroutine ALLOC2. IHULLA (150, 2) is in common block |B|.

IIHOSP (150, 3): The first two columns of this array are used to store the original first two columns of IHOSP (150, 5), since the information contained in those columns of IHOSP is destroyed during the calls to BOUND1 and REFIN1 in subroutine TEST. IIHOSP (I, 1) and IIHOSP (I, 2) are used in subroutine ALTER2, in determining the two adjacent nodes to node I ∊ EX (j₁, j₂) (see section 4.3 of OR [1976]). These two columns of IIHOSP are updated in subroutine UPDATE. The third column of IIHOSP is an indicator vector, IIHOSP (I, 3) = -1 if hospital I has never been in an exchange, IIHOSP (I, 3) = 0 if hospital I has already been permanently assigned to another region, IIHOSP (I, 3) = J if the pair (I, J) has been in an
exchange before. This convention is helpful in avoiding duplicate tests in subroutine ALTER2. IIHOSP (150, 3) is in common block $|F|$. 

$N$: $N$ is the number of hospitals in the whole system. It is determined in subroutine READ1.

$NALTER (5, 5)$: $NALTER (I, J)$ contains the number of elements in the set EX $(I, J)$ (or equivalently in the vector $IALTER (I, J, \cdot)$). It is determined in subroutine IMPROVE and used in subroutines ALTER1 and ALTER2. 
$NALTER (5, 5)$ is in common block $|F|$. 

$NBANK$: This is the number of central blood banks in the whole system. It is read from the keyboard in the main program, BTP.

$NUM (50)$: This vector contains the number of stops for each vehicle. It is in common block $|C|$. 

$NUMTR (50)$: $NUMTR (I)$ contains the number of hospitals in region I. It is in common block $|F|$. 

$SCAL$: It gives the scale of the plots. It is read from the keyboard in the main program BTP.

$SYCOST (120)$: $SYCOST (K)$ gives the systems cost estimate for a region containing $K$ hospitals. It is read from TAPE5 in subroutine READ1. It is in common block $|K|$. 

$TSCOST (5)$: $TSCOST (I)$ gives the total periodic delivery costs in region I. $TSCOST (5)$ is in common block $|K|$. 
X (150): X (I) contains the x-coordinate of hospital I. It is read from TAPE5 in subroutine READ1. X (150) is in common block |E|.

Y (150): Y (I) contains the y-coordinate of hospital I. It is read from TAPE5 in subroutine READ1. Y (150) is in common block |E|.

There are 19 subroutines and one function in the library BTAP (Blood Transportation-Allocation Problem). The following is a brief description of them.

SUBROUTINE READ1 (N)

The function of this subroutine is to read the input data from TAPE5. First X, Y coordinates and the parameter γ_I for each hospital are read and stored in vectors X (150), Y (150) and ALFA (150), respectively. Number of hospitals in the whole system, N, is determined. The blood usage of hospitals are read and stored in IBLAD (150). The system costs for all feasible system sizes are read and stored in SYCOST (120).

SUBROUTINE DISMAT1 (N)

The function of this subroutine is to calculate the distance matrix. For each pair of hospitals (I, J), the Euclidean distance between them is computed and stored in ECS (Extended Core Storage) at address DATX (I, J). Information storage to (and retrieval from) ECS is done one column at a time.

SUBROUTINE ALLOCI (N, NBank)

The function of this subroutine is to determine the hospital assignments that will minimize the expected emergency blood delivery costs (i.e.,
the assignments given by $y^0$ as defined in section 4.2 of OR [1976]). Thus, the closest bank $J$ to each hospital $I$ is determined. $I$ is assigned to bank $J$ by storing the group number of $J$ in IHOSP $(I, 4)$. Then, for each group $K$, the number of hospitals in it, the amount of blood used in it and the total expected emergency delivery costs for it are calculated and stored in NUM $(J)$, NBLAD $(J)$ and EMCOST $(J)$, respectively.

SUBROUTINE ALLOC2 (N, NBANK)

This subroutine corresponds to the assignment algorithm described in section 3.5 of OR [1976]. The group number of each hospital $I$ is determined (as described in section 3.5) and stored in IHOSP $(I, 5)$. Then for each group, $J$, the number of hospitals in it, the amount of blood used in it, and the total expected emergency delivery costs for it are calculated and stored in NUM $(J)$, NBLAD $(J)$ and EMCOST $(J)$, respectively.

SUBROUTINE TRAVEL (N, NBANK, KX)

Once the assignments of the hospitals (either $y^0$ given by ALLOC1 or $y^*$ given by ALLOC2) are completed, this subroutine determines the routings for the MTSP (by calling the subroutines related to the convex hull algorithm) and stores them in column KX of IHULLA $(150, 2)$ (column 1 if the given allocation is $y^0$ and column 2 if the given allocation is $y^*$) in the form that is required by subroutines PRINTS and PLOTING.

SUBROUTINE DISPACH (N, NBANK, KX, STOPS, UNITS, SCAL)

Once the assignments of the hospitals are completed (either $y^0$ given by ALLOC1 or $y^*$ given by ALLOC2 or $\tilde{y}$ given by ALTER1 or ALTER2 or $\hat{y}$ supplied externally), this subroutine determines the routings for the MVDP (using the sweep algorithm discussed in section 2.3.4.3 of OR [1976]) and stores
them in column KK of IHULLA (150, 2) in the form that is required by sub-
routines PRINTS and PLOTTING (KK is 1 if the assignments are given by $y^0$, $\hat{y}$ or $\tilde{y}$ and KK is 2 if the assignments are given by $y^*$). Number of vehicles
serving each region, IBTR (1) to IBTR (NBANK); the total number of vehicles,
JSTAR; the number of stops for each vehicle, NUMTR (1) to NUMTR (JSTAR);
the amount of blood carried by each vehicle, IBLADTR (1) to IBLADTR (JSTAR);
the total expected emergency cost for each region, EMCO$\text{ST}$ (1) to
EMCO$\text{ST}$ (NBANK); the total periodic delivery cost for each region, TSCOST (1)
to TSCOST (NBANK); the total amount of blood used per year in each region,
NBLAD (1) to NBLAD (NBANK) are also determined in the subroutine. (Note
that the calls to subroutines CHULL1, BOUND1 and REFINE1 from this subrou-
tine determine just single vehicle routes.)

SUBROUTINE CHULL1 (N, IC, KK, IX)

The function of this subroutine is to determine the graph, $G$, which
is the boundary of the convex hull of the nodes in region IX. The edges
in $G$ are stored in IHULL (300, 3). Throughout the subroutine (in fact,
throughout the whole program), "IHOSP (I, 4+KK)" is compared with "IX",
to screen away the nodes that are not in region IX; in case of an in-
equality hospital I is disregarded. KK is a 0, 1 variable allowing the
user to select either the fourth or the fifth column of IHOSP for screen-
ing. IC gives the number of edges in the existing graph, it is updated
in subroutine PFIND.

The procedure used to determine $G$ is as follows:

First the nodes with the smallest and the largest x-coordinates are
determined (let them be $H_{11}$ and $H_{1p}$, respectively). Then, starting with
the following step is applied until H_{ip} is reached: "Given H_{i}, find H_{j} such that the slope of the line passing from H_{i}, H_{j} is the maximum over all nodes that have a larger x-coordinate than H_{i}. Store H_{j}, let H_{i} = H_{j}."

When H_{ip} is reached, the following step is applied until H_{il} is reached:
"Given H_{i} find H_{j} such that the slope of the line passing from H_{i}, H_{j} is the maximum over all nodes that have a smaller x-coordinate than H_{i}. Store H_{j}, let H_{i} = H_{j}." Once H_{il} is reached, the graph G will be completely determined.

SUBROUTINE PPIND (N, NSIZE, K1, K2, KPOINT, KX, IX)

The function of this subroutine is to determine the "closest" node, IBEST, out of all the nodes in region IX that are not in the graph yet, to the edge defined by the end nodes K1 and K2. Then the "distance" from IBEST to edge (K1, K2), IBEST, K1 and K2 are stored in row NSIZE of the candidate list CANDID (600, 4). NSIZE which contains the total number of candidates (rows) in CANDID (600, 4) is updated in this subroutine. KPOINT is a pointer passed on to subroutine ADRES.

If this subroutine is entered through the calling statement CALL PPIND (which is the case in subroutines CHULL and BOUND1), the distance criterion used is "Rx D" (Ratio x Difference - see section 3.4.2 of OR [1976] for detailed explanations). On the other hand, if this subroutine is entered through the calling statement CALL AFIND (which is the case in subroutine ALLOC2), the distance criterion used is "ANG x SH (see section 3.5 of OR [1976] for explanations).

SUBROUTINE ADRES (NSIZE, KPOINT)

The function of this subroutine is to order the candidates (rows) in the array CANDID according to their first attribute (the first attribute
contains the "distance" of the candidate node, which is the second attribute, from the edge defined by the third and fourth attributes). The ordering is done by rearranging the entries of the adres vector to the candidate list, IADRES (600).

In theory, the best candidate (the row with the smallest value in column 1) is taken out of the candidate list and considered for insertion into the graph. In practice, however, it is much faster and easier not to delete anything from CANDID and IADRES, but instead to increase KPOINT, the pointer to the starting point in IADRES, by one unit. So at any instant in the algorithm, IADRES (1) to IADRES (KPOINT) contains the row numbers, in CANDID (600, 4), of candidates that have already been considered (i.e., the candidates that should have been deleted). Hence, IADRES (1) to IADRES (KPOINT) is disregarded and the rest of the vector IADRES is rearranged such that IADRES (KPOINT+1) contains the row number of the candidate node that is $i^{th}$ closest to the existing graph.

SUBROUTINE BOUND1 (N, IC, COST, KX, IX)

The function of this subroutine is to modify $G$, the boundary of the convex hull of all nodes in group IX, into a circuit passing from all of the nodes in group IX. In other words, this subroutine executes step 3, 4, 5 and 6 of the convex hull algorithm (see section 3.4.2 of OR [1976]). Subroutine PFIND is called to find new candidates and to store and order them. The modified graph is stored in IHULL (300, 3). IC keeps track of the number of edges stored in IHULL. In theory, every time the existing graph is modified one edge is deleted and two new edges are added. In practice, however, no physical deletion from IHULL occurs. The third column of IHULL is considered to be a vector of flags, and every time
the algorithm calls for a deletion the flag of the edge that is to be deleted is changed from 0 to 1.

Once the traveling salesman route for group IX is determined, the cost is computed and stored in COST (IX). Also the two adjacent nodes to each node I are stored in IHOSP (I, 1) and IHOSP (I, 2) and IHULL is rearranged so that columns 1 and 2 in the first NUM (IX) rows contain the resulting traveling salesman route.

SUBROUTINE REFINE1 (N, COST, KX, IX)

The function of this subroutine is to implement extension 2 of the convex hull algorithm (see section 3.4.4 of OR [1976]). If an improvement is found, subroutine CHANGE is called to make the change permanent. At the end of all the tests and changes, the resulting traveling salesman tour is stored in the first NUM (IX) rows of columns 1 and 2 of IHULL.

SUBROUTINE CHANGE (I1, I2, J1, J2, K1, K2, N)

The function of this subroutine is to modify the first two columns of IHOSP so that the path from I1 to I2 lying between nodes J1 and J2 in the existing graph would be placed between nodes K2 and K1.

SUBROUTINE IMPROVE (N, NBANK, SCAL, EXINDP)

The function of this subroutine is to determine and order the sets EX (j₁, j₂), j₁, j₂ = 1, ..., NBANK (see section 4.3 of OR [1976] for the description of EX (j₁, j₂)). NALTER (j₁, j₂, *) correspond to set EX (j₁, j₂), and can have at most 10 elements. The routings given by x* are saved in the first two columns of I1HOSP (150, 3). EXINDP is a logical parameter provided by the user from the keyboard that is used in
deciding between algorithm 1 and algorithm 2 (see section 4.3 of OR [1976] for the descriptions of these algorithms). If EXINDP = .T., subroutine ALTER1 is called and algorithm 1 is applied; if EXINDP = .F., subroutine ALTER2 is called and algorithm 2 is applied.

**SUBROUTINE ALTER1 (N, NBANK, SCAL)**

This subroutine corresponds to step 9 of algorithm 1 for the MTSP. The node that is to be temporarily placed in a different region is stored in LIST (1) and subroutine TEST is called for the computation of \( (\bar{x}, \bar{y}) \) and the comparison of \( z_{\min} \) with \( z(\bar{x}, \bar{y}) \) (see section 4.3 of OR [1976] for the definitions of these terms). If the test is positive (INDIC = .T.), then subroutine UPDATE is called to make the temporary change permanent. After each permanent change the solution obtained is plotted.

**SUBROUTINE ALTER2 (N, NBANK, SCAL)**

This subroutine corresponds to step 9 of algorithm 1 for the MTSP. The nodes that are to be temporarily placed in a different region are stored in LIST (1) to LIST (3) and subroutine TEST is called for the computation of \( (\bar{x}, \bar{y}) \) and the comparison of \( z_{\min} \) with \( z(\bar{x}, \bar{y}) \). If the test is positive (INDIC = .T.), then subroutine UPDATE is called to make the temporary change permanent. After each permanent change, the solution obtained is plotted. The third column of IIHOSP is used in this subroutine to determine whether the temporary exchange under consideration has been tested before or not. If it is, a duplicate test is avoided.

**SUBROUTINE TEST (I, J, N, COST, LIST, NBR, KX, INDIC)**

In this subroutine the NBR hospitals given by LIST (1) to LIST (NBR) are deleted from region J and added to region I. The change in systems
costs and expected emergency delivery costs are determined. New traveling salesman routes for the regions I and J are found (by calling subroutines CHULL1, BOUND1 and REFIN1) and their costs COST (I) and COST (J) are computed. The change in periodic delivery costs is determined by comparing COST (I) + COST (J) with TSCOST (I) + TSCOST (J). Then the total change in costs is determined. If there is a decrease in total costs, the test is successful (INDIC = .T.). If the test is not successful, the NBR hospitals given by LIST (1) to LIST (NBR) are reassigned to region J.

SUBROUTINE UPDATE (I, J, N, NBANK, COST, LIST, NBR, KX)

The function of this subroutine is to make the temporary change that led to a successful test (caused a decrease in total costs) permanent. The vector NUM and column 4 + KX of IHOSP have already been updated in subroutine TEST. In this subroutine IHULLA, TSCOST (I), TSCOST (J), EMCOST (I), EMCOST (J), NBLAD (I), NBLAD (J) and IIHOSP are updated.

SUBROUTINE PRINTS (N, NBANK, KX, NX, ISTOP, IUNIT)

The function of this subroutine is to provide an output in the format of the tables presented in section 4.4 of OR [1976]. ISTOP and IUNIT contain the constraints on the vehicles (maximum number of stops per vehicle and maximum number of units per vehicle) if a MVDP solution is being provided. KX selects the column of IHULLA that is to be printed and the appropriate title for the hospital assignments. "NX" is a 0, 1 variable showing whether the solution obtained is a MTSP solution or a MVDP solution. At the time this subroutine is entered the information contained in NUM (50), IBLADTR (50) and IBTR (5) should correspond to the routing that is to be printed (i.e., IBTR (1) contains the number of vehicles in
region 1; NUM (1) to NUM (IBTR (1)) contain the number of stops for the
vehicles in region 1; IBLADTR (1) to IBLADTR (IBTR (1)) contain the number
of units carried by vehicles in region 1; IHULLA (1, 1+KX) to
IHULLA (NUM (IBTR (1)), KX+1) contain the circuit corresponding to the
first vehicle in region 1; and so on).

SUBROUTINE PLOTING (SCAL, KX, NBANK, NTRUCK)

The function of this subroutine is to plot (using the CALCOMP plotting
package) the solution obtained. Subroutine PLOTING uses the information
contained in either column of IHULL (150, 2) for plotting. "KX" is a
variable allowing the user to plot either the first or the second column
of IHULLA, and to select the appropriate title for the graph. "SCAL" con-
tains scaling information; it should be 70 for the wide plotter and 220
for the small plotter. Arrays w, z contain the coordinates for the Chicago
shoreline. The number of active vehicles in the routing to be plotted is
stored in NTRUCK. At the time subroutine PLOTING is entered, the informa-
tion contained in NUM (1) to NUM (NTRUCK) should correspond to the routing
that is to be plotted (i.e., IHULLA (1, 1+KX) to IHULLA (NUM (1), 2+KX)
contains the circuit corresponding to the first vehicle; IHULLA (NUM (1)+1,
1+KX) to IHULLA (NUM (1)+NUM (2), 1+KX) contains the circuit corresponding
to the second vehicle; and so on).

INTEGER FUNCTION DATX (I, J)

This function computes the address of the entry (I, J) of the distance
matrix in the extended core storage.
References


Appendix

The following pages contain the actual Fortran codings of the subroutines in the library BTAP and of the main program BTP.
PROGRAM ATTN, INPUT, OUTPUT, TAPE5, TAPE6, TAPE99, KEYBRO, CONSO
1TAPE7, TAPE1=KEYBRO, TAPE2=CONSO
COMMVC, IMBK(5), 4KMB(5), TVHLL(300, 3)
DIMENSION PAIR(1), KTYPE(1), ITYPE(1), CONST(2), JTYPE(2)
LOGICAL PVRD, VPRTIMES
DATA JTYPE(1,1)
DATA KTYPE(1,3,3,3)
DATA ITYPE(2,2,2,2)
10 303 FORMAT(2X, *NUMBER OF BANKS AT MOST 5*)
20 310 FORMAT(2X, *IDENTIFICATION OF BANKS*/2X, *PROVIDE AS MANY NUMBERS*/
30 105 NUMBER OF BANKS*/
40 302 FORMAT(/5X, *ALLOC1 IS ALLOCATION BASED ON EMERGENCY COSTS*
50 12X, *ALLOC2 IS ALLOCATION BASED ON ROUTING COSTS*/
60 22X, *FOUND TRUE OR FALSE*/
70 15X, *PLOT FOR ALLOC1, PLOT FOR ALLOC2, ALLOC1, ALLOC2*)
80 303 FORMAT(2X, *SCALE OF PLOT*/2X,
90 12X, *FOR SMALL PLOTTER, 70 FOR LARGE PLOTTER*/
100 314 FORMAT(/5X, *ALTER1 IMPROVES ALLOCATIONS BY TESTING CANDIDATES 1
110 *1 AT A TIME*/5X, *ALTER2 IMPROVES ALLOCATIONS BY TESTING CANDIDATES 2
120 2UP T) 3 AT A TIME*/5X, *CHOOSE ONE OR NEITHER OF THE FOLLOWING*
130 3/2X, *CONTINUE WITH ALTER1, CONTINUE WITH ALTER2*/
140 318 FORMAT(/5X, *ARE TYPE CONSTRAINTS ON THE VEHICLES, IF SO WRITE*/
150 *15X, *T AND A MULTIPLE VEHICLE SOLUTION WILL BE PROVIDED*/)
160 316 FORMAT(/5X, *GIVE MAXIMUM NUMBER OF STOPS AND VEHICLE CAPACITY*/)
170 25 311 FORMAT(/2X, *ARE INTERMEDIATE EXECUTION TIMES DESIRED*/
180 12X, *IF SO WRITE T AND THEY WILL BE PROVIDED*/
190 311 FORMAT(/2X, *COCPLATIONS ARE NOT ALLOWED, TIME IS */F8.3*)
200 311 FORMAT(/2X, *ROUTINGS BASED ON ALLOC1 ARE COMPUTED, TIME IS */F8.3)
210 311 FORMAT(/2X, *ROUTINGS BASED ON ALLOC2 ARE COMPUTED, TIME IS */F8.3)
220 311 FORMAT(/2X, *ALL INDEPENDENT EXCHANGES ARE TESTED, TIME IS */F8.3)
230 311 FORMAT(/2X, *ALL MULTIPLE VEHICLE SOLUTION IS COMPUTED, TIME IS */F8.3*)
240 WRITE(1, *11)
250 CALL F=FRM1(TIMES, 1, 1)
260 WRITE(1, 300)
270 CALL F=FRM1(NBANK, 2, 1)
280 WRITE(1, 311)
290 CALL F=FRM1(NBANK, 1, 1)
300 WRITE(1, 312)
310 CALL F=FRM1(4, KTYPE, 1, 1)
320 WRITE(1, 313)
330 CALL F=FRM1(4, SCAL, 1, 1)
340 CALL RAN1(N)
350 CALL DISP1(N)
360 X=SEQUENTIAL)
370 IF (TIMES) WRITE(1, *111) X
380 IF (NOT, PARM(3)) GO TO 20
390 CALL ALLOC1(N, NBANK)
400 CALL TRAVEL(N, NBANK, 0)
410 IF (PARM(11)) CALL FLOATING(SCAL, N, NBANK, NBANK)
420 X=SEQUENTIAL)
430 IF (TIMES) WRITE(1, *112) X
440 IF (NOT, PARM(4)) GO TO 40
450 CALL ALLOC2(N, NBANK)
460 CALL TRAVEL(N, NBANK, 1)
470
IF (PARAM(2)) CALL PLOTOING(SCAL,1,NBANK,NBANK)
X=SECOND(X)
IF (TIMES) WRITE(2,313) X
40 PPM(1)=PARAM(2)=F.
60 IF (.NOT. PARAM(3)) .AND. (.NOT. PARAM(4)) KK=-2
   IF (.NOT. PARAM(3)) .OR. (.NOT. PARAM(4)) GO TO 41
   WRITE (2,314)
   CALL FREFM(2,PARAM,KTYPE,1,2)
   IF (PARAM(1)) CALL IMPROVE(NBANK,SCAL,T)
   IF (PARAM(2)) CALL IMPROVE(NBANK,SCAL,F)
   X=SECOND(X)
   IF (PARAM(1) .AND. TIMES) WRITE(2,314) X
   IF (PARAM(2) .AND. TIMES) WRITE(2,315) X
70 41 WRITE(2,325)
   CALL FREFM(1,VDP,1,2)
   IF (.NOT. VDP) STCF
   WRITE(2,316)
   CALL FREFM(2,CONST,JTYPE,1,2)
75 IF (PARAM(4)) KK=1
   IF (PARAM(1) .OR. PARAM(2)) KK=-1
   CALL DISPATCH(NBANK,KK,CONST(1),CONST(2),SCAL)
   X=SECOND(X)
80 IF (TIMES) WRITE(2,316) X
END
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUBROUTINE DISMAT1(N)</td>
<td>Statement to start subroutine</td>
</tr>
<tr>
<td>2</td>
<td>COMMON /F/X(150),Y(150),ALFA(150),IBLAD(150)</td>
<td>COMMON block definition</td>
</tr>
<tr>
<td>3</td>
<td>INTEGER DATX</td>
<td>Declaration of variable</td>
</tr>
<tr>
<td>4</td>
<td>COMMON/CH/DMT1(150),DMT2(150)</td>
<td>COMMON block definition</td>
</tr>
<tr>
<td>5</td>
<td>DO 16 I=1,N</td>
<td>Loop statement</td>
</tr>
<tr>
<td>6</td>
<td>DO 15 J=1,N</td>
<td>Loop statement</td>
</tr>
<tr>
<td>7</td>
<td>DUM=(Y(I)-Y(J))**2+(X(I)-X(J))**2</td>
<td>Calculation of DUM</td>
</tr>
<tr>
<td>8</td>
<td>DMT1(J)=SORT(DUM)</td>
<td>Sorting DUM</td>
</tr>
<tr>
<td>15</td>
<td>CONTINUE</td>
<td>Loop continue statement</td>
</tr>
<tr>
<td>16</td>
<td>CALL WRITEC(DMT1+DATX(1)+I)*N</td>
<td>Call to WRITEC function</td>
</tr>
<tr>
<td>17</td>
<td>RETURN</td>
<td>Program return</td>
</tr>
<tr>
<td>18</td>
<td>END</td>
<td>End of program</td>
</tr>
</tbody>
</table>
SUBROUTINE ALLOC1(N,NBANK)
COMMON/A/IHOSP(150+5)/E/X(150+y(150)+ALFA(150)+IBLAD(150)
COMMON/K/TICST(5)+FMCST(5)+NBLAD(5)+SYCOST(120))
INTEGER NAX
COMMON/NX/DMT1(150)+DMT2(150)/C/IRANK(5)+NUM(50)+IMULL(300,3)

100 FORMAT(5x,1d15)
DO 1 I=1,NBANK
EMCOST(I)=0
NBLAD(I)=0
NUM(I)=0
1 CONTINUE
DO 5 I=1,N
II=IRANK(I)
CALL READEC(DMT1+DATX(1+I)*N)
DUM=DMT1(I)
NBR=1
DO 4 J=2,NBANK
K=IRANK(J)
IF (DMT1(K)-GE,DUM) GO TO 4
DUM=DMT1(K)
NBR=J
4 CONTINUE
IHOSP(I+4)=NBR
NUM(NBR)=NUM(NBR)+1
EMCOST(NBR)=EMCOST(NBR)+ALFA(I)*DUM
NBLAD(NBR)=NBLAD(NBR)+IBLAD(I)
5 CONTINUE
WRITE(7+100) (IHOSP(I+4)*I=1,N)
RETURN
END
EMCOST(IR) = EMCOST(IB) + ALFA(I) * DMTX
39 NBLAD(IB) = NBLAD(IB) + 1
40 CONTINUE
        WRITE(7,100) (IHOSP(I,5)*I=1,N)
60 RETURN
END
SUBROUTINE TRAVEL(N,NBANK,KX)
COMMON/C/HANK(51),NUM(50),IHULL(300,3)/;IHULLA(150,2)
COMMON/K/TSCOST(5),EMCOST(5),NBLAD(5),SCOST(120)
ISTAR=0
DO 5 II=1,NBANK
      MEM=NUM(II)
      CALL CHULL(N,IC,KX,II)
      CALL BOUND(N,IC,TSCOST,KX,II)
      IF (NUM(II) .GE. 5) CALL REFINE1(N,TSCOST,KX,II)
      DO 6 I=1,5
      CALL HULLA(ISTAR*I+KX+1)=IHULL(I+1)
      ISTAR=ISTAR+NUM(II)
      5 CONTINUE
      CALL PRINTS(N,NBANK,KX)
RETURN
END
SUBROUTINE DISPAC(N, NBANK, KK, STOPS, UNITS, SCAL)
COMMON/CMP/NPT1(15), DMTZ(150), A/IHOSP(150), B/IHULLA(150), I
COMMON/F/I/RADTR(50), I/RTH(5), I/P(150), I/TETA(150), I/IPOL(100), I/NUMTR(50)
COMMON/K/X(150), Y(150), ALFA(150), I/BLAD(150)

10 FORMAT(5X, 10S, 15X, 'THE FOLLOWING ALLOCATION IS SUPPLIED EXTERNALLY/*')

20 FORMAT(3X, 'NHOSPITAL', 14X, 'RANK', 13X)

30 FORMAT(3X, 'N-X', 6X, 'N-RAK', 13X)

KK = X

IF (KK .LT. 0) KK = 0
IF (KK .NE. -2) GO TO 3
READ (7, 100) (IHOSP(I), I = 1, N)
WRITE (6, 200) (I, IHOSP(I), I = 1, N)
WRITE (6, 210) (I, IHOSP(I), I = 1, N)
DO 5 IR = 1, N
EMCOST(IR) = I/RADTR(IR) = 0
25 CALL HEADEC(DMT, DATX(1, INUM), I)
DO 4 I = 1, N
IF (IHMSP(I), I = 4, KK), .NE., IB) GO TO 4
EMCOST(IR) = EMCOST(IR) + DMT(I)
30 IF (INUM .EQ. I) GO TO 4
R(I) = DMT(I)
IF (X(I), I = 4, X(INUM)) GO TO 1
40 TETA(I) = TETA(I) + I/2
35 IF (Y(I), I = 4, Y(INUM)) GO TO 1
40 TETA(I) = TETA(I) + 3.141593
45 GO TO 4
1 IF (Y(I), I = 6, Y(INUM)) TETA(I) = 3.141593/2
IF (Y(I), I = 7, Y(INUM)) TETA(I) = (3.141593)/2
GO TO 4
2 IF (X(I), I = 6, X(INUM)) TETA(I) = 3.141593
4 CONTINUE

5 CONTINUE

ISTAR = JSTAR = 1
DO 5 IB = 1, N
5 CONTINUE
7 CONTINUE
10 IFPOL(NS)=J
20 GO TO 15
8 JEND=NS+1
60 DO 9 JJ=1,JEND
9 IFPOL(NS+1-JJ)=IFPOL(NS-JJ)
15 CONTINUE
65 WRITE(*,100)(IFPOL(J),J=1,NS)
70 IBLD=NM=0
75 DO 30 J=1,NS
80 I=IFPOL(J)
85 IF(IPERDAY=IBLAD(J)/260.0
90 IF((IBLD+IPERDAY +GT, UNITS), OR (NM+1, GT, STOP)) GO TO 25
30 CONTINUE
25 NTR=NTR+I
85 WRITE(*,100) NTR,(NUM(J),J=1,NTR)
90 DO 40 J=1,NTR
95 IHOSP(I+K-KK)=NTR
40 CONTINUE
90 CALL CHULL1(NS,IC+1-KK,J)
95 CALL BOUND1(NS,IC,COST+1-KK,J)
100 IF(NUM(J), RE, 6) CALL REFIN1(NS,COST+1-KK,J)
105 NS=NUM(J)
110 DO 41 I=1,NS
115 IHULLA(I,STAR+I-KK+1)=IHULLA(I,1)
120 TSCOST(IN)=TSCOST(IN)+COST(J)
125 NUMT(I,STAR+J)=NUM(J)
130 IBLMTJ(I,STAR+J)=IBLMTJ(J)
135 ISTAR=ISTAR+NUM(J)
40 CONTINUE
140 JSTAR=ISTAR+NTR
150 CONTINUE
50 DO 55 J=1,JSTAR
55 NUM(J)=NUMTR(J)
60 CONTINUE
65 CALL PRINTS(NS,NANK,KK,2,I,J)
75 CALL PLOTTIN(SCAL,XX,NBANK,JSTAR)
80 END
SUBROUTINE CHULL1(N, IC*, KX, IX)
COMMON/A, IHOSP(150, J), E/X(150), Y(150), ALFA(150), IBLAD(150)
COMMON/C, IRANK(5), NUM(50), IHULL(300, 3)

IC=0
DO 1 I=1, N
IF (IHOSP(I, 4+KX) .NE. IX) GO TO 2
1 CONTINUE
STOP
2 BEST=X(I)
MEND=I
XLAST=X(I)
MSTAR=I
DO 5 S=1, N
5 IHOSP(I, 3)=0
ISTART=I+1
DO 10 J=ISTART+N
IF (IHOSP(J, 4+KX) .NE. IX) GO TO 10
IF (X(J) .LE. XLAST) GO TO 8
MSTAR=J
BEST=X(J)
10 CONTINUE
8 IF (X(J) .LE. XLAST) GO TO 10
MEND=J
XLAST=X(J)
10 CONTINUE
75 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)) .LT. 16.17
16 IF (Y(I) .LT. Y(M)) GO TO 20
IF (Y(I) .GT. Y(M)) SLOPE=0.999
IF (Y(I) .LT. Y(M)) SLOPE=-0.999
GO TO 18
17 SLOPE=(Y(I)-Y(M))/(X(I)-X(M))
18 IF (SLOPE .LE. REST) GO TO 20
BEST=SLOPE
MNEXT=I
40 CONTINUE
IHOSP(MNEXT, 3)=IC+2
CAL_PFIND(0, IC+3, MNEXT, 0+KX, IX)
IHULL(IC, 1)=M
IHULL(IC, 2)=MNEXT
M=MNEXT
45 IF (M .LT. 0, END) 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)) .LT. 15.3
23 IF (Y(I) .LT. Y(M)) SLOPE=0.999
IF (Y(I) .LE. Y(M)) SLOPE=-0.999
GO TO 25
24 SLOPE=(Y(I)-Y(M))/(X(I)-X(M))
25 IF (SLOPE .LE. REST) GO TO 30
30 CONTINUE
IC+1
SURROUTINE  CHULLI  NORMAL

BEST=SLOPE  1290
WEXT=I  1300

30 CONTINUE  1310
HOSP(MNEXT,3)=IC*2  1320
CALL PFIND(h,IC,M,MNEXT,0,KK,IX)  1330
IMULL(IC,1)=M  1340
IMULL(IC,2)=MEXT  1350
M=MEXT  1360
IF (M,IF, KSTAR) GO TO 21  1370
RETURN  1380
END  1390
SUBROUTINE PFIN(N,NSIZE,K1,K2,KPOINT,IX)
COMMON/AV/ICSP(150),IA/DANDT(600,4),IADRES(600)
COMMON/AV/XT(150),Y(150),X(150),Y(150)
INTEGER/NSIZE(150),X(150),Y(150),0RTA(150),1BLAD(150)
INTEGER/NSIZE(150),X(150),Y(150),0RTA(150),1BLAD(150)
COMMON/CM/DMT1(150),DMT2(150)
COMMON/CM/DMT1(150),DMT2(150)
BE=1000000
BE=1000000
CALL REALC(DMT2+DMT1(I),K2,N)
CALL REALC(DMT2+DMT1(I),K2,N)
CALL REALC(DMT2+DMT1(I),K2,N)
CALL REALC(DMT2+DMT1(I),K2,N)
10 I=1,N
I=1,N
IF (IMSP(I,4),NE.,IX) GO TO 20
IF (IMSP(I,4),NE.,IX) GO TO 20
IF (IMSP(I,4),NE.,IX) GO TO 20
IF (IMSP(I,4),NE.,IX) GO TO 20
19 DO 20.I=1,N
DO 20.I=1,N
IF (IMSP(I,3).GE.,1) GO TO 20
IF (IMSP(I,3).GE.,1) GO TO 20
IF (IMSP(I,3).GE.,1) GO TO 20
IF (IMSP(I,3).GE.,1) GO TO 20
20 CONTINUE
CONTINUE
GO TO 25
GO TO 25
ENTRY AFIND
ENTRY AFIND
BE=1000000
BE=1000000
CALL REALC(DMT2+DMT1(I),K2,N)
CALL REALC(DMT2+DMT1(I),K2,N)
CALL REALC(DMT2+DMT1(I),K2,N)
CALL REALC(DMT2+DMT1(I),K2,N)
70 I=1,N
I=1,N
IF (IMSP(I,3).GE.,1) GO TO 10
IF (IMSP(I,3).GE.,1) GO TO 10
IF (IMSP(I,3).GE.,1) GO TO 10
IF (IMSP(I,3).GE.,1) GO TO 10
12 DO 10.I=1,N
DO 10.I=1,N
IF (IMSP(I,3).GE.,1) GO TO 10
IF (IMSP(I,3).GE.,1) GO TO 10
IF (IMSP(I,3).GE.,1) GO TO 10
IF (IMSP(I,3).GE.,1) GO TO 10
30 CONTINUE
CONTINUE
10 CONTINUE
CONTINUE
25 NSIZE=NSIZE+1
NSIZE=NSIZE+1
CANDID(NSIZE)=BE
CANDID(NSIZE)=BE
CANDID(NSIZE)=BE
CANDID(NSIZE)=BE
40 CANDID(NSIZE,1)=K1
CANDID(NSIZE,1)=K1
CANDID(NSIZE,2)=K2
CANDID(NSIZE,2)=K2
CANDID(NSIZE,3)=K1
CANDID(NSIZE,3)=K1
CANDID(NSIZE,4)=K2
CANDID(NSIZE,4)=K2
CALL ADPFS(NSIZE,KPOINT)
CALL ADPFS(NSIZE,KPOINT)
RETURN
RETURN
END
END
SUBROUTINE ADRES(NSIZE,KPOINT)
COMMON/A/INOSP(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-1
DO 9 J=1,JEND
9 IADRES(NSIZE+1-J)=IADRES(NSIZE-J)
15 IADRES(1)=NSIZE
RETURN
END
SUBROUTINE ROUND1(N, IC, COST, KX, IX)

COMMON/A, IHOSP(150, 5), O/CANDIN(600, 4), IADRES(600)

INTEGER DATX

COMMON/CH, DMT(150), DMT2(150), C/IBANK(5), NUM(5), IHULL(300, 3)

DIMENSION COST(9)

KPOINT=0

NSIZE=IC

ICOUNT=IC

DO 1 I=1, 300

1 IMULL(I, 3)=0

MM=NUM(I, X)=IC

IF (MM * EQ. 0) GO TO 31

DO 30 J=1, MM

30 IF (J * EQ. 0) GO TO 11

KPOINT=KPOINT+1

ISTAR=IADRES(KPOINT)

IBAR=CANDIN(ISTAR+2)

K1=CANDIN(ISTAR+3)

K2=CANDIN(ISTAR+4)

IF (IHOSP(IBAR+3) * EQ. 0) GO TO 25

CALL PFIN1(IN, NSIZE, K1, K2, KPOINT, KX, IX)

GO TO 11

25 DO 26 I=1, IC

IF (IMULL(I, 1) * EQ. 0) K1 IMULL(I, 3)=1

26 CONTINUE

ICOUNT=ICOUNT+1

IMULL(IBAR+3)=ICOUNT

IC =1+1

IMULL(IC, 1)=K1

IMULL(IC, 2)=IBAR

CALL PFIN1(IN, NSIZE, K1, IBAR, KPOINT, KX, IX)

IC=IC+1

IMULL(IC, 1)=IBAR

IMULL(IC, 2)=K2

CALL PFIN1(IN, NSIZE, IBAR, K2, KPOINT, KX, IX)

30 CONTINUE

COST(IX)=0

DO 40 I=1, IC

IF (IMULL(I, 3) * EQ. 1) GO TO 40

30 CONTINUE

40 IHOSP(ISTAR+2)=IMULL(I, 2)

IE=IMULL(I, 2)

IHOSP(IEND, 1)=IMULL(I, 1)

CALL MEANE(CMTX, DATX, (ISTAR, IEND), 1)

COST(IX)=COST(IX)+CMTX

40 CONTINUE

41 DON I=1, IX

43 IPREV=I

IEXT=IHOSP(I+2)

IEND=NUM(I, X)

DO 45 I=1, IEND

45 IMULL(I, 1)=IPREV

IMULL(I, 2)=IEXT

44 CONTINUE

STOP
SUBROUTINE REFIN1(N,COST,KK,IX)
COMMON/C,IRANK(5),/IJM(50),/HMULL(300,3)
INTEGER DATX
COMPCM/C,C,M1(I),(150),/OMT2(150)/,A/NIOSP(I,(150,5)
DIMENSION COST(9)
DO 11 J=1,2
K=3-JJ
11 I=1,N
IF (IHOSP(I,KK)<IX) GO TO 12
10 I=1
II=I
J=J+IHOSP(I,1)
DO 6 IN=1,KK
6 I=I+IHOSP(I,2)
J=J+IHOSP(I,2)
CALL HFADEC(DMT2,CATX(1,J)*N)
CALL HFADEC(DMT1,CATX(1,J)*N)
DIF1=OMT1(I)+OMT2(I)-OMT1(J)
ISTAR=J
IEN=NUM(IX)-K-1
20 IN=1,ITEND
K=ISTAR
CALL HFADEC(DMT2,CATX(1,K)*N)
CALL HFADEC(DMT1,CATX(1,K)*N)
DIF2=OMT1(I)+OMT2(I)-OMT1(K)
IF (DIF2<GT, DIF2) GO TO 7
30 CONTINUE
GO TO 12
35 CONTINUE
CALL CHANGE(I1*I2,J1*I2,K1*K2,N)
30 CONTINUE
10 CONTINUE
COST(IX)=0
DO 30 J=1,N
35 IF (IHOSP(J,KK)<IX) GO TO 14
30 CONTINUE
STOP
14 IPREV=J
IEN=IHOSP(J,2)
END=NUM(IX)
15 CONTINUE
STOP
40 RETURN
END
SUBROUTINE CHANGE(I1,I2,J1,J2,K1,K2,N)
COMMON /X/ IMOSP(156,5)
IMOSP(J1,K2)=J2
IMOSP(J2,K1)=J1
IMOSP(K1,J2)=I2
IMOSP(K2,J1)=I1
IMOSP(I1,J1)=K2
INEXT=I2
IPREV=K1
10 IMOSP(INEXT,2)=IMOSP(INEXT,1)
IMOSP(INEXT,1)=IPREV
IF (INEXT .EQ. I1) GO TO 15
IPREV=INEXT
INEXT=IMOSP(INEXT,2)
GO TO 10
15 CONTINUE
RETURN
END
SUBROUTINE IMPROVE(N,NRANK,SCAL,EXINDP)
INTEGER DAX
LOGICAL FXINDP
COMMON/F/IALTER(5,5,10),NALTER(5,5),IHOST(150*3),IDUM(150)
COMMON/COMD1(150),DM1Z(150)/A/IHOST(150,5)
COMMON/C/IANK(5),NUM(50),IULL(300*3),HULLA(150,2)
110 FORMAT(1HI//5X*LIST OF POSSIBLE EXCHANGES*)
120 FORMAT(1/5X*HOSPITALS IN GROUP**13** TO BE TRIED IN GROUP**
1,13/)
10 FORMAT(2X,10I5)
DO 1 I=1,NRANK
DO 1 J=1,NRANK
IF(IHOST(I,J))=N
DO 1 K=1,10
IALTER(I,J,K)=0
1 CONTINUE
DO 10 I=1,N
IHOST(I,4)=1
IF(IHOST(I,4)) .EQ. IHOST(I,5) GO TO 10
II=IHOST(I,4)
JJ=IHOST(I,5)
IB=IANK(JJ)
CALL READEC(DMT2,DAX(1+IB)+N)
IB=IANK(JJ)
CALL READEC(DMT1,DAX(1+IB)+N)
IALTER(I,J,J)=IALTER(I,J,J)+1
IF(ALTFK(I,J,J) .GT. 10) IALTER(I,J,J)=10
JN=IALTER(I,J,J)
JEND=NN-1
30 IF(NM .EQ. 1) GO TO 7
DO 6 J=1,JEND
JX=IALTER(I,J,J)
IF(DMT1(I,J) .LE. DMT1(JX)-DMT2(JX)) GO TO 8
6 CONTINUE
35 IF(NM .LT. 10) IALTER(I,J,J,NM)=1
JX=IALTER(I,J,J,NM)
IF(DMT1(I,J) .LE. DMT1(JX)-DMT2(JX)) IALTER(I,J,J,NM)=1
GO TO 10
8 JEND=NN-1
40 DO 9 J=1,JEND
IALTER(I,J,J,NM+1)=IALTER(I,J,J,NM)-J
9 CONTINUE
7 IALTER(I,J,J,NM+JEND)=1
10 CONTINUE
WRITE(6,110)
DO 12 I=1,NRANK
DO 12 J=1,NRANK
IF(ALTFK(I,J)) .EQ. 0 GO TO 11
WRITE(6,112) J+I
11 CONTINUE
12 CONTINUE
DO 14 I=1,N
IHOST(I,4)=IHOST(I,5)
14 CONTINUE
<table>
<thead>
<tr>
<th>SUBROUTINE IMPROVE NORMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1MOSP(I,1)=I1MOSP(I-1)</td>
</tr>
<tr>
<td>I1MOSP(I,2)=I1MOSP(I,2)</td>
</tr>
<tr>
<td>IHULLA(I,1)=IHULLA(I,2)</td>
</tr>
</tbody>
</table>

15 CONTINUE

60 IF (EX.INOP) CALL ALTER1(N,NBANK*SCAL)
    IF (.NOT. EX.INOP) CALL ALTER2(N,NBANK*SCAL)
    CALL PRINTS(N,NRANK,-1,1)
    RETURN
    END
SUBROUTINE ALTER1(N,NBANK,SCAL)
INCLUDE 'INDIC'
COMMON/F/ALTER(5*5*10),NALTERT(5*51),IIHOSP(150*3),IDUM(150)

DIMENSION COST(9),LIST(5)

IEND=NBANK+1

DO 30 I=1:IEND

JSTART=I+1

DO 29 J=JSTART,NBANK

ITER=MAX0(NALTER(I,J),NALTER(J,I))

IF (ITER .LT. 0) GO TO 29

DO 25 K1=1,ITER

IF (K1 .GT. NALTER(I,J)) GO TO 20

LIST(I)=IALTER(I,J,K1)

CALL TEST(I,J,N,COST,LIST,NBANK+NBANK)

IF (INDIC) CALL UPDATE(I,J,N,NBANK+COST,LIST,1+1)

IF (INDIC) CALL PLOTING(SCAL+1,NBANK+NBANK)

25 CONTINUE

29 CONTINUE

20 CONTINUE

IF (INDIC) CALL UPDATE(J,I,N,NBANK+COST,LIST,1+1)

IF (INDIC) CALL PLOTING(SCAL+1,NBANK+NBANK)

RETURN

END
SUBROUTINE ALTER2(N,NBANK,SCAL)
LOGICAL INDIC
COMMON/I,ALTER(5,5),NALTER(5,5),IIHOSP(150,3),IDUM(150)
END=NRANK=1
DO 30 J=1,NRANK
JSTAR=I+1
DO 25 JF,JSTAR,NBANK
ITER=MAXO(NALTER(I,J),NALTER(J,I))
IF (ITER .EQ. 0) GO TO 29
DO 25 K1=1,ITER
IZ=I
JZ=J
25 CONTINUE
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,N,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
75 LIST(2)=J2
NBR=2
CALL TEST(IZ,JZ,N,COST,LIST,NBR,KX,INDIC)
IF (INDIC) GO TO 19
J3=IIHOSP(J1,2)
16 IF (IIHOSP(J3,3) .EQ. J1) GO TO 17
IIHOSP(J3,3)=J1
NBR=2
LIST(2)=J3
CALL TEST(IZ,JZ,N,COST,LIST,NBR,KX,INDIC)
17 LIST(2)=J2
LIST(3)=J3
NBR=3
CALL TEST(IZ,JZ,N,COST,LIST,NBR,KX,INDIC)
IF (INDIC) GO TO 19
GO TO 20
19 CALL UPDATE(IZ,JZ,N,NBANK,COST,LIST,NBR,KX)
CALL PLOTSING(SCAL,-1,NBANK,NRANK)
26 IF (IZ .EQ. J) GO TO 25
IZ=J
JZ=1
KX=0
GO TO 15
25 CONTINUE
29 CONTINUE
30 RETURN
END
SUBROUTINE TEST(I,J,N,COST,LIST,NBR,K,KX,INDIC)

INTEGER NAX

LOGICAL INDIC

COMMON/F/ALTER(5,5,10),NALTER(5,5),IHHOSP(150,3),IDUM(150)
COMMON/K/TSCOST(5),MCOST(5),NBLAD(5),SYCOST(120)
COMMON/C/WM1(150),DM2(150),CI/BANK(5),NUM(50),IHELL(300,3)
COMMON/F/X(150),Y(150),ALFA(150),IRLAD(150),IHHOSP(150,5)
DIMENSION COST(5),LIST(5)

100 FORMAT(11H1/16X*TEST DATA*1/)

110 FORMAT(11X*ORIGINAL ROUTING COST FOR GROUP*I3=***,F9.3/)
12X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3/)
120 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3///)
130 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3/)
140 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3///)
153 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3/)
160 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3///)
170 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3/)
180 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3///)
190 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3/)
200 FORMAT(11X*REVISED ROUTING COST FOR GROUP*I3=***,F9.3///)

INDIC=1.
NUM(I)=NUM(I)+NBR
NUM(J)=NUM(J)+NBR
DO 20 II=1,NBR
JJ=LIST(II)
1 IMOSP(JJ,KK)=II
CALL CHUI1(N*IC+KK,II)
CALL BOUND1(N*IC+K,II)
IF (NUM(J) .GE. 5) CALL REFIN1(N*IC+K,II)
NJ=NUM(J)
DO 5 II=1,NJ
5 ITDUM(II)=IHELL(II)
CALL CHULL1(N*IC+KK,J)
CALL BOUND1(N*IC+K,J)
IF (NUM(J) .GE. 5) CALL REFIN1(N*IC+K,J)
NJ=NUM(J)
DO 10 II=1,NJ
10 DIF=SDIF=ALFA(II)*(DM1(II)-DM2(II))
10 CONTINUE

WRITE(6,100)
WRITE(6,100,II=1,NBR)
WRITE(6,200,I,J,,LIST(II),II=1,NBR)
WRITE(6,200)
WRITE(6,153)(IDUM(II),II=1,NJ)
WRITE(6,153)
WRITE(6,110)(I+TSCOST(I),II=1,NJ)
WRITE(6,110)
WRITE(6,120)SDIF
WRITE(6,130)SDIF

END
SUBROUTINE TEST  NORMAL

WRITE(*,140) DIF
IF (INDIC) RETURN
NUM(I)=NUM(I)+NBR
NUM(J)=NUM(J)+NBR
DO 15 II=1,NBR
   J=LIST(II)
  15 IHOSP(JJ+XX)=J
RETURN
END
SUBROUTINES

SUBROUTINE PRINTS(N, NBANK, KK, ISTOP, IUNIT)
COMMON/C/IANK(5), NUM(50), IHULLA(150), BURA(820)
COMMON/K/SCOST(5), ECGST(5), BLAG(5), SCOST(120)
COMMON/F/IRLADTR(50), IBTR(50), ITR(120)

110 FORMAT(1X, 6X, OPTIMAL ALLOCATION AND ROUTING, 6X, ALLOCATION BASE)
111 FORMAT(1X, 3X, BANK, 12X, IDENTIFICATION, HOSPITAL, 14, ROUTING)

109 FORMAT(2X, 5F10.3)
110 FORMAT(1X, 10X, 1015)
111 FORMAT(1X, 3X, EMERGENCY COST, 6X, ROUTINE DELIVERY COST)

120 FORMAT(1X, 5X, OPTIMAL ALLOCATION AND ROUTING, 6X, ALLOCATION BASE)
112 FORMAT(1X, 3X, SYSTEM COST, 3X, TOTAL COST)
113 FORMAT(1X, 3X, SYSTEM COST, 3X, TOTAL COST)
114 FORMAT(1X, 3X, SYSTEM COST, 3X, TOTAL COST)

125 FORMAT(1X, 5X, NUMBERS OF HOSPITALS IN THE SYSTEM)
126 FORMAT(1X, 5X, AMOUNT OF BLOOD USED IN THE SYSTEM)

210 FORMAT(H1/4AX, SINGLE VEHICLE SOLUTION)
220 FORMAT(H1/4AX, MULTI VEHICLE SOLUTION, 5X, CONSTRAINTS)

10 MAXIMUM NUMBER OF STOPS = 21 + MAXIMUM NUMBER OF UNITS = 14

220 FORMAT(1X, TRUCK NO, 13X, NUMBER OF STOPS, 13X, NUMBER OF UNITS)

220 FORMAT(6X, TRUCK NO, 13X, NUMBER OF STOPS, 13X, NUMBER OF UNITS)

KK = KK

IF (KK = 0) THEN
WRITE(6, 210)
END IF

IF (KK = 1) THEN
WRITE(6, 220)
IF (KK = 1) THEN
WRITE(6, 230)
END IF

END IF

DO 6 II = 1, NBANK
WRITE(6, 111) II, IBANK
END DO

WRITE(6, 112) ISTOP, IUNIT
END IF

NTR = ITR
END DO 3 J = 1, NTR
WRITE(6, 130) J, NUM(ISTAR + J), IRLADTR(ISTAR + J)
JEND = NUM(ISTAR + J)
WRITE(6, 130) (HULLA(ISTAR + KK), I = 1, JEND)

ISTAR = ISTAR + JEND
CONTINUE

MEMBER = ISTAR, IPREV = NTR + 1
IPREV = ISTAR
WRITE(6, 125) MEMBER
JSTAR = JSTAR + NTR
GO TO 5

WRITE(6, 130) (HULLA(ISTAR + KK), I = 1, MEMBER)
WRITE(6, 130) NNUM(II)

WRITE(6, 130) (HULLA(ISTAR + I), I = 1, MEMBER)
WRITE(6, 130) NNUM(II)

5 WRITE(6, 130) NNUM(II)
WRITE(6, 130) RCOST(RCOST + I)
SCOST = SCOST + MEMBER
END IF

CONTINUE
SUBROUTINE PLOTTING

COMMON/E/XX(150),YY(150),ALFA(150),IBLAD(150)/B/HULLA(150,2)
COMMON/E/XX(150),YY(150),ALFA(150),IBLAD(150)/

DIMENSION W(130),Z(130),H(600)

DATA (W(I),I=1,126)/1479,1437,1428,1423,1389,1343,1336,1329,
*1280,1285,1265,1252,1244,1232,1217,1204,1192,1177,1159,
*1080,996,940,900,895,895./

DATA (Z(I),I=1,538)/570,556,549,606,696,758,792,900,956,
*1020,1106,1112,1116,1117,1235,1288,1313,1400,1465,1537,
*1826,1930,1998,2078,2188,2218./

KK=KK

IF (KK LT 0) KK=0
CALL NAMPLT

CALL YLIMIT(35.0)
CALL XLIMIT(30.0)
W(27)=Z(27)=10.0
W(28)=Z(28)=SCAL
CALL LINE(W,7,26,10+3)
DO 2 I=1,NBANK
I=IPANK(I)
X(I)=XX(I)
Y(I)=YY(I)
2 CONTINUE

X(NBANK+1)=Y(NBANK+1)=10.0
X(NBANK+2)=Y(NBANK+2)=SCAL
CALL LINE(X,Y,NBANK+2)=1+3
ISTAR=1
IEND=0
DO 5 I=1,NTRUCK
IEND=IEND+NUM(I)
DO 4 I=1,ISTAR+IEND
J=(HULLA(I)+1),1):
JJ=I+1-ISTAR
X(JJ)=XX(J)
Y(JJ)=YY(J)
4 CONTINUE

N=NUM(I)+1
X(N)=X(I)
Y(N)=Y(I)

CONTINUE

5 CONTINUE

CALL SETPEN(2)
YAX=(1333+10)/SCAL
XEND(1)=1102+10)/SCAL
CALL PLOT(X,Y,YAX)/

CALL SYMBOL(0.2,YAX+0.1,10M7200 NORTH+0.2+10)
YAX=(1126+10)/SCAL
XEND(1)=1124+10)/SCAL
CALL PLOT(X,Y,YAX)/

CALL SYMBOL(0.2,YAX+0.1,10M3200 NORTH+0.2+10)
YAX=( 95A-10)/SCAL
XEND=(128C-10)/SCAL
CALL PLOT(0.2,YAX+3)
CALL PLOT(XEND,YAX+2)
CALL SYMBOL( 0.2,YAX+0.10,10H 800 SOUTH+0.2+10)
YAX=( 76)-10/SCAL
XEND=(1343-10)/SCAL
CALL PLOT(0.0,YAX+3)
CALL PLOT(XEND,YAX+2)
CALL SYMBOL( 0.2,YAX+0.10,10M+500 SOUTH+0.2+10)
YAX=( 53A-10)/SCAL
XEND=(1410-10)/SCAL
CALL PLOT(0.0,YAX+3)
CALL PLOT(XEND,YAX+2)
CALL SYMBOL( 0.2,YAX+0.10,11H11000 SOUTH+0.2+11)
YAX=(125A-10)/SCAL
XEND=(1950-10)/SCAL
CALL PLOT(YAX,0.0,3)
CALL PLOT(XAX,YEND,2)
CALL SYMBOL(YAX,0.0,10+6H0 WEST+90.0+6)
YEND=(1537-10)/SCAL
CALL PLOT(XAX,0.0,3)
CALL PLOT(XAX,0.0,3)
CALL SYMBOL(XAX,0.0,10+9H4000 WEST+90.0+9)
XAX=( 93A-10)/SCAL
XEND=(1876-10)/SCAL
CALL PLOT(YAX,0.0,3)
CALL PLOT(YAX,YEND,2)
CALL SYMBOL(XAX,0.0,10+9H7500 WEST+90.0+9)
CALL SYMBOL(X.2+1.0+0.1+20M METROPOLITAN CHICAGO+0.2+20)
CALL SYMBOL(X.2+0.5+0.1+20M HOSPITAL CRISIS+0.2+20)
CALL SYMBOL(X.2+0.2+0.1+20TRANSPORTATION NETWORK+0.2+22)
YSTAR=2285/SCAL
IF (KX = 0) CALL SYMBOL(0.7,YSTAR+0.14100 ALLOCATION BASED ON EMERGENCY COSTS ONLY+0.40)
IF (KX = 1) CALL SYMBOL(0.7,YSTAR+0.14100 ALLOCATION BASED ON ROUTING COSTS ONLY+0.38)
IF (KX = 0) CALL SYMBOL(0.2,YSTAR+0.14100 ALLOCATION BASED ON EMERGENCY COSTS ONLY+0.50)
CALL ENDPLT
RETURN
END
<table>
<thead>
<tr>
<th>FUNCTION DATX NORMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER FUNCTION DATX(I+J)</td>
</tr>
<tr>
<td>DATX=(J-1)*150+I-1</td>
</tr>
<tr>
<td>RETURN</td>
</tr>
<tr>
<td>END</td>
</tr>
</tbody>
</table>

```
4800
4900
4910
```