Program Description for a Zero-One Mixed Integer Programming Bounding Algorithm

THE MIPBA PROGRAM

by

David Kendrick

and

Martin Weitzman

January 1967

Memorandum No. 52

PROJECT FOR QUANTITATIVE RESEARCH
IN ECONOMIC DEVELOPMENT
Center for International Affairs
Harvard University
Cambridge, Mass.

The development of this program was supported in part by the National Science Foundation under Grant GS1415.

TABLE OF CONTENTS

ERRATA SHEET

Page 2, 1st line, last paragraph, "one slack variable"

Page 3, footnote amended: $*\gamma$ is any negative constant large enough to make $W > 0$. γ should be set to zero unless $W < 0$ in the original problem. When $\gamma \neq 0$ the true value of the objective function, Z', will be $Z' = Z + \gamma$.

Page 6, 3rd paragraph, second line, "tape and the tape on which"

1. Introduction

This write-up describes a code for solving zero-one mixed integer programming problems. The algorithm used employs a set of additional constraints which are progressively tightened to drive the program to the optimal mixed integer solution. It is designed for solving linear programming problems in which a subset of the variables are constrained to take on only the values zero and one while all other variables in the problem can take on any non-negative values.

The algorithm is coded for FORTRAN II as a single subroutine, named EXTRA4, to R. J. Clasen's all-in-core product-form-of-the-inverse linear programming code, MFOR. This write-up assumes a familiarity with the November 12, 1962 version of MFOR which is available through the SHARE library. Since the algorithm involves the solution of a sequence of linear programming problems, familiarity with the write-up to MFOR will be most helpful in understanding how MIPBA functions.

2. Historical Development

The MIPBA program arose out of the authors' dissatisfaction with the computational inefficiency of existing algorithms for solving zero-one mixed integer programming problems. A detailed description of the theoretical underpinnings will soon be forthcoming; this report is limited to describing how to use the MIPBA computer program.

The method underlying MIPBA is a heuristic "rounding" algorithm which is an extension and refinement of the authors' approach as embodied in [4]. The chief difference is that now penalties are stored internally rather than preserved externally as part of a "tree" structure. As such this algorithm bears a family resemblance to Healy [2], an important difference being the incorporation of a more sophisticated and powerful rule for determining the penalties than the rule first proposed by Healy [2] and used by Driebeek [3].

3. An Example Problem

An explanation of the input to and output from the MIPBA code will be given by using the following example problem, which was taken from a problem in Alan S. Manne's Economic Analysis for Business Decisions, McGraw-Hill, 1963, pp. 103-106:

Minimize W

$$W = .64x_1 + .41x_2 + 48x_3 + .307x_4 + .311x_5 + .579x_6$$

subject to:

(1.1) $x_1 + x_5 \geq 2$

(1.2) $x_1 + x_2 + x_6 \geq 5$

(1.3) $5x_3 \geq x_1$

(1.4) $5x_4 \geq x_2$

(1.5) $x_3, x_4 = 0 \text{ or } 1$

(1.6) $x_1, x_2, x_5, x_6 \geq 0$

To convert the problem to the required format for MIPBA the following changes must be made:

1. Addition to the problem of one stack variable for each zero-one variable. These additional variables along with the constraints discussed in 5. below are required to force the integer variables to zero or one. This is accomplished by assigning a high cost in the objective function to the variable to force it to zero or to the slack of the variable to force the variable to one.

2. The establishment of a new objective function which includes only the zero-one variables and their slacks and the pseudo-objective function variable, W. Equation (2.0).

3. The restatement of the original objective function to include it as an identity in the problem. Constraint (2.3).

4. The addition of one bounding constraint for each zero-one variable. These constraints provide lower bounds on the cost that would be incurred by forcing a zero-one variable to the zero or to the one level. Constraints (2.1) and (2.2).

5. The addition of a constraint for each zero-one variable to force the variable plus its slack to equal to one. Constraints (2.8) and (2.9).

Thus the example problem is rewritten as:

Minimize Z

(2.0) $Z = a_{03}x_3 + a_{04}x_4 + b_{03}s_3 + b_{04}s_4 + W$

subject to

(2.1) $a_{13}x_3 + b_{13}s_3 - W + t_1 = 0$

(2.2) $a_{24}x_4 + b_{24}s_4 - W + t_2 = 0$

(2.3)* $.64x_1 + .41x_2 + .48x_3 + .307x_4 + .311x_5 + .549x_6 - W = \gamma$

(2.4) $x_1 + x_5 - t_4 = 2$

(2.5) $x_1 + x_2 + x_6 - t_5 = 5$

(2.6) $-x_1 + 5x_3 - t_6 = 0$

(2.7) $-x_2 + 5x_4 - t_7 = 0$

---

*γ is any constant large enough to make $W > 0$.

(2.8) $x_3 + s_3 = 1$

(2.9) $x_4 + s_4 = 1$

The coefficients a_{03}, a_{04}, b_{03} and b_{04} of the zero-one variables, x_3 and x_4, and their slacks, s_3 and s_4 in the objective function (2.0) and the coefficients a_{13}, b_{13}, a_{24}, and b_{24} of the zero-one and slack variables in the bounding constraints, (2.1) and (2.2), are originally zero but they must be explicitly input as zero (i.e., the number 0.) along with the other parameters of the problem in order to reserve space in core memory so that they can be changed by the program.

The MPOR routine also requires that the slack and surplus variables be explicitly introduced with the problem. For this example problem the slack variables t_1 and t_2 and the surplus variables t_4, t_5, t_6, and t_7 are included in the corresponding constraints to convert them to equality form.

4. Input Deck

Page 5 provides a listing of the input deck for the sample problem. The column names used are the same as the variable names and the row names used are C1 through C9 to correspond to constraints (2.1) through (2.9).

Let NI be the number of zero-one variables in the problem. The first NI columns in the matrix must be the zero-one variables and the next NI columns must be the corresponding slacks. That is, the first column must be the first zero-one variable and the $(NI+1)^{st}$ column must be the slack for the first zero-one variable. The $(NI+2)^{nd}$ variable is the slack for the second integer variable, etc.

```
BEGIN
EXTRA2
0209
PRMODE
10777777
TOLERA
      REJECT  1.0E-04
      COST   -1.0E-06
END
FREQUE
      INVERT        05
END
EXAMPLE PROBLEM
ROW   Z
RHS
            C1       0.0
            C2       0.0
            C3       0.0
            C4       2.0
            C5       5.0
            C6       0.0
            C7       0.0
            C8       1.0
            C9       1.0
END
MATRIX
      X3    Z        0.0
      X3    C1       0.0
      X3    C3        .480
      X3    C6       5.0
      X3    C8       1.0
      X4    Z        0.0
      X4    C2       0.0
      X4    C3        .307
      X4    C7       5.0
      X4    C9       1.0
      S3    Z        0.0
      S3    C1       0.0
      S3    C8       1.0
      S4    Z        0.0
      S4    C2       0.0
      S4    C9       1.0
      W     Z        1.0
      W     C1      -1.0
      W     C2      -1.0
      W     C3       1.0
```

X1	C3	.640
X1	C4	1.0
X1	C5	1.0
X1	C6	-1.0
X2	C3	.410
X2	C5	1.0
X2	C7	-1.0
X5	C3	.311
X5	C4	1.0
X6	C3	.549
X6	C5	1.0
T1	C1	1.0
T2	C2	1.0
T4	C4	-1.0
T5	C5	-1.0
T6	C6	-1.0
T7	C7	-1.0

END
SOLVE
02

Also the first NI constraints in the matrix must be the bounding constraints [(2.1) and (2.2) in our example] and they must be in the same order as the order of the zero-one variables in the first NI columns. That is, the first bounding constraint (2.1) which concerns the first zero-one variable x_3 (as determined by 2.0) must come first. To assure that this occurs it is necessary to enter zero entries in the right hand side for these constraint rows, even though it is not normally necessary with MFOR to explicitly introduce parameters that are zero, and to list them in the appropriate order beginning immediately after the control card RHS.

The BEGIN control card is used to begin each new problem.

The EXTRA2 control card is used to designate a scratch type and the tape on which the current basis is written. The line following EXTRA2 contains in 2I2 format the numbers of these two tapes. The scratch tape is not used in the present version of the program so the first two columns of this card may contain any tape number. The third and fourth columns contain the number of the tape on which the current basis is written (right justified) after each linear program is solved and from which that basis is read as the starting point for the rest of the program. In the sample problem the basis is written on tape 9.

The PRMODE control card controls the length and frequency of output in the manner described in the MFOR write-up. For use with MIPBA the third column of the card following PRMODE must have a 5, 6, 7, 8, or 9, i.e., the basis must be punched (and written on the basis tape) when optimality is achieved on each linear program.

The magnitude of round-off errors in the problem can be controlled to a degree through the use of the REJECT and COST controls with the TOLERA control card. Round-off errors can also be decreased by more frequent inversion through the use of the INVERT control of the FREQUE set.

Mistakes in the use of the ROW control card are a frequent source of error in the use of MFOR. Care must be taken to right or left justify the objective function row name in the 7-12 column space in exactly the same way that it is justified in the right hand side and matrix input sections. Thus if Z is used as the objective function name and is placed in column 7 of the ROW card, it must be placed in column 13 of the matrix input cards for objective function entries.

The RHS control card should always be placed in front of MATRIX in the input deck to insure that the first NI rows of the matrix will contain the bounding constraints.

SOLVE is used in the example problem as the control to solve the problem. The BASIS, ARTROW, INVERT, and GO sequence may be used if it is desirable to specify a starting basis for the problem. Also any other sequence permitted by MFOR may be used.

The card immediately following SOLVE or GO should have the number of zero-one variables in the problem in the I2 format, i.e., in the example problem, these are the two zero-one variables.

5. Output

The output format of MIPBA is exactly that of MFOR with the exceptions described in this section.

First, the continuous problem is solved and the results printed out. The continuous problem is the mixed integer programming problem with the zero-one variables constrained to be less than or equal to one, but left free to take on any value in the zero to one range.

Figure 1 shows the output format following the continuous solution. In the first line NI gives the number of zero-one variables in the problem and NSTART an indicator of whether or not the run is a restart. (NSTART = 0 indicates the run is <u>not</u> a restart.) The restart procedure is explained in 6.

The following lines show the activity levels of the zero-one variables in the continuous solution. These lines are followed by an indication of the mode of the solution. NQQ = 1 indicates that the problem is in "free" mode, i.e., that all of the zero one variables are free to take on any value between zero and one, so long as those values do not violate the bounding constraints. NQQ = 2 and NQQ = 3 indicate that the problem is in "forced" mode, i.e., that one of the zero-one variables is being forced to either the zero or one level through the device of assigning a high cost to that variable or to its slack.

The following lines give the zero and one penalties which are associated with each of the zero-one variables. These penalties are lower bounds on the increase in the objective function which would result from forcing the variables to

Figure 1

NI = 2	NSTART = 0		
X3	0.400		
X4	0.600		
NQQ = 1	VARIABLE	ZERP	ONEP
	X3	0.0928	0.2880
	X4	0.2328	0.0928
NQQ = 1	VARIABLE	ACZER	ACONE
	X3	2.9790	3.1742
	X4	3,1190	2.9790
	NQQ = 1	S3	SET TO ZERO

---

the respective levels. The ACZERS and ACONES in the following lines are the values of the ZERPS and ONEPS plus the value of the objective function plus (in some cases) the MAXMIN. The MAXMIN is the maximum over all the zero-one variables of the minimum over each ZERP and ONEP when the program is in the "forced" mode. Thus the MAXMIN provides a lower bound on the cost above the current objective function level for a mixed integer solution to the problem with one of the zero-one variables forced to its current level.

The ACZERS and ACONES are maintained at the highest level reached at any time, i.e., they are only changed when a higher ACZER or ACONE is obtained. The ZERPS and ONEPS on the other hand are current and specific to the previous linear program solution.

After the continuous solution is obtained and at the completion of each "free" mode solution one of the zero-one variables is selected to be forced to first one level (the NQQ = 2 stage) and then to the other (the NQQ = 3 stage).

A message is printed to indicate that either the variable or its slack is being forced to the zero level.

The "free" and "forced" modes follow one another until all zero-one variables are within .01 of integrity in the "free" mode, at which time a CALL EXIT is executed and the program is halted.

6. Restart Procedure

The program may be restarted from the level reached at any solution in the "free" mode. This is accomplished by inserting a one in the 4th column of the input card following the SOLVE control card. This is the same card that contains in its first two columns a number indicating the number of zero-one variables in the problem.

In addition, the ACZERS and ACONES from the "free" mode solution must be included in the input on subsequent cards in (18X, 2F12.5) format. They must be in the same order as printed in the output.

REFERENCES

[1] Driebeek, Norman J., "An Algorithm for the Solution of Mixed Integer Programming Problems," Management Science, Vol. 12, No. 7, March, 1966.

[2] Healy, W. C. Jr., "Multiple Choice Programming," Operations Research Journal, Vol. 12, No. 1, January-February 1964, pp. 122-138.

[3] Land, A. H., and Doig, A. G., "An Automatic Method of Solving Discrete Programming Problems," Econometrica, 28, No. 3, July 1960, pp. 497-520.

[4] Kendrick, D. A. and Weitzman, Martin, "Progress Report on a Branch and Bound Algorithm for Zero-One Mixed Integer Programming Problems," D/66-3, Center for International Studies, M.I.T., Cambridge, Mass., June 1966.