1987-05
Spring 1987
Modification and Enhancement of LOCSYS and LOCGEN
Roger Murry

SOUTHERN METHODIST UNI

MODIFICATION AND ENHANCEMENT OF LOCSYS AND LOCGEN

BY ROGER MURRY

FOR DR. DICK BARR
SOUTHERN METHODIST UNIVERSITY

DEPARTMENT OF OPERATIONS RESEARCH AND ENGINEERING MANAGEMENT
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
DALLAS, TEXAS 75275

MODIFICATION AND ENHANCEMENT OF LOCSYS AND LOCGEN

BY ROGER MURRY

FOR DR. DICK BARR
SOUTHERN METHODIST UNIVERSITY

PRESENTED MAY 8, 1987

MODIFICATION AND ENHANCEMENT OF LOCSYS

It's one problem when a program I wrote myself needs to be debugged and/or enhanced. It's quite another when someone ELSE'S program isn't working properly, and I have to find the problem. Yet that was the project to which I was committed for my senior design project. It consisted of debugging and modifying two separate, but related, programs. LOCSYS, written by Terry Ross in Austin, Texas, is a client/site resourse allocation goal programming algorithm. The second program, LOCGEN, designed by John Deschner, was the LOCSYS data base generator for LOCSYS. It was also intended to be the front end of LOCSYS. However, when I first received my project assignment, LOCGEN was still a separate module from LOCSYS.

Problems popped up almost immediately after I started working on my project. Space limitations prohibited me from compiling LOCGEN or LOCSYS. After raising my disk allocation from 1 to 3 cylinders, I still couldn't hold a compiled version of both programs. Until more space could be obtained, I had to be content with cleaning up LOCGEN.

LOCGEN was actually fairly clean code when I first tried to compile it. It was written for a different computer, and therefore a different compiler. As a result, a couple of minor errors occurred, such as variable names longer than 6 characters, and format statements missing commas where the IBM 3081 compiler expected them to be. Those problems were quickly corrected, and

LOCGEN compiled with no errors.

By then, my personal disk space had been increased, and I set out to compile and test LOCSYS. It compiled with no errors. To test LOCSYS, and get familiar with how it operates, I tried running LOCSYS using a data file which had already been set up. The results surprised me. I received messages generated by LOCSYS that said "BECAUSE OF MISSING OR EXCLUDED CLIENT OR SITE DATA, THE FOLLOWING CRITERION WAS INDIRECTLY EXCLUDED." This message appeared for every criterion given in the model. Since all criteria had been "excluded," LOCSYS no longer had a problem to solve. The initial solution table was blank.

The explanation given in the documentation for this error stated that this message occurs when site or client data is left out of part C of the data base. (The data base is divided into 3 sections, A, B, and C, depending on the type of data and its format.) I compared the canned data base I was using with the documentation explaining how the information should appear in the file, and all looked well. I tried running LOCSYS with a second canned data base. The same error message recurred. Just to make sure the problem was not with the data bases themselves, I created my own file using an example provided in the LOCSYS documentation. It, too, failed with the same message in the same manner. At this point, some extensive table-tracing was in order.

Logically, I began looking for the place in the program where it determined that a criterion is excluded. After lots of digging, I surmised that a set of decision variables had been reversed--a .GE. should be .LT., and a later .LT. should be .GE. I made those changes and tried to run the program again. LOCSYS no longer considered the criterion to be excluded, and the first screen appeared just as the documentation indicated it should. At this point, I issued the SOLVE command, hopeful that the problem had been located. Strangely, however, the "solution" LOCSYS came up with was exactly the same as the initial table, except that the results in the "actual values" column were all negative.

After further investigation, I found a line of code which arbitrarily, it seemed to me, multiplied actual values by -1. LOCSYS assumes that a negative actual value for a criterion indicates that criterion has been excluded. It seemed logical to me that, if LOCSYS thought a criterion is excluded, it would leave the original criterion parameters unchanged. I removed that line and ran the program again. The actual values were now positive, but otherwise still unchanged from the initial values. As Lewis Carroll wrote in Alice in Wonderland, "Curiouser and curiouser."

At this point I reasoned that production code, which LOCSYS was intended to be, shouldn't have this many bugs in it, and I considered the possibility that I was chasing computer-generated phantoms. So my search for the problem went to a broader scope. I found one mistake in the INCLUDE module, resides outside of LOCSYS and is merged into the main program object code at compile time, and corrected it. Then I returned LOCSYS to its initial state and tried again.

At this point, things really got strange. I tried running LOCSYS with each of the three data bases at my disposal: the two canned data files, and the one I entered myself. Each one failed with a different LOCSYS-generated messages, but none of the messages had anything to do with excluded criteria, the original problem. But the clue to the real cause of all the problems I had been experiencing lay in those error messages.

Because of the way the error-handling subroutine had been designed, it was easy to track down the points from which the messages had been issued. At each point, the program issued a message immediately after an:

IF variable .EQ. 0

hadpfaiised.had been judged to be false. Strategically placed PRINT statements showed, without exception, that the variable in question equaled 0. Therefore, the program should have carried out the rest of the statement. Instead, the program "fell through" to the next line and issued the corresponding error message.

I've never had a problem dealing with computer errors caused by telling the computer to do the wrong thing and having the computer do it. But watching a computer not do what it is told to do was a situation I hadn't encountered before. Then I learned that in FORTRAN, constants aren't. Instead, they are addresses to memory locations where, usually, the corresponding numeric value is stored. However, this makes it possible under certain circumstances to change the value represented by a constant in a FORTRAN program. Based on that information, I printed the value of 0 at each decision point mentioned above. Each time, 0 did not equal 0. It equaled 3.

The most likely scenerio for changing the value of a constant involves passing it as a parameter to a subroutine. If the subroutine accepts a constant into a variable name and assigns a different value to it, the change will be passed back to the calling program when the subroutine ends. At that point, the constant takes on the value of the variable in the subroutine. Of course, this shouldn't happen. But this is exactly what LOCSYS was doing.

The flow of the program at the point where this takes place goes something like this:

```
SUBROUTINE READC
     0 = 0
     CALL SIJ(NT1,0)
     0 = 3
SUBROUTINE SIJ(ICLM,JFROM)
     CALL INDX(JFROM)
     RETURN
SUBROUTINE INDX(KPOS)
     KPOS=3
     RETURN
```

The indended purpose of this section of LOCSYS is to determine if part C of the data base has been entered properly, and to assign the proper S(i,j) values for the different combinations of sites and clients. But it also changes the value of 0 at the same time, thereby causing the program to make all kinds of wrong decisions.

The solution was embarrassingly simple--introduce a dummy variable into subroutine SIJ, set it equal to JFROM, and pass the dummy to INDX. In that way, JFROM doesn't get changed when INDX returns a different value via KPOS. That assures that 0 in READC remains 0 when SIJ ends. With that correction, LOCSYS worked as expected, and the results obtained with the sample data match those given in the documentation.

With LOCSYS working, my attention returned to LOCGEN. The program had originally been written for SMU's old CDC computer. Therefore, some of the program's functions didn't work well on the IBM. For example, LOCGEN stored criteria, site, and client names in an A8 format. The IBM is set up to efficiently handle character strings of an A4 format. As a result, LOCGEN would truncate the first 4 characters of a string, leading to unpredictable results. To remedy this, I spliced all 8-character strings into 2 4-character strings. This worked fine.

Also, many of the commands, such as those listed in the data base requirement instructions, are given for the CDC, not the IBM. Those instructions have been changed to explain to a new user how to use LOCGEN on the IBM.

Next, two major modifications needed to be made to LOCGEN to make it more useful. First, a MODIFY subroutine needed to be added. The original author had written a fine routine for entering data for a model and storing in the format LOCSYS requires. He even included a provision for the user to end a session before all the data had been entered. However, he never put in a modifying routine to make permanent changes to a model, or to finish a model which had already been started.

During this process, I determined that some of John's formats for building the data file did not work as expected, or were incompatable with LOCSYS. For example, LOCGEN's type 6 record generator displaced location names by one character. This lead to LOCSYS not recognizing valid location names in part C of the data base. A rearranged format statement fixed that problem. Also, LOCSYS requires location maximums to be stored in F10.0 format. LOCGEN stored them as F10.2. A minor adjustment solved that as well.

Right now, the user has the ability to finish an incomplete data file, and make changes to the following: criterion weights, limits, ideals and steps; maximum site capacities; and client/site demands. The ability to change the values for f(i,j) and s(i,j) can be added later with some difficulty. The varying formats of these records makes their efficient update tricky. The same holds true for adding/deleting/renaming clients, sites, and criteria. To accomodate these types of changes from within LOCGEN's present structure would be a formidable challenge.

The second major update involved merging LOCGEN into LOCSYS.

That was the original intent of LOCGEN's author--to make LOCGEN the front-end processor for LOCSYS. Compared with creating the modification subroutine, this conversion was fairly simple. Some of the considerations included making sure variable names and types were compatible, and coordinating the entrances and exits of subroutines. For example, when a user would type "STOP" in LOCSYS, the program ended and returned to the R: prompt. With the introduction of LOCGEN, "STOP" now needed to return to LOCGEN's main menu.

One asthetic change made to LOCSYS was the addition of a type of full-screen processing. FORTRAN is an excellent number-crunching language, but it's screen processing leaves something to be desired. When I first received copies of LOCSYS and LOCGEN, all screen formats printed sequentially. That resulted in frequent screen overruns. This inconveniences the user by forcing him or her to constantly press the clear key to refresh the screen. By adding the intrinsic function CALL CLRSCR, at appropriate times, LOCSYS approximates full-screen processing, and makes trips to the clear much rarer.

Finally, I had to change the documentation to reflect these new functions. I expanded the instruction manual to include information on the modification portion of the program, and expanded explanations of the other functions as well. All commands given in the instructions have been changed from CDC terms to IBM terms.

I've encluded in this report a copy of the user's manual in final form, and a printout of the MODIFY subroutine in LOCSYS which I authored. If you would like to make further changes/enhancements in the manual or the program, please feel free to contact me.

LOCSYS USER'S GUIDE

Based on <u>LOCSYS Reference Manual</u>
G. Terry Ross
Analysis, Research, and Computation, Inc.
Austin, Texas

Edited by John Deschner and Roger Murry

GETTING ON THE SYSTEM

To gain access to the LOCSYS algorithm, type the following command from the R: prompt:

```
FILEDEF 1 DISK filename filetype filemode
```

where the filename is either the name of the data file which you want LOCSYS to analyze, or the name of the new data file which you want to create using LOCSYS. This command links the data file to the LOCSYS program. (If you are just using LOCSYS to print out the data base requirements, this step may be omitted.)

After the R: prompt returns, type the following instructions:

```
LOAD LOCSYS
START
```

This will execute the LOCSYS program.

If you wish, you can create an EXEC file using XEDIT which includes each of these commands. Running that EXEC file will then initiate LOCSYS automatically.

CREATING A NEW DATA FILE USING LOCSYS

The LOCSYS algorithm requires a data file in a special format in order to operate. The CREATE/MODIFY portion of LOCSYS helps you organize your data into a file which LOCSYS understands.

To create a new data file, take an option 2 from the main LOCSYS menu. A secondary menu will appear next:

```
#1. IF YOU WOULD LIKE A HARD COPY OF THESE REQUIREMENTS AND
    AN EXPLANATION OF EACH, ENTER 1
#2. IF  YOU WANT THE SAME INFORMATION AT YOU TERMINAL ONLY,
    ENTER 2
#3. IF YOU ARE READY TO CREATE A DATA BASE FILE, ENTER 3
#4. IF YOU WANT TO MODIFY THE FILE NAMED IN YOU FILEDEF
    STATEMENT, ENTER 4
#5. IF YOU WANT TO RETURN TO THE MAIN MENU, ENTER 5

ENTER A 1, 2, 3, 4, OR 5
```

Taking an option 3 from that menu will allow you to enter the data for your model one step at a time. At the end of each step, you will receive the following prompt:

```
DO YOU NEED TO STOP ?
ENTER 1 TO CONTINUE, ENTER 2 TO STOP.
```

This gives you the option of continuing to enter data into the data file, or ending the session. (NOTE: If you enter 2 to stop the session, you must choose the MODIFY EXISTING DATA FILE option in order to add more data to the file.)

Once all of the information has been entered, LOCSYS will automatically arrange the data into the proper format and place it in the file named in the FILEDEF command. The system will return you to the secondary menu. If you want to change information you have entered into the data file, choose option 3. To return to the main menu, choose option 5.

MODIFYING AN EXISTING DATA FILE USING LOCSYS

The interactive portion of LOCSYS allows you to change limits, and weights to see how the new values affect the results. However, these changes are temporary, and do not alter the data file itself. The MODIFY portion of LOCSYS allows you to permanently change criterion weights, ideals, limits, and limit steps. maximum location capacities, and location/client demands.

To modify an existing data file, take an option 2 from the main LOCSYS menu. A secondary menu will appear next:

#1.

Taking an option 4 from that menu will start the modification process. The system will show the following menu:

```
ENTER THE TYPE OF DATA YOU WISH TO MODIFY
     1:  TO CHANGE CRITERIA VALUES
     2:  TO CHANGE LOCATION CAPACITIES
     3:  TO CHANGE CLIENT/LOCATION DEMANDS

PLEASE ENTER 1, 2, OR 3:
```

Choosing option 1 allows you to change criteria weights, ideals, limits, and limit steps. Choosing option 2 allows you to increase or decrease location capacities. Choosing option 3 allows you to change any combination of client/location demands. (NOTE: When changing any of these values, be sure to include the decimal point. Omitting the decimal point will yield unpredictable results.)

Any changes made will be reflected in the data file given in the FILEDEF statement. Once a change has been made, you will return to the secondary menu. Choose option 4 to make further changes to the data file. Choose option 5 to return to the main menu.

PRINTING A COPY OF THE DATA FILE REQUIREMENTS

LOCSYS will print a list of the data file requirements if you need one.

To print a copy of the data file requirements, take an option 2 from the main LOCSYS menu. A secondary menu will appear next:

Taking an option 1 from that menu will automatically send the data file requirements to the printer. Taking an option 2 will send the same information to the screen only.

If this is the only task you will perform in LOCSYS during this session, you do not need to issue a FILEDEF command.

RUNNING THE INTERACTIVE PORTION OF LOCSYS

Once a data file exists and you want LOCSYS to solve a model, take an option 1 from the main LOCSYS menu. This will begin the interactive portion of LOCSYS.

More specific information on how to operate LOCSYS is located elsewhere in this manual.

ENDING A LOCSYS SESSION AND EXITING THE PROGRAM

When you are finished with LOCSYS and want to end the session, take option 3 from the main LOCSYS menu. An END LOCSYS message informs you that you have successfully exited the system..

LOCSYS

THE MULTICRITERIA FACILITY LOCATION MODEL AND INTERACTIVE ALGORITHM

In essence, the model for which LOCSYS was designed, represents a problem of selecting agents (facilities, people or machines) and assigning them tasks (customers or jobs) in the face of agent capacity constraints and multiple conflicting criteria. In the command language, and in the following description of the model, the problem is cast as that of choosing sites at which to establish facilities, in order to serve all clients.

The facility location model has the following characteristics. There exists a finite number of client groups (up to 50), each one representing potential users who are located geographically close to each other and are homogeneous enough in their demands that they can be considered as single entities. There are also a finite number of sites (up to 50), at some subset of which facilities of known limited capacity are to be established. The problem is to choose a subset of sites at which to establish facilities and to assign each client group to exactly one of the facilities. In choosing sites and assigning client groups, multiple criteria (e.g., service and cost criteria--up to 20 criteria functions may be considered simultaneously) and facility capacity constraints are to be considered explicitly.

To formalize the model, let

I = the set of possible sites for facilities,

J = the set of client groups which must be served,

$$z_i = \begin{cases} 1 \text{ if a facility is established at site } i \\ 0 \text{ otherwise,} \end{cases}$$

$$x_{ij} = \begin{cases} 1 \text{ if client group } j \text{ is assigned to the facility at site } i \\ 0 \text{ otherwise,} \end{cases}$$

b_i = the known maximum capacity of the facility to be established at site i.

d_{ij}= the demand placed on the facility at site i by client group j if it is assigned to that facility.

Feasible solutions of the location model must satisfy the following constraints:

$$\sum_{i\epsilon I} x_{ij} = 1 \qquad \text{for all } j\epsilon J, \qquad (1)$$

$$\sum_{j\epsilon J} d_{ij}x_{ij} \leq b_i z_i \qquad \text{for all } i\epsilon I, \qquad (2)$$

$$z_i,\ x_{ij} = 0 \text{ or } 1 \qquad \text{for all } i\epsilon I,\ j\epsilon J \qquad (3)$$

Constraints (1) insure that every client group is assigned to a facility, and constraints (2) insure that facility capacity limitations are enforced. Clearly, $b_i > 0$ for all i, and it is permissible for b_i to equal a very large number whenever there is no capacity limitation on the facility at site i. The model is sufficiently general to represent a demand generation effect that often accompanies the establishment of facilities. That is, the numeric values of d_{ij} for some j can be heavily dependent on i to indicate that demand from client group j varies with the site i to which the client group is assigned. In applications where demand generation is insignificant, the parameter d_{ij} might be replaced by d_j to emphasize the distinction between the two cases. Different values of j need not refer only to geographically different client groups; they may serve to distinguish different services required by the same client group. Thus, the model can be used to decide which services to offer at respective facilities. In this case, the unit of measurement of the parameter d_{ij} must be defined so that the quantity $\sum_{j\epsilon J} d_{ij}x_{ij}$ is meaningful.

Several cost and service criterion functions can be formulated in the following general form,

where:

K denotes the set of criterion functions,

f_{ik} is the contribution to criterion k when a facility is established at site i,

s_{ijk} is the contribution to criterion k when client group j is assigned to a facility at site i,

z_i is a decision variable assigned a value of 1 when site i is selected (and 0 otherwise),

x_{ij} is a decision variable assigned a value of 1 when client group j is assigned to the facility at site i (and 0 otherwise),

$$\text{TYPE} = \underset{I}{\text{SUM}}\ f_{ik}z_i + \underset{I}{\text{SUM}}\ \underset{J}{\text{SUM}}\ s_{ijk}x_{ij} \leq \text{LIMIT}_k$$

or in terms of one of the following special cases of the general form:

$$\text{TYPE } 2_k = \underset{I}{\text{SUM}}\ f_{ik}z_i \leq \text{LIMIT}_k$$

$$\text{TYPE } 3_k = f_{ik}z_i \leq \text{LIMIT}_k$$

$$\text{TYPE } 4_k = \underset{I}{\text{SUM}}\ s_{ijk}x_{ij} \leq \text{LIMIT}_k$$

For example, a criterion representing the noxiousness of establishing a facility at site i can be expressed in the form TYPE 3 where f_{ik} quantifies the noxious effect. A criterion of the form TYPE 4 may represent total demand served from a client group when each s_{ijk} quantifies the number of persons from client group j who use a facility at site i. Analogously, if t_{ijk} denotes the travel time for persons in client group j if they are assigned to a facility at site i, then TYPE 4 represents a criterion function measuring travel time for client group j when each $s_{ijk}=t_{ijk}$. TYPE 2 may be used to minimize or limit the total number of sites chosen by locsys by setting the f(ik) for each site equal to 1. As a final example, TYPE 2 may be used to represent total fixed investment cost when each f_i is the investment cost of establishing a facility at site i. Although the parameters, constraints and criterion functions in the model have been defined here in the context of facilities location, it is clear that the model is applicable in a variety of contexts. For example, the model may represent problems in such diverse contexts as machine loading and inspector assignment.

Because it is unlikely that a single solution will yield the minimum value for all criteria simultaneously, one seeks instead efficient solutions (i.e. solutions with the characteristic that it is possible to decrease the value of one criterion only by increasing the value of at least one other criterion). Efficient solutions are generated through interaction between the user and special purpose optimization software. In particular, the user must provide relative WEIGHTS for each of the various criteria and upper LIMITS on the value that each criterion function may attain in order for the efficient solution to satisfy the user. Using these values, LOCSYS constructs a problem which when solved yields an efficient solution to the original problem. Further efficient solutions may be generated when the user changes criterion WEIGHTS and/or criterion LIMITS and the optimization software solves the resulting problem.

The purpose of LOCSYS is to facilitate this iterative process alternating between the choice of parameter values (WEIGHTS and LIMITS) by the user and the solution of problems by an optimizing algorithm. At each iteration, the system reports the values of the criterion functions and, at the user's option, the choice of sites and assignments of clients to facilities. A simple command language provides the means through which the user provides his inputs and directs the next iteration of the algorithm.

Several points regarding the algorithm should be emphasized. First, an assumption underlying the algorithm is that the decision maker can select his most preferred efficient solution. The purpose of the algorithm is to generate a number of efficient solutions and to facilitate his choice among them.

Second, the algorithm is not very demanding of the decision maker. The user need not have any awareness of the mathematical structure of the problem itself. There is very little formal procedure for the user to learn. Input to be provided by the decision maker is minimal. Perhaps even more important, the numbers the decision maker must supply reflect his judgment and have, therefore, no roght or wrong values. Because each iteration of the algorithm is independent of the last, the user is free to reverse at will his judgments on the relative WEIGHTS of the criteria or to adjust LIMITS as he becomes aware of what is attainable or as curiosity might motivate him.

Third, the algorithm is heuristic, for only a subset of the set of efficient solutions may be examined. The user is free to examine as few or as many of these solutions as he desires. The decision maker is in complete control of the solution process. The user may have little rationale behind his choice of parameters in the first few iterations. However, as additional results are generated he should gain increasing insight into what is attainable and thus finally assure himself that further efforts are unlikely to yield a still more preferable solution than what he has already found.

For further information on the model and the formulation of additional criteria in the form TYPE1-TYPE4, see:

Ross, G.T. and R.M. Soland, "A Multicriteria Approach to the Location of Public Facilities, " to appear in European Journal of Operational Research.

for the inspector assignment case, see:

Trippi, R.R., "The Warehouse Location Formulation as a Special Type of Inspection Problem," Management Science, 21 (1975), 986-988.

for the machine loading case, see:

Klingman, D., G.T. Ross and R.M. Soland, "Optimal Lot Sizing and Machine Loading For Multiple Produces," Proceedings of the Conference on the Problems of Disaggregation in Manufacturing and Service Organizations, The Ohio State University, Columbus, Ohio, March 1977, 243-244.

PROCEDURE

LOCSYS requires one data file containing the parameters of the mathematical model and initial values for certain parameters used in the interactive algorithm.

The contents of this data file are not modified in any manner by the system. LOCSYS utilizes a scratch file for temporary storage of results generated during a run.

The tasks that the user must perform are divided into two phases: data base editing and problem solution. In the first phase, the user has the option to alter the components of the problem (as it is defined in the data base) by removing certain criterion functions, sites or client groups from consideration in the solution phase. The reasons for this option are that a) the data base may include a far greater volume of data than the user wishes to consider at one time and b) the user may wish to solve several different variations of one multicriteria problem in several separate program runs.

In the second phase, the user directs a search for efficient solutions to the multiple criteria problem defined by the first phase. To generate an efficient solution, the user must supply two parameters for each criterion. These parameters are a WEIGHT to denote the relative importance of the k^{th} criterion (larger WEIGHTS imply greater importance), and an upper LIMIT on the value that the k^{th} criterion can attain. The LIMIT is the largest value of the k^{th} criterion function that the user considers to be satisfactory. Following the selection of these values, the optimization algorithm will find an efficient solution that reflects the relative importance of the criteria (as indicated by the user) while not allowing any criterion to exceed its limiting value (assigned by the user). Of course, if one (or more) of the LIMITS is set too low, the algorithm may be unable to find an efficient solution that satisfies the LIMITS. Assuming an efficient solution exists, the user must then evaluate the solution in terms of the values of the criterion functions and in relation to the previously generated solution (if any) which he found most preferable. The new solution may become his most preferred or he may still prefer his previous choice. In either case, the user must then alter one or more criterion WEIGHTS and/or LIMITS to define new conditions for the next efficient solution to satisfy.

The WEIGHTS and LIMITS provide a simple mechanism through which the user's preferences can be expressed. Initially, the user may have little or no rationale for his choices. However, the sequence of efficient solutions generated should provide him with ever increasing insight into what is attainable and should provide the basis upon which to make subsequent choices of WEIGHTS and LIMITS. The decision to stop the search for efficient solutions is the user's. That point is reached when the user has some feeling for the impact of alternative WEIGHTS and LIMITS and is content that further effort is unlikely to generate a solution still more preferable than the incumbent most preferred solution that he has identified.

The content and volume of output generated by the system is entirely dependent upon the user's actions. The output generated at each iteration includes the parameters specified by the user, the values of the criterion function associated with the latest efficient solution, and the values of the criterion functions associated with the user's most preferred solution generated to date. At the user's option the system will report the sites chosen and the assignments of client groups to sites or the results from any earlier iteration of the algorithm.

When the program is initiated the output in Figure 1 is provided to summarize the information contained in step one of the data base.

```
THE NAMES OF 4 CRITERIA, 3 CLIENTS AND 2 SITES HAVE BEEN READ.
INITIAL WEIGHTS, IDEALS, LIMITS AND STEPS ARE:
CRITERION          WEIGHT          IDEAL          LIMIT        STEP
NOXIOUS             1.00            1.00          50.00         5.00
DISTANCE            1.00            3.00           6.00         1.00
COST                1.00          100.00         600.00       100.00
NUMSITES            1.00            1.00           3.00         1.00
IT IS RECOMMENDED THAT YOU CHANGE ONE OR MORE LIMITS AT LEAST
EVERY 3 ITERATIONS.  YOU WILL BE REMINDED TO DO SO IF NECESSARY.
```

Figure 1 Initial Program Output

When an efficient solution is generated, the output in Figure 2 is generated. That table enables the user to compare the values of the criterion functions associated with the most recently generated efficient solution (CURRENT ACTUALS) to those associated with the user's most preferred efficient solution (BEST ACTUALS). Similarly the user can compare the CURRENT ACTUALS to the IDEAL and LIMIT values for the criterion. To facilitate the user's choice of new relative WEIGHTS, the WEIGHTS that were used to generate the latest efficient solution are also provided.

```
**SOLUTION 2**
                 CURRENT                 CURRENT    CURRENT    BEST
CRITERION        WEIGHT      IDEAL       ACTUAL     LIMIT      ACTUAL
NOXIOUS           1.00        1.00       40.00      50.00      40.00
DISTANCE          1.00        3.00        0.00       6.00       0.00
COST              1.00      100.00      335.00     600.00     335.00
NUMSITES          1.00        1.00        2.00       3.00       2.00

DO YOU PREFER THIS SOLUTION OR YOUR BEST PREVIOUS SOLUTION?
ENTER BEST OR 2
```

Figure 2 Program Output Generated for Each Efficient Solution

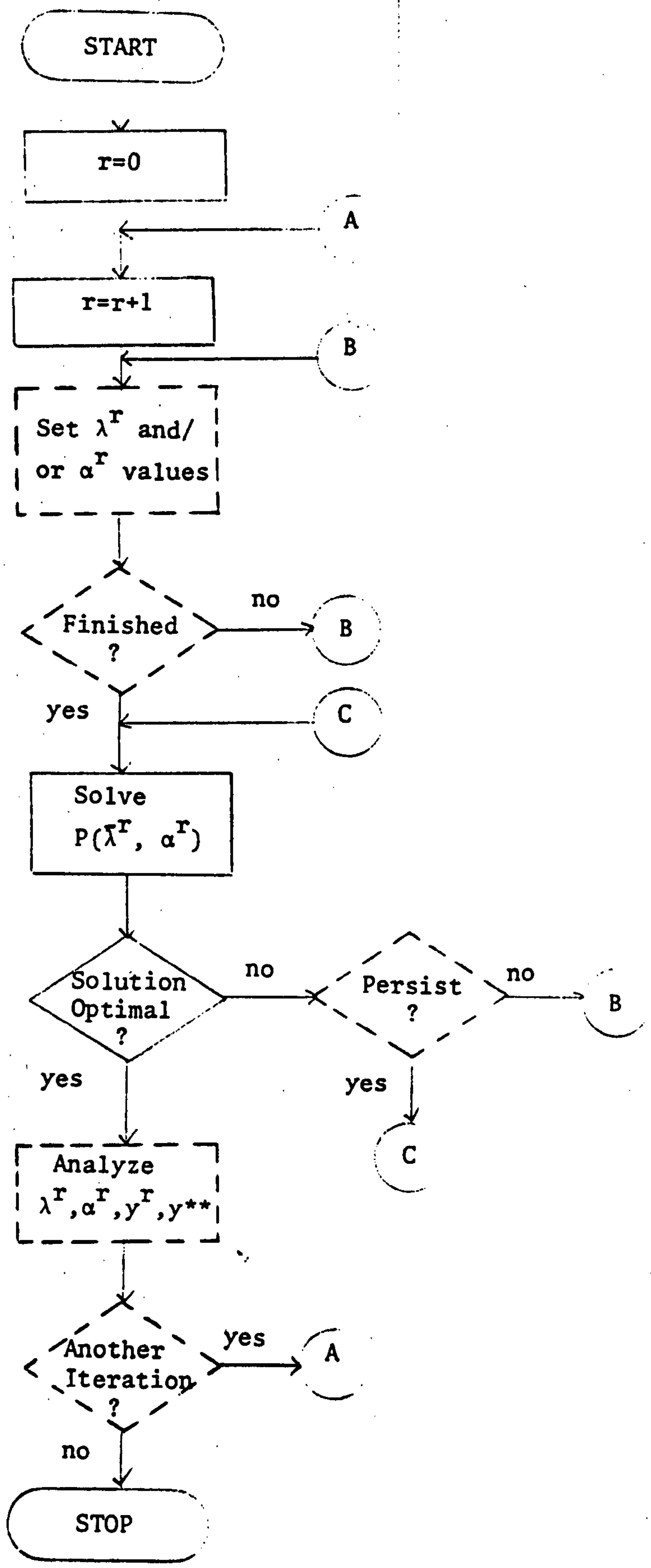
The user may wish to know which sites were selected and which client groups were assigned to each site. Figure 3 shows the output generated when the DISPLAY RESULTS command is entered. As these examples illustrate, the output provided by the program is self-explanatory.

Figure 3 Output Generated Following DISPLAY RESULTS Command

In choosing WEIGHTS and LIMITS from iteration to iteration, the user should keep the following conditions in mind. First, the work of the optimization routine is simplified if the user evaluates all alterative criterion weightings for a given set of LIMITS one after another. The reason for this is that a preceeding solution which is known to satisfy the LIMITS can be used as a starting point in the search for the solution to the next problem with different WEIGHTS. This advantage is not realized when one evaluates several sets of LIMITS for a given set of WEIGHTS. Second, the particular mathematical problem which the optimization routine must solve can be very time consuming to solve. Consequently, at fixed intervals, the user is given the choice of abandoning the attempt to solve a problem. Thus, an efficient solution need not always be found at either because one does not exist or because it has become too time consuming from the user's standpoint for the user to be interested in the solution.

To insure that the user has the dominant role and the freedom to choose the direction in which the algorithm will proceed, the computer system does not initiate any action on its own except to read the data base, request commands from the user and list criterion function information for the latest efficient solution generated. All other actions are carried out in response to commands from the user.

A flowchart for the problem solution phase is given in Figure 4.


START
$r=0$
A
$r=r+1$
B
Set λ^r and/ or α^r values
Finished ?
no
B
yes
C
Solve $P(\lambda^r, \alpha^r)$
Solution Optimal ?
no
Persist ?
no
B
yes
yes
C
Analyze $\lambda^r, \alpha^r, y^r, y^{**}$
Another Iteration ?
yes
A
no
STOP

Legend: – – – – Step performed by decision maker

——— Step performed by computer system

FIGURE 2. Flow of Control During Interactive Algorithm

with

$\lambda_k \equiv$ WEIGHT FOR CRITERIA K

$\alpha_k \equiv$ LIMIT FOR CRITERIA K

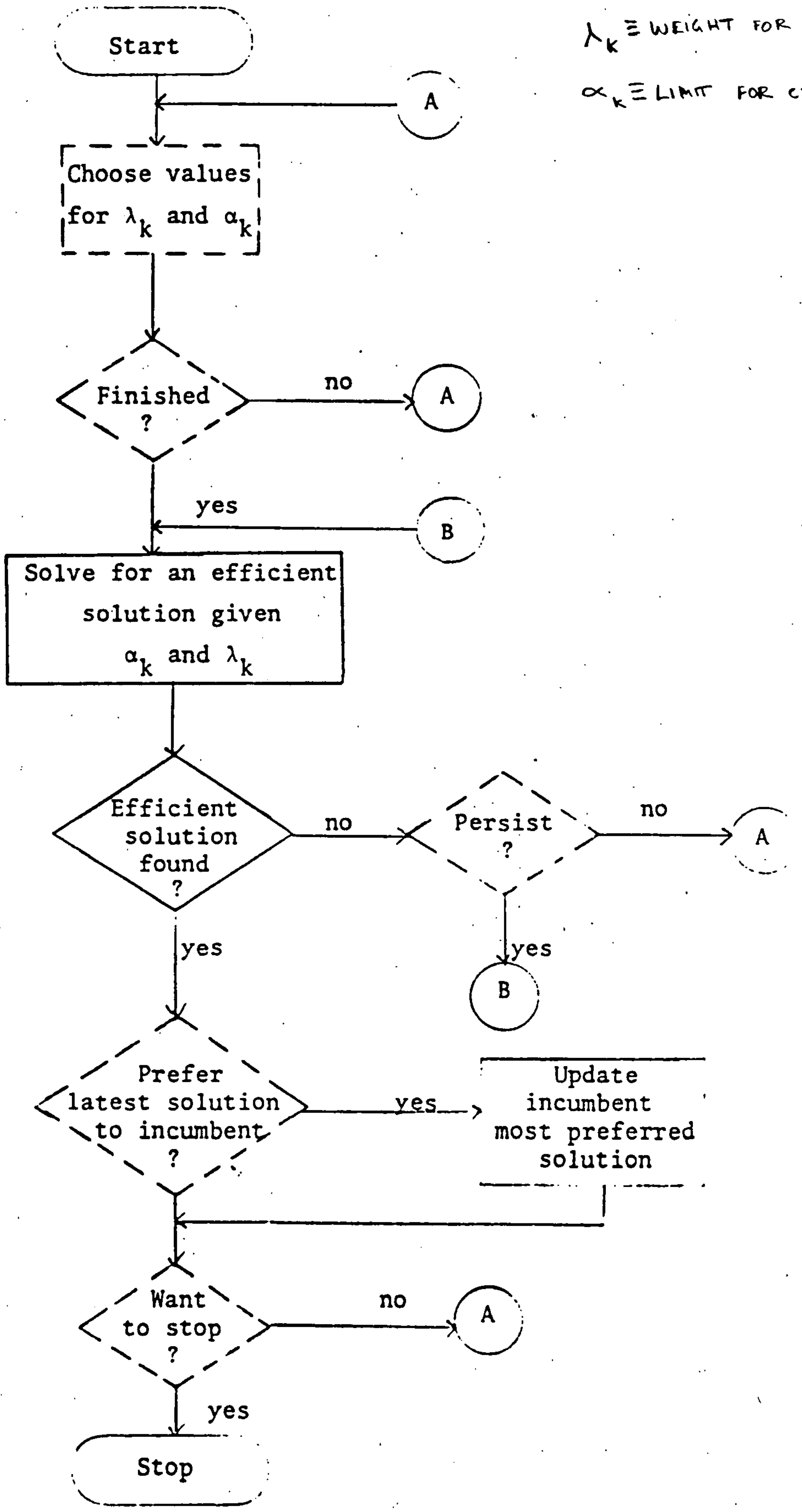

Start
A
Choose values for λ_k and α_k
Finished ?
no
A
yes
B
Solve for an efficient solution given α_k and λ_k
Efficient solution found ?
no
Persist ?
no
A
yes
B
yes
Prefer latest solution to incumbent ?
yes
Update incumbent most preferred solution
Want to stop ?
no
A
yes
Stop

Figure 4 Problem Solution Phase

Legend: ______ Step performed by computer system

----- Step performed by user

PROGRAM INPUTS

Program inputs comprise two major types:

1) commands entered by the user, and

2) parameters in the data base that define the facility location model

Each of these two types of input are described in detail below.

USER COMMANDS

To enable the user to exercise control over the algorithm, a simple command language has been developed. A standardized syntax has been adopted which is English-like and uncomplicated. Every command adheres to a general form involving the elements verb, direct object, adjective, preposition and object of a preposition in that order. For example, SET WEIGHT COST TO 2 is a valid command in the general form. Some verbs provide data management capabilities (e.g. DISPLAY) while others are necessitated by the logic of the interactive procedure (e.g. SOLVE). Most commands do not require all elements of the general form. The most common variants of the general form are a) verb, b) verb, direct object, and c) verb, direct object, adjective. Examples of these command forms are a) SOLVE, b) DISPLAY RESULTS and c) EXCLUDE SITE BOSTON.

To simplify entering commands, the spacing between elements and the positioning of the command on an input record are immaterial. The order of command elements and the validity of each element are the only factors with which the user need be concerned.

The legitimate verbs and keywords used in the commands are given below. A complete list of <u>all</u> admissible commands is given later.

<u>Verbs</u>

DISPLAY	Report the values of various parameters or variables
EXCLUDE	*Ignore data on a criterion, client group or site contained in the data base

INCLUDE	*Reverse effect of the EXCLUDE verb
LOWER	Decrease the value of a parameter
RAISE	Increase the value of a parameter
RESTART	Stop the program and start over again using the same data base
SET	*Assign a value to a parameter
SOLVE	Solve the problem to determine an efficient solution
STOP	Terminate execution of the program

*EXCLUDE, INCLUDE, AND SET IDEAL ARE VALID ONLY UNTIL THE SOLVE COMMAND IS FIRST GIVEN.

Keywords

ACTUAL	A numeric value attained by a criterion function
BEST	The iteration of the algorithm that yielded the user's most preferred efficient solution found to date
CLIENT	A client group
CRITERIA	A criterion function
IDEAL	An estimate of the numeric value a criterion function would attain in the absence of any other criterion
LIMIT	An upper limit on the value that a criterion function may attain
RESULT	A decision made relative to a site or client group
SITE	A potential location for a facility
STEP	A value representing a minimal change in the value of a criterion function
WEIGHT	A numeric value representing the relative importance of a criterion

The system was designed to be forgiving in the sense that incorrect commands affect neither the execution of the program not the results obtained by the interactive algorithm. The user need not be particularly cautious in entering input.

The system does not require the user to perform his tasks in any fixed order except that the system will not follow illogical commands (e.g. to report a solution before it has been obtained.) The user can, in fact, enter a command and then immediately enter a second command to negate the effect of the first command without any adverse consequence. Ordinarily, the system neither prompts the user for specific information nor demands a response from the user. Two exceptions are that after each subprob is solved the user is required to decide whether or not to update his most preferred solution, and should the optimization routine encounter difficulty solving a particular version of the problem, the user must decide whether or not to abandon work on the problem.

The primary advantage in standardizing the command structure is to facilitate parsing the commands. Syntactically incorrect commands are recognized easily, and the cause of the error identified for the user.

A secondary advantage of the command structure is that it admits variant forms which have differing degrees of power. For example, assuming COST is a valid criterion name, DISPLAY WEIGHT COST is a valid command to the system to print the value of a single WEIGHT. By leaving off the qualifying adjective (COST) and pluralizing the direct object WEIGHT, the expression DISPLAY WEIGHTS is a command to print the entire vector, WEIGHTS. Thus, the command structure encompasses many variant forms and allows the user to tailor the system's efforts to his needs.

ALL ADMISSIBLE COMMANDS , AND THEIR MEANINGS

Program Control Commands

STOP	Terminates execution of the program
RESTART	Stops the interactive algorithm and starts the algorithm over again using the same data base.

<u>Data Base Editing Commands</u>

EXCLUDE SITES "list"

EXCLUDE CLIENTS "list"

EXCLUDE CRITERIA "list"

The element "list" is the word ALL or one or more valid site, client or criterion names (depending upon the particular command) separated by commas. Although present in the data base, data on the named sites, clients, and criteria are not to be considered as a part of the problem. Clearly the problem cannot be solved if all of the sites, clients or criteria are excluded. The purpose of the ALL option is to simplify selecting those sites, clients or criteria to be included when they are a small fraction of the total number in the data base. The EXCLUDE command is valid only until the SOLVE command is first given.

INCLUDE SITES "list"

INCLUDE CLIENTS "list"

INCLUDE CRITERIA "list"

Reverses the effect of an exclude command. The program assumes all clients, sites and criteria are included unless otherwise directed. The INCLUDE verb is valid only until the SOLVE command is first given.

<u>Algorithm Control Commands</u>

SOLVE

The current values of weights and limits are used to define a problem for the system to solve. Information on the solution is reported automatically.

SET WEIGHT "name" TO a

SET LIMIT "name" TO a

SET STEP "name" TO a

Assigns the value a (real or integer numeric) to the current value of the weight, limit or step for the named criterion.

SET IDEAL "name" TO a

Assigns the value a(real or integer numeric) to the ideal of the named criterion. This command is valid only until the SOLVE command is first given.

SET WEIGHTS ALL TO i

SET LIMITS ALL TO i

The element i can be an integer number denoting an iteration number or i can be the word BEST. In either case, the values of all weights or limits are set to those that were used at iteration i or at the iteration where the user's most preferred solution was found (BEST).

SET REMINDER TO i

The element i is a positive integer denoting the number of consecutive iterations of the algorithm that may be preferred using a given set of limits before the system will remind the user to change one or more limits. (Default value for REMINDER is 3).

RAISE LIMIT "name" BY "item"

The element "item" can be a real integer number or the word STEP. In any case, "item" specifies the amount by which the limit of the criterion specified by "name" should be increased. When STEP is specified, the step size specified for the criterion is used.

LOWER LIMIT "name" BY "item"

This command has the opposite effect of RAISE.

Output Commands

DISPLAY CLIENTS

DISPLAY CRITERIA

DISPLAY SITES

Lists the names of all clients, criteria or sites that are included in the problem.

DISPLAY IDEALS

DISPLAY STEPS

Lists the value of the ideal or step for every criterion included in the problem.

DISPLAY WEIGHTS "item"

DISPLAY ACTUALS "item"

DISPLAY LIMITS "item"

DISPLAY RESULTS "item"

The element "item" may be blank, an integer number or the word BEST. Blank implies current. Integer number indicates iteration number. BEST indicates those quantities associated with the problem solved when the best solution was found. For the iteration specified by "item" the values of the weights, limits or actuals for all criteria will be listed. In the case of results, the clients assigned to each site in the efficient solution will be listed.

DISPLAY WEIGHT "name"

DISPLAY LIMIT "name"

DISPLAY ACTUAL "name"

DISPLAY IDEAL "name"

DISPLAY STEP "name"

The current value of the wright, limit, actual, ideal or step associated with the criterion specified by "name" will be listed.

DISPLAY RESULT SITES

Lists the names of the sites that were selected in the current solution.

DISPLAY RESULT SITE "name"

DISPLAY RESULT CLIENT "name"

Lists the names of the clients (if any) assigned to the specified site "name", or lists the name of the site to which the specified client "name" is assigned.

COMMAND ERRORS

Command errors may arise after the user has entered a command because the command is not expressed in proper syntax or because it is illogical or impossible to carry out the command. A command may contain two or more errors, but upon finding one error, the system ignores the remainder of the command and gives only one error diagnostic. Errors in commands do not affect program execution.

1. INVALID OR MISSING VERB

   The term in the verb field of the command is misspelled or not a verb.

2. INVALID OR MISSING DIRECT OBJECT

   The term in the direct object field is misspelled, missing or inappropriate for the verb.

3. INVALID OR MISSING ADJECTIVE

   The term in the adjective field is misspelled, missing or inappropriate for the given verb and direct object.

4. INVALID OR MISSING PREPOSITION

   The term in the preposition field is misspelled, missing or inappropriate for the command.

5. INVALID OR MISSING OBJECT OF PREPOSITION

   The object of the preposition is misspelled, missing or inappropriate for the command.

6. ITERATION NUMBER IS OUT OF RANGE

   The numeric value supplied for an iteration number exceeds the number of iterations that have been performed or is negative.

7. COMMAND HAS TOO MANY TERMS

   There are more terms in the command than there should be. No extraneous characters or terms can be entered at the end of a command.

8. CRITERION WAS EXCLUDED

   The user has issued a command that refers to an excluded criterion. (A criterion may have been excluded directly by a command from the user or indirectly because data was not supplied in the data base. In the latter case, the user would have received a message earlier telling him the criterion had been indirectly excluded.)

9. ILLEGAL CRITERION WEIGHT

   The user has attempted to set a criterion weight to a non-positive value. All criterion weights must be positive.

10. ILLEGAL CRITERION IDEAL

   The user has attempted to set a criterion ideal to a non-positive value. All criterion ideals must be positive.

11. NO SOLUTION AVAILABLE

   The user has issued a command that refers to results which have not yet been derived.

12. INVALID OR MISSING SITE NAME

   The site name in a DISPLAY RESULTS command is misspelled, missing, or not defined in the data base.

13. INVALID OR MISSING CLIENT NAME

   The client name in a DISPLAY RESULTS command is misspelled, missing or not defined in the data base.

14. ILLEGAL COMMAND AFTER FIRST PROBLEM HAS BEEN SOLVED

   The INCLUDE and EXCLUDE commands may not be used after the first problem has been solved. (If it is necessary to INCLUDE or EXCLUDE after the first problem has been solved, the program must be RESTARTed.) Also, the user may not use the SET command to change a criterion ideal after solving the problem once.

15. ILLEGAL CHARACTER IMBEDDED IN A NUMBER

   The data item in a numeric field on a command is not a legitimate number.

16. PROBLEM MUST INCLUDE AT LEAST ONE CLIENT

   This error arises when the user has attempted to solve a problem for which all clients have been excluded from the problem or when Part B of the data base is missing. The error may be corrected by using the INCLUDE command to include the desired client(s) or by correcting XDB.

17. PROBLEM MUST INCLUDE AT LEAST ONE SITE

   This error indicates that the user tried to solve the problem with all sites excluded. This error may be corrected by using the INCLUDE command to include the desires site(s).

18. PROBLEM MUST INCLUDE AT LEAST ONE CRITERION

   This error indicates that all criteria have been excluded from the problem either directly (via the EXCLUDE command) or indirectly (because data was not included for the criterion in Part C of the data base). In the first case, the user may issue the INCLUDE command. In the second case the error can be corrected by including data for at least one criterion in XDB.

DATA FILE DESCRIPTION

In order to execute LOCSYS, the following data must be specified in the external data file. Entry is broken down into the following steps when using the CREATE/MODIFY procedure within LOCSYS:

Step 1

NAMES OF CRITERION FUNCTIONS TO BE MINIMIZED
INITIAL WEIGHT FOR EACH CRITERION
INITIAL IDEAL FOR EACH CRITERION
INITIAL LIMIT FOR EACH CRITERION
INITIAL STEP FOR EACH CRITERION

Each criterion name must be 1-8 alphanumeric characters long. The first character must be a letter. No embedded or leading blanks are allowed. Also, all criterion names must be unique. All numeric values for each criterion should be entered in F10.0 format (xxxxxxxxx.).

Step 2

LIST OF CLIENTS

Each client name must be 1-8 alphanumeric characters long. The first character must be a letter. No embedded or leading blanks are allowed. Also, all client names must be unique.

Step 3

LIST OF LOCATIONS
MAXIMUM CAPACITY FOR EACH LOCATION

Each location name must be 1-8 alphanumeric characters long. The first character must be a letter. No embedded or leading blanks are allowed. Each location name must be unique, and no location can have the same name as any client. Each capacity should be entered in F10.0 format (xxxxxxxxx.).

Step_4

DEMANDS FOR EACH LOCATION WITHIN EACH CLIENT

The order in which the location names appear for each client is immaterial. If a client can be served by four or more locations, additional records are needed. In this case, each subsequent record has the same format as the first, except columns 2-9 MUST be blank. (NOTE: LOCSYS handles this situation automatically.) Each demand must be entered in F10.0 format (xxxxxxxxx.)

Step_5

TYPE OF EACH CRITERION FUNCTION
F(I,K) VALUES FOR EACH CRITERION FUNCTION OF TYPE
1, 2, OR 3
S(I,J,K) VALUES FOR EACH CRITERION FUNCTION OF TYPE
1 OR 4

The subscript i represents locations; j refers to clients; and k indicates criteria. The function F(i,k) quantifies criterion k when location i is established. The function S(i,j,k) quantifies the effect of criterion k when client j is established at location i.

Each of the four types of criteria functions are defined below:

A printout of these data file requirements is available from within LOCSYS. Follow the instructions given the section called PRINTING A COPY OF THE DATA FILE REQUIREMENTS to obtain the printout.

EXAMPLE DATA FILE

Figure 5 shows a goal programming model in algebraic form. The four criterion names are NOXIOUS, DISTANCE, COST, and NUMSITES. Weights for these criteria are all equal to 1. Limits for these criteria are 50, 6, 600, and 3, respectively. The ideal values for these criteria are 1, 3, 100, and 1, respectively. The initial limit step for these functions are 5, 1, 100, and 1, respectively. Client names include ATLANTA1, COLUMBUS, AND AUGUSTA. Locations are ATLANTA2 and MACON. Figure 6 illustrates how this model would look after LOCSYS puts this information in the form of the external data file.

Minimize $63z_1 + 30z_2 + 169x_{11} + 182x_{21} + 16x_{12} + 18x_{22} + \infty x_{13} + 55x_{23}$

Minimize $1z_1 + 1z_2$

Minimize $40z_2$

Minimize $0x_{11} + 5.9x_{21}$

Subject to:

$$
\begin{array}{llllllll}
x_{11} + x_{21} & & & & & & = 1 \\
& & x_{12} + x_{22} & & & & = 1 \\
& & & & x_{13} + x_{23} & & = 1 \\
131x_{11} & & + 8x_{12} & & + 27x_{13} & & \le 135z_1 \\
& 100x_{21} & & + 10x_{22} & & + 27x_{23} & \le 100z_2
\end{array}
$$

$x_{ij} = 0$ or 1 for all $i\varepsilon I$, $j\varepsilon J$

$z_i = 0$ or 1 for all $i\varepsilon I$.

FIGURE 3.3 Example Multicriteria Location Problem

```
1 4
1NOXIOUS                    1.        1.       50.        5.
1DISTANCE                   1.        3.        6.        1.
1COST                       1.      100.      600.      100.
1NUMSITES                   1.        1.        3.        1.
2 3
2ATLANTA1
2COLUMBUS
2AUGUSTA
3 2
3ATLANTA2      135.
3MACON         100.
4ATLANTA1 ATLANTA2        131.MACON          100.
4COLUMBUS ATLANTA2          8.MACON           10.
4AUGUSTA  ATLANTA2         27.MACON           27.
5COST     1
6ATLANTA2       63.MACON          30.
7ATLANTA1 ATLANTA2        169.MACON          182.
7COLUMBUS ATLANTA2         16.MACON           18.
7AUGUSTA  MACON            55.
5NUMSITES 2
6ATLANTA2        1.MACON           1.
5NOXIOUS  3
6MACON          40.
5DISTANCE 4
7ATLANTA1 ATLANTA2        0.0MACON           5.9
0
```

FIGURE 3.4 Data Base For Example Problem

DATA FILE ERRORS

Data file errors arise when the system discovers omissions, inconsistencies or illegalities in input data. In general, errors discovered in the data file are fatal, and the program terminates upon discovering them. The data file may contain additional errors, but the system will stop processing after finding the first error.

1. BECAUSE OF MISSING OR EXCLUDED CLIENT OR SITE DATA, THE FOLLOWING CRITERION WAS INDIRECTLY EXCLUDED

   This error message results from not including the appropriate data for the criterion in Step 5 of the data file.

2. MAXIMUM NUMBER OF TYPE 1 CRITERIA EXCEEDED

   The system allows at most three Type 1 criteria to be considered simultaneously.

3. MAXIMUM NUMBER OF TYPE 2 CRITERIA EXCEEDED

   The system allows at most two Type 2 criteria to be considered simultaneously.

4. MAXIMUM NUMBER OF TYPE 3 CRITERIA EXCEEDED

   The system allows at most two type 3 criteria for any one location to be considered simultaneously.

5. MAXIMUM NUMBER OF TYPE 4 CRITERIA EXCEEDED

   The system allows at most two Type 4 criteria for any one client to be considered simultaneously.

```
[redacted] PL 1986) [redacted] VS FORTRAN [redacted] DATE: [redacted] 10 [redacted] 87 [redacted] TIME: 2 [redacted] 0:4 [redacted] PAGE: [redacted] 
.CT: NOLIST NOMAP NOXREF NOGOSTMT NODECK SOURCE TERM OBJECT FIXED IRMFLG SRCFLG NOSYM NORENT
     SDUMP(ISN) AUTODBL(NONE) NOSXM NOVECTOR IL NOTEST NODC NODIRECTIVE
          OPT(0) LANGLVL(77) NOFIPS    FLAG(I)  NAME(MAIN)    LINECOUNT(60)      CHARLEN(500)

*....*..1.........2.........3.........4.........5.........6.........7.*.......8
C
      SUBROUTINE MODIFY                                                 LOC31590
C  SUBROUTINE TO MODIFY CRITERIA PARAMETERS, MAXIMUM SITE CAPACITIES,   LOC31600
C  AND DEMANDS FOR EACH SITE/LOCATION COMBINATION                       LOC31610
C  AUTHOR:  ROGER MURRY                                                 LOC31620
      CHARACTER*4 CRIT1,CRIT2,CLI1,CLI2,SITE1,SITE2                     LOC31630
      CHARACTER*4 CCRIT1,CCRIT2,TCRIT1,TCRIT2                           LOC31640
      CHARACTER*4 CSITE1,CSITE2,CCLI1,CCLI2,DCLI1,DCLI2                 LOC31650
      CHARACTER*4 DDUM1,DDUM2,ISPACE,DDST1,DDST2                        LOC31660
      CHARACTER*4 DSITE1,DSITE2                                         LOC31670
      COMMON NCRITS,CRIT1(20),CRIT2(20),NNCLI,CLI1(50),CLI2(50)         LOC31680
      COMMON NSITES,SITE1(50),SITE2(50),DMD(50,50)                      LOC31690
      COMMON LINER(200,80),JCNTR,DSITE1,DSITE2,CCLI1,CCLI2              LOC31700
      COMMON DDST1(3),DDST2(3)                                          LOC31710
      DIMENSION INMBR(200),DD(3)                                        LOC31720
      INTEGER FLAG                                                      LOC31730
C                     INITIALIZE VARIABLES                              LOC31740
      ICOUNT = 0                                                        LOC31750
      NCRITS=0                                                          LOC31760
      NSITES=0                                                          LOC31770
      NNCLI=0                                                           LOC31780
      JCNTR = 0                                                         LOC31790
      KCNTR=0                                                           LOC31800
      ISPACE='    '                                                     LOC31810
      REWIND 1                                                          LOC31820
C                         READ IN ENTIRE DATA BASE FILE                 LOC31830
10    ICOUNT = ICOUNT + 1                                               LOC31840
      READ (1,15) INMBR(ICOUNT),(LINER(ICOUNT,I),I=2,80)                LOC31850
C                         CHECK FOR END-OF-FILE                         LOC31860
      IF (INMBR(ICOUNT) .EQ. 0) GO TO 200                               LOC31870
15    FORMAT (I1,79A1)                                                  LOC31880
      IF (INMBR(ICOUNT).EQ.4) GO TO 12                                  LOC31890
C                         IF DATA BASE FILE HAS NOT BEEN FULLY ENTERED  LOC31900
      IF (INMBR(ICOUNT) .GT. 7) GO TO 19                                LOC31910
      ITEMP=ICOUNT-1                                                    LOC31920
C                   IF THE NEXT RECORD IS A DIFFERENT TYPE              LOC31930
      IF (INMBR(ICOUNT).NE.INMBR(ITEMP)) GO TO 10                       LOC31940
C                   BRANCH BASED ON CRITERIA,CLIENT, OR SITE RECORD     LOC31950
12    GO TO (16,17,18,80,100,100,100)INMBR(ICOUNT)                      LOC31960
C                   COUNT THE NUMBER OF CRITERIA                        LOC31970
16    NCRITS=NCRITS+1                                                   LOC31980
      WRITE(CRIT1(NCRITS),700)(LINER(ICOUNT,I),I=2,5)                   LOC31990
      WRITE(CRIT2(NCRITS),700)(LINER(ICOUNT,I),I=6,9)                   LOC32000
      GO TO 10                                                          LOC32010
C                   COUNT THE NUMBER OF CLIENTS                         LOC32020
17    NNCLI=NNCLI+1                                                     LOC32030
      WRITE(CLI1(NNCLI),700)(LINER(ICOUNT,I),I=2,5)                     LOC32040
      WRITE(CLI2(NNCLI),700)(LINER(ICOUNT,I),I=6,9)                     LOC32050
      GO TO 10                                                          LOC32060
C                   COUNT THE NUMBER OF SITES                           LOC32070
18    NSITES=NSITES+1                                                   LOC32080
      WRITE(SITE1(NSITES),700)(LINER(ICOUNT,I),I=2,5)                   LOC32090
                                                                        LOC32100
```

```
[redacted]LEV[redacted]2.1[redacted](SE[redacted]19[redacted]  [redacted]VS FORTRAN[redacted]   DATE: MAY 10, 1987    TIME: 21:40:49      NAME: MODIF
  IF DO    ISN       *....*...1.........2.........3.........4.........5.........6.........7.*.......8
           40              WRITE(SITE2(NSITES),700)(LINER(ICOUNT,I),I=6,9)                       LOC32110
           41              GO TO 10                                                              LOC32120
                 C                    OBTAIN LOCATION/CLIENT DEMANDS                             LOC32130
           42    80        BACKSPACE 1                                                           LOC32140
           43    82        READ(1,426)IDUM,CCLI1,CCLI2,DDST1(1),DDST2(1),                        LOC32150
                        1     DD(1),DDST1(2),DDST2(2),DD(2),DDST1(3),DDST2(3),DD(3)              LOC32160
                 C                IF MORE THAN ONE TYPE 4 RECORD FOR THIS CLIENT                 LOC32170
           44              IF (CCLI1.EQ.ISPACE .AND. CCLI2.EQ.ISPACE) GO TO 88                   LOC32180
                 C                WHICH CLIENT?                                                  LOC32190
           45              DO 85 I=1,NNCLI                                                       LOC32200
    1      46                IF (CCLI1.EQ.CLI1(I) .AND. CCLI2.EQ.CLI2(I)) GO TO 88               LOC32210
    1      47    85        CONTINUE                                                              LOC32220
                 C                       WHICH LOCATION?                                         LOC32230
           48    88        DO 95 K=1,3                                                           LOC32240
    1      49                 DO 90 J=1,NSITES                                                   LOC32250
    2      50                    IF (DDST1(K).EQ.SITE1(J) .AND. DDST2(K).EQ.SITE2(J))            LOC32260
    2                   1                  DMD(J,I)=DD(K)                                        LOC32270
    2      52    90           CONTINUE                                                           LOC32280
    1      53    95        CONTINUE                                                              LOC32290
           54              GO TO 10                                                              LOC32300
           55   100        GO TO 10                                                              LOC32310
           56   700        FORMAT (4A1)                                                          LOC32320
                 C                   ASK IF OPERATOR WANTS TO FINISH ENTERING DATA               LOC32330
                 C                           INTO AN UNFINISHED FILE                             LOC32340
           57    19        CALL CLRSCR                                                           LOC32350
           58              WRITE(6,20)                                                           LOC32360
           59    20        FORMAT (//' THIS DATA FILE IS INCOMPLETE. DO YOU WANT TO COMPLE',LOC32370
                        1'TE IT NOW?'//)                                                         LOC32380
           60    21        WRITE(6,22)                                                           LOC32390
           61    22        FORMAT (20X,'1: TO COMPLETE THE DATA FILE'/                           LOC32400
                        120X,'2: TO CHANGE PART OF THE EXISTING FILE'/                           LOC32410
                        120X,'3: TO RETURN TO THE PREVIOUS MENU')                                LOC32420
           62              READ(5,*)NEXT                                                         LOC32430
           63              CALL CHECK(3,NEXT,FLAG)                                               LOC32440
           64              GO TO (30,19)FLAG                                                     LOC32450
                 C                  BRANCH BASED ON DECISION TO FINISH OR UPDATE                 LOC32460
                 C                          INCOMPLETE FILE                                      LOC32470
           65    30        CALL CLRSCR                                                           LOC32480
           66              GO TO (45,200,900)NEXT                                                LOC32490
                 C                  FINISH ENTERING INTO INCOMPLETE DATA FILE                    LOC32500
                 C                       HOW FAR DID THEY GET BEFORE?                            LOC32510
           67    45        ITEMP=ICOUNT-1                                                        LOC32520
           68              JCNTR=INMBR(ITEMP)                                                    LOC32530
           69              JNUM = JCNTR + 1                                                      LOC32540
           70              WRITE(6,40)JCNTR,JNUM                                                 LOC32550
           71    40        FORMAT(//10X,'YOU PREVIOUSLY ENDED THE SESSION AT STEP',I2,'.',       LOC32560
                        1 /10X,'PLEASE START ENTERING NEW DATA AT STEP',I2,'.'//)                LOC32570
           72              PAUSE '     PRESS ENTER TO CONTINUE...'                               LOC32580
           73              BACKSPACE 1                                                           LOC32590
                 C                  GO TO 'CREATE NEW DATA BASE' SECTION BASED ON                LOC32600
                 C                          WHERE THEY LEFT OFF BEFORE                           LOC32610
           74              CALL CREATE(JNUM)                                                     LOC32620
           75              GO TO 900                                                             LOC32630
                 C                  CHANGE EXISTING INFORMATION                                  LOC32640
           76   200        WRITE(6,210)                                                          LOC32650
           77   210        FORMAT(//10X,'ENTER THE TYPE OF DATA YOU WISH TO MODIFY',             LOC32660
[redacted]LEV[redacted] 2.[redacted] (S[redacted] 1[redacted]) [redacted] [redacted]S F[redacted]RAN[redacted] [redacted] [redacted]E: [redacted]Y 1[redacted]198[redacted] [redacted]E: [redacted]40[redacted] [redacted]NA[redacted] MO
  IF DO   ISN       *....*...1.........2.........3.........4.........5.........6.........7.*.......8

                    1/20X,'1:  TO CHANGE CRITERIA VALUES',                                      LOC32670
                    1/20X,'2:  TO CHANGE LOCATION PARAMETERS',                                  LOC32680
                    1/20X,'3:  TO CHANGE DEMAND VALUES',                                        LOC32690
                    1//10X,'PLEASE ENTER 1, 2, OR 3:')                                          LOC32700
          78          READ(5,*)NEXT                                                             LOC32710
          79          CALL CHECK(3,NEXT,FLAG)                                                   LOC32720
          80          GO TO (212,200) FLAG                                                      LOC32730
          81   212    REWIND 1                                                                  LOC32740
          82          CALL CLRSCR                                                               LOC32750
          83          GO TO (220,320,400)NEXT                                                   LOC32760
               C                  CHANGE CRITERIA VALUES                                        LOC32770
          84   220    WRITE(6,222)                                                              LOC32780
          85   222    FORMAT(//10X,'PLEASE ENTER THE CRITERIA WITH VALUES TO BE ',              LOC32790
                    1 'CHANGED:')                                                               LOC32800
          86          READ(5,224)CCRIT1,CCRIT2                                                  LOC32810
          87          READ(1,223)IDUM                                                           LOC32820
          88   223    FORMAT(I1)                                                                LOC32830
          89   224    FORMAT(2A4)                                                               LOC32840
               C                 CHECK FOR VALID CRITERIA NAME, OBTAIN VALUES                   LOC32850
          90          DO 230 KT=1,NCRITS                                                        LOC32860
    1     91          READ(1,226)CCWT,CCID,CCLM,CCST                                            LOC32870
    1     92   226    FORMAT(20X,4F10.0)                                                        LOC32880
               C                 IF A VALID CRITERIA GO TO 240                                  LOC32890
    1     93          IF (CCRIT1.EQ.CRIT1(KT).AND.CCRIT2.EQ.CRIT2(KT)) GO TO 240                LOC32900
    1     94   230    CONTINUE                                                                  LOC32910
               C                 INVALID CRITERIA                                               LOC32920
          95          WRITE(6,232)                                                              LOC32930
          96   232    FORMAT(10X,'THIS CRITERIA NAME DOES NOT MATCH UP. ENTER AGAIN')           LOC32940
          97          REWIND 1                                                                  LOC32950
          98          GO TO 220                                                                 LOC32960
               C                 VALID CRITERIA ENTERED                                         LOC32970
          99   240    WRITE(6,242) CCRIT1,CCRIT2,CCWT,CCID,CCLM,CCST                            LOC32980
         100   242    FORMAT(//10X,'TO CHANGE A VALUE, ENTER THE FIELD NUMBER',                 LOC32990
                    1 /10X,'PRESENT VALUES FOR CRITERIA:',2A4,                                  LOC33000
                    1 //20X,'1.  WEIGHT: ',F10.0,                                               LOC33010
                    1  /20X,'2.   IDEAL: ',F10.0,                                               LOC33020
                    1  /20X,'3.   LIMIT: ',F10.0,                                               LOC33030
                    1  /20X,'4.    STEP: ',F10.0)                                               LOC33040
         101          READ(5,245)NEXT                                                           LOC33050
         102   245    FORMAT(I1)                                                                LOC33060
         103          CALL CHECK(4,NEXT,FLAG)                                                   LOC33070
         104          GO TO (248,240) FLAG                                                      LOC33080
               C                  BRANCH BASED ON VALUE TO BE CHANGED                           LOC33090
         105   248    GO TO (250,270,290,300)NEXT                                               LOC33100
         106   250    WRITE(6,255)                                                              LOC33110
               C                  TO CHANGE WEIGHTS                                             LOC33120
         107   255    FORMAT(//10X,'ENTER THE NEW VALUE FOR THE WEIGHT:')                       LOC33130
         108          READ(5,258)ZNEW                                                           LOC33140
         109   258    FORMAT(F10.0)                                                             LOC33150
         110          CCWT=ZNEW                                                                 LOC33160
         111          BACKSPACE 1                                                               LOC33170
               C                  REWRITE THE NEW RECORD                                        LOC33180
         112          WRITE(1,260)INMBR(KT),CCRIT1,CCRIT2,CCWT,CCID,CCLM,CCST                   LOC33190
         113   260    FORMAT(I1,2A4,11X,4F10.0)                                                 LOC33200
               C                  GO TO THE END                                                 LOC33210
         114          GO TO 900                                                                 LOC33220
```

```
 IF DO    ISN    *....*...1.........2.........3.........4.........5.........6.........7.*.......8

                 C                  CHANGE IDEALS                                           LOC33230
          115    270     WRITE(6,275)                                                       LOC33240
          116    275     FORMAT(5X,'ENTER THE NEW VALUE FOR THE IDEAL:')                    LOC33250
          117            READ(5,258)ZNEW                                                    LOC33260
          118            CCID=ZNEW                                                          LOC33270
          119            BACKSPACE 1                                                        LOC33280
                 C                  REWRITE THE NEW RECORD                                  LOC33290
          120            WRITE(1,260)INMBR(KT),CCRIT1,CCRIT2,CCWT,CC ID,CCLM,CCST           LOC33300
                 C                  GO TO THE END                                           LOC33310
          121            GO TO 900                                                          LOC33320
                 C                CHANGE LIMITS                                             LOC33330
          122    290     WRITE(6,295)                                                       LOC33340
          123    295     FORMAT(//10X,'ENTER THE NEW VALUE FOR THE LIMIT:')                 LOC33350
          124            READ(5,258)ZNEW                                                    LOC33360
          125            CCLM=ZNEW                                                          LOC33370
          126            BACKSPACE 1                                                        LOC33380
                 C                REWRITE THE NEW RECORD                                    LOC33390
          127            WRITE(1,260)INMBR(KT),CCRIT1,CCRIT2,CCWT,CC ID,CCLM,CCST           LOC33400
                 C                GO TO THE END                                             LOC33410
          128            GO TO 900                                                          LOC33420
                 C                CHANGE LIMIT STEP                                         LOC33430
          129    300     WRITE(6,305)                                                       LOC33440
          130    305     FCRMAT(//10X,'ENTER THE NEW VALUE FOR THE LIMIT STEP:')            LOC33450
          131            READ(5,258)ZNEW                                                    LOC33460
          132            CCST=ZNEW                                                          LOC33470
          133            BACKSPACE 1                                                        LOC33480
                 C                REWRITE THE NEW RECORD                                    LOC33490
          134            WRITE(1,260)INMBR(KT),CCRIT1,CCRIT2,CCWT,CC ID,CCLM,CCST           LOC33500
                 C                   GO TO THE END                                          LOC33510
          135            GO TO 900                                                          LOC33520
                 C                 TO CHANGE MAXIMUM SITE CAPACITIES                        LOC33530
          136    320     WRITE(6,322)                                                       LOC33540
          137    322     FORMAT(//10X,'ENTER THE LOCATION WITH VALUES TO BE CHANGED:')      LOC33550
          138            READ(5,224)CSITE1,CSITE2                                           LOC33560
                 C                 CHECK FOR VALID LOCATION NAME AND OBTAIN CAPACITIES      LOC33570
          139            JCNTR=0                                                            LOC33580
          140    325     JCNTR=JCNTR+1                                                      LOC33590
          141            READ(1,223)IDUM                                                    LOC33600
          142            IF (IDUM.NE.3) GO TO 325                                           LOC33610
          143            DO 330 KT=1,NSITES                                                 LOC33620
     1    144            READ(1,326)CNCAP                                                   LOC33630
     1    145    326     FORMAT(10X,F10.0)                                                  LOC33640
                 C                  IF LOCATION NAME IS VALID, CHANGE THE VALUE             LOC33650
     1    146            IF (CSITE1.EQ.SITE1(KT).AND.CSITE2.EQ.SITE2(KT)) GO TC 340         LOC33660
     1    147    330     CONTINUE                                                           LOC33670
                 C                  INVALID LOCATION NAME ENTERED                           LOC33680
          148            WRITE(6,332)                                                       LOC33690
          149    332     FORMAT(10X,'THIS LOCATION NAME DOES NOT MATCH ANY EXISTING ',      LOC33700
                      1  'LOCATION.'/10X,'PLEASE RE-ENTER:')                                LOC33710
          150            REWIND 1                                                           LOC33720
          151            GO TO 320                                                          LOC33730
                 C                 A VALID LOCATION NAME WAS ENTERED                        LOC33740
          152    340     WRITE(6,342) CSITE1,CSITE2,CNCAP                                   LOC33750
          153    342     FORMAT(//10X,'THE PRESENT CAPACITY FOR LOCATION ',2A4,':',         LOC33760
                      1  //20X,'MAXIMUM CAPACITY: ',F10.0)                                  LOC33770
          154            WRITE(6,355)                                                       LOC33780
```

```
[redacted] LE[redacted] 2.[redacted] ([redacted]T 1[redacted]) [redacted] [redacted]S F[redacted]RA[redacted] [redacted] [redacted]E:[redacted]Y [redacted]19[redacted] [redacted]E:[redacted]40[redacted] [redacted]NA[redacted] MC
  IF DO    ISN     *....*...1.........2.........3.........4.........5.........6.........7.*.......8

          155     355       FORMAT(//10X,'ENTER THE NEW CAPACITY:')                           LOC33790
          156               READ(5,358)ZNEW                                                    LOC33800
          157     358       FORMAT(F10.0)                                                      LOC33810
          158               CNCAP=ZNEW                                                         LOC33820
          159               BACKSPACE 1                                                        LOC33830
                  C                     WRITE THE NEW RECORD                                   LOC33840
          160               WRITE(1,360)IDUM,CSITE1,CSITE2,CNCAP                               LOC33850
          161     360       FORMAT(I1,2A4,1X,F10.0)                                            LOC33860
                  C                     GO TO THE END                                          LOC33870
          162               GO TO 900                                                          LOC33880
                  C                        TO CHANGE CLIENT/SITE DEMAND                        LOC33890
          163     400       WRITE(6,402)                                                       LOC33900
          164     402     FORMAT(//10X,'ENTER THE CLIENT WHICH HAS DEMANDS TO BE CHANGED:')    LOC33910
          165               READ(5,224)DCLI1,DCLI2                                             LOC33920
                  C                     CHECK FOR A VALID CLIENT NAME                          LOC33930
          166     407       JCNTR=JCNTR+1                                                      LOC33940
          167               READ(1,223)IDUM                                                    LOC33950
          168               IF (IDUM.NE. 4) GO TO 407                                          LOC33960
          169               BACKSPACE 1                                                        LOC33970
          170               DO 415 J=1,NNCLI                                                   LOC33980
     1    171     408          READ(1,410)CCLI1,CCLI2                                          LOC33990
     1    172                  JCNTR=JCNTR+1                                                   LOC34000
     1    173     410          FORMAT(1X,2A4)                                                  LOC34010
                  C          IF VALID CLIENT NAME ENTERED, OBTAIN THE DESIRED LOCATION         LOC34020
     1    174                  IF (CCLI1.EQ.DCLI1 .AND. CCLI2.EQ.DCLI2) GO TO 420              LOC34030
     1    175                  IF (CCLI1 .EQ. ISPACE .AND. CCLI2.EQ.ISPACE) GO TO 408          LOC34040
     1    176     415       CONTINUE                                                           LOC34050
                  C                     INVALID CLIENT NAME ENTERED                            LOC34060
          177               WRITE(6,418)DCLI1,DCLI2                                            LOC34070
          178     418       FORMAT(/5X,2A4,//10X,'THIS CLIENT DOES NOT MATCH ANY EXISTING',    LOC34080
                         1  ' CLIENTS.  PLEASE ENTER AGAIN:')                                  LOC34090
          179               REWIND 1                                                           LOC34100
          180               JCNTR=0                                                            LOC34110
          181               GO TO 400                                                          LOC34120
          182     420       WRITE(6,422)DCLI1,DCLI2                                            LOC34130
          183     422       FORMAT(//10X,'ENTER THE LOCATION DEMAND TO BE CHANGED FOR CLIE',   LOC34140
                         1  'NT ',2A4)                                                         LOC34150
          184               READ(5,224) DSITE1,DSITE2                                          LOC34160
          185     425       BACKSPACE 1                                                        LOC34170
                  C                     CHECK FOR VALID LOCATION NAME                          LOC34180
          186               READ(1,426)IDUM,CCLI1,CCLI2,DDST1(1),DDST2(1),DD(1),               LOC34190
                         1  DDST1(2),DDST2(2),DD(2),DDST1(3),DDST2(3),DD(3)                   LOC34200
          187     426       FORMAT(I1,2A4,1X,3(2A4,2X,F10.2))                                  LOC34210
          188               DO 430 I=1,3                                                       LOC34220
                  C                     IF LOCATION NAME IF VALID, GET THE DEMAND              LOC34230
     1    189                  IF (DDST1(I).EQ.DSITE1 .AND.DDST2(I).EQ.DSITE2) GO TO 438       LOC34240
     1    190     430       CONTINUE                                                           LOC34250
          191               READ(1,224)DDUM1,DDUM2                                             LOC34260
          192               IF (DDUM1.EQ.ISPACE .AND. DDUM2.EQ.ISPACE) GO TO 425               LOC34270
                  C                     INVALID LOCATION NAME ENTERED                          LOC34280
                  C                     MUST RE-ENTER CLIENT,LOCATION NAMES                    LOC34290
          193               WRITE(6,432)DSITE1,DSITE2,DCLI1,DCLI2                              LOC34300
          194     432       FORMAT(//10X,'LOCATION ',2A4, 'DOES NOT MATCH ANY LOCATIONS ',     LOC34310
                         1  'GIVEN WITH CLIENT ',2A4,                                          LOC34320
                         1  /10X, 'PLEASE RE-ENTER CLIENT AND LOCATION NAMES')                 LOC34330
          195               REWIND 1                                                           LOC34340
```

```
LEVEL 2.1.1 (SEPT 1986)          VS FORTRAN             DATE: MAY 10, 1987        TIME: 21.40.          NAME: MODI
  IF DO      ISN    *....*...1.........2.........3.........4.........5.........6.........7.*.......8

             196               JCNTR=0                                                            LOC34350
             197               GO TO 400                                                          LOC34360
                    C                      VALID LOCATION NAME ENTERED                            LOC34370
             198    438        WRITE(6,440)DCLI1,DCLI2,DSITE1,DSITE2,DD(I)                        LOC34380
             199    440        FORMAT(//10X,'FOR CLIENT ',2A4,' AND SITE ',2A4,':',               LOC34390
                         1     /20X,'DEMAND = ',F10.2,//10X,                                      LOC34400
                         1     'ENTER THE NEW DEMAND:')                                           LOC34410
             200               READ(5,442)ZNEW                                                    LOC34420
             201    442        FORMAT(F10.2)                                                      LOC34430
             202               DD(I)=ZNEW                                                         LOC34440
             203               BACKSPACE 1                                                        LOC34450
                    C                      REWRITE THE NEW RECORD                                 LOC34460
             204               WRITE(1,426)IDUM,CCLI1,CCLI2,(DDST1(I),DDST2(I),DD(I),I=1,3)       LOC34470
                    C                      GO TO THE END                                          LOC34480
             205               GO TO 900                                                          LOC34490
             206    900        CONTINUE                                                           LOC34500
             207               RETURN                                                             LOC34510
             208               END                                                                LOC34520
*STATISTICS*    SOURCE STATEMENTS = 207, PROGRAM SIZE = 10956 BYTES, PROGRAM NAME = MODIFY    PAGE:    80.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

**MODIFY** END OF COMPILATION 29 ******
```