A COMPUTER PROGRAM FOR THE STAIRCASE INTEGER PROGRAMMING PROBLEM—ETC(U)
JUL 80  L J POLLENZ
UNCLASSIFIED  TR-95
N00014-76-C-0418
NL
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
A COMPUTER PROGRAM FOR THE STAIRCASE INTEGER PROGRAMMING PROBLEM

BY

LYNNE J. POLLENZ

TECHNICAL REPORT NO. 95
JULY 1980

PREPARED UNDER CONTRACT
N00014-76-C-0418 (NR-047-061)
FOR THE OFFICE OF NAVAL RESEARCH

Frederick S. Hillier, Project Director

Reproduction in Whole or in Part is Permitted for any Purpose of the United States Government
This document has been approved for public release and sale; its distribution is unlimited.

DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

Also issued as Technical Report No. 80-16, Dept. of Operations Research Stanford University, under National Science Foundation Grant MCS76-81259
A Computer Program for the Staircase Integer Programming Problem

1. Introduction

The computer code SDA to be described in this report solves the staircase integer linear programming problem, \((SP)\) given by:

\[
\begin{align*}
\text{maximize} & \quad \sum_{t=1}^{T} c_t x_t \\
\text{subject to:} & \quad A_t x_t \leq b_t \quad t=2,3,\ldots,T \\
& \quad A_t x_t + B_t x_{t-1} \leq b_t \quad t=2,3,\ldots,T \\
& \quad L_t \leq x_t \leq U_t \quad t=1,2,\ldots,T \\
& \quad x_t \text{ integer,} \quad t=1,2,\ldots,T.
\end{align*}
\]

This formulation arises in multiplant production allocation problems, multisector economic planning models, and multitime period production and inventory problems.

The solution method used by the program SDA relies upon decomposition of the problem \((SP)\) into smaller subproblems \((S_t)\), each of which can then be efficiently solved by an LP-based branch-and-bound routine. Each subproblem \((S_t)\) is specified by a vector of costs \(c_t\), the "diagonal" submatrix \(A_t\), the "offdiagonal" submatrix \(B_{t-1}\), the right-hand side \(b_t\), and upper and lower bounds \(U_t\) and \(L_t\) on the variables \(x_t\). The actual formulation is as follows:

\[
\begin{align*}
\text{maximize} & \quad c_t x_t \\
\text{subject to:} & \quad A_t x_t \leq b_t - B_{t-1} x_{t-1} \\
& \quad L_t \leq x_t \leq U_t \\
& \quad x_t \text{ integer.}
\end{align*}
\]
The algorithm proceeds by finding a solution to a subproblem (say $S_t$) and moving forward to the next subproblem $(S_{t+1})$ with a new right-hand side determined by the solution to $S_t$. When a set of solutions to all the subproblems is found, this solution will be feasible for the original problem. Prices are calculated to help guide the search for subproblem solutions toward an optimal value. Bounds on the maximum objective value obtainable given the current solutions to a subset of the subproblems are utilized to speed fathoming. See Pollenz (1980) for a detailed discussion of the staircase decomposition algorithm.

The branch-and-bound search procedure, which is used to solve the subproblems obtained via decomposition of (SP), is a slightly modified version of the computer program BB written by Gary Kochman as part of his dissertation at Stanford University. This computer code solves pure integer programming problems with general upper and lower bounds on the variables using a branch-and-bound technique similar to that presented by Dakin (1965). Tomlin's (1971) improved penalties are employed to guide the choice of branching variable at unfathomed nodes, with the branch being taken in the direction opposite to the maximum penalty. Nodes are removed from the branch-and-bound list according to a last-in-first-out (LIFO) strategy, and reoptimization after a branch is accomplished by the dual simplex method. See Kochman (1976) for further details about this program.

The linear programming portions of the code BB were developed by John Tomlin of the Systems Optimization Laboratory at Stanford University and adapted by Kochman to efficiently deal with simple upper and lower bounds on the variables. The most important features of LPM-1 are: storage of the basis inverse in product form (see Orchard-Hays (1968)); LU decomposition of the basis inverse (see
Benichou et al. (1977); and storage of all data, including the inverse, in compressed form (i.e., zeroes are not saved).

In order to maintain compatibility with BB, the computer code SDA was written in FORTRAN. Testing and evaluation of this program was carried out on the SCORE DEC System 20 computer at Stanford University. Due to the restrictive nature of the FORTRAN-10 compiler at SCORE, few of the features of SDA would be unacceptable to another system. The sole exception is the usage of the system clock routine IHPTIM, which is employed to record total CPU time and the percentage of execution time devoted to various portions of the algorithm. Such information is valuable, but if no system clock is available, the appropriate sections of the program can be eliminated without affecting the whole.

The main program and input requirements for SDA are discussed in the next section. The current restrictions on problem size are given in section 3. The various subroutines which comprise the bulk of the program SDA are outlined in section 4. The output generated by SDA is described in section 5, and sample input and output are given at the end of the appendix, following the listing of the program.

2. Main Program and Input Requirements

The main program calls the input subroutine, a subroutine which generates a few initial bounds, and then iteratively calls the branch-and-bound program to solve the appropriate subproblems. After the optimal solution has been discovered (and optimality verified), control passes to the next section of the main program, which computes some timing information and calls the output subroutine.

The first input card must contain values for the parameters
IFPROB, NP, IOBJ, INVFRQ, ITRFRQ, and INITBD, in (4I4, I5, I10) format. These variables have the following significance:

IFPROB = problem identification number (must be nonzero).

NP = number of time periods (equivalently, number of subproblems).

IOBJ = row number of objective row. Default is 1. (Currently, IOBJ must be 1 for subroutine BOUNDR to operate correctly; this can be corrected.)

INVFRQ = frequency with which basis inversion is carried out in the linear programming portions of the code. If INVFRQ = k, basis reinversion will occur after every k simplex iterations. INVFRQ should not be set greater than the number of rows in the smallest subproblem.

ITRFRQ = upper limit on total number of simplex iterations. Computations are terminated if the total number of simplex iterations exceeds ITRFRQ. If ITRFRQ = 0, the default value of 999999 is used.

INITBD = initial lower bound estimate for maximum objective value. It is used in fathoming tests until the first feasible solution with objective value greater than INITBD is found.

Following the initial input card, data for each subproblem is read in sequentially. The data must be input in the following order:

1. A leading card with "SUB nnnn" in columns 5-12, where nnnn
is replaced by the appropriate subproblem number. Subproblem numbers must be sequential, beginning with 1.

2. Nonzero entries of the offdiagonal submatrix $B_{k-1}$ (except for subproblem 1, for which no offdiagonal matrix is needed).

3. Row name and type for each constraint.

4. Nonzero entries of the diagonal matrix $A_k$.

5. The right-hand side $b_k$.

6. Lower bounds $L_k$, followed by upper bounds $U_k$, in format 15F5.0.

The type of constraint is denoted by a single letter preceding the row name. "E" stands for an equality, "L" and "G" identify less than or equal to and greater than or equal to inequalities respectively, and "N" marks the objective row. The format for this row identification input is (1X, A1, 2X, A8).

For the reading of all coefficients, the following information must be specified: column name, row name, and entry value. The format used for this is (4X, A8, 2X, A8, 2X, F12.4, 3X, A8, 2X, F12.4). The pattern for row name and value is repeated so that two entries for the same column may be input on one line. The row names must match a name read in earlier (input step §3). Zero coefficients may be omitted completely. This format is the same as used for LPM-1, and is consistent with standard MPS format.

The only restriction on input data is that the cost coefficients are assumed to be integral. Furthermore, these coefficients must be input with the opposite sign (i.e., if you wish to maximize $cx$, you must input $-c$ for the objective row). This is an unfortunate result of the fact that LPM-1 is a minimization routine.

After all data has been read in, an END card signifies the end of the data file.

The input for a sample run of SDA is included in the appendix.
following the program listing.

3. Restrictions on the Use of SDA

For current dimensioning, the following restrictions apply to use of the SDA:

1. The number of subproblems (NP) must be \( \leq 10 \).
2. The total number of rows, excluding the objective row, must not exceed 60.
3. The total number of columns (including slacks) must be \( \leq 120 \).
4. The total number of nonzero elements in the constraint matrix (excluding the elements of \( A_1 \)) must not exceed 1000. (This restriction is necessary only for subroutine BOUNDR.)
5. The total number of nonzero elements in the first diagonal submatrix, \( A_1 \), must not exceed 1000.
6. No right-hand side \( b_k(i) \) may exceed 1000 (due to default upper bounds on the slack variables).
7. The number of nodes in the branch-and-bound tree must never exceed 500. (For \([0,1]\) problems, the number of nodes will never exceed the total number of columns, so this restriction will not be a factor.)

4. Subroutines of SDA

BLOCK DATA (from LPM-I): sets initial values for many global program constants, including maximum problem dimensions and minimum
tolerances.

SUBROUTINE INPUT (IFPROB, INITBD) (from LPM-1): reads in all problem data, checks that problem dimensions do not exceed current specifications.

Parameters:

   IFPROB - nonzero problem identification number (output)
   INITBD - initial lower bound estimate of maximal objective value (output parameter)

SUBROUTINE INPSTO: stores all data relevant to subproblem whose number is stored in variable NS. This subroutine is called after reading the data initially (from INPUT) and before each forward step of the algorithm.

Entry points:

   STORE - entry point from subroutine TESTX. After an integral solution is found to subproblem NS, the current state of the corresponding search is saved by a call to STORE.

SUBROUTINE RESTOR (MNFLAG): restores all data from subproblem NS in preparation for either a forward or a backtracking step. For a forward step, the LP relaxation of this subproblem, with new right-hand side determined by a call to FIXRHS, is solved.

Parameters:

   MNFLAG - input parameter set at 0 if backtracking, 1 if taking a forward step.

SUBROUTINE FIXRHS: computes new right-hand side for subproblem NS, given the (newly established) setting of variables of the previous
SUBROUTINE BOUNDR: calculates two LP-based bounds on the maximum objective values for some aggregations of the subproblems. These bounds are used for fathoming in subroutines TESTX, BKTRAK, and PENLTS. Also, a vector of prices for use in guiding the branch-and-bound search (see subroutine PENLTS) is computed.

SUBROUTINE UPDATX: updates right-hand side by taking into consideration variables which are nonbasic at their upper or lower (nonzero) bounds.

SUBROUTINE FTRAN (IPAR) (from LPM-1): performs forward transformation of matrix column (stored in vector Y) by basis inverse.

Parameters:

IPAR - input parameter indicating which eta-vectors are used to update the matrix column.

SUBROUTINE RTRAN (from LPM-1): performs backward transformation on column stored in vector Y.

SUBROUTINE FORMC (from LPM-1): forms objective function vector for Phase I of primal simplex method.

SUBROUTINE PRICE (from LPM-1): prices out nonbasic columns for primal simplex method and chooses pivot column (stored in variable JCOLP). Also checks for dual feasibility.

SUBROUTINE CHUZR (from LPM-1): chooses pivot row for primal simplex method using min ratio test; stores pivot row in variable IROWP.
SUBROUTINE WRETA (from LPM-1): forms a new eta-vector for the product form of the inverse.

SUBROUTINE SHIFTR (IOLD, INEW) (from LPM-1): rearranges data storage.
Parameters:
   IOLD, INEW - input parameters indexing storage locations. Move from location designated by IOLD to that designated by INEW.

SUBROUTINE INVERT (from LPM-1): Creates basis inverse by LU decomposition.

SUBROUTINE UNPACK (IV) (from LPM-1): expands compressed matrix columns by inserting zeroes appropriately.
Parameters:
   IV - input parameter indexing the matrix column to be expanded.

SUBROUTINE SHFTE (from LPM-1): Subroutine for INVERT.

SUBROUTINE UPBETA (from LPM-1): updates right-hand sides following a primal or dual simplex pivot.

SUBROUTINE NORMAL (ITSINV) (from LPM-1): directs execution of primal simplex method.
Parameters:
   ITSINV - counts number of simplex iterations since last basis inversion (for comparison with INVFRQ).
SUBROUTINE BANDB (INITBD) (from BB): master program for branch-and-bound integer programming routine. It also serves as master program for reoptimization via the revised dual simplex method after a forward branch.

Parameters:

INITBD - initial lower bound estimate on maximum objective value.

Entry points:

BRENTR - reentry point from main program to apply branch-and-bound search to any subproblem after the first.

SUBROUTINE DCHUZC (from BB): selects pivot column for dual simplex method and stores it in variable JCOLP. Also checks for dual feasibility.

SUBROUTINE TESTX (from BB): tests LP-optimal solution at current node for integrality and for fathoming (fathoming tests and branching strategy modified substantially from those of BB). Any new complete solution found is immediately printed out and saved in the array INCUNM; any integral solution to a subproblem (other than the last) causes a call to entry STORE in preparation for a forward step. Variable MSTAT flags the result of testing and is checked in subroutine BANDB.

SUBROUTINE PENLTS (from BB): computes Tomlin's improved up- and down-penalties and the Gomory penalty at each node. Checks for forced branches on both basic and non-basic variables. If fathoming does not
occur, a branch variable is chosen in accordance with the combination of Tomlin's penalties and the prices computed in BOUNDR, and subroutine BRANCH is called. (Substantial modifications have been made from the version of this subroutine given in BB.)

SUBROUTINE BRANCH (from BB): Performs necessary bookkeeping for branching on variable indexed by ICOL. Increments list of stored nodes, revises bounds on branching variable, and saves opposite branch direction and a bound on the maximum objective value on that opposite branch.

SUBROUTINE BKTRAK (from BB): backtracks to a promising (unfathomed) node from the list of stored nodes. Employs last-in-first-out selection rule. If backtracking brings us back to the previous subproblem, the appropriate data and status of the search of that subproblem are restored by a call to subroutine RESTOR.

SUBROUTINE WRAPUP (from BB): Outputs final solution information. (See output from sample run at the end of the appendix.)

For further information on subroutines derived from LPM-1 see Tomlin [1975].

5. Description of Output

The output produced by this program falls onto 3 sections. The initial output contains the problem identification number and dimensions, followed by the prices and bounds computed by subroutine BOUNDR. At this point the iterative portion of the algorithm is
begun. Each time an improved feasible solution is discovered, the time of discovery, total number of branches taken, and new maximum objective value (called INCVAL) are printed out by subroutine TESTX. After termination, some information regarding the total time taken and time spent at certain tasks is output. If termination occurs normally, the final section of output contains the optimum solution and its objective value. The solution is printed in 2015 format in the following order: objective value, slack variables, and integer variables for subproblem 1, then subproblem 2, ..., and finally for subproblem NP. If computation is halted because the limit on iterations (ITRFRQ) has been exceeded, then the best solution found so far and the total number of simplex iterations used are printed.
REFERENCES


Appendix:

Program Listing, Sample Input and Output.
### Sample Input

**Sub 1**

<table>
<thead>
<tr>
<th>ROWS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9999</td>
<td>0</td>
</tr>
</tbody>
</table>

**Columns**

<table>
<thead>
<tr>
<th>COL 101 OBJ</th>
<th>COL 101 ROW 102</th>
<th>COL 101 ROW 104</th>
<th>COL 102 OBJ</th>
<th>COL 102 ROW 102</th>
<th>COL 102 ROW 104</th>
<th>COL 103 OBJ</th>
<th>COL 103 ROW 102</th>
<th>COL 103 ROW 104</th>
<th>COL 104 OBJ</th>
<th>COL 104 ROW 102</th>
<th>COL 104 ROW 104</th>
<th>COL 105 OBJ</th>
<th>COL 105 ROW 102</th>
<th>COL 105 ROW 104</th>
</tr>
</thead>
<tbody>
<tr>
<td>-10.00</td>
<td>-6.00</td>
<td>7.00</td>
<td>-19.00</td>
<td>-9.00</td>
<td>15.00</td>
<td>-7.00</td>
<td>-9.00</td>
<td>-12.00</td>
<td>15.00</td>
<td>4.00</td>
<td>-8.00</td>
<td>-6.00</td>
<td>15.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**RHS**

<table>
<thead>
<tr>
<th>RHS1 ROW 101</th>
<th>RHS1 ROW 103</th>
<th>RHS1 ROW 105</th>
</tr>
</thead>
<tbody>
<tr>
<td>39.00 ROWN 102</td>
<td>25.00 ROWN 104</td>
<td>35.00 ROWN 105</td>
</tr>
</tbody>
</table>

**Bounds**

0. 0. 0. 0. 0.
1. 1. 1. 1. 1.

**Sub 2**

<table>
<thead>
<tr>
<th>ROWS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9999</td>
<td>0</td>
</tr>
</tbody>
</table>

**Offdiagonal Columns**

<table>
<thead>
<tr>
<th>COL 101 ROW 201</th>
<th>COL 101 ROW 203</th>
<th>COL 101 ROW 205</th>
<th>COL 102 ROW 201</th>
<th>COL 102 ROW 203</th>
<th>COL 102 ROW 205</th>
<th>COL 103 ROW 201</th>
<th>COL 103 ROW 203</th>
<th>COL 103 ROW 205</th>
<th>COL 104 ROW 201</th>
<th>COL 104 ROW 203</th>
<th>COL 104 ROW 205</th>
<th>COL 105 ROW 201</th>
<th>COL 105 ROW 203</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>17.00</td>
<td>3.00</td>
<td>-2.00</td>
<td>16.00</td>
<td>13.00</td>
<td>-8.00</td>
<td>0.00</td>
<td>5.00</td>
<td>17.00</td>
<td>-2.00</td>
<td>3.00</td>
<td>-4.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Columns**

<table>
<thead>
<tr>
<th>COL 105 ROW 205</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8.00</td>
</tr>
<tr>
<td>COL</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHS</th>
<th>ROW</th>
<th>COL</th>
<th>OBJ</th>
<th>ROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHS1</td>
<td>31.00</td>
<td>201</td>
<td>-1.00</td>
<td>203</td>
</tr>
<tr>
<td>RHS1</td>
<td>26.00</td>
<td>203</td>
<td>26.00</td>
<td>205</td>
</tr>
<tr>
<td>RHS1</td>
<td>23.00</td>
<td>205</td>
<td>23.00</td>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BOUNDS</th>
<th>ROW</th>
<th>COL</th>
<th>OBJ</th>
<th>ROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUB</th>
<th>3</th>
<th>L ROW</th>
<th>ROW</th>
<th>ROW</th>
<th>ROW</th>
<th>ROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFDIAGONAL COLUMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>204</td>
</tr>
<tr>
<td>204</td>
</tr>
<tr>
<td>204</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLUMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>301</td>
</tr>
<tr>
<td>301</td>
</tr>
<tr>
<td>301</td>
</tr>
<tr>
<td>302</td>
</tr>
<tr>
<td>302</td>
</tr>
<tr>
<td>302</td>
</tr>
<tr>
<td>303</td>
</tr>
<tr>
<td>303</td>
</tr>
<tr>
<td>303</td>
</tr>
<tr>
<td>304</td>
</tr>
<tr>
<td>304</td>
</tr>
<tr>
<td>304</td>
</tr>
<tr>
<td>305</td>
</tr>
</tbody>
</table>
COL 305 ROW 302  5.00  ROW 303  2.00
COL 305 ROW 304 -4.00  ROW 305 -10.00

RHS
RHS1  ROW 301  22.00  ROW 302  38.00
RHS1  ROW 303  21.00  ROW 304  23.00
RHS1  ROW 305  21.00

BOUNDS
0. 0. 0. 0. 0.
1. 1. 1. 1. 1.

SUB 4

ROWS
N OBJ
L ROW 401
L ROW 402
L ROW 403
L ROW 404
L ROW 405

OFFDIAGONAL COLUMNS
COL 301 ROW 401  1.00  ROW 402 -2.00
COL 301 ROW 403 -6.00  ROW 404  14.00
COL 301 ROW 405  4.00
COL 302 ROW 401  1.00  ROW 402  4.00
COL 302 ROW 403  15.00  RCW 404  19.00
COL 302 ROW 405  10.00
COL 303 ROW 401 -4.00  ROW 402  11.00
COL 303 ROW 403  18.00  ROW 404 -3.00
COL 303 ROW 405  17.00
COL 304 ROW 401  5.00  ROW 402  6.00
COL 304 ROW 403  9.00  ROW 404  16.00
COL 304 ROW 405  14.00
COL 305 ROW 401  4.00  ROW 402  20.00
COL 305 ROW 403  10.00  ROW 404  3.00
COL 305 ROW 405  16.00

COLUMNS
COL 401 OBJ  -1.00  ROW 401  7.00
COL 401 ROW 402  17.00  ROW 403  3.00
COL 401 ROW 404  12.00  ROW 405  16.00
COL 402 OBJ  -16.00  ROW 401  12.00
COL 402 ROW 402  14.00  RCW 403  11.00
COL 402 ROW 404  12.00  ROW 405 -10.00
COL 403 OBJ  -12.00  ROW 401  17.00
COL 403 ROW 402  6.00  ROW 403  4.00
COL 403 ROW 404  -8.00  ROW 405  12.00
COL 404 OBJ  -14.00  ROW 401  5.00
COL 404 ROW 402  10.00  ROW 403  11.00
COL 404 ROW 404  -4.00  ROW 405  18.00
COL 405 OBJ  -10.00  ROW 401  16.00
COL 405 ROW 402  17.00  ROW 403  6.00
COL 405 ROW 404  -6.00  ROW 405  3.00

RHS
RHS1  ROW 401  38.00  ROW 402  29.00
RHS1  ROW 403  29.00  ROW 404  36.00
RHS1  ROW 405  27.00

BOUNDS
0. 0. 0. 0. 0.
1. 1. 1. 1. 1.

EOT
**** Sample Output ****

PROBLEM 1
20 ROWS 20 COLUMNS 4 PERIODS

PRICE OF OFFDIAGONAL COLUMNS IN SUB. 4
-2.29 4.57 12.57 6.86 22.86

PRICE OF OFFDIAGONAL COLUMNS IN SUB. 3
0.00 0.00 0.00 0.00 0.00

PRICE OF OFFDIAGONAL COLUMNS IN SUB. 2
2.12 9.18 3.53 2.12 -5.65

MAXIMUM CUMULATIVE OBJECTIVE VALUES
FOR LAMBDA = .5: 196 140 89
FOR LAMBDA = 0: 90 49 42

INTERMEDIATE SOLUTIONS FOUND
TIME = 0.90 SECONDS; BRANCHES = 21; INCVAL = 80
TIME = 0.99 SECONDS; BRANCHES = 28; INCVAL = 84
TIME = 0.99 SECONDS; BRANCHES = 31; INCVAL = 90
TIME = 1.32 SECONDS; BRANCHES = 76; INCVAL = 92
TIME = 1.84 SECONDS; BRANCHES = 145; INCVAL = 105

TOTAL SOLUTION TIME = 2.12 SECONDS
TIME FOR INPUT = 0.44 SECONDS
TIME FOR LP SOLUTIONS = 0.31 SECONDS
TIME FOR BOUNDING & PRICING = 0.32 SECONDS
TIME FOR BOOKKEEPING OPERATIONS = 0.33 SECONDS
NUMBER OF FORWARD STEPS TAKEN = 80
TOTAL NUMBER OF BRANCHES TAKEN = 174

OPTIMAL INTEGER SOLUTION
SUBPROBLEM 1
10 25 6 23 16 43 0 0 0 1 1
SUBPROBLEM 2
39 2 0 20 0 1 0 1 0 1 1
SUBPROBLEM 3
20 1 24 23 13 20 1 0 1 0 0
SUBPROBLEM 4
28 12 0 2 21 4 0 1 1 0 0
MAX OBJECTIVE VALUE = 105
STAIRCASE-STRUCTURED MATRIX BRANCH-AND-BOUND CODE
PURE-INTEGER LINEAR PROGRAMMING IN GENERAL INTEGER VARIABLES
SDB: TWO BOUNDING PROCEDURES, EXTRA TIMERS IF NEEDED
WRITTEN BY LYNNE POLLENZ, LAST UPDATED MAY 1979

IMPLICIT REAL*4 (A,E-H,0, P, R-W, Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
COMMON/GEStencil/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
1 JS,NP,JPCOL(11),JFROW(11),JFECOL(11),MAXC(10),MAXC2(10)
COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV

"MAIN PROGRAM"

"START TIMER"
100  ITOT = IHPTIM(1)

INPUT PROBLEM DATA
CALL INPUT(IFPROB,INITBD)
ITIMIN = IHPTIM(1)
IF (IFPROB .EQ. 0) GO TO 1000
CALL BOUNDR
ITIMLP = IHPTIM(1)
TSTORE = 0.
TIMELP = 0.

UNDER THE SCORE COMPILER, "D" IN THE FIRST COLUMN IS READ AS "C"
UNLESS A SPECIAL OPTION IS USED, IN WHICH CASE IT IS TREATED AS A "".

TIMINV = 0.
TIMEDC = 0.
TIMEDR = 0.
NBRANC = 0
NFORWD = 1

APPLY BRANCH-AND-BOUND SEARCH ROUTINE, STARTING IN PERIOD 1

IF = 1
CALL RESKPR(1);
CALL 3ANDR(INITBD)
30  IF (LISTL.EQ.0) GO TO 500

A PARTIAL SOLUTION HAS BEEN FOUND. GO ON TO THE NEXT PERIOD.

VS = VS + 1
NPOPYO = NFORWD + 1
CALL RESKPR(1)
CALL 3ANDR
GO TO 30

STOP TIMER, ALL DONE. REPORT TIMING INFORMATION.

1100  ITWP = IHPTIM(1)
IT = ((ITIM2-ITOT)/100000.
#ITII: (21,1) TOT
#IMINT = IT = "TOTAL SOLUTION TIME = " F7.2, " SECONDS"
ITIMLP = (ITIMIN - ITOT)/100000.
#ITII: (21,2) TIMELP
#IMINT = TIMELP " INPUT = " F7.2, " SECONDS"
ITIMFRC = (ITIMLP - ITIMIN)/100000.
#ITII: (21,3) TIMEDR
#IMINT = TIMEDR " LP SOLUTIONS = " F7.2, " SECONDS"
#ITII: (21,4) TIMEDC
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPN-1, WRITTEN BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

***DESCRIPTION OF PARAMETERS***

1) IFPROB = NONZERO PROBLEM ID NUMBER (OUTPUT)

2) INITBO = INITIAL LOWER BOUND ESTIMATE FOR MAXIMAL OBJECTIVE VALUE (OUTPUT)

3) DATA MUST BE INPUT IN THE FOLLOWING ORDER: PROBLEM CONSTANTS, FOR EACH PERIOD: ROWS, OFFDIAG. COLS, DIAGONAL COLS, RMS, BOUNDS

THE OBJECTIVE ROW MUST BE THE FIRST ROW IN EACH SUBPROBLEM VARIABLES LOCAL TO ONE PERIOD MUST APPEAR AFTER VARIABLES OF THAT PERIOD WHICH ALSO APPEAR IN THE FOLLOWING PERIOD.

IMPLICIT REAL*4 (A,C,E-H,O,P-R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q), COMMON/TOTS1Z/ MAXTRW,MAXTCL,MAXNP,MAKLE,MAXNL,
COMMON/CONSTS/ZTOLZE,ZTOLPV,ZTCOST,ZTOLSH,NEGINF,NMAX,NRMAX,QBL,
1 NLES,NMAX,OA,QI,QF,QN,QSUB,QB,OC,FE,QH,QL,QQ,QR,QM,OC,
DATA MAXTRW/7000/,MAXTCL/130/,MAXNP/100/,MAKLE/2000/,MAKNL/1000/
DATA ZTOLZE/1.E-5/,ZTOLPV/1.E-4/,ZTCOST/1.E-3/,ZTOLSH/1.E-10/
DATA NMAX/600/,NMAX/500/,NEGINF/-1000000/
DATA QQ/" /,QI/" A " /,QF/" F " /,QN/" N " /,
1 QSUB/" /,QB/" B " /,QC/" C " /,QF/" E " /,QR/" H " /,
2 QH/" L " /,QO/" D " /,QR/" K " /,QM/" M " /,QG/" C " /
END

IMPLICIT REAL*4 (A,C,E-H,O,P-R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q), INTEGER JH,KINBAS,LA,LE,IA,IE
1 INTEGER NAME(6),NAMCOL(130,2)
DOUBLE PRECISION E(2000),ATEMP1,ATEMP2
REAL A(1000)
***DESCRIPTIONS OF LOCAL VARIABLES***

**JCPTR** points to current offdiagonal input column

**NAMCOL(I,*):** name of column I

**NTCOL:** total number of columns so far

**NCOL1:** first column of current subproblem

**NROWL:** last slack column (and last row) of current subproblem

**MAXTRW:** max total number of rows (including NS objective rows)

**MAXTCL:** max total number of columns including slack columns

**MAXNP:** max number of subproblems

**MAXAFL:** max number of nonzero elements of any diagonal block

**MAXELE:** max. sum of nonzero entries in all diagonal blocks

***DESCRIPTIONS OF SOME IMPORTANT VARIABLES IN BLANK COMMON***

**B(I):** right hand side of row I (in current subproblem)

**X(I):** LP value for JH(I), which is the variable basic in row I

**A:** contains the nonzero elements of the constraint matrix, incl. the objective row IOBJ. LA(J) = location in A of the first element of col. J. IA(I) = row in which element I of A belongs.

**E:** contains the nonzero elements of the current LP basis inverse in ETA vector form. LE, IE are to E as LA, IA are to A.

**MSTAT:** flags feasibility of current LP

**ITCNT:** no. of simplex iterations so far; if > ITRFRQ, stop

**INVRQ:** number of simplex iterations before E is inverted

**KINBAS(J):** (1 if J is basic in row I, i.e. J = JH(I)

**-1** if J is nonbasic at its upper bound XUB(J)

***DESCRIPTIONS OF SOME VARIABLES IN GESTALT COMMON***

**JFCOL(I), JFROW(I), JFELEM(I):** first col, row, element of A, respectively, for period I

**NP:** total number of periods (or equivalently, subproblems)

**NS:** current subproblem number

**INITIALIZATIONS***

ITCNT = 0
NS = 0
NOELEM = 0
NTCOL = 0
JFCOL(1) = 1
JFROW(1) = 1
JFELEM(1) = 1
D7 1 = 1, MAXTCL
1
C

READ (20,7000,END=9999) IFPROB, NP, IOBJ, INVFQ, ITRFRQ, INITBD

7000 FORMAT (414,15,110)

IF (IFPROB .EQ. 0) RETURN
IF (NP.GT.NAXNP) GO TO 9996
NMAX = 4000/NP
MLK = 1000/NP
IF (IOBJ .EQ. 0) IOBJ = 1
IF (IOBJ .NE. 1) GO TO 9995
IF (INVFRQ .EQ. 0) INVFRQ = 99999
IF (ITRFRQ .EQ. 0) ITRFRQ = 999999
WRITE(21,8010) IFPROB
8010 FORMAT(/19X, "PROBLEM ", I4)
C
C  INITIALIZE FOR READING EACH SUBPROBLEM
C
5 IF (NS .EQ. NP) GO TO 25
NROW = 0
ICS1 = 0
ICS2 = 0
DO 10 I=1,NMAX
10 B(I) = 0.
NS = NS + 1
NCOL1 = JFCOL(NS)
READ(20,99) (NAME(I),I=1,3)
99 FORMAT(2A4,I4)
IF (NAME(2).NE.QSUB .OR. NAME(3).NE.NS ) GO TO 9998
C
C  READ IN DATA FOR SUBPROBLEM NS
C
25 READ(20,101) K1,K2,K3,K4,(NAME(I),I=1,4),ATEMP1,NAME(5),NAME(6),
1 ATEMP2
101 FORMAT(4A1,2A4,2X,2A4,2X,$F12.4,3X,2A4,2X,12.4)
IF(K1 .EQ. QBL) GO TO 50
IF(K1 .EQ. QR .AND. K2 .EQ. QO) L=1
IF(K1 .EQ. QR .AND. K2 .EQ. QO) GO TO 25
IF(K1 .EQ. QR) GO TO 150
IF(K1 .EQ. QO) GO TO 160
IF(K1 .EQ. QR .AND. K2 .EQ. QR) L=4
IF(K1 .EQ. QR .AND. K2 .EQ. QR) GO TO 25
IF(K1 .EQ. QB) GO TO 600
IF(K1 .EQ. QB) GO TO 700
WRITE(21,8020) K1,K2,K3,K4,NS
8020 FORMAT(" IMPROPER INPUT ",4A1, " IN SUBPROBLEM ", I4)
GO TO 9999
50 GO TO(210,320,400,500),L
C
150 L = 2
NROWL = NCOL1 + NROW - 1
GO TO 25
C
160 L = 3
ICS1 = -9999
GO TO 25
C
C  READ ROW NAMES ( = SLACK COLUMN NAMES)
C
210 NROW= NROW+1
NCOL= NROW
NTCOL = NCOL + 1
NANCOL(NTCOL,1) = NAME(1)
NANCOL(NTCOL,2) = NAME(2)
C
C  TEST ROW TYPE: (<,=,>=, OR OBJ. ROW) SET SLACK BOUNDS
C
IF(K2.EQ.QL .OR. K3.EQ.QL) GO TO 220
IF(K2.EQ.QL .OR. K3.EQ.QE) GO TO 230
IF(K2.EQ.QG .OR. K3.EQ.QG) GO TO 240
IF(K2.EQ.QN .OR. K3.EQ.QN) GO TO 250
GO TO 230
220 XLB(NROW) = 0.
XUB(NROW) = 1.E4
GO TO 250
230 XLB(NROW) = 0.
XUB(NROW) = 0.
GO TO 250
240 XLB(NROW) = 0.
XUB(NROW) = 1.E4
A(NROW) = -1.
GO TO 260
250 A(NROW) = 1.
260 IA(NROW) = NROW
LA(NROW) = NROW
NELEM = NROW
GO TO 25
C
C MATRIX ELEMENTS
C
320 J = 3
K = 4
IF (DABS(ATEMP1) .GT. ZTOLZE) GO TO 324
J = 5
K = 6
IF (DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
ATEMP1 = ATEMP2
C
TEST FOR COLUMN MATCH
324 IF (NAME(1) .EQ. ICS1 .AND. NAME(2) .EQ. ICS2) GO TO 330
NCOL = NCOL + 1
NTCOL = NTCOL + 1
ICS1 = NAME(1)
ICS2 = NAME(2)
NAMCOL(NTCOL1) = ICS1
NAMCOL(NTCOL2) = ICS2
LA(NCOL) = NELEM + 1
C
RECORD OBJECTIVE VALUE
330 IF (NAME(J) .EQ. NAMCOL(NCOL1+OBJ-1,1)) COST(NTCOL) = ATEMP1
C
TEST FOR ROW MATCH
DO 340 I = NCOL1,NROWL
IF (NAME(J) .NE. NAMCOL(I,1) .OR. NAME(K) .NE. NAMCOL(I,2)) GO TO 340
NELEM = NELEM + 1
IA(NFLEM) = I - NCOL1 + 1
A(NFLEM) = ATEMP1
LA(NCOL+1) = NELEM + 1
IF (K .GT. 5) GO TO 25
IF (DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
J = 5
K = 6
ATEMP1 = ATEMP2
GO TO 330
340 CONTINUE
C
*RITE(21,830C) NAME(J),NAME(K),NAME(1),NAME(2)
H300 FORMAT(17CHAR MATCH FOR ROW ,2A4,11H AT COLUMN ,2A4)
GO TO 9999
C
1FDIAGONAL MATRIX ELEMENTS

! J = 3
! K = 4
IF (DABS(ATEMP1) .GT. ZTOLZE) GO TO 420
! J = 5
! K = 6
IF (DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
ATEMP1 = ATEMP2

C TEST FOR COLUMN MATCH

! IFSATCATEPI) .GT. ZTOLZE)
! GO TO 420
! J = 5
IF IA5(TM2 - LE. ZTOLZI) GO TO 25
ATEMPI = ATEMP2

C TEST FOR COLUMN MATCH

! IF (NAMF(1).EQ.ICS1 .AND. NAME(2).EQ.ICS2) GO TO 450
ICS1 = NAME(1)
ICS2 = NAME(2)
IREG = JCPTRE + 1
IPM0 = JFCOL(NS) - 1
IF (IREG.GT.IEND) GO TO 435
DO 430 = I,IBF,1,E1ND
LCOLOI(I) = NOELEM + 1
IF (ICS1.EQ.NAMCOL(I,1) .AND. ICS2.EQ.NAMCOL(I,2)) GO TO 440
430 CONTINUE

! IF (IPTF(21,8250).ICSI,ICS2.IS)
! 8250 IF(NAMF(1).EQ.ICS1 .AND. NAME(2).EQ.ICS2) GO TO 9999
ICS1 = ICS2

C TEST FOR ROW MATCH

! DO 460 = I,=COL1,1NCOL
IF (NAME(J).NE.NAMCOL(I,1).OR.NAME(K).NE.NAMCOL(I,2)) GO TO 460
NOELEM = NOELEM + 1
IRROW(NOELEM) = 1 - NCOL1 + 1
OFFD(NOELEM) = ATEMP1
LCOLOD(JCPTRE + 1) = NOELEM + 1
IF (K.GT.5) GO TO 25
IF (DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
J = 5
K = 6
ATEMP1 = ATEMP2
GO TO 450

460 CONTINUE

! IF (IPTF(21,8300).NAME(J),NAME(K),NAME(1),NAME(2)
! GO TO 9999
C
C RIGHT HAND SIDE

! J = 3
! K = 4
IF (DABS(ATFMP1) .GT. ZTOLZE) GO TO 530
! J = 5
! K = 6
IF (DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
ATEMP1 = ATEMP2

C TEST FOR ROW MATCH

! DO 540 = I,=ROWL,1NCOL
IF (NAME(J).NE.NAMCOL(I,1).OR.NAME(K).NE.NAMCOL(I,2)) GO TO 540
R(I-NCOL1+1) = ATEMP1
IF (K.GT.5) GO TO 25
IF (DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
J = 5
K = 6
ATEMP1 = ATEMP2
GO TO 530
CONTINUE
WRITE(21,8300) NAME(J),NAME(K)
GO TO 9999
C
C  BOUNDS ON INTEGER VARIABLES
C
K = NROW + 1
C  INPUT LOWER AND UPPER BOUNDS ON DECISION VARIABLES
READ (20,9000) (XLB(J), J=K,NCOL)
READ (20,9500) (XUB(J), J=K,NCOL)
C  FORMAT (15PE.0)
C
C  CHECK PROBLEM SIZE
JFCOL(NS+1) = JFCOL(NS) + NCOL
JFROW(NS+1) = JFROW(NS) + NROW
JFELEM(NS+1) = JFELEM(NS) + NELEM
IF (NROW.GT.1.*NRMAX) GO TO 9996
IF (NROW.GT.MAXAEL) GO TO 9996
IF (JFCOL(NS+1) .GT. (MAXROW + 1)) GO TO 9996
IF (JFCOL(NS+1) .GT. (MAXTCL + 1)) GO TO 9996
IF (JFELEM(NS+1) .GT. (MAXAEL + 1)) GO TO 9996
JCHTE = JFCOL(NS) + NROW - 1
CALL INPUT
GO TO 5
C
C  END OF INPUT
C
K = JFROW(NP+1) - 1 - NP
NSCOLS = JFCOL(NP+1) - JFROW(NP+1)
WRITE(21,8970) NROW,NSCOLS,NP
C  FORMAT (5A,16, "ROWS",4X,14, "COLUMNS",2X,14, "PERIODS",/)
RETURN
C
C  ERROR MESSAGES FOR INPUT ERRORS
C
9995 WRITE(21,9960)
9960 FORMAT("OBJECTIVE ROW MUST BE THE FIRST ROW")
GO TO 9999
9996 WRITE(21,9970)
9970 FORMAT("PROBLEM IS TOO LARGE FOR CURRENT DIMENSIONING")
9997 WRITE(21,9975) MAXNP,MAXROW,MAXTCL,MAXAEL
9975 FORMAT("MAX SUBS=" ,13,"MAX ROWS=" ,13,"MAX COLS=" ,14,"MAX ELE=" ,16)
GO TO 9999
9998 WRITE(21,9990) NS
9990 FORMAT("SUBPROBLEM",14,"NOT IN PROPER POSITION")
9999 IF(KRF0M = 0 RETURN
C
C--------------------------------------------------------------------------
SUBROUTINE INFEST
C
C  STORE ALL DATA RELVANT TO SUBPROBLEM NS
C
C  IMPLICIT REAL*4 (A,C,F-H,O,P,F-M-Z), REAL*8 (B,D,X,Y),
C  INTEGER*4 (I-N)
C  INTEGER JUNIX,JNUM,JIA,JEP,JCOLA,JROWA,JESET,LESTOR,LESTOR
C  DOUBLE PRECISION E(2000),ESTOR(400)
C  REAL AI(1000)
C
C  C : UN/CONSTZ, ZTOLZE,ZTOLPV,ZTCSRZ,ZTULSM,NEGINF,NEMAX,NRMAX,QBL,
C  LNRNS,NMAX,QA,W1,QP,QN,QSUN,QB,JC,QF,QR,QL,QD,QR,QM,QG
COMMON XLR(122), XUR(122), DE, UP, R(60), T(60), Y(60), YTEMP(50), A,
1 E, KETA, I0HJ, I0KWP, ITCNT, INTHRQ, JCOLP,
2 NROW, NCOL, NLE, NELEM, NETA, NLELEM, NLETA, NULEM, NUETA, JH(60),
3 KINPAS(122), LA(122), LE(502), LA(1000), LE(2000)
COMMON/G: SLT/PRICE(130), [CURX(130), ISUMC, ITSYV, LISTL, UBRANC,
1 NS, NP, JFCOL(11), JFROW(11), JFLE(11), MAXC(10), MAXC(10)
COMMON/H: TIMEH, ITR, TSTORE, THELP, TIMEDK, TIMEDC, TIMINV
COMMON/SUBSAV/BMW(70), BORIG(70), XSTORE(70), XUBSTO(130),
1 XIESTO(130), XUBORG(130), XLBORG(130), ASTORE(2000), ESTOPE,
2 IPWA(2000), LCOLA(140), LTEST(4000), LESTOR(1002), JHSTOR(70),
3 INBSTO(130), IPARTC(10), INSTO(10), NELSTO(10), NETSTO(10),

C COMMON BLOCK SUBSAV IS A STORAGE AREA FOR THE SUBPROBLEM VARIABLES
C XLBORG, XUBORG, BORIG STORE THE ORIGINAL VALUES OF THE CORRESP. VARS
C JMOD(I) = RHS FOR ROW I AFTER OFFDIAGONAL VALUES HAVE BEEN ADDED
C
C AFTER READING INPUT, STORE THE DIAGONAL MATRIX ELEMENTS AND RHS
C
IPTR = JFLEM(NS)
LENGTH = JFLEM(NS + 1) - IPTR
DO 10 I = 1, LENGTH
    ASTORE(IPTR) = A(I)
    IPTR = IPTR + 1
10
C STORE ORIGINAL RHS B; BOUNDS XUR, XLB
IPTR = JFROW(NS)
DO 20 I = 1, NROW
    BORIG(IPTR) = B(I)
    IPTR = IPTR + 1
20
C STORE THE CURRENT STATE OF SUBPROBLEM NS
C
ENTRY STORE
C STORE THE PARTIAL OBJ. VALUE ISUMC; RHS L SOLUTION X
ITIME = IHETIM(1)
IPARTC(NS) = ISUMC
IPTR = JFROW(NS)
DO 300 I = 1, NROW
    IMOD(IPTR) = B(I)
    XSTORE(IPTR) = X(I)
    JHSTOR(IPTR) = JH(I)
300
C STORE BASIC VAR INDICATOR KINHAS; BOUNDS XUR, XLB
IPTR = JFCOL(NS)
DO 350 I = 1, NCOL
    INBSTO(IPTR) = KINHAS(I)
    XURSTO(IPTR) = XUB(I)
    XLBSTO(IPTR) = XLI(I)
350
C STORE BASIS INVERSE. IF TOO LARGE, REINVERT.
IF (META.LT.NLES) GOTO 390
CALL INVERT
ITSINV = 0
390 INVSTO(NS) = ITSINV
NELSTO(NS) = NELEM
VESTO(NS) = NETA
LEPTR = (NS-1) * NLES
JEPT = (NS-1) * NMAX
DO 400 I=1,NELEM
   JEPTR = JEPTR + 1
   ESTORE(JEPTR) = E(I)
400 IESTOP(JFPT) = IE(I)
ENDIF = META + 1
DO 450 I=1,ENDIF
   LEPTR = LEPTR + 1
450 IESTOP(LEPTR) = LE(I)
    ITIME2 = ITHEMT(1)
    TSTORE = TSTORE + (ITIME2-ITIME)/10000.
RETURN
END

SUBROUTINE RESTORE(MNFLAG)

RESTORE ALL DATA RELEVANT TO SUBPROBLEM NS

**DESCRIPTION OF PARAMETER**

MNFLAG = 0 IF BACKTRACKING, 1 IF TAKING A FORWARD STEP (INPUT)

IF TAKING A FORWARD STEP, RESTORE ORIGINAL LP-OPT. IF TAKING

THIS STEP FOR THE FIRST TIME, COMPRE ORIGINAL LP SOLUTION.

IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
1 INTEGER JH,KIDS,JL,IE,IA,IECOLA,ICOWMA,JESTOR,LESTOR
1 DOUBLE PRECISION E(2000),ESTORE(4000)

REAL (A(1000))

COMMON/CONSTS/ ZTOLZLZTOLPV/PTOLPSZTOLSMW/NEMINF/NEMAX,NMAX,QBL,
1 NLES,YMAX,YA,YQ, YQSUB, YQ, YQ, QL, QL, QL, QL,
1 COMMON XLR(122),XUR(122),DE,DP,P(60),X(60),Y(60),YTEMP(60),A,
1 XMINSTAT,TUW,TWOP,ITCN,TINVFR,THEFT,KCOLP,
2 XROW,XCOL,NELEM,NETA,NLEM,NLTA,NLETA,NLELM,NETA,JH(60),
2 KINDAS(122),LA(122),LE(502),IA(1000),IE(2000)
2 COMMON/PRICE1/PRICE(130),ICUR(130),TSUMC,ITSINV,LISTL,WBRANC,
1 NS,NP,JFCOL(11),JFROW(11),JPELEM(11),MAXC(10),MAXC2(10)
1 COMMON/TIME1/ITOT,TSTORE,TIME1P,TIME1D,TIMEDC,TIMINV
1 COMMON/BSAV/BSAV(70),BSOFF(70),XSTORE(70),XUBSTO(130),
   XLASTO(130),XUHORG(130),XLEORG(130),ASTORE(2000),ESTORE,
   1 INOWA(2000),LCOLA(140),JESTOR(4000),LESTOR(1002),JHSTOR(70),
   1 INSTO(130),IPARTC(10),TINSTORE(10),NELSTO(10),NETSTO(10)
C RESTORE SUBPROBLEM DIMENSIONS; A MATRIX

   ITIME = ITHEMT(1)
   JPOWER = JFPOWER(NS+1) - JFPOWER(NS)
   NCOL = JFCOL(NS+1) - JFCOL(NS)
   NPTR = JFELMT(NS)
   LENGTH = JFELMT(NS+1) - IPTR
C RESTORE ELEMENTS OF A MATRIX FROM DIAGONAL BLOCK NS
   DO 10 I=1,LENGTH
      A(I) = ASTORE(IPT)
10 IA(I) = TINSTORE(IPT)
10  \( IPTR = IPTR + 1 \)
20  \( IPR = JFCOL(\text{NS}) + \text{NS} - 2 \)
30  \( IPRD = NCOL + 1 \)
40  \( 15 = I=1, IPRD \)
50  \( L(A(I)) = LCOLA(IPTR + 1) \)
60  \( J(PTR) = (\text{NS}-1) \times \text{NEMAX} \)
70  \( L(PTR) = (\text{NS}-1) \times \text{NCL} \)
80  \( \text{MSTAT} = \text{QRL} \)
90  \( \text{IF (MFLAG.EQ.1) GO TO 200} \)

C BACKTRACK STEP: RESTORE RHS H; LP SOLUTION X, JH, KINBAS; BOUNDS

C \( IPR = JFRM(N) \)
100  \( DO 20 I=1, NROW \)
110  \( \text{F(I)} = \text{MOD(IPR)} \)
120  \( \text{X(I)} = XSTOR(EIPTR) \)
130  \( \text{J(I)} = JSTOR(IPTR) \)
20  \( \text{IPTR = IPTR + 1} \)
30  \( \text{IPTR = JFCOL(N)} \)
40  \( \text{DO 40 I} = 1, NCOL \)
50  \( \text{XUB(I)} = XUBSTO(IPTR) \)
60  \( \text{XLB(I)} = XLSTO(IPTR) \)
70  \( \text{KINBAS(I)} = NKSTO(IPTR) \)
80  \( \text{IPTR = IPTR + 1} \)
90  \( \text{RESTORE PARTIAL OBJ. VALUE; IF BACKTRACKING (NOT FOR FORWARD STEPS)} \)
100  \( \text{ISUMC = 1 (IPARTC(NS))} \)

C RESTORE LP BASIS INVERSE

C \( \text{ITSINV} = \text{INVSTO(N)} \)
110  \( \text{KELEM} = \text{KLSTO(N)} \)
120  \( \text{META} = \text{MLSTO(N)} \)
130  \( \text{DO 110} I = 1, \text{NELEM} \)
140  \( \text{JEPTP = JEPTP + 1} \)
150  \( \text{F(I)} = \text{ESTORE(JEPTP)} \)
160  \( \text{DO 110} I = 1, \text{NELEM} \)
170  \( \text{I2PTR = LEPTR + 1} \)
180  \( \text{LE(I)} = \text{LSTOR(LEPTR)} \)
190  \( \text{CTO} \) TO 500

C FORWARD STEP

C RESTORE ORIGINAL BOUNDS FOR A FORWARD STEP

200  \( \text{IPTR = JFCOL(N)} \)
210  \( \text{DO 210} I = 1, \text{NROW} \)
220  \( \text{XUB(I)} = \text{XHORG(IPTR)} \)
230  \( \text{XLB(I)} = \text{XLBORG(IPTR)} \)
240  \( \text{IPTR = IPTR + 1} \)
250  \( \text{IPTR = JFROW(N) - 1} \)
260  \( \text{DO 220} I = 1, \text{NROW} \)
270  \( \text{F(I)} = \text{NORG(IPTR + 1)} \)

C SOLVE LP RELAXATION OF SUBPROBLEM NS

C \( \text{IF (NS.GT.1) CALL FIXPHS} \)

C LP BASIS STARTS OFF AS ALL SLACK BASIS

C \( \text{ITILP = 1 (IP1IM(N))} \)
280  \( \text{DO 310} I = 1, \text{NROW} \)

C
JH(I) = I  
DO 320 I=1,NK0
320  KINHAS(I) = I
    NROWP1 = NROW + 1
    DJ 330 I=NROWP1,NCOL
    KINHAS(1) = 0
C
C SOLVE LP
C
    ITSINV = 99999
    CALL NORMAL(ITSINV)
    IF (MSTAT.EQ.QN) GO TO 2000
    ITIML2 = IHPTIM(1)
    TIMELP = TIMELP + (ITIML2-ITIMLP)/100000.
C
C ITIME2 = IHPTIM(1)
C TSTOKF = TSTOKF + (ITIME2-ITIME)/100000.
C RETURN
C
C L? IS INFEASIBLE
C
2000 IF (NS.EQ.1) WRITE(21,2010)
2010 FORMAT(" SUPROALFM 1 IS INFEASIBLE")
    GO TO 500
C END

C---------------------------------------------------------------

C SUBROUTINE FIXRHS
C
C GIVEN SETTING OF VARIABLES FOR PERIOD (NS-1), COMPUTE NEW
C RHS FOR PERIOD NS IN PREPARATION FOR A FORWARD STEP.
C
C IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (R,D,X,Y),
C INTEGER I,J,K,L,N,F,NS,NSP,ROW
C DOUBLE PRECISION S(2000)
C REAL A(1000)
C
C COMMON XLR(122),XUD(122),DE,DP,R(60),X(60),Y(60),YTEMP(60),A,
C K,N Tot, I, J, L, M, N, COL, NLEM, NETA, NLELM, NUETA, NULEM, NUET,
C 4 KINHAS(122),LA(122),LE(502),IA(1000),IE(2000)
C COMON/STLOT/PRICE(130),TCRXX(130),ISUMC,ITSINV,LSTL,WRANC,
C 1 ISNP,IFCOL(K),IFROW(K),IFCOL(K),MAXC(10),MAXC(10)
C CWMN/FFDG4/JFDG(2000),LCOLND(130),IROWOD(2000),COST(130)
C
C START AT FIRST NONSLACK COLUMN OF LAST PERIOD
C JSEG = IFCOL(NS-1) + (IFROW(NS) - IFROW(NS-1))
C JSTD = JFCOL(K3) - 1
C
C COMPUTE CONTRIBUTION TO RHS FROM LAST PERIOD
C
C 50 100 J=JSEG, JFEND
C 50 100 IFPACK OFFDIAGONAL COLUMN J IF NONZERO
C 50 100 ISEG = LCOLND(J)
C 50 110 JEND = LCOLND(J+1) - 1
C 50 120 IF ( ISEG .GT. JEND ) GO TO 100
C 50 130 J = ISEG, JROW
C 50 50 10 IF J=1, JROW
C 50 50 20 YTEMP(I) = 0.
C 50 50 30 I=1, JEND
C 50 50 40 = JROWOD(I)
Calculate 2 LP based bounds on MAX objective values for periods.

For \( k = n_p, \ldots, 2 \) including offdiagonal columns, also price out the offdiagonal columns.

This subroutine assumes objective row = 1 in each subproblem, and that variables local to period \( n_p \) are numbered after this. Variables which have non-zero entries in periods \( n_p \) and \( n_p+1 \) also total number of nonzero elements in the constraint matrix including offdiagonal columns must not exceed 1000 (for currently dimensioning).

```fortran
IMPLICIT REAL*4 (A,C,E-H,O,P-R-W,Z), REAL*8 (B,D,K,Y)
1 INTEGER*4 (T-N,W)
INTEGR JH, KINBAS, LA, LE, IA, IE, LCOLA, IROWA, IESTOR, LESTOR
DOUBLE PRECISION K(2000), FSTOPE(4000)
REAL A(1000)

COMMON/CONSTS/ZTOLZE, ZTOLPV, ZTCOST, ZTOLSM, NEGINF, NEMAX, NRMAX, QBL,
1 NLES, NMAX, QA, QL, QF, QU, QSU3, QB, QC, QE, QH,QL,Q0, QR,QN, QC
COMMON XLB(122), XUB(122), DE, DP, R(60), X(60), Y(60), YTEMP(60), A,
1 EJSTAT, ITHJ, ITHOW, ITCTR, ITNFRQ, ITRFRQ, ITCOLP,
2 NHOW, YCOL, KELEM, META, KELEM, YMETA, NUELEM, NUETA, JH(60),
3 KINBAS(122), LA(122), LP(502), IA(1000), IE(2000)
COMMON/GSTLT/PRICE(130), ICURX(130), ISUMC, IITSINV, LISTL, NBRANC,
1 NS, NP, JFCOL(11), JFROW(11), JFELEM(11), MAXC(10), MAXC2(10)
COMMON/SISAV/EMOD(70), HURIG(70), XSTORE(70), XUBSTO(130),
1 KLSTO(130), XUBKG(130), KLSTG(130), ASTORE(2000), ESTORE,
2 IPUMA(2000), LCOLA(140), IESTOR(4000), LESTOR(1002), JHSTOR(70),
3 IINSTO(130), IFJAC(10), INVSTO(10), MELSTO(10), NETSTO(10)
COMMON/TFDAC/FFD(2000), LCOLD(130), IROWD(2000), COST(130)

MAXC(K) = WEIGHTED LP BOUND ON MAX OBJ VALUE FOR SUBS K+1,..,NP
OFFDIAGONAL VARIABLES OF SUBPROBLEM \( \star \star \) HAVE A WEIGHT OF 1,
ALL OTHER VARIABLES HAVE A WEIGHT OF 2. THUS AN LP BOUND ON
THE OBJ. VALUE FOR DIAGONAL BLOCK \( K \) \( \star \star \) MAXC(K) IS A VALID
BOUND ON DOUBLE THE OBJ. VALUE FOR PERIODS \( K \), \( K+1 \),..,NP.
(IN TECH. REPORT BY POLLENZ [1980], THIS CORRESPONDS TO
MAXC(1/2, K).)
MAXC2(K) = LP BOUND ON MAX OBJ VALUE FOR SUBS K+1,..,NP. PERIOD K
VARIABLES HAVE COST 0. THOSE FROM PERIODS \( \geq K \) HAVE ORIGINAL COSTS.
(THIS CORRESPONDS TO MAXC(0,K) IN THE TECH. REPORT.)

MAXC(NP) = 0
MAXC2(NP) = 0
LISTC = JFCOL(NP+1) - 1
DO 20 J = 1, LISTC
20 PRICE(J) = 0.
NHOW = 0
READ IN SLACK COLUMNS; SKIP OBJ. SLACKS FOR PERIODS 2,..,NP
```
DO 40 NS=1,NP
   IBEG = JFLELEM(NS)
   IEND = IBEG + JFROW(NS+1) - JFROW(NS) - 1
   IF (NS.EQ.1) IBEG = IBEG + 1
   DO 30 I=IBEG,IEND
       NRW = NROW + 1
       A(NROW) = ASTORE(I)
       IA(NROW) = NRW
       IH(NROW) = NRW
       JH(NROW) = NRW
       KINBA(NROW) = NRW
       Z(NROW) = 0.
       JCOL = JFCOL(NS) + 1 + I - IBEG
       IF (NS.EQ.1) JCOL = JCOL + 1
       XUB(NROW) = XUBORG(JCOL)
       30 CONTINUE
   NS = NP
   NCOL = NROW
   NAPTR = NROW + 1
C READ RIGHT HAND SIDE FOR PERIOD NS
C
100 IFXTRA = NS - 1
   IREG = JFROW(NS) - NS + 2
   IFND = JFROW(NS+1) - NS
   DO 150 I=IREG,IEND
       150 J(NROW) = RORIG(IROW + IEXTRA)
C ADD ON FULL COLUMN FOR NONSLACK PERIOD NS VARIABLES
C
   JREG = JFCOL(NS) + (JFROW(NS+1) - JFROW(NS))
   JEND = JFCOL(NS+1) - 1
   DO 300 J=JFREG,JEND
   C FIRST ADD ON OFFDIAGONAL ELEMENTS OF COLUMN J IF ANY
   NCOL = NCOL + 1
   LA(NCOL) = NAPTR
   IF (NS.EQ.NP) GO TO 220
   IBEG = LCOLOD(J)
   IEND = LCOLOD(J+1) - 1
   IF (JFREG.GT.IEND) GO TO 220
   DO 210 I=IREG,IEND
       210 A(NAPTR) = OFFD(I)
       IA(NAPTR) = 1ROWOD(I) + JFROW(NS+1) - (NS+1)
       NAPTR = NAPTR + 1
   CONTINUE
C NEXT ADD ON DIAGONAL ELEMENTS OF COLUMN J
220 IFNS = LCOLA(J+NS-1) + JFLELEM(NS) - 1
   IEND = LCOLA(J+NS) + JFLELEM(NS) - 2
   DO 250 I=IFNS,IEND
       A(NAPTR) = ASTORE(I)
       IF (1ROWA(I).NE.1ROWJ) GO TO 230
       IA(NAPTR) = A(NAPTR) * 2.O
       GO TO 240
230 IA(NAPTR) = 1ROWA(I) + JFROW(NS) - NS
240 NAPTR = NAPTR + 1
250 CONTINUE
   KINBS(NCOL) = 0
**Fortran Code**

```fortran
XUB(NCOL) = XUBORG(J)
XLB(NCOL) = XLBORG(J)
300 CONTINUE
LA(NCOL+1) = NAPTR
C RESOLVE LP WITH PERIOD NS VARIABLES ADDED
ITSINV = 99999
CALL NORMAL(ITSINV)
C ADD OFFDIAGONAL COLUMNS FROM PERIOD NS-1 TO LP
C NAPTRD = NAPTR
NTEMPC = NCOL
C START AT FIRST NONSLACK COLUMN OF PERIOD NS-1
JBEQ = JFCOL(NS-1) + (JFROW(NS) - JFROW(NS-1))
JEND = JFCOL(NS) - 1
DO 430 J = JBEQ, JEND
   IEQ = LCOLOD(J)
   IEND = LCOLOD(J+1) - 1
   NCOL = NCOL + 1
   LA(NCOL) = NAPTR
   LKBAS(NCOL) = 0
   DO IN COST OF 0 AND BOUNDS OF COLUMN J
   IA(NAPTRD) = 1
   A(NAPTRD) = 0.
   NAPTRD = NAPTRD + 1
   XLB(NCOL) = XLBORG(J)
   XUB(NCOL) = XUBORG(J)
   IF (IEQ.GT. IEND) GO TO 430
   DO 420 I = IEQ, IEND
      IF (IPWOD(I).EQ.IOBJ) GO TO 420
      IA(NAPTRD) = IPWOD(I) + JFROW(NS) - NS
      A(NAPTRD) = JFFD(I)
      NAPTRD = NAPTRD + 1
C CONTINUE
420 CONTINUE
430 CONTINUE
LA(NCOL+1) = NAPTRD
C PRICE OUT EACH OFFDIAGONAL COLUMN OF SUB NS AND STORE IN PRICE.
C CALL UPDATE
CALL FORMC
CALL RTRAN
DO 500 J = JBEQ, JEND
   DFPRICE = 0.
   IEQ = LCOLOD(J)
   IEND = LCOLOD(J+1) - 1
   IF (IEQ.GT. IEND) GO TO 470
   DO 450 I = IEQ, IEND
      IP = IPWOD(I)
      IF (IP.EQ.IOBJ) GO TO 450
      IF (IP .GT. IEND) GO TO 450
      IH = IP + JFROW(NS) - NS
      DF = OFFD(I)
      DFPRICE = DFPRICE + (DE * Y(IP))
450 CONTINUE
C REDUCE REDUCED COSTS. RECALLING THAT OBJECTIVE VALUES WERE DOUBLED.
470 PRICE(J) = DFPRICE / 2.0
500 CONTINUE
WRITE(21,1011) NS
WRITE(21,1026)(PRICE(J), J = JBEQ, JEND)
1010 FORMAT(* PRICE OF OFFDIAGONAL COLUMNS IN SUB.*, I3)
```
RESOLVE LP WITH OFFDIAGONAL COLUMNS ADDED ON. CALCULATE MAXC2.

ITSINV = 99999
CALL NORMAL(ITSINV)
DOJ = X(IDOJ) + ZTOLZE
IF (DOBJ.GE.0.) GO TO 510
DOBJ = X(IDOJ) - ZTOLZE
MAXC2(NS-1) = IDINT(DOBJ/2.0)

ADD IN OFFDIAGONAL COSTS FOR CALCULATION OF MAXC
NCOL = NTEMPC
DO 520 J=JLEG,JEND
NCOL = NCOL + 1
520 A(LA(NCOL)) = COST(J)
RESOLVE LP WITH OFFDIAGONAL COLUMNS ADDED ON AND COSTS SET.

ITSINV = 99999
CALL NORMAL(ITSINV)
IF (WSAT.EQ.QN) GO TO 2000
WS = NS - 1
NCOL = NTEMPC
DOJ = X(IDOJ) + ZTOLZE
MAXC(NS) = IDINT(DOBJ)
IF (DOBJ.LT.0.) MAXC(NS) = MAXC(NS) - 1
IF (WS.EQ.1.) GO TO 100
MAXC AND PRICE HAVE BEEN CALCULATED. RETURN.

K = UP - 1
1505 FORMAT(15150I)
1515 FORMAT(" MAXIMUM CUMULATIVE OBJECTIVE VALUES")
1516 FORMAT(2I15.6)
1517 FORMAT(" FOR LAMBDA = 0.5",9I6)
1518 FORMAT(2I15.6)
1519 FORMAT(" FOR LAMBDA = 0.1",9I6)
1520 FORMAT("")
RETURN

SUPPROBLEM NS IS INFEASIBLE. QUIT
2000 WRITE(21,7C10) NS
2010 FORMAT(" SUPPROBLEM",13, " IS INFEASIBLE")
STOP
END

C-----------------------

SUBROUTINE UDATA
UPDATE AE'S USING VARIABLE HOUNDS ACCORDING TO KINBAS, THEN
USE BASIS INVERSE TO TRANSFORM INTO CORRECT X VECTOR.

IMPLICIT REAL*4 (A,C,F=H,D,P,R-W,Z), REAL*8 (R,D,X,Y),
INTEGER*4 (I-N,Q), INTEGRAL..KINCS,LA,LE,IA,16,
DOUBLE PRECISION *(2000)
REAL A(1000)
SUBROUTINE FTRAN(IPAR)

! PERFORM FORWARD TRANSFORMATION ON COLUMN STORED IN VECTOR Y
! SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
! BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
! **DESCRIPTION OF PARAMETERS**
! IPAR = PARAMETER INDICATING WHICH ETA-VECTORS MATRIX (ALL E OR
! JUST 1 OF LU DECOMP) IS USED TO UPDATE COLUMN Y (INPUT)
!
! IMPLICIT REAL*4 (A,C,F-H,0,P,R-W,Z), REAL*8 (B,D,X,Y),
! INTEGER*4 (I-N),
! INTEGER JH,KINBAS,LA,LE,IA,IE
! DOUBLE PRECISION F(2000)
! REAL A(1000)
!
! COMMON XLB(122),XUB(122),DF,DP,B(60),X(60),Y(60),YTEMP(60),A,
! EP,STAT,INDM,ITM0P,ITCT,INFRQ,ITMRQ,JCOLP,
! ? NROW,NCOL,NLITE,NETA,NLITE,NLITE,NLITE,NLITE,NLITE,NLITE,NLITE,NLITE,
! JH(60),
! KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
!
! NLE = NETA
! NFE = 1
! IF (IPAR .EQ. 2) NFE = NLETA + 1
! IF (NFE .GT. NLF) RETURN
!
! DO 1000 IK = NFE,NLE
! LL = LE(IK)
! KA = LF(IA+1) - 1
! IPIV = IE( LL)
! Y = Y(IPIV)
! Y(IPIV) = Y
! IF (KA .LE. LL) GO TO 1000
!
! LL = LL + 1
! DO 500 J = LL,KK
! Y(IP) = Y(IR) - E(J) * DY
!
! 500 CONTINUE
SUBROUTINE BTRAN

PERFORM BACKWARD TRANSFORMATION ON COLUMN STORED IN VECTOR Y
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-I, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A, C, E, H, Q, P, R, W, Z), REAL*8 (B, D, X, Y),
1 INTEGER*4 (I, N, Q)
1 INTEGER JH, KINBAS, LA, LE, IA, IE

DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON XLB(122), XUB(122), DE, DP, R(60), X(60), Y(60), YTEMP(60), A,
1 F, NSTAT, IOKW, ITPNT, INVFRO, ITRFRO, JCMPL,
2 NG, NCOL, NLEN, NETA, NLELEM, NLETA, NNUETA, JH(60),
3 KINBAS(122), LA(122), LE(502), IA(1000), IE(2000)

IF (NETA .LE. 0) RETURN

DO 1600 I = 1, NETA

IK = NETA - I + 1
LL = LF(IK)
KK = LE(IK+1) - 1
IPIV = IF(LL)
DP = E(LL)
DY = Y(IPIV)
DSUM = 0.

IF (KK .LE. LL) GO TO 600

LL = LL + 1

500 CONTINUE

DO 600 J = LL, KK

IK = IE(J)
DP = E(J)
DPROD = DE * Y(IK)
DSUM = DSUM + DPROD

600 CONTINUE

1600 DSQ = (DY - DSUM) / DP

CONTINUE

RETURN

END

SUBROUTINE FORMC

FORM OBJECTIVE FUNCTION VECTOR; IF BASIS IS INFEASIBLE, SET
OBJECTIVE FUNCTION TO BE TYPICALITY FORM FOR PHASE I.
CALL: FROM SUBROUTINES NORMAL AND ROUND.
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-I, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A, C, F, H, Q, P, R, W, Z), REAL*8 (B, D, X, Y),
1 INTEGER*4 (I, N, Q)
1 INTEGER JH, KINBAS, LA, LE, IA, IE

DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON C, JSTS, ZTULZ, ZTULPV, ZTCONS, ZTULSM, NECTIF, NMAX, NMAX, QBL,
1 MLF, NT, QA, WF, QP, QH, OSDH, ID, QC, QR, QH, QL, QG, QR, QM, QC
COMMON XLH(122), XUB(122), DE, DP, A(6C), X(60), Y(60), YTEMP(60), A,
C

MSTAT = IF
V(IOBJ) = 0.
DO 30 I=1,NROW
   IF (I.EQ. IOBJ) GO TO 30
   ICOL = JH(I)
   IF (X(I) .LE. (XLB(ICOL) - ZTOLZE)) GO TO 10
   IF (Y(I) .GE. (XUB(ICOL) + ZTOLZE)) GO TO 20
   Y(I) = 0.
   GO TO 30
10  Y(I) = 1.
   MSTAT = JH
   GO TO 30
20  Y(I) = -1.
   MSTAT = QH
   CONTINUE
30  IF (MSTAT.EQ.QF) V(IOBJ) = 1.
RETURN
END

C----------------------------------------------------------------------------------
SUBROUTINE PRICE
C
PRICE OUT NONBASIC COLUMNS; CHOOSE PIVOT COLUMN JCOLP FOR
CURRENT PRIMAL SIMPLEX ITERATION. JCOLP=0 ==> DUAL FEASIBLE.
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
IMPLICIT REAL*4 (A,C,E-H,0,P,R-W,Z), REAL*8 (B,X,Y),
1 INTF7JER*4 (I-NQ),
1 INTEGER JH,KINBAS,LA,LL,IA,IE
1 INTEGER IPAR,INCUMB,IVBND,IVTD,IOBND
1 DOUBLE PRECISION E(2000)
1 REAL A(1000)
1 COMMON/CONSTS/ ZTOLZE,ZTULPV,ZTCOST,ZTOLSM,NEGINF,NMAX,NRMAX,QBL,
1 NLFLS,VMAX,GQ,QL,QQ,QQSUB,QA,QG,QE,QH,QL,QO,QR,QM,QQ
1 COMMON XLR(122),XUB(122),DE,DP,R(60),X(60),Y(60),YTEMP(60),A,
1 E,MSTAT,IOBJ,IKWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
1 NROW,NCOL,NELEM,NLELEM,NLELM,NLETA,NUELEM,NUETA,JH(60),
1 KINBAS(122),LA(122),L5(502),IA(1000),IE(2000)
1 CMIN = 1.E10
1 CMAX = -1.E10
D7 1000 D = (ICOL) - JCOLP
IF (KINBAS(J) .GT. 0) GO TO 1000
IF ((XUB(J) - XLR(J)) .LT. ZTOLZE) GO TO 1000
C CALCULATE DPRICE = PRICE OF BRINGING COLUMN J INTO THE BASIS
DPRICE = 0.
LL = LA(J)
KK = LA(J+1) - 1
DO 500 I = LL,KK
   IR = IA(I)
   IF = A(I)
500  DPRICE = DPRICE + (DE * Y(IR))
IF (KINBAS(J) .EQ. -1) GO TO 600
IF (DPRICE .GE. CMIN) GO TO 1000
CMIN = DPRICE
JCOL1 = J
GO TO 1000
600 IF (DPRICE .LE. CMAX) GO TO 1000
CMAX = DPRICE
JCOL2 = J
1000 CONTINUE
C
CHOOSE PIVOT COLUMN JCOLP BASED ON PRICES
C
IF (CMIN .LE. -ZTCOST) GO TO 1500
IF (CMAX .GE. ZTCOST) GO TO 2000
JCOLP = 0
RETURN
1500 IF (CMAX .GE. ZTCOST) GO TO 2500
1600 JCOLP = JCOL1
RETURN
2000 JCOLP = JCOL2
RETURN
2500 IF (ABS(CMIN) - CMAX) 2000, 2000, 1600
END

SUBROUTINE CHZR
C PERFORM MIN-RATIO TEST FOR PIVOT COLUMN JCOLP DETERMINED IN
C SUBROUTINE PREP, THEN SELECT PIVOT ROW IF PIVOT FOR CURRENT
C PRIMAL SIMPLEX ITERATION.
C SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-I, WRITTEN
C BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A,E-H), REAL*8 (B,D,F), INTEGER*4 (I-Q)
DOUBLE PRECISION R(2000)
REAL A(1000)
COMMON/CONSIST/ ZTOLZE, ZTOLPV, ZTCOST, ZTOLSM, NEGINF, NEMAX, NRMAX, QB1,
1 NMAX, NTMAX, GAU, LI, UQ, QG, QSUB, QR, QC, QE, QH, QL, QO, QG, QM, QG
COMMON/RULIC/ DFPART(60), RREVAND, IJCYVAL, ICOLYVAL, IDIR, IPART(122),
1 JTPART(130), IVAND(500), IVID(500), JOBAND(500), MPV, IIP, ITYPE, IFEAS
COMMON XL(122), XUB(122), DE, DP, BC(60), X(60), Y(60), YTEMP(60), A,
1 P, STAT, IPOW, ITCPNT, INFR, ITFRKQ, JCOLP,
2 NMAX, NCOL, NREL, MTAUGH, NLELM, NLET, NULEM, NUETA, JH(60),
3 JKNRAS(127), LA(122), LE(502), I5(1000), IL(2000)
COMMON/WJPFL/ WJPFL, X(IPUDP) AFTER PIVOT.
WJPFL Y CONTAINS FORWARD TRANSFORM OF COLUMN JCOLP.
UP = MIN. RATIO SO FAR. IT IS PASSED TO UPBETA FOR THE PIVOT STEP.
ALSO PASSED TO UPBETA IS DF C = X(IPUDP) AFTER PIVOT STEP.
IFIT = 0 IFF JCOLP IS NONBASIC (AT OPPOSITE BOUND) AFTER PIVOT.
IF (YJKNRAS(JCOLP) .EQ. -1) GO TO 1000
C
INTIAL VARIABLE AT LOWER BOUND; COMPUTE MIN RATIO DP
C
DP = 1.0
DO 500 I= 1, NMAX
IF (IT .EQ. 176J) GO TO 500
JCOL = IP(I)
500 IF (Y(I) .GT. ZTOLPV) GO TO 100
IF (Y(I) .LT. ZTOLPV) GO TO 200
C POSITIVE COEFFICIENT A(I,JCOLP)
100 IF (X(I) .LT. (XLB(ICOL) - ZTOLZE)) GO TO 500
   Dp = (X(I) - XLB(ICOL))/Y(I)
   IF (DE .GT. DP) GO TO 500
   IPTYPE = 0
   GO TO 250
C NEGATIVE COEFFICIENT A(I,JCOLP)
200 IF (X(I) .GT. (XUB(ICOL) + ZTOLZE)) GO TO 500
   Dp = (X(I) - XUB(ICOL))/Y(I)
   IF (DE .LT. DP) GO TO 500
   IPTYPE = -1
250 Dp = DP
   IFDP = 1
500 CONTINUE
   DP = DP + XLB(JCOLP)
   IF (DE .LT. XLB(JCOLP)) GO TO 600
   DP = XUB(JCOLP) - XLB(JCOLP)
   NPIVOT = 0
   RETURN
600 NPIVOT = 1
   RETURN
C INCOMING VARIABLE AT UPPER BOUND; COMPUTE MAX RATIO DP
1000 DP = -1.E10
   DO 1500 I=1,HROW
      IF (I .EQ. IDRJ) GO TO 1500
      JCOL = JH(I)
      IF (Y(J) .GT. ZTOLPV) GO TO 1100
      IF (Y(J) .LT. -ZTOLPV) GO TO 1200
      GO TO 1500
   GO TO 1500
C POSITIVE COEFFICIENT A(I,JCOLP)
1100 IF (X(J) .GT. (XUB(ICOL) + ZTOLZE)) GO TO 1500
   Dp = (X(J) - XUB(ICOL))/Y(J)
   IF (DE .LE. DP) GO TO 1500
   IPTYPE = -1
   GO TO 1250
C NEGATIVE COEFFICIENT A(I,JCOLP)
1200 IF (X(J) .LT. (XLB(ICOL) - ZTOLZE)) GO TO 1500
   Dp = (X(J) - XLB(ICOL))/Y(J)
   IF (DE .GE. DP) GO TO 1500
   IPTYPE = 0
1250 Dp = DE
   IFDP = 1
1500 CONTINUE
   DP = DP + XUB(JCOLP)
   IF (DE .GT. XUB(JCOLP)) GO TO 1600
   DP = XLR(JCOLP) - XUB(JCOLP)
   NPIVOT = 0
   RETURN
1600 NPIVOT = 1
   RETURN
END
C---------------------------------------------
SUBROUTINE WETA
C FORM NEW LTA-VECTORS FOR PRODUCT FORM OF BASIS INVERSE
C SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
IMPLICIT REAL*4 (A, C, E-H, O, P, R-W, Z), REAL*8 (B, D, X, Y),
1 INTEGER*4 (I-N, Q)
INTEGER JH, KINBAS, LA, LE, IA, IE
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZE, ZTOLPV, ZTCOST, ZTOLSM, NEGINF, WEMAX, WMAX, QBL,
1 WELS, WMAX, QA, QI, QF, QG, QSB, QR, QC, QE, QH, QL, QQ, QD, QR, QM, QC
COMMON XLB(122), XUB(122), DE, DP, B(60), X(60), Y(60), YTEMP(60), A,
1 MSAT, 108J, IROWP, ITCNT, INVFPQ, ITRFRQ, JCOLP,
2 NROW, NCOL, NELPM, NELEMA, NLETA, NUELM, NUEMA, JH(60),
3 KINBAS(122), LA(122), LE(502), IA(1000), IE(2000)

NFLEM = NELFM + 1
IE(NFLEM) = IROWP
DO 1000 I = 1, NROW
   IF (I * EQ. IROWP) GO TO 1000
   IF (ABS(Y(I)) .LE. ZTOLZE) GO TO 1000
   NELE4 = NELEM + 1
   IF (NELEM) = 1
   E(NELEM) = Y(I)
1000 CONTINUE
NETA = NETA + 1
LE(NETA+1) = NELEM + 1
RETURN
END

SUBROUTINE SHIFTR( OLD, NEW)

REAL ARRANGE DATA STORAGE; USED BY SUBROUTINE INVERT
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
***DESCRIPTION OF PARAMETERS***
OLD, NEW = PARAMETERS INDEXING STORAGE LOCATIONS IN WHICH
DATA IS TO BE TRANSFERRED (INPUT)

IMPLICIT REAL*4 (A, C, E-H, O, P, R-W, Z), REAL*8 (B, D, X, Y),
1 INTEGER*4 (I-N, Q)
INTEGER JH, KINBAS, LA, LE, IA, IE
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZE, ZTOLPV, ZTCOST, ZTOLSM, NEGINF, WEMAX, WMAX, QBL,
1 WELS, WMAX, QA, QI, QF, QG, QSB, QR, QC, QE, QH, QL, QQ, QD, QR, QM, QC
COMMON XLB(122), XUB(122), DE, DP, B(60), X(60), Y(60), YTEMP(60), A,
1 MSAT, 108J, IROWP, ITCNT, INVFPQ, ITRFRQ, JCOLP,
2 NROW, NCOL, NELPM, NELEMA, NLETA, NUELM, NUEMA, JH(60),
3 KINBAS(122), LA(122), LE(502), IA(1000), IE(2000)

DIMENSION BARRAY(240)
EQUIVALENCE (BARRAY(1), P(1))
IF0 = (IULE - 1) * NRMAX
IFN = (IUNE - 1) * NRMAX
DO 1000 I = 1, NROW
   BARRAY(IF0 + I) = BARRAY(IFN + I)
1000 CONTINUE
RETURN
END
SUBROUTINE INVERT

INVERSE OF CURRENT BASIS BY LU DECOMPOSITION
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TUMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A, C, F-H, O, P-R, W-Z), REAL*8 (R, D, X, Y),
INTEGER JH, KINBAS, LA, LF, IA, IE
DOUBLE PRECISION R(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZ, ZTOLP, ZTOLM, NEGINF, WMAX, OBJMAX, QBL,
1 NLES, NMAX, QA, QD, QF, QN, QSUB, QC, QG, QH, QL, QQ, QR, QM, QG
COMMON XLB(122), XUB(122), D, DP, B(60), X(60), Y(60), YTEMP(60), A,
1 F, YSTAT, IOF, JROW, ITCNT, INVFRQ, ITRFRQ, JCOLP,
2 NROW, NCOL, LELEM, NLEA, NLEB, NUEA, NUEB, JH(60),
3 KINBAS(122), LA(122), LE(502), IA(1000), IE(2000)
COMMON/TIMERS/ITOT, TSTOR, TIMFLP, TIMEDR, TIMEDC, TIMINV

INTEGER MREG, HREG, VREG
DIMENSION MREG(60), HREG(60), VREG(60)
EQUVALENCe (MREG(1), YTEMP(1)), (HREG(1), YTEMP(31)), (VREG(1), X(1))

ITIME = IHPTIM(1)

SET PARAMETERS

META = 0
NLETA = 0
NUETA = 0
4ELEM = 0
NLELEM = 0
NULEM = 0
NAVE = 0
LE(1) = 1
K1 = 0
LH4 = NROW + 1

PUT SLACKS AND ARTIFICIALS IN PART 4 AND REST IN PART 1

DO 100 I = 1, NROW
IF (JH(I) .LT. NROW) GO TO 50
LP4 = LF4 - 1
HREG(KR1) = JH(I)
VREG(KR1) = JH(I)
GO TO 90
50 KR1 = K1 + 1
HREG(KR1) = JH(I)
90 CONTINUE

DO 200 I = LF4, NROW
IF = HREG(I)
HREG(IF) = 0
JH(IF) = IR
KINBAS(IF) = IR

C
C
CONTINUE
C PULL OUT VECTORS BELOW BUMP AND GET NONZ COUNTS
C
NBNDNZ = NROW - LR4 + 1
IF (KRI .EQ. 0) GO TO 1190
J = 1

IV = VREG(J)
LL = LA(IV)
KK = LA(IV+1) - 1
IRCNT = 0
DO 220 I = LL, KK
   NBNDNZ = NBNDNZ + 1
   IF = IA(T)
   IF (HREG(IR) .GE. 0) GO TO 220
   IRCNT = IRCNT + 1
   HREG(IR) = HREG(IR) - 1
   IRP = IR
   CONTINUE
220 CONTINUE
IF (IRCNT - 1) 230, 250, 300

WRITE(21, 8000)
9000 FORMAT(16H MATRIS SINGULAR )
KTHMAS(IV) = 0
VREG(J) = VPEG(KRI)
KRI = KRI - 1
IF (J .GT. KRI) GO TO 310
G7 TO 210
C
VREG(J) = VREG(KRI)
KRI = KRI - 1
LR3 = LR3 - 1
VPEG(LR3) = IV
KREG(LR3) = IRP
HREG(IRP) = 0
JH(IRP) = IV
KTHMAS(IV) = IRP
IF (J .GT. KRI) GO TO 310
G7 TO 210
300 IF (J .GE. KRI) GO TO 310
J = J+1
G1 TO 210
C
PULL OUT REMAINING VECTORS ABOVE AND BELOW THE BUMP AND ESTABLISH MERIT COUNTS OF COLUMNS
C
310 MWREM = 0
IF(KRI .EQ. 0) GO TO 1190
J = 1

IV = VREG(J)
LL = LA(IV)
KK = LA(IV+1) - 1
IRCNT = 0
DO 400 I = LL, KK
   IA = IA(T)
   IF(HPEG(IR) .NE. -2) GO TO 400
C
PIVOT ABOVE BUMP (PART OF L)
C
NABOVE = NABOVE + 1
INDBP = IK
CALL UNPACK(IV)
CALL WRITE
NL = T I = NETA
JH(1R) = IV
KINBAS(IV) = IR
VREG(J) = VREG(KR1)
KR1 = KR1 - 1
NVREM = NVREM + 1
HREG(IR) = IV
G0 TO 940
C
400 IF (HREG(IR) .GE. 0) GO TO 900
IRCN T = IRCNT + 1
IRP = IR
C
500 CONTINUE
C
WRI TE(21,8000)
KINBAS(IV) = 0
VREG(J) = VREG(KR1)
NVREM = NVREM + 1
KR1 = KR1 - 1
IF (J .GT. KR1) GO TO 1010
G0 TO 320
C
PUT VECTOR BELOW BUMP
C
900 VREG(J) = VREG(KR1)
NVREM = NVREM + 1
KR1 = KR1 - 1
LR = LR - 1
VREG(LR) = IV
NREG(LR) = IRP
HREG(IRP) = 0
JH(IRP) = IV
KINBAS (IV) = IRP
C
CHANGE ROW COUNTS
C
940 DO 950 J = 1, KK
II = IA(J)
IF (HREG(IIR) .GE. 0) GO TO 950
HREG(IIR) = HREG(IIR) + 1
950 CONTINUE
C
1000 IF (J .GT. FR1) GO TO 1010
G0 TO 320
C
1010 IF (NVREM = 0) GO TO 310
C
GET PIVOT COUNTS
C
1020 IF (KR1 = 0) GO TO 1190
DO 1100 J = 1, KR1
IV = VREG(J)
LL = LA(IV)
K1 = LA(IV+1) - 1
IMCNT = 0
DO 1050 I = LL, KK
IR = IA(I)
IF (HREG(IR) .GE. 0) GO TO 1050
INCNT = INCNT - (HREG(IR) + 1)
1050 CONTINUE
    MREG(J) = INCNT
1100 CONTINUE
C SORT COLUMNS INTO MERIT ORDER USING SHELL SORT
C
    ISD = 1
1106 IF (KRI .LT. 2*ISD) GO TO 1108
    ISD = 2*ISD
    GO TO 1106
1108 ISD = ISD - 1
C END OF INITIALIZATION
1101 IF (ISD .LE. 0) GO TO 1107
    ISK = 1
1102 ISJ = ISK
    ISL = ISK + ISD
    ISY = MREG(ISL)
    ISZ = VREG(ISL)
1103 IF (ISY .LT. MREG(ISJ)) GO TO 1104
1104 ISL = ISJ + ISD
    MREG(ISL) = ISY
    VREG(ISL) = ISZ
    ISK = ISK + 1
    IF ((ISK + ISD) .LE. KRI) GO TO 1102
    ISD = (ISD - 1) / 2
    GO TO 1101
1106 IF (ISD .LT. ISJ) GO TO 1103
1104 ISL = ISJ + ISD
    MREG(ISL) = MREG(ISJ)
    VREG(ISL) = VREG(ISJ)
    ISJ = ISJ - ISD
    IF (ISJ .GT. 0) GO TO 1103
    GO TO 1106
1107 CONTINUE
C END OF SORT ROUTINE
C PUT OUT BELOW BUMP ETAS (PART OF U)
C
1195 NSLCK = 0
    NSFLOW = 0
    NELAST = NMAX
    NTLAST = NMAX
    LR*(NTLAST + 1) = NELAST + 1
    LR = LR3
    IF (LR3 .GE. LR4) LR = LR4
    IF (LR .GT. NROW) GO TO 2050
    JK = NMAX + 1
1200 JJL = LR4,NROW
    JJ = JK - 1
    IV = VREG(JK)
    T = MREG(JK)
    NSFLOW = NSFLOW + 1
    IF (IV .GT. NROW) GO TO 1200
    NSLCK = NSLCK + 1
    GO TO 2000
1300  MNUETA = NUFTA + 1
       DO 1400 J = LL, KK
          IR = IA(J)
          IF (IR .EQ. I) GO TO 1390
          IF (NELAST) = IF
          E(NELAST) = A(J)
          NELAST = NELAST + 1
          NUELEM = NUELEM + 1
          GO TO 1400
1390  KP = I(J)
1400  CONTINUE
          IF (NFLAST) = I
          E(NFLAST) = EP
          L(NFLAST) = NELAST
          NELAST = NELAST - 1
          NUELEM = NUELEM - 1
          CONTINUE
2000  CONTINUE
2050  IF (KR1 .EQ. 0) GO TO 3500
C
C   DO L-U DECOMPOSITION OF BUMP
C
D7 3000  J = 1, KP1
          IV = VREC(J)
          CALL UNPACK(IV)
          CALL FTRAN(2)
          IROWP = 0
          INCMIN = -999999
          DO 2100 I = 1, NROW
             IF (DABS(Y(I)) .LE. ZTOLPV) GO TO 2100
             IF (HREG(I) .GE. 0) GO TO 2100
             IF (HREG(I) .LE. IRCMIN) GO TO 2100
             INCMIN = HREG(I)
             IROWP = I
2100  CONTINUE
          IF (TROWP .GT. 0) GO TO 2150
          WRITE(21, 8000)
          KINBIAS(IV) = 0
          GO TO 3000
2150  INCR = HREC(IROWP) + 3
C
C   WRITE L AND U ETAS
C
          IF (J .EQ. KR1) GO TO 2160
          NLELEM = NELEM + 1
          IE(NLELEM) = IROWP
          E(NLELEM) = Y(IROWP)
2160  DO 2300 I = 1, NROW
             IF (I .EQ. IROWP) GO TO 2300
             IF (DABS(Y(I)) .LE. ZTOLZE) GO TO 2300
             IF (HREG(I) .EQ. 0) GO TO 2200
C
C   L ETA ELEMENTS
C
          NELEM = NELFM + 1
          IF (NELEM) = 1
          EL(NELEM) = Y(I)
          GO TO 2300
C
C   U ETA ELEMENTS
C
C 2200 IE(NELAST) = I
   E(NELAST) = Y(I)
   NELAST = NELAST - 1
   NUELEM = NUELEM + I
   CONTINUE
C
   JH(IROWP) = IV
   KINBAS(IV) = IROWP
   NUETA = NUETA + 1
   IE(NELAST) = IROWP
   IF (J .NE. KRI) GO TO 2330
   E(NELAST) = Y(IROWP)
   GO TO 2340
2330 E(NELAST) = 1.
   NUETA = NUETA + 1
   LE(NUETA+1) = NUELEM + 1
2340 NUELEM = NUELEM + 1
   LE(NULAST) = NELAST
   NELAST = NELAST - 1
   NULAST = NULAST - 1
C
   UPDATE ROW COUNTS
C
   DO 2350 I = 1,NROW
      IF (DABS(Y(I)) .LE. ZTOLZE) GO TO 2350
      IF (HREG(I) .GE. 0) GO TO 2350
      HREG(I) = HREG(I) - INCR
      IF (HREG(I) .GE. 0) HREG(I) = -1
2350 CONTINUE
   HREG(IROWP) = 0
3000 CONTINUE
C
   MERGE L AND U ETAS
C
   3500 NLETA = NUETA
      NUETA = NLETA + NUETA
      NUELEM = NUELEM
      NLELEM = NUELEM + NUELEM
      IF (NUELEM .EQ. 0) GO TO 3550
      CALL SHFTE
C
   INSERT SLACKS FOR DELFTED COLUMNS
C
   3550 DO 3600 J = 1,NROW
      IF (JH(I) .NE. 0) GO TO 3600
      JH(I) = I
      IROWP = I
      CALL UNPACK(IROWP)
      CALL FTRANS(I)
      CALL WPSTA
3600 CONTINUE
C
   UPDATE X
C
   CALL UPDATX
   LTIMF2 = IMPTIM(1)
   JTINJV = TIMINV + (ITIME2-ITIME)/100000.
   RETURN
END
SUBROUTINE UNPACK(IV)

SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

***DESCRIPTION OF PARAMETERS***

IV = PARAMETER INDEXING COLUMN TO BE EXPANDED (INPUT)

IMPLICIT REAL*4 (A,C,F-H,P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
INTEGER JH,JINDA,LELA,IAE
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON XLH(122),XUB(122),DEP,R(60),X(60),Y(60),YTEMP(60),E,
1 E,JSTAT,OBJ,ITRWP,ITCNT,INVFRQ,ITRFRQ,TCOLP,
2 NROW,NCOL,NELEM,NLE4,MLELEM,MLETA,NUELEM,NUETA,JH(60),
3 KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)

DO 100 I = 1,NROW
  Y(I) = 0.
100 CONTINUE

LL = LA(IV)
KK = LA(IV+1) - 1
DO 200 I = LL,KK
    IF = IA(I)
    Y(IF) = A(I)
200 CONTINUE

RETURN
END

SUBROUTINE SFFT

SUBROUTINE FOR INVFRT
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A,C,F-H,P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
INTEGER JH,JINDA,LELA,IAE
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/XLH(122),XUB(122),DEP,R(60),X(60),Y(60),YTEMP(60),E,
1 E,JSTAT,OBJ,ITRWP,ITCNT,INVFRQ,ITRFRQ,TCOLP,
2 NROW,NCOL,NELEM,NLE4,MLELEM,MLETA,NUELEM,NUETA,JH(60),
3 KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)

SHIFT IE AND F OF U ELEMENTS

NF = NMAX - NUELEM + 1
INCR = 0
DO 1000 I = NF,NMAX
    INCR = INCR + 1
    IF(NUELEM + INCR) = IE(I)
1000 CONTINUE
C
IDIF = NELEM - NLELEM - NUELEM
NF = MTMAX - NUFTA + 1
INCR = 0
DO 2000 I = NF,MTMAX
    INCK = INCR + 1
    LE(NETA + INCR) = LE(I) - IDIF
2000    CONTINUE
LE(NETA+1) = NELEM + 1
RETURN
END

SUBROUTINE UFBETA
C
SUBROUTINE UPDATED RIGHT-HAND SIDES TO REFLECT NEW BASIS RESULTING FROM
CURRENT SIMPLEX PIVOT
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
IMPLICIT REAL*4 (A,G,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
INTEGER JH,KINAS,LA,LE,IA,IE
INTEGER IPART,INCUMB,IVBD,IVID,IOBD
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/BBLIST/DFPART(60),REVBD,INCVAL,ICOL,IVAL,IDIR,IPART(122),
1 INCUMB(130),IVBD(500),IVID(500),IOBD(500),NPVOT,ITYPE,IFEAS
COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
1 F,ASTAT,IBJ,IPMP,ITCNT,INFQF,IRFQJ,JCOLP,
2 NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
3 KINAS(122),LA(122),LE(502),I4(1000),IE(2000)
C
DO 1000 I=1,NROW
1000   X(I) = X(I) - Y(I)*DP
IF (NPVOT .EQ. 1) GO TO 2000
KINAS(JCOLP) = -(KINAS(JCOLP) + 1)
RETURN
2000   A(IPMPI) = DE
IPMPI = JH(IPMPI)
KINAS(JCOLP) = IROWS
IROWS = IPTYPE
JH(IPMPI) = JCOLP
RETURN
END

SUBROUTINE NORMAL(ITSINV)
C
THIS IS THE MASTER PROGRAM FOR LINEAR PROGRAMING COMPONENT
(REVISRED PRIMAL-SIMPLEX METHOD) OF BRANCH-AND-BOUND ROUTINE.
SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
***DESCRIPTION OF PARAMETERS***
ITSINV = NUMBER OF SIMPLEX ITERATIONS SINCE LAST BASIS
INVERSION (INPUT/OUTPUT)
C
IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
INTEGER JH,KINAS,LA,LE,IA,IE
INTEGER IPART,INCUMB,IVBD,IVID,IOBD
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZ, ZTOLPV, ZTCOST, ZTLSM, NWSINFO, NMAX, NRMAX, QBL,
1 NLES, NMAX, QA, IQ, IF, QN, QSUS, QA, QC, QE, QH, QL, QO, QW, QM, QG
COMMON/RLIST/ RPART(60), REVAND, INCVAL, ICOL, IVAL, IDIR, IPART(122),
1 INCUMB(130), IVBND(500), IVD(500), I0BND(500), NPIVOT, IPTYPE, IFEAS
COMMON XLM(122), XUB(122), DE, DPA(60), X(60), Y(60), YTEMP(60), A,
1 R, STAT, I0BJ, IROW, ICTNT, INVFRQ, ITFRQ, JCOLP,
2 NROW, NCOL, NLEM, NLE, NLTA, NUELM, NUTA, JH(60),
3 KINBAS(122), LA(122), LE(502), IA(1000), IE(2000)

IF (ITSINV .LT. INVFRQ) GO TO 1500
1000 CALL INVERT
ITSINV = 0
C
C SIMPLEX CYCLE
C
1500 CALL FORMC
CALL RTRAN
CALL PRICE
IF (JCOLP .GT. 0) GO TO 3000
IF (MSTAT .EQ. 0) GO TO 2000
MSTAT = 283
RETURN
2000 MSTAT = 284
RETURN
C
C PIVOT ON COLUMN JCOLP.
C
3000 CALL UNFACK(JCOLP)
CALL RTRAN(1)
CALL CH24R
CALL UPDATA
ITCNT = ITCNT + 1
ITSINV = ITSINV + 1
IF (NPIVOT .EQ. 0) GO TO 4010
IF (NLEM .GT. (NMAX-NROW)) GO TO 1000
CALL WDATA
4010 IF (ITSINV .GE. INVFRQ) GO TO 1000
IF (ITCNT .GE. ITFRQ) RETURN
61 TO 1500
END
C---------------------------------------------------------------
C SUBROUTINE RANDB(INITBD)
C
C MASTER PROGRAM FOR BRANCH-AND-BOUND INTEGER PROGRAMMING
C routine. Also serves as master program for reoptimization
C via revised dual-simplex method after a forward branch.
C SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
***DESCRIPTION OF PARAMETERS***
C INITBD = INITIAL LOWER BOUND ON MAX. OBJECTIVE VALUE (INPUT)
C
IMPLICIT REAL*4 (A,F,E-H,0,P,R-W,Z), REAL*8 (R,D,X,Y),
1 INTEGER*4 (I-H,Q)
INTEGER JH, KINAS, LA, LE, IA, IE
INTEGER IPART, INCUMB, IVBND, IVD, I0BND
DOUBLE PRECISION E(2000)
REAL A(1000)
C COMMON/CUNSTS/ ZTOLZE, ZTOLPV, ZTCOST, ZTOLSM, NEGINF, NMAX, WMAX, QBL, 
1 NLE, NMAX, QA, QP, GQ, QSUB, QP, QC, QG, QL, GO, QR, QM, QG 
COMMON/BALIST, DPART(60), RFVAND, INCVAL, ICOL, IVAL, IDIR, IPART(122), 
1 INCUR(130), IVAND(500), IVID(500), IORD(500), NPVOT, ITYPE, IFEAS 
COMMON XLB(122), XUB(122), DE, DP, R(60), X(60), Y(60), YTEMP(60), A, 
1 EMSTAT, TOLJ, IROWP, ITCNT, INVFRQ, ITRFRQ, JCMLP, 
2 HROW, NCOL, NLEM, NETA, NLELEM, NLETA, NLBETA, JH(60), 
3 KINBASE(122), LA(122), LE(502), IA(1000), IE(2000) 
COMMON/GSTLT/PPICE(130), ICURX(130), ISUMC, ITSIMV, LISTL, NBRANC, 
1 NS, NP, JPCOL(11), JFROM(11), JFELEM(11), MAXC(10), MAX2(10) 

C IFEAS = 0 
LISTL = 0 
ISUMC = 0 
INCVAL = INITBD 

C TEST FOR FATHOMING 
C 
ENTRY BBENTR 
100 CALL TESTX 
IF (MSTAT .NE. QBL) GO TO 200 
IF (MSTAT .NE. QE) RETURN 
C CURRENT NODE FATHOMED; BACKTRACK TO LAST PROMISING NODE ON LIST 
150 CALL BKTRAK 
C IF LIST IS EMPTY, RETURN TO MAIN (COMPUTATIONS COMPLETED) 
IF (LISTL .EQ. 0) RETURN 
C USE PRIMAL SIMPLEX METHOD FOR REOPTIMIZATION AT NEW NODE 
C CALL NORMAL(ITSINV) 
IF (ITCNT .GE. ITRFRQ) GO TO 2000 
GO TO 100 
C CURRENT NODE NOT FATHOMED; COMPUTE PENALTIES 
C BRANCHING AT CURRENT NODE IS DONE FROM SUBROUTINE PENLTS 
200 CALL PENLTS 
IF (IDIR) 400, 150, 400 
C C INVERT CURRENT BASIS 
1000 CALL IINVERT 
ITSINV = 0 
C 
C DUAL SIMPLEX CYCLE 
C 
C CHOOSE PIVOT ROW IROWP 
300 CALL DCHUZR 
IF (IROWP .GT. 0) GO TO 400 
MSTAT = 2PL 
GO TO 100 
C CHOOSE PIVOT COLUMN JCOLP 
400 CALL DCHUZC 
IF (JCOLP .GT. 0) GO TO 150 
C UPDATE RIGHT-HAND SIDES TO REFLECT NEW BASIS RESULTING FROM 
C CURRENT SIMPLEX PIVOT 
CALL UPEETA 
ITCNT = ITCNT + 1 
IF (ITCNT .GE. ITRFRQ) GO TO 2000 
ITSINV = ITSINV + 1 
IF (NLELEM .GT. (NMAX-NROW) .OR. (ITSINV .GE. INVFRQ)) GO TO 1000 
C WRITE OUT NEW ETA-VECTOR FOR CURRENT SIMPLEX PIVOT 
CALL WPEETA 
GO TO 300
SUBROUTINE DCIIUZR

SELECT PIVOT NOW IROWP FOR CURRENT DUAL-SIMPLEX ITERATION.
SFT IROWP=0 IF CURRENT BASIS IS OPTIMAL. OTHERWISE, CHOOSE
PIVOT TO BE THE ROW WITH GREATEST PRIMAL INFEASIBILITY.
SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
BY GARY A. KOCH (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL *4 (A,C,F,E-H,Q,P,P-P,W,Z), REAL *8 (B,D,X,Y),
1 INTEGER *4 (I-N,Q)
INTEGER JH,KINRAS,LA,LE,IA,IE
INTEGER IPH1,INCUMB,IVRND,IVID,ID8ND
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/CCNSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTULSM,NEGINF,NEMAX,MRMAX,QBL,
1 NLIN,NMAX,ILMAX,QLMAX,QLQMAX,QLQBL,
COMMON/RBLIST/ DPART(60),REVND,INCVAL,ICOL,IVAL,DIR,IPART(122),
1 INCUMP(130),IVBND(500),IVD(500),IOBND(500),NP VOT,IP TYPE,IFEAS
COMMON/XLB(122),XUB(122),DE,DP,PI(60),Q(60),V(60),XTEMP(60),A,
1 E,STAT,INJ,IROWP,ITCNT,INVRQ,ITRFRQ,ICOLP,
2 JROW,NCOL,KELEM,NETA,NELEM,NELETA,NUELEM,NUETA,JH(60),
3 KINRAS(122),LA(122),LE(502),IA(1000),IE(2000)
COMMON/TIMERS/ITOT,TSTURE,TIMEDT,TIMEDR,TIMEDC,TIMINV

C ITIME = INPTIM(1)
IROWP = 0
DP = -1.0E10
D) 1000 I=I+1,NROW
   IF (T .EQ. YOBJ) GO TO 1000
   ICOL = JH(I)
   IF (X(I) .LT. (XLB(ICOL) - ZTOLZE)) GO TO 100
   IF (X(I) .GT. (XUB(ICOL) + ZTOLZE)) GO TO 200
   GO TO 1000
C
C BASIC VARIABLE ON ROW I FALLS BELOW ITS LOWER BOUND
100 DE = XLB(ICOL) - X(I)
   IF (DE .LE. DP) GO TO 1000
   IPTYPE = 0
   GO TO 250
C
C BASIC VARIABLE ON ROW I EXCEEDS ITS UPPER BOUND
200 DE = X(I) - XUB(ICOL)
   IF (DE .LE. DP) GO TO 1000
   IPTYPE = -1
C
250 IROWP = I
   DP = DE
1000 CONTINUE
C ITIME2 = ITIME1
D TIMEDR = TIMEDR + (ITIME2-ITIME)/100000.
RETURN
C
SUBROUTINE DCIIUZR2
SELECT PIVOT COLUMN JCOLP FOR CURRENT DUAL-SIMPLEX ITERATION.
SET JCOLP = 0 IF LP-PROBLEM AT CURRENT NODE IS INFEASIBLE,
OTHERWISE CHOOSE JCOLP TO MAINTAIN PRIMAL-OPTIMALITY.
SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BD, WRITTEN
BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A,C,E-H,N,P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
INTEGER JH,KINBAS,LA,LE,IA,IE
INTEGER IPART,INUMB,IVBND,IVID,IODBD
DOUBLE PRECISION P(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSH,NEGINF,NEMAX,MRMAX,QBL,
1 NLES,NMAX,QA,QF,QM,QSUB,QB,QE,QH,QL,QO,QR,QM,QG
COMMON/BLLIST/ DFPART(60),KEVBD,INCVAL,ICOL,IVAL,IDIR,IPART(122),
1 INUMB(130),IVBND(500),VID(500),IODBD(500),NPVOT,IPTYPE,IFEPAS
COMMON XLB(122),XUB(122),DF,BP(60),X(60),Y(60),YTEMP(60),A,
1 E,STF,STOJ,TRWP,ITCNT,INVFQR,ITFRQR,JCOLP,
2 NROW,NCOL,NELEM,NETA,NLELE,NLTA,NLELM,NEUET,JH(60),
3 KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
COMMON/TIMERS/ITOT,STSTORE,STHELP,TIMDRV,TIMEDC,TIMINV

C ITIME = IHPRTM(1)
JCOLP = 0
IF (IPTYPE .EQ. -1) GO TO 1000

C LEAVING VARIABLE FALLS BELOW ITS LOWER BOUND; COMPUTE MAX RATIO DP
C
DP = -1.E10
100 J=1,NCOL
   IF (KINBAS(J) .GT. 0) GO TO 500
   IF (((XUB(J) - XLB(J)) .LE. ZTOLZE) GO TO 500
   F = J
   CALL UNPACK(K)
   CALL FTRAN(1)
   IF (KINBAS(J) .EQ. -1) GO TO 200
   IF (Y(IRQWP) + ZTOLPV) 225,225,500
200 IF (Y(IRQWP) + ZTOLPV) 500,225,225
225 DL = Y(10BJ)/Y(IRQWP)
   IF (DL .LE. DP) GO TO 500
   JCOLP = J
   DP = DF
500 CONTINUE

C STORE PIVOT COL JCOLP IN Y; STORE CHANGE IN INCOMING VAR. ICOL IN DP
C IF (JCOLP .EQ. 0) RETURN
C
CALL UNPACK(JCOLP)
CALL FTRAN(1)
JCOLP = JH(IROWP)
DP = (Y(IROWP) - XLR(icol))/Y(IROWP)
G1 TO 1000

C LEAVING VARIABLE EXCEEDS ITS UPPER BOUND; COMPUTE MIN RATIO DP
C
1000 DP = 1.E10
500 J=1,NCOL
   IF (KINBAS(J) .GT. 0) GO TO 1500
   IF (((XUB(J) - XLB(J)) .LE. ZTOLZE) GO TO 1500
K = J
CALL UNPACK(K)
CALL FTK(1)
IF (KINBAS(J) .EQ. -1) GO TO 1200
IF (Y(JROWP) - ZTOLPV) 1500, 1225, 1500
1200 IF (Y(JROWP) + ZTOLPV) 1725, 1225, 1500
1225 DE = Y(JROWP) / Y(JROWP)
IF (DE .GE. DP) GO TO 1500
JCOLP = J
DP = DE
1500 CONTINUE
C
STOR PIVOT COL JCOLP IN Y; STORE CHANGE IN INCOMING VAR. ICOL IN DP
IF (JCOLP .EQ. 0) RETURN
CALL UNPACK(JCOLP)
CALL FTRAN(1)
ICOL = JH(IROWP)
DP = (X(IROWP) - XUB(ICOL))/Y(IROWP)
C
2000 IF (KINBAS(JCOLP) .EQ. 0) DE = DP + XLB(JCOLP)
IF (KINBAS(JCOLP) .EQ. -1) DE = DP + XUB(JCOLP)
NPIVOT = 1
D ITIME2 = ITIM(1)
D TMEC = TIMEDC + (ITIME2-ITIME)/10000.
C SUBROUTINE TESTX
C TEST LP-OPTIMAL SOLUTION AT CURRENT NODE FOR FATHOMING.
C FATHOMING OCCURS IF:
C (1) LP PROBLEM AT CURRENT NODE IS INFEASIBLE (MSTAT = QN);
C (2) LP-OPT OBJ. VALUE + OBJ. VALUE FOR PREVIOUS PERIODS +
OBJ. BOUND ON SUCCESSING PERIODS <= OBJ. OF INCUMBENT
C (3) LP-OPT. SOL. SATISFIES INTEGER RESTRICTIONS AND NS = MP
C IF THE LP-OPT. SOL. IS INTEGER BUT FATHOMING DOES NOT OCCUR,
BRANCH ON (FIX) ALL NONSLACK VARS AND STORE SUBPROBLEM NS IN
PREPARATION FOR A FORWARD STEP. SET MSTAT = QE TO FLAG THIS.
C SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
IMPLICIT REAL*4 (A,C,E-H,S,T-P,R-W,Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-N,Q)
1 INTEGER JH,KINBAS,LA,LE,IA,IE
1 INTEGER IPART,INCUMB,IVBND,IVID,IOBND
1 DOUBLE PRECISION E(2000)
1 REAL A (1000)
C COMMON/CONST/ ZTOLZE,ZTOLPV, ZTCOST,ZTOLS, WEGINF, WEMAX, WMAX, QBL,
1 HKS, YMAX, Q, QL, QN, QSUB, QL, QC, QC, QH, QF, QQ, QM, QG
C COMMON/AMLST/ DFPART(60), RKVBD, INCVAL, IVAL, IDIR, IPART(122),
1 INCUMB(130), IVBND(500), IVID(500), IOBND(500), NPIVOT, ITPYPE, IF, S
C COMMON XLB(127), XUB(122), DE, DP, R(60), X(60), Y(60), YTEMP(60), A,
1 E, MSTAT, IOBJ, IROWP, ITCNT, INVFQ, ITRFRQ, JCOLP,
2 NROW, NCOL, NELM, NEFA, NLEL, NLEFA, NLELFA, NLELA, JH(60),
3 KINBAS(122), LA(122), LE(502), IA(1000), IE(2000)
C COMMON/GESLI/PRICE(130), ICRX(130), ISUMC, ITSINV, LISTL, WRANC,
1 NS, MP, JPCOL(11), JFROW(11), JPELEM(11), MAXC(10), MAXC2(10)
C COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
INCUMB = BEST SOLUTION FOUND SO FAR; INCVAL IS ITS OBJECTIVE VALUE
ICURX CONTAINS CURRENT VALUES FOR VARIABLES IN SUBS. 1, . . . , NS-1
IPART(I) (IPART(I)) IS THE FRACTIONAL (INTEGRAL) PART OF X(I)
IVAL = LP-OPT. OBJECTIVE VALUE FOR SUBPROBLEM NS ON CURRENT BRANCH
JBOUND = BOUND ON MAX. OBJECTIVE VALUE (USING MAXC)
ISUMC = OBJECTIVE VALUE FOR SUBPROBLEMS 1, . . . , NS-1 ON THIS BRANCH
IPFAS = 1 IF AN INTEGRAL SOL. WITH INCVAL > INITBD HAS BEEN FOUND

TEST FOR FATHOMING IN WAYS (1) AND (2)

IF (MSTAT .EQ. 0) GO TO 2000
DP = X(IUBJ) + ZTOLZE
IVAL = IDINT(DP)
IF (DP .LT. 0.) JVAL = IVAL - 1
IF (((MAXC(NS)+IVAL)+ISUMC) .LE. INCVAL) GO TO 2000
IF (NS.EQ.NP) GO TO 50
JBOUND = IVAL + MAXC(NS)
IF (JBOUND .LT. 0.) JBOUND = JBOUND - 1
JBOUND = JBOUND / 2
IF (((JBOUND + ISUMC) .LE. INCVAL) GO TO 2000

COMPUTE INTEGRAL AND FRACTIONAL PARTS OF EACH BASIC VAR.

DO 100 I=1,NROW
IPART(I) = IDINT(X(I) + ZTOLZE)
IVAL = IDINT(I)
TEMP1 = IPART(I)
100 DFPART(I) = X(I) - FLOAT(TEMP1)

CHECK FOR ALL-INTEGRAL SOLUTION

DO 200 K=1,NROW
IF (JH(I) .LE. NROW) GO TO 200
IF (DFPART(I) .GE. ZTOLZE) RETURN
200 CONTINUE

SOLUTION ALL-INTEGRAL; CHECK FOR COMPLETE SOLUTION

IF (NS.EQ.NP) GO TO 400

NEW IMPROVED INTEGER SOLUTION TO ALL PERIODS REACHED.
OUTPUT LP. VAL. AND COMPUTATION TIME REQUIRED TO REACH IT.

INCVAL = IVAL + ISUMC
JTIME = 100*1N(1)
TOPT = (JTIME+JTOT)/100000.
IF (IFPAS.EQ.0) WRITE(21,1)
1 FORMAT ("INTERMEDIATE SOLUTIONS FOUND")
WRITE (21,2) TOPT,NBRANC,INCVAL
2 FORMAT ("TIME =",F7.2," SECONDS; BRANCHFS =","110," INCVAL =","110")
IFPAS = 1

STORE NEW INCURRENT SOLUTION
K = JFCOL(KS) - 1
DO 300 J=1,K
300 INCUMB(J) = ICURX(J)
DO 350 J=1,KCOL
IF (PINHAS(J)) 320,330,350
320 INCUMB(K+J) = IDINT(XUB(J))
GOTO 350
330 INCUMB(K+J) = XUB(J)
350
530  INCMJ(K+J) = IDINT(XLB(J))
350  CONTINUE
   DO 360 J=1,NROW
      ICOL = JH(I)
   360  INCUM(J+ICOL) = IPART(I)
      GO TO 200
C A PARTIAL INTEGER SOLUTION HAS BEEN REACHED. BRANCH ON ALL
C NONSLACK, UNFIXED VARIABLES AND SAVE SOLUTION IN ICURX.
C
400  IC = ICOL(NB) - 1
C BRANCH ON ALL NONBASIC NONSLACK VARIABLES FIRST
   DO 500 J=1,ICOL
      IF (KINGAS(J)) 420,440,500
C VARIABLE NONBASIC AT UPPER BOUND; BRANCH UP
   420  ICURX(K+J) = IDINT(XUB(J))
      IF (J.LE.NROW .OR. (XUR(J)-XLB(J)).LE.ZTOLZE) GO TO 500
      IDIR = -1
      RFVND = SNGL(XUR(J))
      GO TO 460
C VARIABLE NONBASIC AT LOWER BOUND; BRANCH DOWN
   440  ICURX(K+J) = IDINT(XLB(J))
      IF (J.LE.NROW .OR. (XUB(J)-XLB(J)).LE.ZTOLZE) GO TO 500
      IDIR = 1
      RFVND = SNGL(XLB(J))
   460  ICOL = J
      CALL BRANCH
   500  CONTINUE
C STORE AND BRANCH ON ALL NONSLACK BASIC VARIABLES
   DO 600 J=1,NROW
      ICOL = JH(I)
      ICURX(K+ICOL) = IPART(I)
      IF (ICOL.LE.NROW) GO TO 600
      IF ((XUB(ICOL)-XLB(ICOL)).LE.ZTOLZE) GO TO 600
      IF ((X(I)-XL8(ICOL)).LE.ZTOLZE) GO TO 520
      IF ((XUB(ICOL)-X(I)).GT.ZTOLZE) GO TO 550
C VARIABLE BASIC AT UPPER BOUND; BRANCH UP
      IDIR = -1
      RFVND = SNGL(XUB(ICOL))
      GO TO 590
C VARIABLE BASIC AT LOWER BOUND; BRANCH DOWN
   520  IDIR = 1
      RFVND = SNGL(XLB(ICOL))
      GO TO 590
C VARIABLE BASIC BETWEEN BOUNDS; BRANCH TO FIX IT
   550  IDIR = -1
      RFVND = FLOAT(IPART(I))
      CALL BRANCH
      IDIR = 1
  590  CALL BRANCH
  600  CONTINUE
C STORE CURRENT SUBPROBLEM AND THE CURRENT OBJ. VALUE
   CALL STORE
   ISUMC = ISUMC + IVAL
   MSTAT = OE
   RETURN
C CURRENT PROBLEM NO LONGER OF INTEREST
SUBROUTINE PENLTS

COMPUTE TOMLIN'S IMPROVED UP- AND DOWN- PENALTIES AND THE
GOMORY PENALTY FOR EACH NONINTEGER BASIC VARIABLE. THEN CHECK
FOR FORCED BRANCHES ON BOTH BASIC AND NONBASIC VARIABLES. IN
THE ABSENCE OF FORCED BRANCHES ON BASIC VARIABLES, ADD TO
EACH PENALTY THE PRICE OF THE CORRESPONDING OFFDIAGONAL COLUMN
OF THE NEXT SUBPROBLEM (NS + 1). THEN CHOOSE AS BRANCHING
VARIABLE THE ONE WITH LARGEST ASSOCIATED UP- OR DOWN-PENALTY.
TAKE THE FORWARD BRANCH IN THE DIRECTION OPPOSITE TO THIS
MAXIMUM PENALTY WHILE ADDING THAT VARIABLE TO THE LIST WITH
THE APPROPRIATE BRANCH DIRECTION (IVID) AND BOUND (IOBND).
THE BRANCING PROCESS ITSELF IS DONE IN SUBROUTINE BRANCH.
SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A-E,H-O,P-W,Z), REAL*8 (B-D,X,Y)
1 INTEGER I,N,Q
INTEGEB JH,KINHAS,LA,LE,IA,IE
INTEGER IPART,INCMRR,IVBND,IVID,IOBND
DOUBLE PRECISION F(2000)
REAL A(1000)
REAL PU(60),PD(60),PG(60)
LOGICAL FORCED

COMMON/CONSTS/ ZTOLZE, ZTOLPV, ZTCST, ZTULZE, NEGINF, NMAX, NRMAX, QBl,
1, LES, NMAX, QA, qi, QP, QN, QSUM, DB, QC, QR, QH, QL, QD, QR, QC, QC
COMMON/PRELIS/ DPART(60), RVPNBD, INIVCAL, INCOL, INVAL, IDIR, IPART(122),
1 INCMRR(130), IVBND(500), IVID(500), IOBND(500), NVIPOI, ICTYPE, IFEAAS
COMMON XLB(122), XUR(122), DE, DP, P(60), X(60), Y(60), YTEMP(60), A,
1 K, STA, TOL, TOLP, ICMUN, ICMUN1, INKROW, JIPRINT, IDIR, ICMUN,
1, KINHAS(127), LA(122), LE(502), IAI(1000), IE(2000)
COMMON/CSSTLT/FRCIC(130), ICURX(130), ISTMC, ITSVINC, LSTL, NBRANC,
1 NS, NP, JCFLH(11), JFRFON(11), JPELEM(11), MAXC(10), MAXC2(10)

PU(I), PD(I), PG(I) ARE THE UP, DOWN, AND GOMORY PENALTIES FOR VAR JH(I)
JTIVCAL = REVISED BOUND ON THE BRANCH VAR ICL; IT IS PASSED TO BRANCH
IF DIR = -1 BRANCH UP, +1 BRANCH DOWN, 0 PATHOMING HAS OCCURRED
IVCAL = LP-OPT. OBJECTIVE VALUE FOR SUBPROBLEM NS ON CURRENT BRANCH
JIVCAL = LOWER ON MAX. OBJECTIVE VALUE FOR SUBPROBLEMS NS,...,NS-1 ON CURRENT BRANCH
JIVCAL Y CONTAINS APPROPRIATE COLUMN OF CURRENT SIMPLEX TABLEAU

01 10 I = 1, NROW
IF (DPART(I) < TOLZE) Go To 5
PU(I) = 1.0E6
PD(I) = 1.0E6
PG(I) = 1.0E6
Go To 10
5
PU(I) = 0.
PD(I) = 0.
PG(I) = 0.
CONTINUE

"11" LOOP: CALCULATE PENALTIES FOR EACH ELIGIBLE INCOMING VAR. J
C
J 1000 J=1,NCOL
IF (KINBAS(J) .GE. 0) GO TO 1000
IF ((XUB(J) - XLN(J)) .LE. ZTOLZE) GO TO 1000
F = J
CALL UNPACK(K)
CALL FTRAN(I)
IF (KINBAS(J) .EQ. 0) GO TO 30
D(J) ZO I=1,NROW
V(I) = -Y(I)

C CHECK FOR FORCED BRANCH ON NONBASIC VARIABLE J

30 IF (J .LE. NROW) GO TO 60
DP = X(IOBJ) - Y(IOBJ) + ZTOLZE
IVAL = IDINT(DP)
IF (DP .LT. 0.) IVAL = IVAL - 1
IF (((MAXC2(NS) +IVAL +ISUMC) .GE. INCVAL) GO TO 50
IF (NS .EQ. NP) GO TO 60
JBOUND = IVAL + MAXC(NS)
IF (JBOUND .LT. 0) JBOUND = JBOUND - 1
JBOUND = JBOUND / 2
IF (((JBOUND + ISUMC) .GT. INCVAL) GO TO 60
IDIR = 2*KINBAS(J) + 1
IF (IDIR .EQ. -1) REVBD = SNGL(XUB(J))
IF (IDIR .EQ. 1) REVBD = SNGL(XLB(J))
ICOL = J
CALL BRANCH
GO TO 1000

C COMPUTE PENALTIES ON BASIC VARIABLES JH(I), FOR ICOL = J

50 DO 50 I=1,NROW
IF (JH(I) .LE. NROW) GO TO 500
IF (DFPART(I) .LT. ZTOLZE) GO TO 500
C COMPUTE UP PENALTY FOR JH(I)
100 IF (Y(I) .GT. -ZTOLZE) GO TO 200
DE = Y(IOBJ)*DFPART(I) - 1)/Y(I)
IF (DE .LT. Y(IOBJ)) DE = Y(IOBJ)
IF (DE .LT. PD(I)) PD(I) = DE
GO TO 300
C COMPUTE DOWN PENALTY FOR JH(I)
200 IF (Y(I) .LT. ZTOLZE) GO TO 300
DE = Y(IOBJ)*DFPART(I)/Y(I)
IF (DE .LT. Y(IOBJ)) DE = Y(IOBJ)
IF (DE .LT. PD(I)) PD(I) = DE
C COMPUTE COMورية PENALTY FOR JH(I)
300 DP = DABS(Y(I))
IF (DP .LE. ZTOLZE) GO TO 500
NTEMP1 = IDINT(DP)
DP = DP - FLOAT(NTEMP1)
IF (((DP .GT. ZTOLZE) .AND. (DP .LT. 1.-ZTOLZE) ) GO TO 330
IF (JH(I) .LT. 0.) GO TO 320
DE = Y(IOBJ)*DFPART(I) / Y(I)
GO TO 350
320 DE = Y(IOBJ)*(1. - DFPART(I)) / (-Y(I))
GO TO 350
330 IF (Y(I) .LT. 0.) DP = 1. - DP
IF (DP .GT. DFPART(I)) GO TO 340
DE = Y(IOBJ)*DFPA(I)/DP
GO TO 350

DP = Y(IOURJ)*(1. - DFP(I))/DP

IF (DE .LT. PG(I)) PG(I) = DE

CONTINUE

1000 CONTINUE

C COMPUTE LARGEST GOMORY PENALTY AND TEST FOR FATHOMING

C PEN = 0.

DO 2000 I=1,NROW
IF (JH(I) .LE. NROW) GO TO 2000
IF (PG(I) .GT. PEN) PEN = PG(I)

2000 CONTINUE

DP = X(IOBJ) - PEN + ZTOLZE
IVAL = IDINT(DP)

IF (DP .LT. 0.) NTEMP1 = NTEMP1 - 1
IF ((MAXC2(NS)+IVAL+ISUMC) .LE. INCVAL) GO TO 2050

IF (NS.EQ.NP) GO TO 3000
JBOUND = IVAL + MAXC(NS)
IF (JBOUND .LT. 0) JBOUND = JBOUND - 1
JBOUND = JBOUND / 2
IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 3000

2050 IFH = 0
RETURN

C PROBLEM NOT FATHOMED: CHECK FOR FORCED BRANCHES ON BASIC X(I)

C FORCED = .FALSE.

DO 3900 I=1,NROW
IF (JH(I) .LE. NPOW .OR. DFP(I) .LT. ZTOLZE) GO TO 3900

IF (PU(I) .GT. PD(I)) GO TO 3600

DP = X(IOBJ) - PD(I) + ZTOLZE

NTEMP1 = IDINT(DP)

IF (DP .LT. 0.) NTEMP1 = NTEMP1 - 1
IF ((MAXC2(NS)+NTEMP1+ISUMC) .LE. INCVAL) GO TO 3050

IF (NS.EQ.NP) GO TO 3900
JBOUND = NTEMP1 + MAXC(NS)
IF (JBOUND .LT. 0) JBOUND = JBOUND - 1
JBOUND = JBOUND / 2
IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 3900

3000 FORCED = TRUE.
RETURN

C FORCED BRANCH UP ON Y(I)

3050 IVAL = NTEMP1

IDI = -1
NTEMP1 = IPART(I) + 1
GO TO 3700

C FORCED BRANCH DOWN ON X(I)

3600 DP = X(IOBJ) - PU(I) + ZTOLZE

NTEMP1 = IDINT(DP)

IF (DI .LT. 0.) NTEMP1 = NTEMP1 - 1
IF ((MAXC2(NS)+NTEMP1+ISUMC) .LE. INCVAL) GO TO 3650

IF (NS.EQ.NP) GO TO 3900
JBOUND = NTEMP1 + MAXC(NS)
IF (JBOUND .LT. 0) JBOUND = JBOUND - 1
JBOUND = JBOUND / 2
IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 3900

3650 IVAL = NTEMP1

IDI = 1
NTEMP1 = IPART(I)
3700 IROWP = 1
  ICOL = JH(IROWP)
  RVBLD = FLOAT(NTEMP1)
  FCWCFD = .TRUE.
  CALL BRANCH
3900 CONTINUE
  IF (FORCED) GO TO 5000
C
C NO FORCED BRANCHES: CHOOSE BRANCHING VAR. AND DIRECTION
C
PEN = 0.
  TRWP = 0
C DETERMINE BASIC VARIABLE JH(IROWP) WITH MAX. UP- OR DOWN-PENALTY,
C ADDING IN A PENALTY FROM THE NEXT SUBPROBLEM (STORED IN PRICE)
DO 4900 I=1,NROW
  IF (JH(I).LE.IROW .OR. DFPart(I).LE.ZTOLZE) GO TO 4900
  UPWPEN = PU(I) + PRICE(JPCOL(NS) + JH(I) - 1)
  IF (UPWPEN .GT. PD(I)) GO TO 4600
  IF (PD(I) .LE. PEN) GO TO 4900
  PEN = PD(I)
  IKWP = I
  IDIR = -1
  NTEMP1 = IPart(I) + 1
  RVBLD = FLOAT(NTEMP1)
  GO TO 4900
4600 IF (UPWPEN .LE. PEN) GO TO 4900
  PEN = UPWPEN
  IKWP = I
  IDIR = 1
  NTEMP1 = IPart(I)
  RVBLD = FLOAT(NTEMP1)
4900 CONTINUE
  IF (IDIR .EQ. 0) GO TO 4950
C EACH UP- AND DOWN-PENALTY = 0. (DUAL-DEGENERACY) CHOOSE ANY
C NONINTEGER BASIC VARIABLE AS BRANCHING VARIABLE ICOL
DO 4910 IROWP=1,NROW
  IF (JH(IROWP) .LE. IROW) GO TO 4910
  IF (DFPart(IROWP) .GE. ZTOLZE) GO TO 4920
4910 CONTINUE
4920 IDIR = 1
  NTEMP1 = IPart(IROWP)
  RVBLD = FLOAT(NTEMP1)
4950 IF (IDIR .EQ. 1) PEN = PU(IROWP)
  ICOL = JH(IROWP)
  DP = X(ION) - PEN + ZTOLZE
  NTEMP1 = IDINT(DP)
  IF (DP .LT. 0.) NTEMP1 = NTEMP1 - 1
  IF (IVAL .GE. NTEMP1) IVAL = NTEMP1
C BRANCH ON CHOSEN VARIABLE
  CALL BRANCH
5000 IF (IDIR .EQ. -1) IPTYPE = 0
  IF (IDIR .EQ. 1) IPTYPE = -1
  RETURN
END
C-----------------------------------------------------------

C SUBROUTINE BRANCH
C
C BRANCH ON VARIABLE ICOL AS DETERMINED IN SUBROUTINE PENLTS
C
SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C
BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C

IMPLICIT REAL*4 (A-C,E-H,O,P-R,S-Z), REAL*8 (B-D,X,Y)
1 INTEGER*4 (I-N,Q)
INTEGER JH,KINDS,LA,LE,IA,IE
INTEGER IPART,INCUMB,IVBND,IVID,IOBND
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,MRMAX,QBL,
1 NLFP,NMAX,OA,QA,P,QF,ON,QSUB,QR,QC,WE,OH,QL,QQ,QR,QM,QG
COMMON/BLIST/ DFPART(60),REVBD,INCVAL,ICOL,IVAL,IDIR,IPART(122),
1 INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFPES
COMMON XLB(122),XUB(122),DE,DP,RE(60),X(60),Y(60),YTEMP(60),A,
1 E,STAT,IOBJJ,ITSEP,ITCHT,JFRQ,ITFRQ,IOCP,
2 NROW,NCOL,NLEL,META,NLEL,META,NUELEM,NUETA,JH(60),
3 KINDS(122),LA(122),LE(502),IA(1000),IE(2000)
COMMON/GENBLIST/PPICE(130),ICUX(130),ISUMC,ITSINV,LISTL,BRANC,
1 NS,NP,JFCOL(11),JFROW(11),JFELM(11),MAXC(10),MAXC2(10)

ICOL INDICES BRANCHING VARIABLE CHOSEN
IDIR = -1 MEANS BRANCH UP, = +1 MEANS BRANCH DOWN
IVID STORES BRANCH VARIABLE, OPPOSITE DIRECTION, AND SUBPROBLEM #
IVBND STORES AN OBJECTIVE FUNCTION BOUND ON THE OPPOSITE BRANCH
IVBND STORES VARIABLE BOUND XUN OR XLR FOR OTHER BRANCH DIRECTION

ADD OPPOSITE DIRECTION TO LIST
WRANC = WRANC + 1
LISTL = LISTL + 1
IF (IDIR = 0. -1) IVBND(LISTL) = IDINT(XLB(ICOL) + ZTOLZE)
IF (IDIR = 0. +1) IVBND(LISTL) = IDINT(XUB(ICOL) + ZTOLZE)
IVAL(LISTL) = IDIR *(ICOL + (NS * 1000))
IVBND(LISTL) = IVAL

REVISE BOUNDS ON BRANCHING VARIABLE FOR FORWARD DIRECTION
IF (IDIR = 0. -1) XUB(ICOL) = DBLE(REVBND)
IF (IDIR = 0. +1) XLB(ICOL) = DBLE(REVBND)
RETURN
END

C-------------------------------

SUBROUTINE AFCHIN

AFTACKTRACK TO A PROMISING (UNPATHMED) NODE FROM THE LIST OF
STORED NODES. EMPLOYS LAST-IN-FIRST-OUT (LIFO) SELECTION RULE
SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
BY GARY A. (OPERATIONS RESEARCH, STANFORD UNIVERSITY)

IMPLICIT REAL*4 (A-C,E-H,O,P-R,S-Z), REAL*8 (B-D,X,Y)
1 INTEGER*4 (I-N,Q)
INTEGER JH,KINDS,LA,LE,IA,IE
INTEGER IPART,INCUMB,IVBND,IVID,IOBND
DOUBLE PRECISION E(2000)
REAL A(1000)

COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,MRMAX,QBL,
1 NLFP,NMAX,OA,QA,P,QF,ON,QSUB,QR,QC,WE,OH,QL,QQ,QR,QM,QG
COMMON/BLIST/ DFPART(60),REVBD,INCVAL,ICOL,IVAL,IDIR,IPART(122),
1 INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFPES
COMMON XLB(122),XUB(122),DE,DP,RE(60),X(60),Y(60),YTEMP(60),A,
1 E,STAT,IOBJJ,ITSEP,ITCHT,JFRQ,ITFRQ,IOCP,
2 NROW,NCOL,NLEL,META,NLEL,META,NUELEM,NUETA,JH(60),
3 KINDS(122),LA(122),LE(502),IA(1000),IE(2000)
COMMON/GENBLIST/PPICE(130),ICUX(130),ISUMC,ITSINV,LISTL,BRANC,
1 NS,NP,JFCOL(11),JFROW(11),JFELM(11),MAXC(10),MAXC2(10)
C  IF LIST IS EMPTY, RETURN (COMPUTATIONS COMPLETED)
50  IF (LISTL .EQ. 0) RETURN
C  GET NEXT NODE FROM LIST. CHECK ITS SURPROBLEM NUMBER.
   ICOL = IVID(LISTL)
   JBOUND = ILPS(ICOL)/1000
   IF (JBOUND .EQ. NS) GO TO 70
C  MAKE BACKTRACKING TO A PREVIOUS SURPROBLEM
60  XBOUND = IBOUND
   CALL RESTOR(0)
70  IF ((MAXC(NS)+IOBND(LISTL)+ISUMC) .LE. INCVAL) GO TO 2000
   IF (XBOUND .LT. NP) GO TO 90
   JBOUND = IOBND(LISTL) + MAXC(NS)
   IF (JBOUND .LT. 0) JBOUND = JBOUND - 1
   JBOUND = JBOUND / 2
   IF ((JBOUND + ISUMC) .LE. INCVAL) GO TO 2000
90  IF (ICOL .LT. 0) GO TO 100
   ICOL = ICOL - (NS * 1000)
C  BRANCH DIRECTION WAS DOWN. RESTORE UP DIRECTION BOUNDS.
   NTEMP1 = IDINT(XUP(ICOL) + ZTOLZE)
   NTEMP2 = IVERN(LISTL)
   XUP(ICOL) = XUP(ICOL) + 1.
   XLR(ICOL) = FLOAT(NTEMP2)
   IF (KINBAS(ICOL) .GT. 0) GO TO 1000
   KINBAS(ICOL) = 0
   NTEMP3 = 1
   GO TO 1000
C  BRANCH DIRECTION WAS UP. RESTORE LOWER DIRECTION BOUNDS.
100  IF (ICOL .LT. (ICOL + (NS * 1000)))
   NTEMP1 = IDINT(XLR(ICOL) + ZTOLZE)
   NTEMP2 = IVERN(LISTL)
   XLR(ICOL) = XLR(ICOL) - 1.
   XUP(ICOL) = FLOAT(NTEMP2)
   IF (KINBAS(ICOL) .GT. 0) GO TO 1000
   KINBAS(ICOL) = -1
   NTEMP3 = 1
C  MARK OLD BRANCH AS FATHOMED
1000  IVID(LISTL) = -IVID(LISTL)
   IVERN(LISTL) = NTEMP1
   IOBND(LISTL) = NEGINF
C  UPDATE X IF NECESSARY
2000  IF (NTEMP3 .NE. 0) CALL UPDATEX
   RETURN
C  NO FATHOMED. UPDATE VAR. BOUNDS AND BACKTRACK AGAIN
2050  IF (ICOL .LT. 0) GO TO 2100
   ICOL = ICOL - (NS * 1000)
   NTEMP1 = IVERN(LISTL)
   IF (KINBAS(ICOL)) 2010, 2050, 2050
SUBROUTINE WRFPUP
C
C  OUTPUT OPTIMAL SOLUTION AND CORRESPONDING OBJECTIVE VALUE.
C  SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C  BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
IMPLICIT REAL*4 (A,E-H,O,P-P-Z), REAL*8 (B,D,X,Y),
1 INTEGER*4 (I-W,Q)
INTEGER JF,JH,K(LP),L,JNL(122),JA,JA1,JA2,
1 INCUMB,IVBND,IVID, IORD, IORDN, IORD2, IORD3, IORD4,
1 LC, M, NF, NP, NBRANCH, NCL, NCP, NCP+1, NCPL, NCPLP,
1 NF, NPL, NPLP, NPLP+1, NPLP, NPLP+1, NPLP, NPLP+1, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
1 NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP, NPLP,
RETURN
40 WRITE (21, 100) ITCNT
100 FORMAT ("SIMPLEX ITERATIONS =", I8,": COMPUTATIONS TERMINATED.")
IF (IFLAP .EQ. 0) GO TO 10
WRITE (21, 101)
101 FORMAT ("BEST INTEGER SOLUTION FOUND IS:")
DO 600 I=1, NP
   WRITE (21, 2) I
   JBEG = JFCOL(I)
   JEND = JFCOL(I + 1) - 1
   WRITE (21, 3) (INCUM(J), J=JBEG, JEND)
600 CONTINUE
WRITE (21, 6) ICVAL
6 FORMAT ("MAX OBJECTIVE VALUE DISCOVERED =", I6)
RETURN
END
**A COMPUTER PROGRAM FOR THE STAIRCASE INTEGER PROGRAMMING PROBLEM**

**LYNNE J. POLLENZ**

This report documents the staircase integer programming computer code SDA. This program is intended for use in solving multitime period integer programming problems with general upper and lower bounds on the variables.