CHAPTER XIX: COMPUTERIZED AIDS FOR PROGRAMMING STUDIES

Data manipulation, model specification, debugging, validation and experimentation within moderate sized programming models can be complex. Computerized procedures may facilitate these tasks. This chapter reviews ways computerized procedures may facilitate programming studies. In addition, material on modeling with GAMS will be presented.

Given the model structure and data, computerized procedures may be used to retrieve necessary data items; calculate coefficients; prepare equations for solver input; check numerical formulations for errors; summarize solution content; debug the model; predict implications of data alterations; and execute model validation tests. Many of these options are available within modeling systems such as GAMS, but some are not and a more general discussion is in order.

Before discussing computerized approaches, three caveats are pertinent:

1) This chapter assumes that a programming formulation is the appropriate analysis tool.

2) The chapter is based upon the authors' experience and does not reflect experience with packages such as OSL or ANALYZE (Greenberg, 1991).

3) The chapter is oriented toward solving moderate to large models (more than 100 rows and/or columns).

19.1 Model Generation Tools

Numerical programming models must be constructed in a form consistent with solver input requirements. Construction of the numeric formulation is commonly called matrix generation. Problems of the size dealt with herein virtually require a computerized matrix generator (MG). Williams (1978a, pg. 35) argues that "the clerical problems (of building moderate to large models by hand) almost always become prohibitive."

Different types of MGs can be employed. These run the continuum from simple specific model MGs to MGs designed to facilitate direct model use by decision makers (see McCarl and Nuthall or McCarl, et al.) on to general purpose matrix generators as in modeling systems (i.e., GAMS-Brooke, et al.). The fundamental distinction made here is between custom, problem specific and general purpose modeling system based MGs.

19.1.1 Potential Functions of a Matrix Generator

Potentially there are several functions MGs can perform. We group these into input preparation, formulation manipulation, documentation and analysis facilitation.

19.1.1.1 Input Preparation

Fundamentally MGs prepare solver input from data and equation specifications. Programming models usually permit a large model to be generated from a smaller set of input (e.g., Kutcher and Meeraus state that 400 lines of data and a 1000 statement FORTRAN matrix generator yielded an 80,000 line numerical model file). Matrix generators calculate and place coefficients avoiding clerical errors.

19.1.1.2 Formulation Manipulation

Initial formulations are almost always different from final formulations due to omissions, errors, improper data or oversights. Model based analysis usually involves new and altered equations and variables, as well as data changes. MGs enable rapid, efficient modification of model structure and/or data usually requiring modification of a few commands or input tables.

19.1.1.3 Documentation

Often, it is very hard for someone other than the model author to use a model. Documentation is often inadequate. MG use can partially solve the documentation problem. MGs utilize specific data, carry out data transformations, and place the coefficients. This contributes to documentation. Nevertheless, accompanying documentation is still important.

19.1.1.4 Analysis Facilitation

MGs may facilitate analysis reducing the time to do a run and correct errors. Thus, a more extensive analysis may be done (since it is easier to do individual runs). Other possible MG functions include: suggesting an advanced basis (Dillon); resolving degeneracies (McCarl, 1977); resolving alternative optimals (McCarl et al., 1977); checking for inconsistent or improper data -- when the modeler codes in such checks; aiding in the discovery of causes of infeasibility (adding artificial variables) or unboundedness (by adding large upper bounds); and automatically setting up approximations of nonlinear phenomena (i.e., generating a separable programming approximation).

19.1.2 When not to use a Matrix Generator

MGs are not always appropriate. MGs should only be used when the time required to do all analysis with the MG is less than or equal to the time to do the analysis without it. (This is an elusive criterion since many uncertain factors enter.) Thus, if the analysis may be done efficiently without a MG, don't use one. This criterion also applies to the choice between a generalized (e.g., GAMS) and a problem tailored MG. Problem specific MGs are more time consuming to construct and can be difficult to change because of accompanying documentation and complexity of the implementation. This situation could well lead to an analysis done with the wrong model. A matrix generator should be used with the attitude that it will change to accommodate the problem rather than vice versa.

19.2 Pre-Solution Formulation Analysis Tools

The first problem formulation is rarely the final formulation used in the study. Computerized tools may facilitate the identification of formulation difficulties. Unfortunately, GAMS does not contain much in the way of such tools although GAMSCHK (McCarl 1994) does contain such tools. The primary aid in verification is a "picture" routine which generates a schematic of a programming problem (Orchard-Hays). A page of "picture" output (e.g., Figure 19.1) typically represents 50-60 columns and 40-50 rows. Characters on the page indicate the location, sign and magnitude of a coefficient. A picture thereby allows verification of whether coefficients with proper sign and magnitude are in their proper places.

Preanalysis routines may also generate data on problem scaling, and check for obvious structural defects. Finally, presolution formulation analysis may use redundancy identification techniques (Luenberger, Ch. 6) which are an integral part of some large commercial packages (e.g., OSL automatically reduces the problem).

19.3 Solution Interpretation Aids

Computerized techniques can be developed to aid solution interpretation. They can be used to summarize solutions, aid in solution debugging, or augment solution information.

19.3.1 Solution Summarization

The primary solution summarization tool is the report writer (RW). A RW takes solution information and reformats it into tables for decision makers.[1] Linear programming solver output is usually an inadequate description of a solution. Substantive summary reports are usually necessary and may be the only effective way to involve nontechnical decision makers in model use. Report writers are frequently based upon rudimentary accounting procedures involving data tabulation, labeling and organization. (McCarl and Nuthall discuss report writers at length.)

RWs may be either problem specific or general purpose. The arguments for a problem specific report writer are weaker than those for a matrix generator. This is due to both the ability to add accounting equations summing up variables of interest and the report writing features of modeling systems like GAMS (see the discussion of GAMS report writing later in this chapter). The proper trade-off in selecting whether to develop a problem specific report writer is between time required to do the whole analysis with the report writer versus other methods (hand and/or generalized report writers).

19.3.2 Solution Debugging

Post-solution computational aids may also help in debugging "bad" solutions. Often, the analyst has doubts about the validity of an LP solution. Specifically, three potential questions are asked: 1) Why is a particular variable in (or not in) the solution?; 2) Why is a particular shadow price so large?; and 3) Why does a particular row exhibit slack resources? The resolution of these questions revolves around the budgeting and row summing procedures discussed above. These items are easily computerized and have

[1] For example, see the GAMS report writing example above, or McCarl et al.

been in GAMSCHK (McCarl 1994).

19.3.3 Solution Interpretation

Post solution computational aids can enhance or augment solution information. Linear programming solver output typically consists of solution levels, shadow prices, reduced costs and slacks. Less often, cost and right hand side ranges are obtained. Post optimality routines (or programs which request parametric analysis) may provide additional information (e.g., the impact of ranging a number of right hand sides simultaneously). Similarly, calculations may be done on, or requests generated for, information from the final tableau. (In larger problems this information is virtually never available and may be difficult to calculate. Although, it is sometimes quite useful.)

19.4 Getting the most from GAMS There are a number of GAMS features which can facilitate a programming exercise. These features will be discussed under the topics: setting up data, changing model structure, providing an advanced basis, report writing, debugging models, conducting a comparative model analysis, sensitivity analysis, speeding up GAMS, minimizing model size, and avoiding GAMS failures.

19.4.1 Setting Up Data

GAMS has excellent facilities for data entry and computation as well as the ability to document data. We recommend that users enter primary data with comments on data sources and long labels. In turn, one should then calculate the needed data. External calculations are usually inadvisable as they are difficult to trace at a future time. For example, in the transportation example the distance and the formula relating distance to cost were entered with the costs computed. Users following such a strategy would find it easy to alter raw data assumptions and reanalyze the problem.

19.4.2 Changing Model Structure

One of the big advantages of using a modeling system is the ability to add constraints or variables and reanalyze the problem. However with large models or comparative studies, it may be desirable to make equations or terms in equations temporarily active or inactive. This can be achieved with $ controls. For example if the following lines were put in a GAMS problem

```
      SCALAR   ISITACTIVE     tells whether items are active /0/
      CONDEQ$ISITACTIVE..     sum(stuff,x(stuff)) =L= 1;
      EQNOTH(index)..         sum(stuff,r(index,stuff)*x(stuff)) +
                              4*sum(stuff,Y(stuff))$ISITACTIVE =L= 50;
```

they would cause the CONDEQ equation and the Y term in the EQNOTH equations to only appear in the empirical model when the ISITACTIVE parameter was nonzero. Thus, the sequence

```
      ISITACTIVE = 0;
      SOLVE MODELNAME USING LP MAXIMIZING OBJ;
      ISITACTIVE =1;
      SOLVE MODELNAME USING LP MAXIMIZING OBJ;
```

would cause the model to be solved with and without the constraint and term.

19.4.3 Providing an Advanced Basis

The importance of an advanced basis has long been recognized in mathematical programming. Models which take more than 24 hours to solve from scratch, can be solved in less than 3 hours from a good basis. GAMS can accept an advanced basis under certain conditions. In any model involving a sequence of solves, GAMS automatically uses the basis from the first solve to restart the second and later solves. But this does not carry over to models solved from scratch.

So what can you do when solving a model from scratch? Three possibilities are available.

1. One can let GAMS solve without worrying with the basis. This should certainly be done for small models.
2. One can try to suggest a basis by setting the levels and marginals for the variables and the marginals for the equations. For example the statements

```
            X.L("VAR1") = 10;
            X.L("VAR2") = 0;
            X.M("VAR1") = 0;
            X.M("VAR2") = 10;
            CONSTRAINT.M("RES1")=20;
            CONSTRAINT.M("RES2")=0;
```

   would result in a basis being suggested with the first variable and the slack from the second constraint.

3. One can save the variable and equation information from previous solves and restart including that information by setting the levels and marginals as above (a small example is given in the context of the Chapter 5 DIET problem in the files SAVBASIS and ADVBASIS) GAMSBAS (McCarl 1994a) does this automatically.

19.4.4 Report Writing

GAMS report writing has been discussed before in this text. However, a few additional comments are relevant. The normal GAMS output format is not always desirable. All *.lst files have the GAMS instructions listed first and may be followed by: a cross reference map; output from pre-solution displays; the equation and variable listings; the solver output; and post-solution displays. Several methods may be employed to manage this output. One may

1. Suppress the cross reference map by using the command

```
$OFFSYMLIST OFFSYMXREF
```

2. Suppress the variable and equation listings by using

```
OPTION LIMROW = 0;

OPTION LIMCOL = 0;
```

3. Limit the amount of GAMS code echoed back by restarting the job with a very short file containing display statements only. For example the batch file (REDUCED.BAT)

```
GAMS MAINPART S=SAVED

GAMS FINALOUT R=SAVED
```

   does this where MAINPART contains all the model statements (in the example on the disc these are from the file EVPORTFO) and FINALOUT is just a display of the relevant output. This generates a much smaller FINALOUT.LST file.

4. Suppress the solution information using

```
OPTION SOLPRINT=OFF;
```

5. Suppress the listing of lines between $OFFLISTING and $ONLISTING commands

One may also wish to control the GAMS output ordering and format. In this context one should be aware that

a. The order in which things appear in terms of elements of sets, variables etc. is determined by the order in which GAMS saw those items. In particular the order that the variables appear in the equation, variable and solution listings depends on the ordering of the VARIABLES declarations. Thus, if you want the TRANSPORT variable to proceed the DEMAND and OBJ variables you need to declare them in that order. The same holds for the equations.

   The ordering of set elements is somewhat more tricky and depends on the order in which the set elements were seen across all sets. Thus the set statements

```
SETS    ONE     / SALT, PROTEIN , TOTAL/

        TWO    / INDUSTRIAL , MUNICIPAL, TOTAL/
```

   would result in any listing over data indexed by the set TWO appearing with the TOTAL element first since that word appeared first in set definitions before the other words.

   The ordering of items in a parameter defined over multiple sets is also an issue. GAMS output is ordered with the sets varied from left to right. Thus, when displaying X(A,B,C) A will vary the slowest. The sets should be ordered in the pattern desired.

b. The page width and page length are at the users control. Both can be specified in the GAMS initiation command. For example, the command

```
GAMS FILE PW=130 PS=9999
```

   would result in 130 character wide output arrayed in 9,999 line long pages.

c. One can control the formatting of output from a display statement using two option commands. The command

```
OPTION DECIMALS = 0;
```

   would result in all displayed numbers being rounded to zero decimals (although GAMS still insists on reporting all decimals for numbers less than one).[2] The command

```
OPTION TRADE:3:2:1;DISPLAY "TRADE FLOWS", TRADE;
```

[2] One can overcome this using the command OUTPUTITEM$(OUTPUTITEM GE 0.5) =OUTPUTITEM before the display statement.

would result in the data in the TRADE array being displayed with three indices displayed in the rows, two in the column headings, and the data would appear with one decimal place. Further, the label TRADE FLOWS would precede the display in the output.

d. Users may create much more customized output files using the GAMS "PUT" command. This is a rather extensive feature (GAMS Development Corporation, 1992) but can generate output such as in Table 19.1 which constitutes a reformatting of the Report Writer output as tabled in chapter 6 (this is generated by the file PUTEXPL)

19.4.5 Debugging Models

GAMS allows one to work with large models. PC based GAMS can be used to solve problems with hundreds of thousands of variables and tens of thousands of equations. However, debugging formulations that big is not easy. The debugging aids in most commercial systems are not available within GAMS.[3] However, the algebraic structure of GAMS does provide one with the ability to **work from small to large**. This should always be exploited. For example, the full structure of a GAMS transportation problem implementation for a problem with N destinations and M sources can be worked out on a 2x2 problem. Data calculations, model equations, report writing tables, comparative model analysis procedures etc. can be worked out in a fast turn around limited output situation. Later the full data set can be entered.

GAMS supports several other debugging techniques:

1. One may examine the individual rows or equations within the model using the option statements for variable and equation listing. For example, the commands:

```
OPTION LIMROW=10;
OPTION LIMCOL=10;
```

would result in the print out of the first 10 elements in each equation and variable block whenever a SOLVE is executed. One may control the ordering of these items so the proper

[3] Moves are now afoot to correct that situation with the development of a GAMS interfaced version of ANALYZE (Greenberg, 1991) and GAMSCHECK (McCarl, 1994).

items appear first by ordering the set elements and variable/equation names as discussed in the output control section above.

2. One may hand generate versions of the tests in the FIXING MODELS chapter using GAMS calculations. For example, in a resource allocation model the lines below could be used to test for unbounded variables.
```
PARAMETER UNBOUN(J)  equals 1 if variable j is unbounded;
UNBOUN(J)=1$(C(J) GT 0 AND SUM(I,1$(A(I,J) GT O) LE 0);

DISPLAY UNBOUN;
```
Similarly, after solution, the following code would provide a budget analysis
```
PARAMETER BUDGET(J,*,*)  BUDGET OF COLUMN J;
BUDGET(J,I,"AIJ")=A(I,J);
BUDGET(J,I,"SHADOWPRIC")=CONSTRAINT.M(I);
BUDGET(J,I,"PRODUCT")=A(I,J)*CONSTRAINT.M(I);
      BUDGET(J,"SUMINDIRCT","PRODUCT")=
      SUM(I,BUDGET(J,I,"PRODUCT"));
BUDGET(J,"OBJECTIVE","PRODUCT")=C(J);
      BUDGET(J,"REDUCECOST,"PRODUCT")=
      BUDGET(J,"SUMINDIRCT","PRODUCT")-
      BUDGET(J,"OBJECTIVE","PRODUCT");

DISPLAY BUDGET;
```

However, such coding is specific to a structure and is easier done using auxiliary aids such as those in McCarl (1994b); and Greenberg.

19.4.6 Conducting a Comparative Model Analysis

Models, once setup, are usually employed in a comparative statics analysis. Such an exercise involves repeated solutions of the same problem. There are several GAMS features which are relevant in such a setting.

1. More than one model can be solved in a run. Thus, one can stack solve statements or loop over solves as in the LP modeling DIET and risk EVPORTFO example models.
2. When solving multiple model versions one needs to be careful with data revisions as the values in the model once changed remain so. For example the commands
```
SCALAR LAND /100/
PARAMETER SAVELAND;
SAVELAND = LAND;
      SET LANDCHANGE    SCENARIOS FOR CHANGES IN LAND
```

```
                                 /R1,R2,R3/
PARAMETER VALUE(LANDCHANGE) PERCENT CHANGE IN LAND
            /R1 10 , R2  20 , R3 30/
LOOP ( LANDCHANGE,

      LAND = LAND * (1 + VALUE ( LANDCHANGE ) / 100. ) );
```

results in land equalling 110, 132 and 171.6 during the loop. However, alteration of the calculation statement, so it operated from a saved parameter value

```
LAND = SAVELAND * (1 + VALUE ( LANDCHANGE ) / 100. )
```

result in values of 110 , 120, and 130.

3. The development of a comparative report writer may be attractive when doing multiple runs. Such a report writer is illustrated in the Risk chapter EVPORTFO example and is in TABLE 19.2. In that case a parameter is defined over the loop set - OUTPUT(*,RAPS). In turn, during loop execution the OUTPUT array is saved with scenario dependent values of variables, shadow prices, data etc. Finally, when the output is displayed a comparison across scenarios appears.

19.4.7 Sensitivity Analysis

A number of users are interested in getting sensitivity analysis information from GAMS usually in the form of LP ranging analysis results. Unfortunately, the base version of GAMS does not yield such information. The user wishing for such information has two alternatives. First, one may cause the model to be repeatedly solved varying a parameter and examine the results (as in Table 19.2). Second, one can use solver dependent features of GAMS (which currently work with OSL or CPLEX) and retrieve the ranging information (GAMS Development Corporation , 1993)

19.4.8 Speeding up GAMS

One can speed up GAMS execution time. Models with a lot of subscripts and a lot of dimensions to those subscripts can be quite slow in performance. The use of $ conditions in such models is essential. For example the report writing equation

```
Y=SUM((A,B,C,D,E,F,G),
(DAT(A)+IT(B,C)+Y(D,E)+W(F,G))*X.L(A,B,C,D,E,F,G))
```

will perform much faster with the addition of a $ condition as follows

```
        Y=SUM((A,B,C,D,E,F,G$X.L(A,B,C,D,E,F,G),
           (DAT(A)+IT(B,C)+Y(D,E)+W(F,G))*X.L(A,B,C,D,E,F,G));
```

this will result in the calculation only being done when nonzero values are involved and will avoid excess work.[4] The ordering of subscripts is also important where the data arrays should be referenced in an order consistent with their definition. For example, summing the above in the order F,D,A,C,E,B,G would be much slower. Also one should compute intermediate products to avoid repetitive and complex calculations (i.e., one could add the DAT and IT items into another parameter ahead of time if they were frequently added). GAMS also gives help in reporting particularly slow statements. During execution, a report appears on the screen giving the line being executed, and one can observe progress making notes of statements which are computed for a long time to see if they can be streamlined. Also one may use the undocumented PROFILE feature which produces a report of the time spent on code segments[5].

19.4.9 Minimizing Model Size

GAMS can generate very large problems when models contain a lot of sets with many elements. The statements above on speeding up GAMS are also relevant when setting up models. It is usually highly desirable to define $ conditions on equations and the sums leading to generation of variables to avoid unneeded model features.

19.4.10 Avoiding GAMS Failures

Finally we need to mention strategies for avoiding GAMS solver failures. Solver failures generally happen because of numerical difficulties caused by scaling, degeneracy, or nonlinearities. Cycling without making progress can often be resolved by adding small numbers to the right hand sides

[4] Such a modification reduced run time for a report writer from 2 hours to 10 minutes.

[5] This is invoked by including the OPTION PROFILE=1, or including PROFILE=1 on the GAMS call.

and/or scaling. Both of these procedures are explained in the fixing models chapter. Gradient problems or a lack of progress may require separable approximation of nonlinear problems as explained in the LP approximation chapter. Improved starting values for the nonlinear variables can also help.

References

Brooke, A., D. Kendrick, and A. Meeraus. GAMS: A User's Guide. Boyd and Fraser Publishers, Version 2.25, 1993.

Dillon, M., "Heuristic Selection of Advanced Bases for a Class of Linear Programming Models," Operations Research, 18(1970):90-100.

GAMS Development Corporation. "Sensitivity Analysis with GAMS/CPLEX and GAMS/OSL." Washington, DC, 1993.

GAMS Development Corporation. "Guide to the 'Put' Writing Facility." Washington, DC, 1990.

Greenberg, H.J. "A Primer for ANALYZE$^{(c)}$: A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions." Mathematics Department, University of Colorado at Denver, Denver, CO, February 1991.

Kutcher, G. and A. Meeraus, "Computational Considerations for Sectoral Programming Models," in The Book of CHAC: Programming Studies for Mexican Agriculture. R. Norton and L. Solis, (Eds.), Johns Hopkins Press for the World Bank, 1983.

Luenberger, D., Introduction to Linear and Nonlinear Programming, Addison Wesley, 1973.

McCarl, B.A. "GAMSCHECK USER DOCUMENTATION: A System for Examining the Structure and Solution Properties of Linear Programming Problems Solved using GAMS." Working Documentation, Department of Agricultural Economics, Texas A&M University, 1994.

McCarl, B.A. "Degeneracy, Duality and Shadow prices in Linear Programming," Canadian Journal of Agricultural Economics, 25(1977):70-73.

McCarl, B.A., W.V. Candler, D.H. Doster and P. Robbins, "Experiences with Farmer Oriented Linear Programming for Crop Planning," Canadian Journal of Agricultural Economics, 25(1977):17-30.

McCarl, B.A. and P. Nuthall, "Linear Programming for Repeated use in the Analysis of Agricultural Systems," Agricultural Systems, 8(1982):17-39.

Optimization Subroutine Library (OSL). IBM Guide and Reference, Release 2. Ed. J.A. George and J.W.H. Liu. IBM Corporation, 1990-91.

Orchard-Hays, W., Advanced Linear-Programming Computing Techniques, McGraw-Hill Book Company, 1968.

Williams, H., Model Building in Mathematical Programming, John Wiley & Sons, 1978.

Figure 19.1 Sample Picture of BLOCKDIAG Problem

```
GAMSCHECK PICTURE -  COEFFICIENT CODES
  LOWER BOUND   CODE     UPPER BOUND
   (INCLUSIVE)            (LESS THAN)
    1000.00000      G     +INFINITY
     100.00000      F      1000.00000
      10.00000      E       100.00000
       1.00000      D        10.00000
       1.00000      C         1.00000
        .50000      B         1.00000
        .00000      A          .50000
        .00000      0          .00000
       -.50000      1          .00000
      -1.00000      2         -.50000
      -1.00000      3        -1.00000
     -10.00000      4        -1.00000
    -100.00000      5       -10.00000
   -1000.00000      6      -100.00000
   -INFINITY        7     -1000.00000
```

```
            |                         M M M M M M                                 N     H     O I   E I   O
            |                         A A A A A A T T T T T T                     E     S     S J   G J   W
            |                         K K K K K K R R R R R R                     T           I ,   A ,
            |                         E E E E E E N N N N N N                     I     C     T S   T S   C
            |                         T T T T T T S S S S S S                     N     O     I     I     N
            | M A K E C H A I R M A K A A A A A A P P P P P P S E L L S E L L S E C     E     V     V     T
            |                         B B B B B B O O O O O O                     O     F     E     E     S
            |                         L L L L L L R R R R R R                     M     F
            |                         E E E E E E T T T T T T                     E     S
            |                   1 1 1                                           1
            | 1 2 3 4 5 6 7 8 9 0 1 2 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 7 8 9 0 1
             ----------------------------------------------------------------------------
      OBJT 1| E E E E E E E E E E E E E F     E F D D E E D D 6 6 6 7 5 6 6 6 5 6 C   = 0     2 3   1 0   3 3
     R     1|                         D D                                             < F       2     0     2
     E     2|                         C C                                             < E       2     0     2
     S     3| B D A D D B                                                             < F       6     0     6
     O     4| B A D B A D                                                             < E       6     0     6
     U     5| A A A C C C                                                             < F       6     0     6
     R     6| C D D B B B                 D D                                         < F       8     0     8
     E     7|                             C C                                         < 0       2     0     2
     Q     8|             B D A D D B                                                 < F       6     0     6
     R     9|             B A D B A D                                                 < F       6     0     6
     E    10|             A A A C C C                                                 < F       6     0     6
     S    11|             C D D B B B         D D                                     < F       8     0     8
     O    12|                                 C C                                     < E       2     0     2
 LINKTABLE 1|                         3               3       C   C                   < 0       2     2     4
 LINKTABLE 2|                           3               3       C   C                 < 0       2     2     4
 LINKCHAIR 1|                                     3       3       D                   < 0       1     2     3
 LINKCHAIR 2|                                       3       3       D                 < 0       1     2     3
TRNCHAIREQ 1| 3 3 3                               C                   C               < 0       2     3     5
TRNCHAIREQ 2|       3 3 3                           C                   C             < 0       2     3     5
TRNCHAIREQ 3|             3 3 3                           C                   C       < 0       2     3     5
TRNCHAIREQ 4|                   3 3 3                       C                   C     < 0       2     3     5
TRNTABLEEQ 1|                                 3       C                   C           < 0       2     1     3
TRNTABLEEQ 2|                                   3       C                   C         < 0       2     1     3
            |----------------------------------------------------------------------------
POSITIVE    | 5   5   5   5   5   5   3   2   3   2   2   2   1   2   1   1   1   1
COLUMN CTS  |   5   5   5   5   5   5   3   2   3   2   2   2   1   2   1   1   1
NEGATIVE    | 1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   0
COLUMN CTS  |   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1
COLUMN      | 6   6   6   6   6   6   4   2   4   3   3   3   2   3   2   2   2   1
COUNTS      |   6   6   6   6   6   6   4   2   4   3   3   3   2   3   2   2   2
             ----------------------------------------------------------------------------
```

Figure 19.1. Sample Picture of BLOCKDIAG Problem(continued)

```
#### Dictionary of Variables
         M   1: MAKECHAIR(PLANT2,FUNCTIONAL,NORMAL)
         A   2: MAKECHAIR(PLANT2,FUNCTIONAL,MAXSML)
         K   3: MAKECHAIR(PLANT2,FUNCTIONAL,MAXLRG)
         E   4: MAKECHAIR(PLANT2,FANCY,NORMAL)
         C   5: MAKECHAIR(PLANT2,FANCY,MAXSML)
         H   6: MAKECHAIR(PLANT2,FANCY,MAXLRG)
         A   7: MAKECHAIR(PLANT3,FUNCTIONAL,NORMAL)
         I   8: MAKECHAIR(PLANT3,FUNCTIONAL,MAXSML)
         R   9: MAKECHAIR(PLANT3,FUNCTIONAL,MAXLRG)
         M  10: MAKECHAIR(PLANT3,FANCY,NORMAL)
         A  11: MAKECHAIR(PLANT3,FANCY,MAXSML
         K  12: MAKECHAIR(PLANT3,FANCY,MAXLRG)
MAKETABLE    1: MAKETABLE(PLANT1,FUNCTIONAL)
MAKETABLE    2: MAKETABLE(PLANT1,FANCY)
MAKETABLE    3: MAKETABLE(PLANT2,FUNCTIONAL)
MAKETABLE    4: MAKETABLE(PLANT2,FANCY)
MAKETABLE    5: MAKETABLE(PLANT3,FUNCTIONAL)
MAKETABLE    6: MAKETABLE(PLANT3,FANCY)
TRNSPORT     1: TRNSPORT(PLANT2,CHAIRS,FUNCTIONAL)
TRNSPORT     2: TRNSPORT(PLANT2,CHAIRS,FANCY)
TRNSPORT     3: TRNSPORT(PLANT3,TABLES,FUNCTIONAL)
TRNSPORT     4: TRNSPORT(PLANT3,TABLES,FANCY)
TRNSPORT     5: TRNSPORT(PLANT3,CHAIRS,FUNCTIONAL)
TRNSPORT     6: TRNSPORT(PLANT3,CHAIRS,FANCY)
         S   1: SELL(PLANT1,TABLES,FUNCTIONAL)
         E   2: SELL(PLANT1,TABLES,FANCY)
         L   3: SELL(PLANT1,DINSETS,FUNCTIONAL)
         L   4: SELL(PLANT1,DINSETS,FANCY)
         S   5: SELL(PLANT2,CHAIRS,FUNCTIONAL)
         E   6: SELL(PLANT2,CHAIRS,FANCY)
         L   7: SELL(PLANT3,TABLES,FUNCTIONAL)
         L   8: SELL(PLANT3,TABLES,FANCY)
         S   9: SELL(PLANT3,CHAIRS,FUNCTIONAL)
         E  10: SELL(PLANT3,CHAIRS,FANCY)
NETINCOME    1: NETINCOME

#### Dictionary of Equations
OBJT         1: OBJT
         R   1: RESOUREQ(PLANT1,LABOR)
         E   2: RESOUREQ(PLANT1,TOP)
         S   3: RESOUREQ(PLANT2,SMLLATHE)
         O   4: RESOUREQ(PLANT2,LRGLATHE)
         U   5: RESOUREQ(PLANT2,CARVER)
         R   6: RESOUREQ(PLANT2,LABOR)
         E   7: RESOUREQ(PLANT2,TOP)
         Q   8: RESOUREQ(PLANT3,SMLLATHE)
         R   9: RESOUREQ(PLANT3,LRGLATHE)
         E  10: RESOUREQ(PLANT3,CARVER)
         S  11: RESOUREQ(PLANT3,LABOR)
         O  12: RESOUREQ(PLANT3,TOP)
LINKTABLE    1: LINKTABLE(FUNCTIONAL)
LINKTABLE    2: LINKTABLE(FANCY)
LINKCHAIR    1: LINKCHAIR(FUNCTIONAL)
LINKCHAIR    2: LINKCHAIR(FANCY)
TRNCHAIREQ   1: TRNCHAIREQ(PLANT2,FUNCTIONAL)
TRNCHAIREQ   2: TRNCHAIREQ(PLANT2,FANCY)
TRNCHAIREQ   3: TRNCHAIREQ(PLANT3,FUNCTIONAL)
TRNCHAIREQ   4: TRNCHAIREQ(PLANT3,FANCY)
TRNTABLEEQ   1: TRNTABLEEQ(PLANT3,FUNCTIONAL)
TRNTABLEEQ   2: TRNTABLEEQ(PLANT3,FANCY)
```

Table 19.1 PUT file Output for TRANSPRT example used in Chapter 6.

```
              Report of Output from Solve of Transport Model
                   Run Done on 01/18/94 at 10:01:13

                   Commodity Movements Between Cities
                             all units in tons

Origin                                 Destination City
 City                MIAMI       HOUSTON     MINEPLIS    PORTLAND    Total

NEWYORK                30          35          15           0          80
CHICAGO                 0           0          75           0          75
LOSANGLS                0          40           0          50          90
                     ----        ----        ----        ----        ----
Total                  30          75          90          50         245

                   Cost of Commodity Movements Between Cities

  Origin      Destination          Quantity                 Cost
   City          City              Shipped    Cost/unit    Incurred
                                    tons         $/ton         $

  NEWYORK      MIAMI                  30          20          600
  NEWYORK      HOUSTON                35          40         1400
  NEWYORK      MINEPLIS               15          35          525
  CHICAGO      MINEPLIS               75          20         1500
  LOSANGLS     HOUSTON                40          35         1400
  LOSANGLS     PORTLAND               50          40         2000
                                                             ----
Total Cost of Shipping                                       7425

                   Report on Status of Commodity Usage by Plant

Plant              Quantity Available  Quantity Shipped  Value of More Supply
                        in tons            in tons            in $ / ton

NEWYORK                    100               80                  0.00
CHICAGO                     75               75                 15.00
LOSANGLS                    90               90                  5.00

                   Report on Status of Supply By Market

Market             Quantity Needed  Quantity Received  Cost of Meeting Demand
                       in tons          in tons             in $ / ton

MIAMI                      30             30                  20.00
HOUSTON                    75             75                  40.00
MINEPLIS                   90             90                  35.00
PORTLAND                   50             50                  45.00

                   Cost of Altering Commodity Movements Between Cities
                                  all units in $/ton

Origin                            Destination City
 City             MIAMI       HOUSTON     MINEPLIS    PORTLAND

NEWYORK              0.00        0.00        0.00       75.00
CHICAGO             45.00       35.00        0.00       40.00
LOSANGLS            75.00        0.00       40.00        0.00
```

Table 19.2 Comparative Report Writer Code from EVPORTFO File

```
PARAMETER RISKAVER(RAPS) RISK AVERSION COEFFICIENTS
/     R0   0.00000,    R1   0.00025,    R2   0.00050,    R3   0.00075,
      R4   0.00100,    R5   0.00150,    R6   0.00200,    R7   0.00300,
      R8   0.00500,    R9   0.01000,    R10  0.01100,    R11  0.01250,
      R12  0.01500,    R13  0.02500,    R14  0.05000,    R15  0.10000,
      R16  0.30000,    R17  0.50000,    R18  1.00000,    R19  2.50000,
      R20  5.00000,    R21  10.0000,    R22  15.    ,    R23  20.
      R24  40.    ,    R25  80./

 PARAMETER OUTPUT(*,RAPS) RESULTS FROM MODEL RUNS WITH VARYING RAP

 OPTION SOLPRINT = OFF;
 LOOP (RAPS,RAP=RISKAVER(RAPS);
      SOLVE EVPORTFOL USING NLP MAXIMIZING OBJ ;
           VAR = SUM(STOCK, SUM(STOCKS,
      INVEST.L(STOCK)* COVAR(STOCK,STOCKS) * INVEST.L(STOCKS))) ;
      OUTPUT("OBJ",RAPS)=OBJ.L;
        OUTPUT("RAP",RAPS)=RAP;
      OUTPUT(STOCKS,RAPS)=INVEST.L(STOCKS);
        OUTPUT("MEAN",RAPS)=SUM(STOCKS, MEAN(STOCKS) *
INVEST.L(STOCKS));
        OUTPUT("VAR",RAPS) = VAR;
        OUTPUT("STD",RAPS)=SQRT(VAR);
      OUTPUT("SHADPRICE",RAPS)=INVESTAV.M;
      OUTPUT("IDLE",RAPS)=FUNDS-INVESTAV.L
            );
 DISPLAY OUTPUT;
```