MCDANIEL, Roger Dale, 1942-
COMPUTATIONAL ASPECTS OF VARIOUS BENDERS-BASED PROCEDURES FOR MIXED INTEGER PROGRAMMING.

The University of Oklahoma, Ph.D., 1973
Operations Research

THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

COMPUTATIONAL ASPECTS OF VARIOUS BENDERS-BASED PROCEDURES FOR MIXED INTEGER PROGRAMMING

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

DOCTOR OF PHILOSOPHY

BY

ROGER D. MCDANIEL

Norman, Oklahoma

1973

COMPUTATIONAL ASPECTS OF VARIOUS BENDERS-BASED PROCEDURES FOR MIXED INTEGER PROGRAMMING

APPROVED BY

Michael D. Devine

Hillel Kumin

B L Foote

DISSERTATION COMMITTEE

ABSTRACT

This dissertation examines possible modifications to Benders' partitioning procedure for the solution of mixed integer problems with the intent of reducing the amount of time required for the solution of such problems. In general this can be done in two ways, by reducing the number of iterations required by the procedure for solution and/or by reducing the time per iteration.

The three modifications to Benders' original algorithm all relied upon changes in the technique of solution of the integer subproblem. On larger problems the majority of the time spent in solving mixed integer problems arises from the integer subproblem and thus this would be an excellent place to reduce solution time.

The changes effected in the integer subproblem were: (a) solution of the integer program to a non-optimal, feasible point, (b) addition of multiple constraints to the integer subproblem at each iteration, and (c) solution of the integer subproblem as a linear program until certain characteristics are recognized as indicating a need for solution as an integer program.

The first of these modifications attempts to reduce the time spent per iteration in the integer subproblem with the hopes that the number of iterations will not be increased. The second modification's efficiency depends chiefly upon reducing the number of iterations required for solution but, to some extent, also results in reduced time per iteration as well. The last suggested modification relies entirely upon reduction of time per iteration, which, hopefully, will offset the necessitated extra iterations which quite frequently arise.

The research for this dissertation consisted of analysis of the effect of the proposed modifications upon time of execution, coding of computer programs to solve problems using Benders' original algorithm and Benders' algorithm plus modifications, and analysis of results obtained on a large number of real-life problems.

ACKNOWLEDGEMENTS

The author would like to express his sincere appreciation for all help given him by his advisor, Dr. Mike Devine. It was through his aid that the original idea of this dissertation was formulated and his continuing efforts which kept it steadfast to the original aim of research. The work could certainly not have been done without his ideas and constant attention to detail.

Appreciation is also expressed for the interest and assistance of Clark A. Mount-Campbell who contributed time, effort and test problems for the research.

Finally, the deepest appreciation goes to the author's wife who suffered through the months of creation of this dissertation.

TABLE OF CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS vi

INTRODUCTION 1

A. Previous Research and Applications . . 2

B. Benders' Algorithm for Mixed Integer Programming 4

INVESTIGATION OF GLOVER'S REDEFINITION 19

SOME PARTITIONING ALGORITHMS FOR THE MIXED INTEGER PROGRAMMING PROBLEM 28

A. Use of Non-Optimal Integer Subproblem Solutions 29

B. Generating Multiple Constraints per Iteration 38

C. Solving of the Integer Program as a Linear Program 47

GENERAL RESEARCH PROCEDURES AND RESULTS 55

APPENDIX A 65

APPENDIX B 69

APPENDIX C 73

APPENDIX D 76

APPENDIX E 93

BIBLIOGRAPHY 99

INTRODUCTION

In 1962 Benders (7) developed an algorithm for the solution of programming problems which involve a mixture of either different types of variables or functions. The general mixed programming problem can be expressed as (28):

$$\begin{aligned} &\text{Minimize:} && X_o = CX + f(Y) \\ &\text{Subject to:} && AX + F(Y) \geq B && \text{(P1)} \\ & && X \geq 0 \quad Y \in S. \end{aligned}$$

A is an m by n matrix, X and C are n-vectors, Y a p-vector, F an m-vector whose components are functions of Y, f a scalar-valued function of Y, B an m-vector and S a subset of E^p, the Euclidean space of dimension p.

As an example of (P1), one can consider a problem in which both functions f(Y) and F(Y) are nonlinear. One would then have a problem in which the functions involve both linear and nonlinear terms. These linear and nonlinear terms are separable by definition and therefore Benders' algorithm may be applied to the problem. As another example of a possible form of (P1), consider the case in which all the functions of Y are linear but Y is constrained to be integer valued, i.e. $S=\{Y \mid Y \geq 0 \text{ and } Y \text{ integer}\}$. This is the type problem to be considered in this dissertation.

A. Previous Research and Applications

Although the procedure outlined by Benders can be used on any mixed integer problem, other more specialized methods have generally been used in the optimization field. With the ready availability of commercial computer codes for mixed integer programming it is seldom that the coding of a general algorithm such as Benders' is attempted for a single application problem.

It appears, therefore, that research might be done into the possible advantages of Benders' algorithm over other solution procedures on specific application problems. It would seem from the literature available that Benders' algorithm has not found wide spread use. This lack of references, however, is not necessarily an indication of lack of research into the algorithm.

Buzby, Stone and Taylor (9) were probably the first persons to get computational experience using Benders' algorithm. In 1964-1965 they solved a type of non-linear distribution problem which was slightly more general than the plant location problem. By randomly generating such problems and using Benders' algorithm, they found that as the problems became more difficult, i.e. more feasible answers, the realative efficiency of the procedure increased.

Geoffrion (17) used Benders' algorithm in the design of optimal distribution systems and (18) generalized Benders' algorithm so as to handle non-separable functions under certain conditions.

Gorry, Shapiro and Wolsey (23) used an adaptation of Benders' algorithm in which group theoreticsmethods were usdd to solve the series of integer subproblems arising in the procedure.

Balinski and Wolfe (3) used Benders' algorithm on a plant location problem and had promising results on small problems.

Finally, Muckstadt and Wilson (34) also developed a mixed integer problem dealing with the scheduling of thermal generating systems and showed good results using Benders' algorithm. Despite their assertions of extensive modifications, the algorithm used was essentially Benders' original algorithm.

This dissertation will place heavy emphasis on the computational comparison of Benders' original algorithm and some other algorithms which are based on Benders' dual-decomposition concepts. The type problems to be investigated will be restricted to the mixed integer programming problem with binary variables. In this way, it will be possible to obtain computational results on a type of problem which has numerous applications.

Glover's redefinition of Benders' algorithm is also examined for the extra insight which it can give. A comparison of the similarities, differences, and the general relative efficiency of the two statements of the same procedure is given. No attempt is made, however, to computationally compare the two.

B. Benders' Algorithm for Mixed Integer Programming

Since this paper will deal specifically with the mixed integer problem, (P1) is here restated as such and development of Benders' algorithm will be in terms of this restatement. The general mixed integer problem is (28):

$$\begin{aligned} &\text{Minimize:} && X_o = CX + C'Y \\ &\text{Subject to:} && AX + A'Y \geq B \qquad\qquad (P2) \\ & && X,Y \geq 0 \quad Y \equiv 0 \text{ (mod 1)}. \end{aligned}$$

Both the objective function and all of the constraints are linear in both X and Y. $Y \equiv 0$ (mod 1) is a mathematical statement of the requirement that Y have only integer values.

Let Y take some specific value, say Y^*. Then (P2) becomes:

$$\begin{aligned} &\text{Minimize:} && X_o = CX \\ &\text{Subject to:} && AX \geq B - A'Y^* \qquad \text{(P3)} \\ & && X \geq 0 \end{aligned}$$

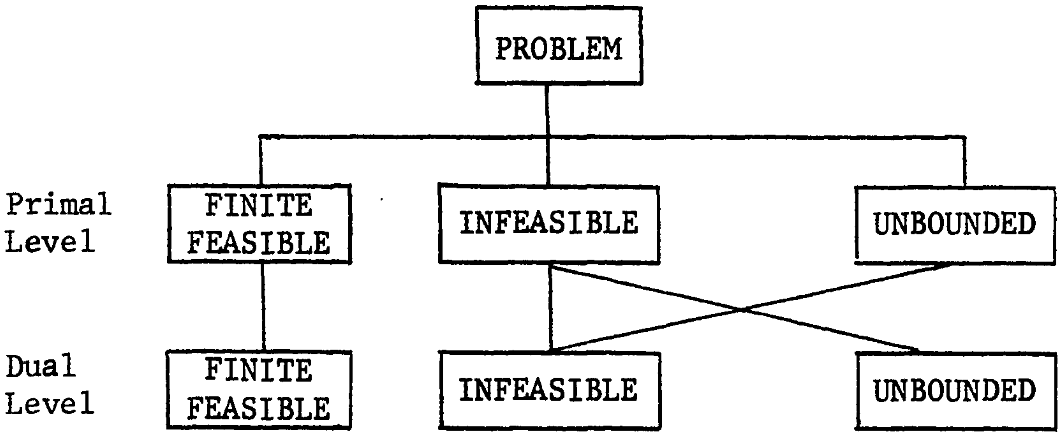
which is merely a linear program since Y^* is a constant. (P3) can, of course, be solved by any linear programming technique. Note that an objective function of $X_o = CX$ may be used in place of $X_o = CX + C'Y^*$ since the addition of a constant ($C'Y^*$) to the objective function in no way affects the solution point of the linear program. After such a solution point has been found, the objective function value calculated would be supplemented by the amount $C'Y^*$ to give the true objective function value at that point.

Now consider the dual of (P3).

$$\begin{aligned} &\text{Maximize:} && U_o = U(B - A'Y^*) \\ &\text{Subject to:} && UA \leq C \qquad \text{(P4)} \\ & && U \geq 0. \end{aligned}$$

Note that the feasible region (as defined by the constraints) is now independent of Y so that regardless of what value Y may assume, the optimal point of (P4) will be a vertex of the space defined by $UA \leq C$ and $U \geq 0$. In the solution of (P4) three cases may occur. First, there may be no feasible solution; second, (P4) may be unbounded; and finally, a feasible finite solution may be derived.

Case 1: Suppose (P4) has no feasible solution for $Y = Y^*$. Since the feasible region is independent of Y, there will be no feasible solution for (P4) for any value of Y. Let us consider the relationships between the primal and dual problems so that infeasibility in the dual problem can be explained in relation to the primal problem.



By inspection of this relationship tree it can be seen that if the dual has no feasible solution, then the primal could be either unbounded or infeasible. One can easily determine which of these two possibilities is true by adding a constraint to the primal problem so as to prevent unboundedness. The most commonly used constraint for this purpose is $X + Y \leq M$, where M is an extremely large number. The addition of this constraint to the primal problem will have the effect of increasing the dimensionality of the dual by one; i.e. adding one variable. If the dual still has no feasible solution, the original mixed integer problem must be infeasible for all values of Y as now the primal problem cannot be unbounded. If it occurs that the dual now has a feasible solution, the original mixed integer problem was unbounded and by using Step 1 of the procedure outlined below it is possible to determine a value of Y at which this occurs. One would then have a solution to the original mixed integer problem, namely $X_0 = -\infty$ (unbounded), $Y = Y^*$ and X such values as necessary to produce the unboundedness.

Case 2: If (P4) is determined to have an unbounded solution, add to (P4) the constraint:

$$\sum_{i=1}^{m} U_i \leq M \qquad \text{(E1)}$$

where M is again an extremely large number. Upon the resolving of (P4), one should obtain a feasible finite solution. Note that unboundedness in the dual results from infeasibility in the primal problem. Thus, for $Y = Y^*$, the primal problem is infeasible. One would therefore like to delete Y^* from any future consideration in the process of the algorithm. By using Benders' algorithm, any future objective function value derived for the original mixed integer problem when $Y = Y^*$ will be very large (due to the addition of (E1) to the set of constraints) and as the problem is being minimized, simply continuing in the algorithm will have the practical effect of Y^* being deleted from future consideration.

Case 3: It is well known that a set of linear constraints results in a space which is convex, i.e. any positive linear combination of two points within the space is also within the space. Also, if the space is bounded, a solution to a linear program lies at one of the extreme points or vertices of that space.

Thus, in the third case, i.e. the obtaining of a finite feasible solution to (P4), the solution will occur at some vertex of the space defined by $UA \leq C$ and $U \geq 0$. Suppose there are a total of N such vertices and let U^P represent the P'th such vertex. One can then write (P4) as:

$$\text{Maximize: } U_o = U^p(B - A'Y^*) \quad \text{for } p = 1,2,\ldots,N. \qquad \text{(P5)}$$

By duality theory, the solution to (P5), say U_o^*, must be equal in value to the value of the objective function of the primal problem at its solution point. Thus, $U_o^* = CX$. Since $X_o = CX + C'Y$, substituting (P5) into (P2) results in:

$$\text{Minimize:} \quad X_o = C'Y + \max_{p=1,\ldots,N} U^p(B - A'Y)$$

$$Y \geq 0 \quad Y \equiv 0 \pmod 1. \qquad \text{(P6)}$$

(P6) can be expressed as:

$$\text{Minimize:} \quad Z$$

$$\text{Subject to:} \quad Z \geq C'Y + \max_{p=1,\ldots,N} U^p(B - A'Y)$$

$$Y \geq 0 \quad Y \equiv 0 \pmod 1. \qquad \text{(P7)}$$

Finally, (P7) can be written:

$$\text{Minimize:} \quad Z$$

$$\text{Subject to:} \quad Z \geq C'Y + U^p(B - A'Y) \quad p=1,\ldots,N$$

$$Y \geq 0 \quad Y \equiv 0 \text{ (Mod 1)}. \qquad \text{(P8)}$$

In (P8), there would be N constraints of the form $Z \geq C'Y + U^p(B - A'Y)$, one constraint for each vertex (U^p) of the hyperspace created by the constraints of (P4). Since there are N vertices in the dual solution space, a maximum of N constraints will completely define the integer problem and hence the original mixed integer problem. As N is typically very large, however, and the number of constraints which are tight at the final solution stage is small, one would like to solve a "relaxed" version of (P8), i.e. one that has only a small subset of the N constraints. Benders' algorithm is an iterative procedure which solves a relaxed version of (P8); at each iteration a new constraint of (P8) is added to the relaxed problem.

Outlined below is the complete Benders' algorithm for mixed integer programming. Following that is a detailed graphical description of (P8), and Benders' algorithm for solving (P8).

<u>Step 0</u>: Set t, the iteration number counter, equal one. Select some $U^1 \geq 0$ such that $U^1 A \leq C$. It is not necessary that this point be a vertex of the space defined by $UA \leq C$. If upon examination of the problem it is found that no such point exists, then the original problem had no feasible solution or was unbounded, and the method previously outlined can be used to determine which case has occurred.

<u>Step 1</u>: Solve the "pure"[1] integer problem:

$$\begin{aligned} &\text{Minimize:} && Z \\ &\text{Subject to:} && Z \geq C'Y + U^i(B - A'Y) \quad i=1,\ldots,t \\ & && Y \geq 0 \quad Y \equiv 0 \pmod{1}. \end{aligned}$$

[1] The use of quotation marks around the word pure in "pure" integer problem throughout this paper arises from the fact that although Y is restricted to integer values, the value of Z, which can be considered as merely an additional variable, may be non-integral. This is due to the possibility of non-integral coefficients in the constraints of the integer problem. The algorithm developed to handle this "almost integer program" is given in Appendix E.

Let Z^t and Y^t be the solution. If Z is unbounded from below, take Y^t to be some point that gives Z^t some arbitrarily large negative value.

Step 2: Solve the linear problem:

$$\text{Maximize:} \quad U_o = U(B - A'Y^t)$$

$$\text{Subject to:} \quad UA \leq C$$

$$U \geq 0.$$

If U goes to infinity, i.e. the problem is unbounded, add the constraint:

$$\sum_{i=1}^{m} U_i \leq M,$$

where M is a large number, and resolve the linear program. Let the solution to this linear program be U_o^{t+1} and U^{t+1}. Determine whether:

$$Z^t - C'Y^t = U^{t+1}(B - A'Y^t).$$

If this equality holds, Y^t is the value for Y such that the original problem is optimized. In that case, proceed to Step 3. If, however:

$$Z^t - C'Y^t < U^{t+1}(B - A'Y^t),$$

not all of the necessary constraints have been derived to find a final solution to the mixed integer problem and the most violated constraint should be added to the existing set of constraints in the integer program. The addition of this most violated constraint means that if $Y = Y^t$ in some future iteration, the value of Z associated with Y^t will be greater than Z^t. This is desirable as Y^t did not result in the optimal solution to the mixed integer problem and thus $Z^* > Z^t$.

Add the constraint:

$$Z \geq C'Y + U^{t+1}(B - A'Y)$$

to the integer problem. Let t = t + 1. Return to Step 1.

Step 3: Using the Y^t obtained in the iteration in which $Z^t - C'Y^t = U^{t+1}(B - A'Y^t)$, return to the original problem and solve:

$$\begin{aligned} \text{Minimize:} \quad & X_o = CX + C'Y^t \\ \text{Subject to:} \quad & AX \geq B - A'Y^t \\ & X \geq 0. \end{aligned}$$

Proof of the finiteness of the algorithm, proof of the optimality of the Y and X values derived and the yielding of the upper and lower bounds on the optimal objective function value during execution of the procedure are given in Appendix A.

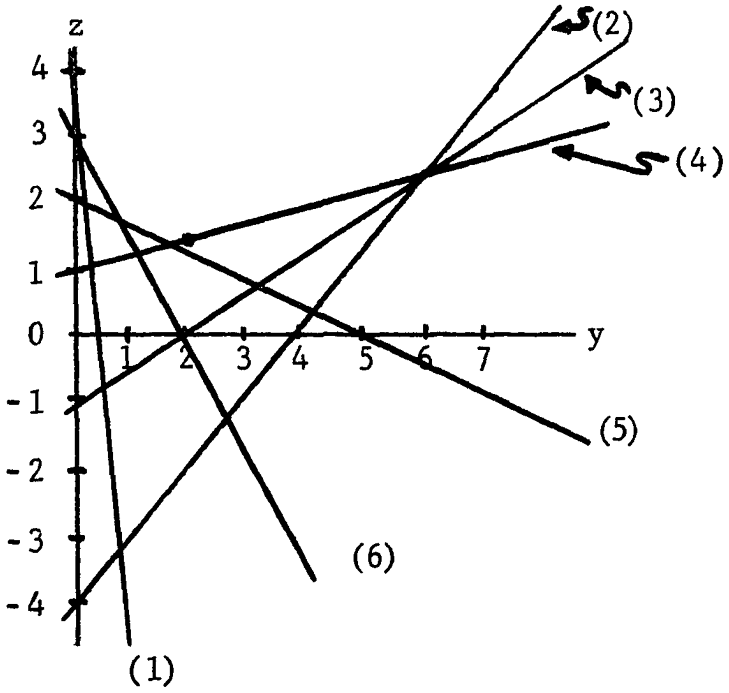
As an aid to help understand further the procedure involved in the relaxation of the integer program, consider the following problem. The mixed integer problem can be expressed as a "pure" integer problem of the form:

$$\begin{aligned} &\text{Minimize:} \quad Z \\ &\text{Subject to:} \quad Z \geq C'Y + U_i^p(B - A'Y) \quad i=1,\ldots,N \\ &\qquad Y \geq 0 \quad Y \equiv 0 \pmod 1. \end{aligned} \qquad \text{(P6)}$$

In order to better visualize the procedure involved, let us consider a graphical display of the full integer problem and note the graphs produced by the procedure at different iterations. Unfortunately, in order to have a graphical display, it is necessary to restrict ourselves to a single integer variable. Thus, consider the case where N = 6 and Y consists of a single variable y. Furthermore, let the points U_i^p, i=1,...,N be such that (P8) becomes:

$$\begin{aligned} &\text{Minimize:} \quad z \\ &\text{Subject to:} \quad z \geq -7y + 4 \quad (1) \\ &\qquad z \geq y - 4 \quad (2) \\ &\qquad z \geq 2/5\, y - 1 \quad (3) \\ &\qquad z \geq 1/6\, y + 1 \quad (4) \\ &\qquad z \geq -2/5\, y + 2 \quad (5) \\ &\qquad z \geq -3/2\, y + 3 \quad (6) \\ &\qquad y \geq 0 \quad y \equiv 0 \pmod 1. \end{aligned} \qquad \text{(P9)}$$

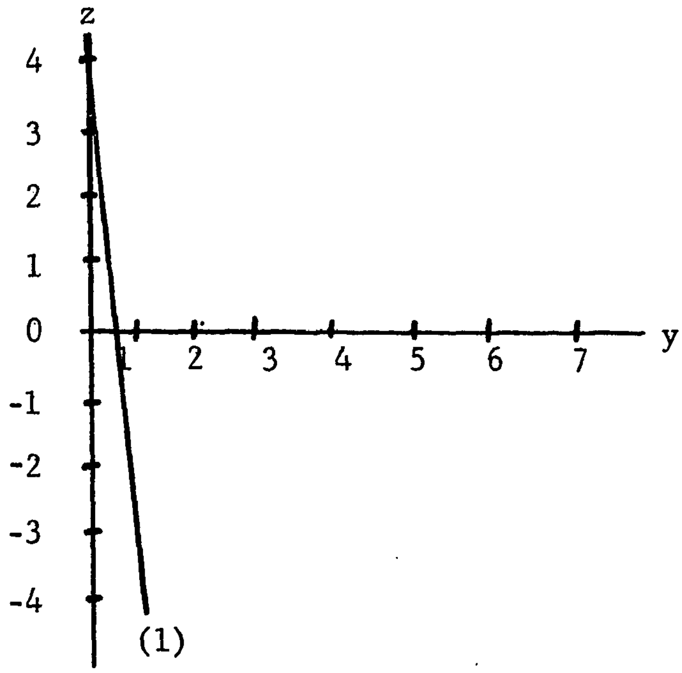
Thus, the complete problem would be:



(D1)

It is easy to see the minimal z over all y from this drawing, namely z = 4/3 for y = 2.

Consider Benders' algorithm which finds this point by relaxation of the problem and the addition of constraints, one at a time. Begin with only the constraint y $\geq$ 0 and set y = 0 as a beginning value. The procedure now would determine the most violated in the set of constraints by the solution of a linear program. Graphically, it can be seen that that constraint is (1). Thus, the problem becomes:



(D2)

When one solves:

Minimize: z

Subject to: $z \geq - 7 y + 4$ (1) (P10)

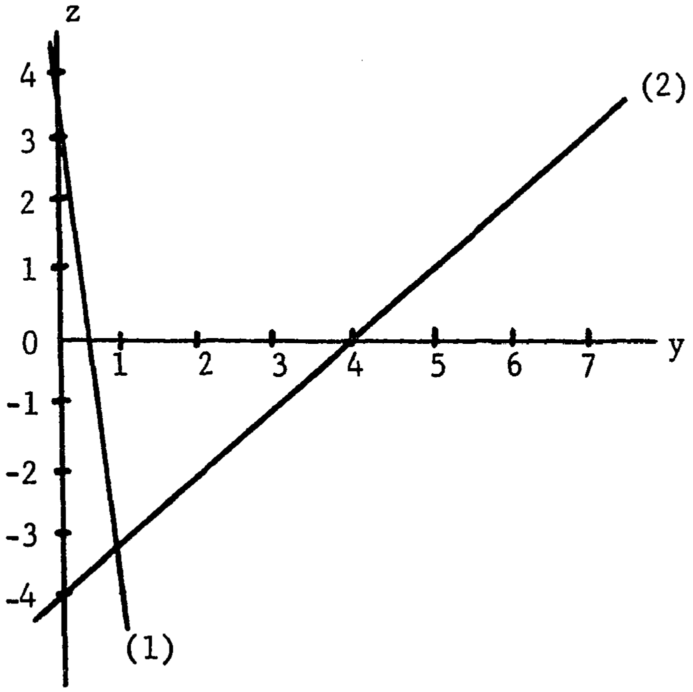
$y \geq 0 \quad y \equiv 0 \pmod 1$,

a solution of $y = \infty$, $z = -\infty$ is obtained. To (P10) a limiting constraint of $y \leq M$, where M is a very large number, is added. (P10) is then resolved for a solution of $y = M$, $z = - 7 M + 4$.

Now, find the most violated constraint at $y = M$ by a linear program. One can see that that constraint is (2). Adding constraint (2), one gets:

$$\begin{array}{lll} \text{Minimize:} & z & \\ \text{Subject to:} & z \geq -7y + 4 & (1) \\ & z \geq y - 4 & (2) \\ & y \geq 0 \quad y \equiv 0 \pmod{1}. & \end{array} \qquad \text{(P11)}$$

Graphically:

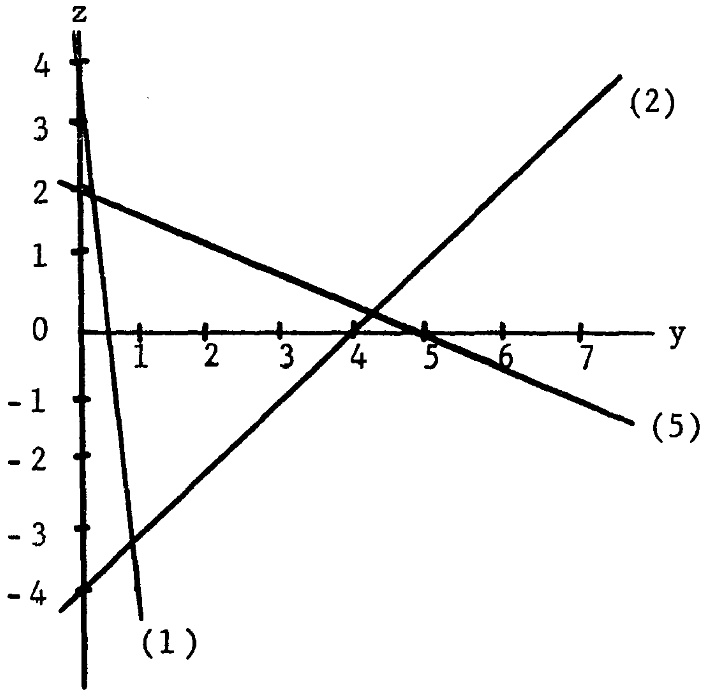


(D3)

Solving (P11), y = 1, z = -3. (5) will be indicated by the linear program to be the most violated constraint at y = 1. Now:

$$\begin{array}{lll} \text{Minimize:} & z & \\ \text{Subject to:} & z \geq -7y + 4 & (1) \\ & z \geq y - 4 & (2) \\ & z \geq -2/5\, y + 2 & (5) \\ & y \geq 0 \quad y \equiv 0 \pmod{1}. & \end{array} \qquad \text{(P12)}$$

Or:



(D4)

Solving (P12), y = 4, z = -2/5 and (4) will be the most violated constraint at y = 4. Next:

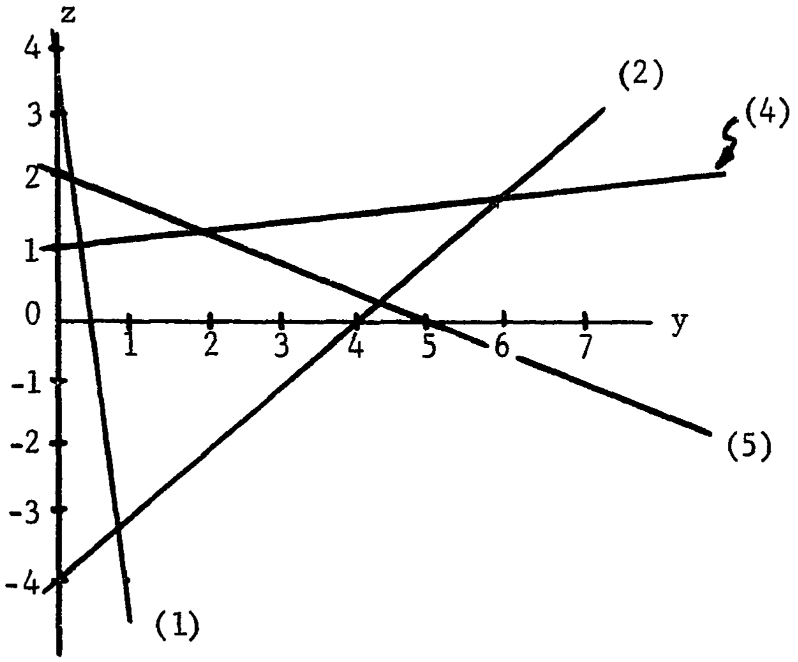
Minimize: z

Subject to:

$$\begin{aligned} z &\geq -7y + 4 && (1)\\ z &\geq y - 4 && (2)\\ z &\geq -2/5\, y + 2 && (5)\\ z &\geq 1/6\, y + 1 && (4) \end{aligned} \qquad (P13)$$

$$Y \geq 0 \quad Y \equiv 0 \pmod 1.$$

Or graphically:



(D5)

Finally, (P13) is solved for an answer of y = 2, z = 4/3 and the constraint indicated to be most violated at y = 2 is (4) once more. At this point the procedure will stop.

This, of course, is an extremely simplistic model, being only one-dimensional but lends insight into the actual behavior of Benders' algorithm.

INVESTIGATION OF GLOVER'S REDEFINITION

Glover (20) has restated Benders' algorithm in a way which lends some insights into and understanding of the basic procedure of the algorithm. Again, consider the problem:

$$\begin{aligned} &\text{Minimize:} && X_o = CX + C'Y \\ &\text{Subject to:} && AX + A'Y \geq B \qquad\qquad \text{(P2)} \\ & && X,Y \geq 0 \quad Y \equiv 0 \text{ (mod 1)}. \end{aligned}$$

The procedure as described by Glover is basically the same as that described by Benders except for the form of the integer program. Consider an integer program which has the form:

$$\begin{aligned} &\text{Minimize:} && Y_o = DY \\ &\text{Subject to:} && FY \leq G \qquad\qquad \text{(P14)} \\ & && Y \geq 0 \quad Y \equiv 0 \text{ (mod 1)}. \end{aligned}$$

(P14) is a general form of an integer program where the process is started with D, F and G such that the space defined by $FY \leq G$ and $Y \geq 0$ will contain at least one Y which is optimal for the original mixed integer program. An example of such a program might be:

$$\begin{aligned} &\text{Minimize:} && Y \\ &\text{Subject to:} && Y \leq M \\ & && Y \geq 0 \quad Y \equiv 0 \pmod 1). \end{aligned} \tag{P15}$$

where M is an extremely large number. As the procedure continues, additional constraints, which will be described later, will be added to this integer program.

Once additional constraints have been added to the integer program, certain rules are followed. If the integer program has no feasible solution, the optimal mixed integer solution is the best found thus far by the procedure. If there have been no mixed integer solutions found thus far, the mixed integer program is itself infeasible.

Suppose there exists a feasible solution to (P14), say Y^*. If Y^* is substituted into (P2) the result is a reduced linear program of:

$$\begin{aligned} &\text{Minimize:} && X_o = CX + C'Y^* \\ &\text{Subject to:} && AX \geq B - A'Y^* \\ & && X \geq 0. \end{aligned} \tag{P16}$$

There are two cases which can occur. Case No. 1: If (P16) has an unbounded solution, the original mixed integer problem is unbounded and has a solution of $X_o = -\infty$, $Y = Y^*$ and X equal to the values found when solving (P16).

If (P16) has no feasible solution, then Y^* is not a feasible value for Y for the mixed integer program. A new constraint (to be described later) for the integer program will then be generated and added to the integer program. The above procedure would then be iterated as many times as necessary.

Case No. 2: If a feasible finite solution to (P16) is found, then that solution, say $X = X^*$ and $Y = Y^*$, is a feasible solution to the original mixed integer problem. If this solution is better than any previously found, it is possible to update the best found thus far to this new solution and generate a new constraint for the integer problem. The process would be continued with iterations of this type until a solution was found.

Consider the method of determination of the constraint to be added to the integer problem and the objective function of that additionally constrained program. The final tableau resulting from the solving of (P16) appears thusly:

$$\begin{aligned} X_o &= a_{oo} + Q^o\ W \\ X &= A_o + Q\ W \end{aligned} \qquad \text{(T1)}$$

where W is the vector of final nonbasic variables.

Consider the representation of the final tableau if the integer variables Y had been included in (P16) while solving the linear program but were not allowed to enter the solution, i.e. the identical pivots were taken as produced (T1). Then one would have had:

$$\begin{aligned} X_o &= a_{oo} + Q^o W + P^o Y \\ X &= A_o + Q W + P Y \end{aligned} \qquad \text{(T2)}$$

First, in the case when an optimal solution to (P16) was found, namely $X_o = a_{oo}$ and $X = A_o$, let X_o' be the best solution found to date for the mixed integer problem. If the newly found solution is better than this best ($a_{oo} < X_o'$), one immediately sets the best found so far to the newly found solution, i.e. set $X_o' = a_{oo}$. It would be advantageous to add a constraint to the integer program so as to force the integer program from the Y^* previously found as one has determined the best possible solution with $Y = Y^*$ and now wishes to consider another value for Y.

At this point, either $X_o = X_o'$ (because X_o' was updated to X_o if $X_o = a_{oo} < X_o'$) or $X_o > X_o'$. If a constraint is added to the integer program such that $X_o < X_o'$, $X_o = a_{oo}$ could not possibly be a feasible value for X_o in any future iterations. This is due to the fact that a_{oo} is the optimal value for X_o when $Y = Y^*$ and $X_o < X_o'$ requires a better value for X_o. Coupling a requirement of $X_o < X_o'$ with (T2), one gets:

$$X_o = a_{oo} + P^o Y + Q^o W < X_o'$$

$$\text{or} \quad P^o Y < X_o' - a_{oo} - Q^o W. \qquad \text{(E2)}$$

Since the linear program at this point is optimal and it is a minimization problem, it is known that $Q^o \geq 0$ and $W \geq 0$, so that $- Q^o W \leq 0$. Thus:

$$P^o Y < X_o' - a_{oo} - Q^o W < X_o' - a_{oo}$$

$$\text{or} \quad P^o Y < X_o' - a_{oo}. \qquad \text{(E3)}$$

Since it is impossible to handle strict inequalities in integer programming, subtract a small positive number e from the right hand side of the inequality. Then:

$$P^o Y \leq X_o' - a_{oo} - e. \qquad \text{(E4)}$$

(E4) will be the new constraint to be added to the integer program and through its use, Y^* will not be derived as a solution to the integer subproblem in the future. Note that all constraints added to the integer program will depend parametrically upon X_o'.

In the second case, when the reduced linear program has no feasible solution, one must add a constraint to the integer program also. As the previously derived integer point Y^* was not a feasible one (as shown by the linear program's infeasibility), one would wish to exclude it from future consideration.

The infeasibility of the problem will have resulted in at least one row, say the r-th one, in the final tableau being such that:

$$X_r = a_{ro} + Q^r W + P^r Y \qquad \text{(E5)}$$

where $a_{ro} < 0$ and $Q^r \leq 0$ with W again the nonbasic variables in the final tableau. Considering (E5) and wishing to have feasibility ($X_r \geq 0$):

$$X_r = a_{ro} + Q^r W + P^r Y \geq 0$$

$$\text{or} \quad P^r Y \geq - Q^r W - a_{ro}. \qquad \text{(E6)}$$

Since $- Q^r W \geq 0$, one can say:

$$P^r Y \geq - Q^r W - a_{ro} \geq - a_{ro}$$

$$\text{or} \quad P^r Y \geq - a_{ro}. \qquad \text{(E7)}$$

(E7), then, is the constraint to be added to the integer program when (P16) has no feasible solution. Note that the previously found integer point Y^* cannot satisfy the requirement that $X_r \geq 0$ and therefore will not be found feasible in future iterations.

It was assumed that Y was bounded ($FY \leq G$) in the initial conditions and therefore there are a finite number of values for Y. Note that the constraints added in Glover's adaptation of Benders' algorithm completely excluded any integer points previously found. Benders' algorithm only changed the value associated with each integer point so that a previously found integer point could reenter as the optimal point, but at a higher objective function value.

Therefore, since in Glover's version of the algorithm adding constraints to the integer program results in the elimination of at least one integer point from consideration in each iteration, there can be only a finite number of iterations. Thus, as far as finite convergence of this procedure is concerned, thé objective function form is immaterial.

Glover considers interesting an objective function of:

$$Y_o = a_{oo} + P^o Y \qquad \text{(E8)}$$

whose coefficients are the same as those of the most recently adjoined constraint.

Consider (E4). By a simple manipulation, one can get:

$$X_o' - e \geq P^o Y + a_{oo}. \qquad \text{(E9)}$$

At the j-th iteration, one would have j of these constraints, thusly:

$$X_o' - e \geq P_i^o Y + a_{oo}^i \quad \text{for } i=1,\dots,j \qquad \text{(E10)}$$

Now, since X_o' - e is greater than or equal to all $P_i^o Y + a_{oo}^i$ for i=1,...,j, it can be seen that:

$$X_o' - e \geq \max_{i=1,\dots,j} (P_i^o Y + a_{oo}^i). \qquad \text{(E11)}$$

But $X_o^{'}$ - e merely indicates that it is desired to reduce the optimal value found thus far by amount e. Let e be considered a variable which one wishes to maximize, i.e. one wishes to reduce the best found solution as far as possible. This would have the effect of minimizing $X_o^{'}$ - e. Let $Z = X_o^{'} - e$. Then:

$$Z \geq \max_{i=1,\ldots,j} (P_i^o Y + a_{oo}^i) \tag{E12}$$

where one wishes to minimize Z.

It is known from the definition of Benders' algorithm that the optimal value of the mixed integer program at the i-th iteration is $C'Y + U_i^p(B - A'Y)$. From Glover's definition, the optimal value is $a_{oo}^i + P_i^o Y$. Since one is the dual of the other:

$$a_{oo}^i + P_i^o Y = C'Y + U_i^p(B - A'Y). \tag{E13}$$

If one substitutes (E13) into (E12):

$$Z \geq \max_{i=1,\ldots,j} (C'Y + U_i^p(B - A'Y)). \tag{E14}$$

Since one wishes Z minimized and (E14) to be in closed form, (E14) can be expanded to:

$$\begin{aligned} &\text{Minimize:} && Z \\ &\text{Subject to:} && Z \geq C'Y + U_i^p(B - A'Y) \qquad i=1,\ldots,j \\ & && Y \geq 0 \quad Y \equiv 0 \pmod 1 \end{aligned} \tag{P19}$$

which is merely Benders' representation for the integer problem at the j-th iteration.

SOME PARTITIONING ALGORITHMS FOR THE MIXED INTEGER PROGRAMMING PROBLEM

In his original work Benders made two primary contributions: (1) development of a "pure" integer problem (P8) which is equivalent to the original mixed integer problem (P2), and (2) development of an algorithm which will solve this resultant "pure" integer problem in a practical procedure. This algorithm involves solving two separate subproblems iteratively, one a relaxed integer problem and the other a linear programming problem involving the non-integral constrained variables of the original problem. Hence, the term "partitioning algorithm."

There have been some encouraging and somewhat surprising results reported by Geoffrion and Marsten (19), which show that Benders' algorithm can generally solve the mixed integer program in relatively few iterations. An investment planning type problem of 378 constraints, 1326 continuous variables and 24 integer variables was solved in 4 iterations. Two project evaluation problems, both involving

350 continuous variables, 27 binary variables and 275 constraints were solved in 10 and 25 iterations respectively. Unfortunately, if the original problem involves many integer variables, each iteration requires the solution of a "pure" integer problem for which the determination of the solution require an extreme amount of time.

The purpose of this research is to propose some alternative schemes for solving (P8) and then to test these schemes computationally. Of course, the way in which to improve upon Benders' algorithm is to either (1) reduce the number of iterations without increasing substantially the time per iteration (although, as mentioned, Benders' algorithm has required surprisingly few iterations for solution), (2) reduce the time per iteration without substantially increasing the number of iterations or (3) reduce both the number of iterations and the time per iteration.

In this chapter three algorithms are presented for solving (P8) which appear to be promising in terms of the saving of time and/or iterations in the solution of mixed integer problems.

(A) Use of Non-Optimal Integer Subproblem Solutions

The first algorithm aims at reducing the time per iteration by reducing the time involved in solving the integer subproblem. There are two primary difficulties in the solving of integer programming problems by any presently available methods. These are the finding of the optimal solution itself and the elusiveness of proof of optimality once the solution is found. Feasible answers to integer problems are, of course, considerably easier to find than the optimal and many times such feasible answers are quite close to the true optimum. Some procedures actually find the optimum fairly rapidly in some problems, but the procedure does not terminate with an indication that the point obtained is optimum for many more iterations. Thus, in integer programming, one is not only concerned with finding an optimum solution, but also recognizing this solution to be optimal as soon as possible.

It would therefore seem to be advantageous (in terms of time spent solving the integer subproblem in Benders' algorithm) to find only a "good solution" as this can generally be done quite rapidly. In fact, Aldrich (1) and Gorry, Shapiro and Wolsey (23) have suggested such a modification as possibly being advantageous. If this feasible point for the integer problem results in a new vertex of the linear programming subproblem (and thus a new constraint for the integer programming subproblem) being generated, one may have possibly gained by not spending large amounts of time in the integer subproblem proving optimality. Thus, what one would hope is that the feasible solutions to the integer subproblem would generate one of the necessary constraints for eventual solution or, failing that, that the number of extraneous constraints, i.e. constraints which are generated in addition to those actually necessary for solution of the mixed integer problem, would be small.

Let us examine some of the differences in procedure that are necessitated by the solution of the integer subproblem to a "good solution" instead of an optimal solution. From Appendix A, (E16) shows that in Benders' algorithm $Z^t \leq Z^*$, where Z^t is the solution to the relaxed integer problem at iteration t and Z^* is the solution to the original mixed integer problem. Since we have not solved the relaxed integer problem completely, this inequality does not necessarily hold. Thus, it is possible that the solution of the relaxed integer problem to a feasible point only will result in $Z^t > Z^*$, i.e. Z^t is a feasible, non-optimal point of the original mixed integer problem space. If this occurs, it follows that $U_o^t = Z^t - C'Y^t$ where U_o^t is the objective function value obtained upon solution of the linear program with $Y = Y^t$. Under normal conditions, i.e. the solution of the relaxed integer program to a true optimum, this equality would indicate final solution of the original mixed integer problem, whereas with the solving of the relaxed integer subproblem to only a feasible solution, it does not necessarily indicate the finding of the final solution. Thus, the stopping rule for Benders' algorithm must be changed so as to have some provision to detect the situation where $U_o^t = Z^t - C'Y^t$ does truly indicate a final solution.

The detection of the situation in which $U_o^t = Z^t - C'Y^t$ does not indicate a final solution consisted of two parts. First, the value of the best (smallest) U_o, call it U_o^*, found thus far, along with associated variable values, was saved and if $Z^t > U_o^*$, then Y^t is obviously an undesirable value for Y, i.e. one which will produce the above mentioned erroneous optimality test. Second, if $Z^t \leq U_o^*$ and $Z^t \geq Z^*$, then as stated above, it follows that $U_o^t = Z^t - C'Y^t$ when the linear program is solved. If either of these two possible error conditions ($Z^t > U_o^*$ or $U_o^t = Z^t - C'Y^t$) occurs, the solution of the previous integer subproblem is continued from the point at which it was left off until a new feasible point is found. This procedure continues until either

$Z^t \leq U_o^*$ and $Z^t \leq Z^*$ or until the true optimum for the previous integer subproblem is found. If the first of these situations occurs, a constraint is generated for the integer subproblem as usual. If the second occurs, we have the final solution to the original mixed integer problem.

A detailed statement of the algorithm suggested follows.

<u>Step 0</u>: Same as in Benders' algorithm.

<u>Step 1</u>: Solve the "pure" integer program:

Minimize: Z

Subject to: $Z \geq C'Y + U^i(B - A'Y) \quad i = 1,\ldots,t$

$Y \geq 0 \qquad Y \equiv 0 \pmod 1$.

until a feasible solution is found for which $Z^t \leq U_o^*$. Let Z^t and Y^t be this solution.

<u>Step 2</u>: Solve the linear program:

Maximize: $U_o = U(B - A'Y^t)$

Subject to: $UA \leq C$

$U \geq 0$.

Let U_o^{t+1} and U^{t+1} be the solution. If $Z^t - C'Y = U^{t+1}(B - A'Y^t)$, there is a possibility that one has found the solution to the original mixed integer problem. It is necessary to determine whether this test truly indicates global optimality. If the problem in step 1 was not solved to optimality, return to step 1 and continue solution to the next discovered feasible solution, or until the present solution to the integer subproblem is proven to be optimal. If the problem in step 1 was solved to optimality and that optimality was recognized as such, go to step 3.

If $Z^t - C'Y^t \ngeq U^{t+1}(B - A'Y^t)$, add the constraint:

$$Z \geq C'Y + U^{t+1}(B - A'Y)$$

to the integer subproblem exactly as in Benders' original algorithm. Let t = t + 1, return to step 1.

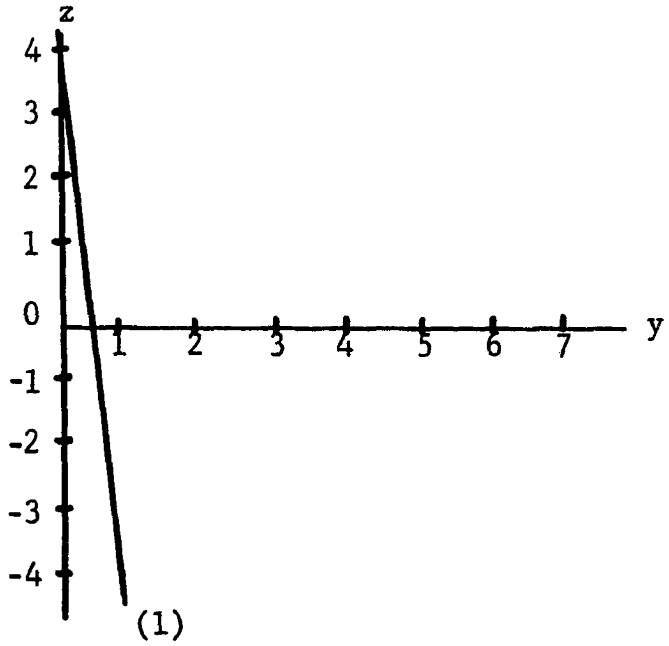
Step 3: Same as Benders' algorithm.

In step 1 of the procedure outlined above, solution of the "pure" integer problem was always to the next discovered feasible point. Since most algorithms "discover" feasible points in such a way that each new feasible point is better (in terms of the objective function value generated) than the last, one will eventually be lead to the true optimum if one returns to step 1 often enough in a single iteration. Of course, if the optimal point is found in step 1, then the iterations are the same as Benders' original algorithm.

In order to better visualize some of the problems and advantages involved in this algorithm, let us consider the problem presented in (D1), where Z^* was 4/3. After the first iteration we had:

$$\begin{array}{lll} \text{Minimize:} & Z & \\ \text{Subject to:} & Z \geq -7y + 4 \quad (1) & \text{(P17)} \\ & Y \geq 0 \quad Y \equiv 0 \pmod 1. & \end{array}$$

or:



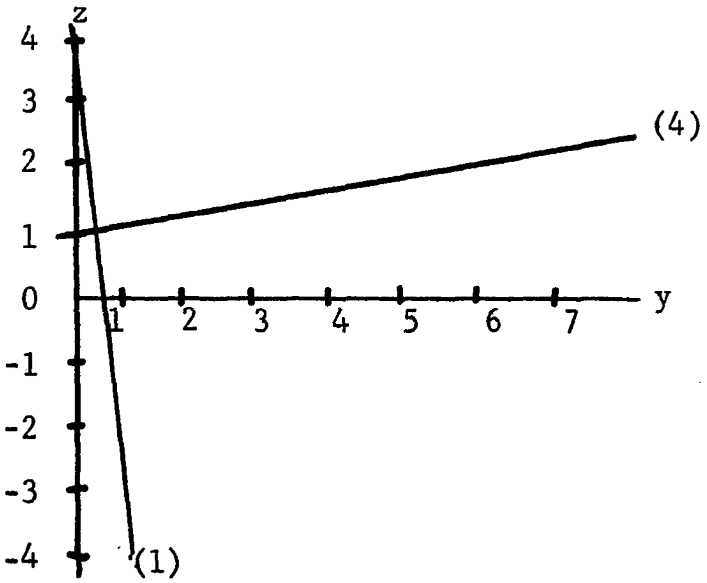
(D6)

The optimal solution, of course, is $y = \infty$, $Z = -\infty$ (computationally, we set y to some very large number), and the most violated constraint at that point was (2) which is $Z \geq y - 4$.

Suppose that (P17) was not solved exactly so that only a feasible solution was found, say $y = 3$. At $y = 3$, the most violated constraint is (4) or $Z \geq 1/6\ y + 1$. At this point ($y = 3$), $Z = -17$ and $Z < Z^*$ so the procedure continues with the problem:

$$\begin{aligned}
&\text{Minimize:} && Z \\
&\text{Subject to:} && Z \geq -7\ y + 4 && (1) \\
& && Z \geq 1/6\ y + 1 && (4) && \text{(P18)} \\
& && y \geq 0 \quad y \equiv 0 \pmod{1}.
\end{aligned}$$

Graphically we have:



(D7)

Now, the solution to (P18) is $y = 1$ and $Z = 7/6$ and, of course, (5) would be added to the integer subproblem as the most violated constraint. By solving for only a feasible point, suppose one got $y = 4$ for a solution. Then $Z = 3/2$ and since $Z > Z^*$, one would get a false test for optimality using Benders' original stopping rule. As can be seen, this situation arises due to the fact that the most violated constraint at $y = 4$ is again (4), a constraint already generated. Therefore, return should be made to the solution of the relaxed integer subproblem and the solution continued until another feasible solution is found. This procedure is continued until a point is found at which the most violated constraint is not (4) or any of the other previously generated constraints or until the true optimal to the relaxed integer problem is found and proven optimal.

In our example, suppose we ended up solving the problem to an optimum. Then at $y = 1$, the most violated constraint is (5). Adding this constraint will eventually force the solution of $y = 2$ and $Z = 4/3$ in the next iteration.

The proof of finiteness of the suggested modification to Benders' algorithm is as follows: If at some iteration $Z^t \leq Z^*$, where Z^* is the optimal value for the original mixed integer problem, then the proof is the same as for Benders' original algorithm. If $Z^t > Z^*$, then one would find that:

$$U^{t+1}(B - A'Y^t) = Z^t - C'Y^t.$$

In that case, return is made to the integer subproblem and another feasible point is found. Since these additional feasible points found for the integer subproblem are such that each is smaller than the preceding one, eventually either $Z^t \leq Z^*$ for some non-optimal point of the integer problem solution space or the optimal point for the integer subproblem is found and thus $Z^t \leq Z^*$.

Thus, the only time that the procedure continues is when $Z^t \leq Z^*$ and thus a new constraint will be generated each time or the solution of the mixed integer problem will be found.

(B) Generating Multiple Constraints per Iteration

A second tested alteration of Benders' original algorithm was to introduce more than one new constraint at each iteration as suggested by Geoffrion and Marsten (17) as possibly being advantageous. Since a large portion of this algorithm's time is generally spent solving integer programs, it would be advantageous to try to cut the iterations to a minimum. Since a certain number of constraints are required to be generated before convergence of the procedure, the faster one can generate those constraints the better; that is, if not too many unnecessary constraints are generated.

The generation of multiple constraints would be accomplished by using not only the U^p which maximized the linear program but other vertices as well. Several alternative methods could be used to derive these additional U^p's.

Due to the fact that during research the primal problem (P3) was solved rather than the dual problem (P4), several correspondences between the primal and dual problems had to be examined to determine exactly how to derive these other U^p's. Since it was desired to find feasible, non-optimal vertices of the dual problem, it was necessary to discover non-feasible, optimal vertices of the primal. With the current tableau reflecting the optimal solution, the following procedure was used to derive a non-feasible, optimal vertex in the primal problem.

First, since the dual variable values are found in the objective function row of the primal tableau under the original basic variables, it was necessary to keep all these values greater than or equal to zero. Also, since the value of the slack variables for the dual problem occur under the beginning non-basic variables, these values also have to be kept greater than or equal to zero.

In order to insure a change in the value of the dual variables, it was decided that the entering primal variable must be one of the original basic variables and must have a strictly positive value in the objective function row. It follows that this guarantees change in the dual variable values as the value in the objective function row under the selected entering variable must change from a positive value to zero.

Any variable having these two properties was tested for the possibility of its being able to enter the solution. Two other conditions also had to be satisfied for the variable to enter the solution. First, the entry of the variable had to make the primal problem non-feasible. This, of course, means that the right-hand side had to become negative for some given row, call it the infeasibility row, and for this to happen the value under the entering variable and in the given infeasibility row had to be negative. Second, the entry of the variable had to leave the solution optimal, i.e. all objective function row values had to stay positive or zero. Thus, all columns having negative coefficients in the infeasibility row were examined and the column having the smallest absolute ratio of objective function row value to infeasibility row value was selected to be the entering variable.

The first variable found satisfying all four of these conditions was entered into the solution and the resulting, changed values for the dual variables used to generate a second constraint at each iteration.

If no variable was found to satisfy all of these conditions, only one constraint was generated in that iteration.

This procedure, as mentioned above, can cut the number of necessary iterations of the algorithm and thus cut the overall time for solution. A detailed explanation follows:

<u>Step 0</u>: Same as Benders' algorithm.

<u>Step 1</u>: Same as Benders' algorithm.

<u>Step 2</u>: Solve the linear program:

$$\begin{aligned} \text{Maximize:} \quad & U_o = U(B - A'Y^t) \\ \text{Subject to:} \quad & UA \leq C \\ & U \geq 0. \end{aligned}$$

Let U_o^{t+1} and U^{t+1} be the solution. Determine another feasible but non-optimal point, say $\bar{U}_o^{t+1}$ and $\bar{U}^{t+1}$, as discussed above. If:

$$Z^t - C'Y^t = U^{t+1}(B - A'Y^t)$$

go to step 3. If:

$$Z^t - C'Y^t < U^{t+1}(B - A'Y^t),$$

add to the integer subproblem the two constraints:

$$Z > C'Y + U^{t+1}(B - A'Y) \quad \text{and}$$
$$Z > C'Y + \bar{U}^{t+1}(B - A'Y).$$

Let t = t + 1. Go to step 1.

Step 3: Same as Benders' algorithm.

For the purposes of this research only two constraints at a time were added to the integer subproblem. Obviously, any reasonable number of constraints could be added to the subproblem at each iteration, each being generated from a different discovered U^p.

No checks were made for duplication of constraints in different iterations as it was deemed to occur so infrequently that the additional calculations involved in doing such checking could not be justified.

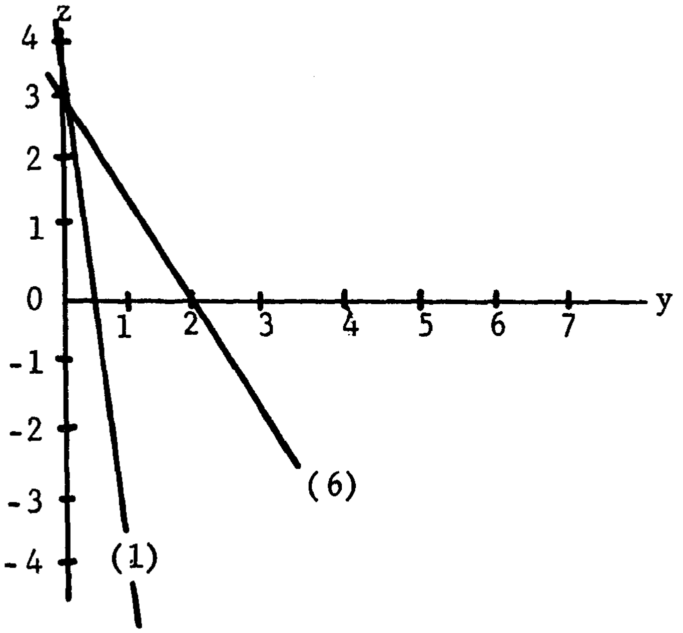
Since this additional constraint is merely an attempt to cut the number of iterations, it can be seen that Benders' original algorithm is still intact within the procedure. And thus, the proof of finiteness of the algorithm must be exactly the same.

To examine this suggested modification more closely, let us look at the example previously examined, namely (D1). Let us assume that at each iteration we not only detect the most violated constraint of the integer problem but also the second most violated.

Thus, at the first iteration, when $y = 0$, the linear subprogram not only generates constraint (1) but also (6). Then, our first integer subprogram to be solved is:

$$\begin{array}{lll} \text{Minimize:} & z & \\ \text{Subject to:} & z \geq -7\,y + 4 & (1) \\ & z \geq -3/2\,y + 3 & (6) \qquad (P19) \\ & y \geq 0 \quad y \equiv 0 \pmod{1}. & \end{array}$$

The graph of this would be:



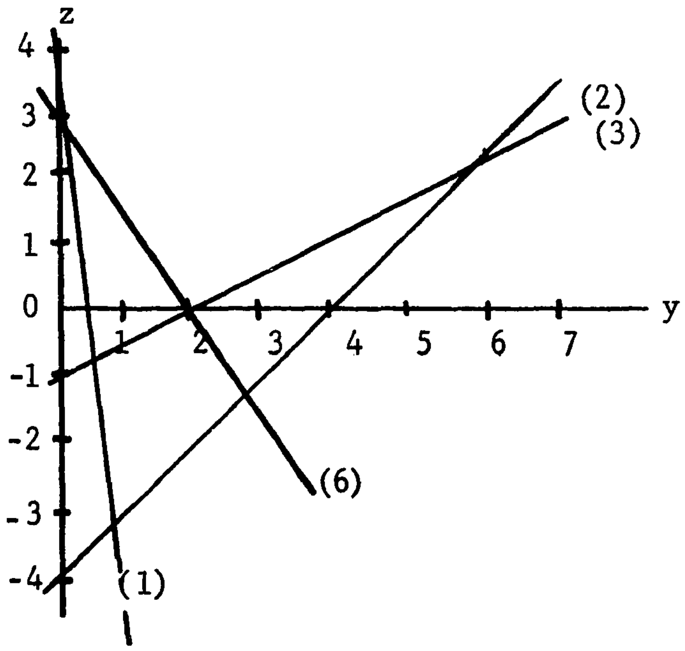
(D8)

The solution to (P19) is, of course, $z = -\infty$ and $y = \infty$, but with the addition of a bounding constraint of $y \leq M$, the solution becomes $z = -3/2\ M + 3$ and $y = M$, where M is a very large number.

At $y = M$, we find the two most violated constraints to be (2) and (3) so both of these are added to the integer subprogram to produce:

$$\begin{array}{lllr} \text{Minimize:} & z & & \\ \text{Subject to:} & z \geq -7\ y + 4 & (1) & \\ & z \geq -3/2\ y + 3 & (6) & \\ & z \geq y - 4 & (2) & \text{(P20)} \\ & z \geq 2/5\ y - 1 & (3) & \\ & y \geq 0 \quad y \equiv 0\ (\text{mod } 1). & & \end{array}$$

Or, graphically:



(D9)

Here the solution is $y = 2$ and $z = 0$. Notice that we have discovered the correct value for y but the value for z is incorrect since an essential constraint, (4), is missing. Thus, Benders' stopping rule will not come into play and the procedure will continue.

At $y = 2$, the most violated constraints are (4) and (5). Adding these two constraints, one gets:

$$\begin{array}{lll} \text{Minimize:} & z & \\ \text{Subject to:} & z \geq -7\,y + 4 & (1) \\ & z \geq -3/2\,y + 3 & (6) \\ & z \geq y - 4 & (2) \\ & z \geq 2/5\,y - 1 & (3) \qquad \text{(P21)} \\ & z \geq 1/6\,y + 1 & (4) \\ & z \geq -2/5\,y + 2 & (5) \\ & y \geq 0 \quad y \equiv 0 \pmod 1. & \end{array}$$

Since (P21) and (P9) are identical, i.e. in this case we have generated the complete integer equivalent to the original mixed integer problem, we know that the answer of $y = 2$ and $z = 4/3$ will be found at this iteration.

Considering the results of this particular example, some points need to be made. In this case, the entire integer problem was generated. This would seem to be due to the small size of the example. Although it is theoretically possible that extra constraints added at each iteration might result in extra iterations, intuition and all of the test problems discussed in this dissertation indicate that it will probably seldom happen. In the main then, one can make the assumption that under this modification the number of iterations will be less than or equal to the number of iterations using Benders' original algorithm. If one allows this assumption, then by using this particular alteration of Benders' algorithm the maximum number of constraints which can be generated is twice the number generated using Benders' original algorithm. Thus, on large problems only a few of the N possible constraints would be generated.

We find that a factor which becomes important in evaluation of the efficiency of the modification is the fact that since at each iteration two constraints are added rather than one, the problem is more restricted, i.e. there are fewer feasible answers, at each iteration, than in Benders' original algorithm. Since the integer subproblem algorithm examines different feasible points, reducing by just a few the number of feasible points can reduce significantly the time of execution of the integer subproblem. Also, of course, with a decreased number of iterations there is a decrease in the number of linear programs solved.

(C) Solving of the Integer Program as a Linear Program

Since the solving of linear programs is generally so much more rapid than the solving of integer programs of the same size and since the whole purpose of the solving of the integer program is to determine a Y^t which in turn produces a new U^p, any method for determining a value for Y which will give a new U^p should be of interest. If the procedure advanced by Benders were modified to solve the integer program as if it were a linear program for some fixed number (k) of iterations before beginning to solve the subproblem as a "pure" integer one, it would seem that the amount of calculations should be cut significantly. Also, if a point is reached where no more new U^p's are generated, a switch must be made to solve the subproblem as an integer problem. Finally, if the objective function value for the integer subproblem (solved linearly, of course) is within a given amount of the objective function value of the linear subproblem, the integer subproblem should be solved as an integer program.

This procedure must be a finite one as, at the worst, after k iterations, return is made to Benders' original algorithm which must generate the necessary constraints eventually.

A more detailed expression of this suggested modification follows.

Step 0: Same as Benders' algorithm.

Step 1: If $t > k$, go to step 2A. If $t \leq k$, solve the linear program:

Minimize: Z

Subject to: $Z \geq C'Y + U^{i}(B - A'Y) \quad i=1,\ldots,t$

$Y \geq 0.$

Let Y^{t} and Z^{t} be the solution.

Step 2: Solve the linear program:

Maximize: $U_{o} = U(B - A'Y^{t})$

Subject to: $UA \leq C$

$U \geq 0.$

Let U^{t+1} and U_{o}^{t+1} be the solution. If $U_{o}^{t+1} - (Z^{t} - C'Y^{t}) \leq \varepsilon$, and Step 2 was entered via Step 1, set $k = 0$ and go to Step 2A. If:

$$Z^{t} - C'Y^{t} < U^{t+1}(B - A'Y^{t}),$$

add the constraint:

$$Z \geq C'Y + U^{t+1}(B - A'Y)$$

to the subproblem of Step 1. Let $t = t + 1$. Go to Step 1. If:

$$Z^t - C'Y = U^{t+1}(B - A'Y^t),$$

and Step 2 was entered via Step 1, set $k = 0$ and go to Step 2A. If Step 2 was entered from Step 2A go to Step 3.

Step 2A: Solve the integer program:

Minimize: Z

Subject to: $Z \geq C'Y + U^i(B - A'Y) \quad i=1,\ldots,t$

$Y \geq 0 \quad Y \equiv 0 \pmod 1$.

Let Y^t and Z^t be the solution. Go to Step 2.

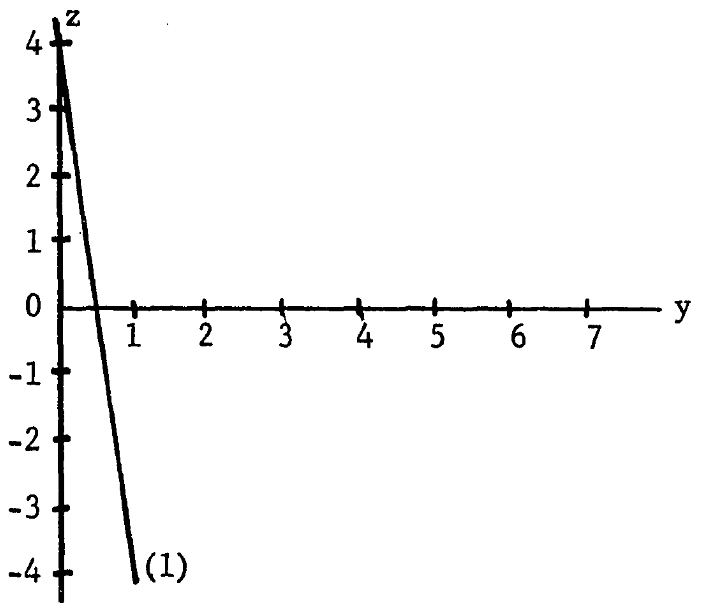
Step 3: Same as Benders' algorithm.

Graphically, one can see the tremendous advantage of the modification. This advantage is due, of course, to the much faster execution times in solving linear programs rather than integer programs. Consider the problem of (D1).

Let $y = 0$, then the most violated constraint is, of course, (1). Then our problem is:

$$\begin{aligned} &\text{Minimize:} \quad z \\ &\text{Subject to:} \quad z \geq -7\,y + 4 \qquad (1) \qquad\qquad \text{(P22)} \\ &\qquad\qquad\qquad y \geq 0. \end{aligned}$$

Or:



(D10)

The solution is $z = -\infty$, $y = \infty$ so that a limiting function ($y \leq M$) is added and the answer becomes $y = M$, $z = -7\ M + 4$.

Of course, the most violated constraint is (2), which is added to the problem so that we have:

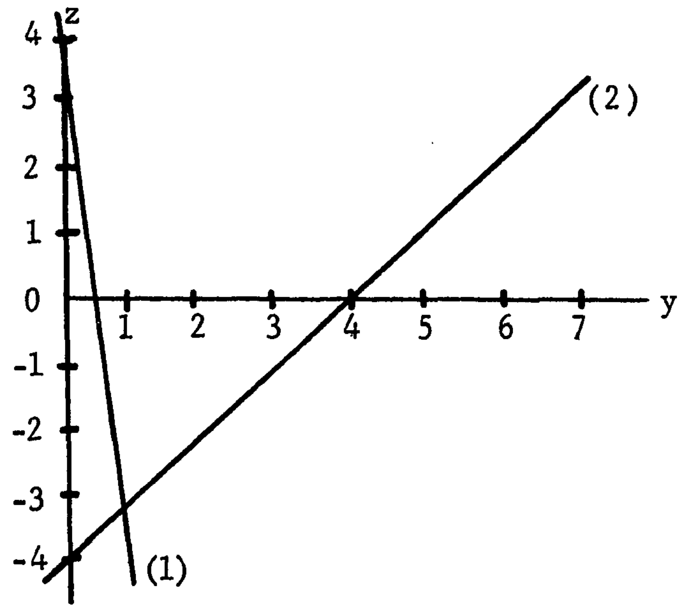
Minimize: z

Subject to: $z \geq -7\ y + 4$ (1)

$z \geq y - 4$ (2) (P23)

$y \geq 0$.

Or:



(D11)

The solution to (P23) is y = 1, z = -3. Now (5) is the most violated constraint. Then our problem becomes:

Minimize: z

Subject to:

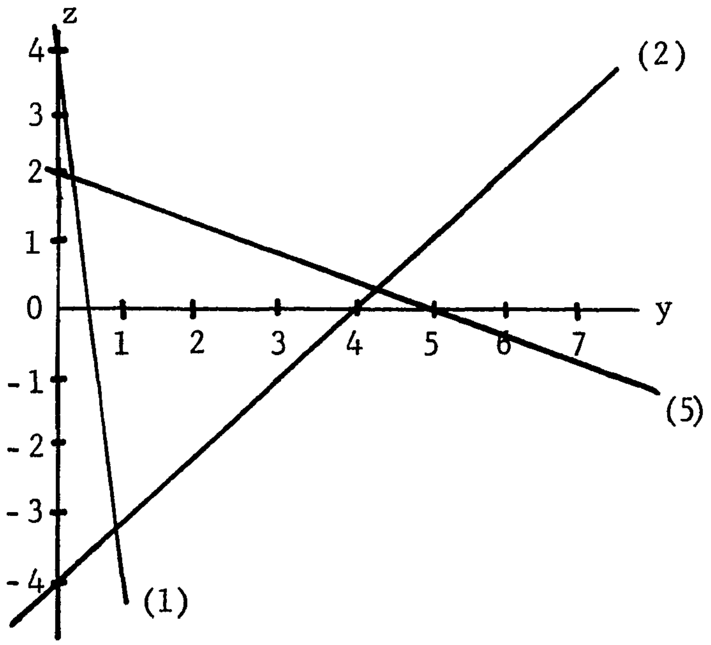
$$z \geq -7\ y + 4 \qquad (1)$$

$$z \geq y - 4 \qquad (2) \qquad \text{(P24)}$$

$$z \geq -2/5\ y + 2 \qquad (5)$$

$$y \geq 0.$$

Or graphically:



(D12)

The solution to (P24) is $y = 4\ 2/7$, $z = 2/7$. At $y = 4\ 2/7$, the most violated constraint is (4). Adding this, we find:

Minimize: z

Subject to: $z \geq -7\ y \pm 4$ (1)

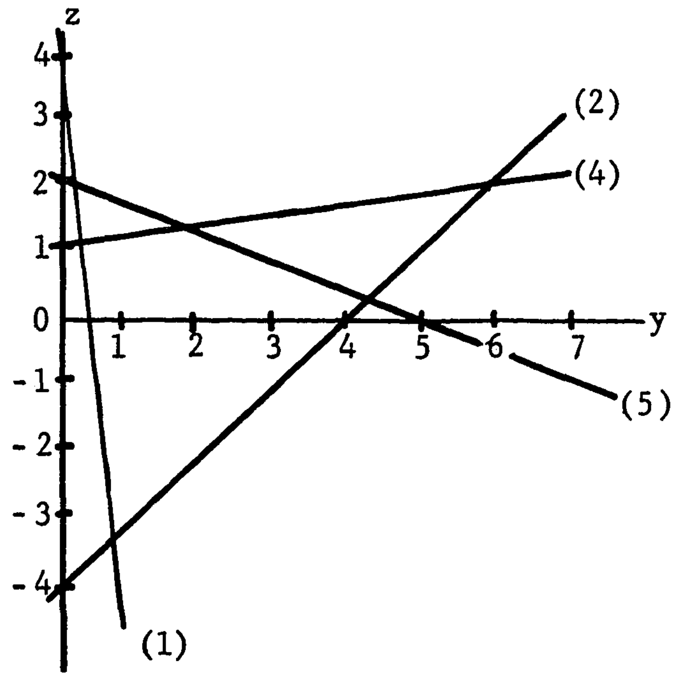
$z \geq y - 4$ (2)

$z \geq -2/5\ y + 2$ (5) (P25)

$z \geq 1/6\ y + 1$ (4)

$y \geq 0.$

Graphically:



(D13)

The solution to (P25) is y = 1 13/17 and z = 1 5/7. The most violated constraint is (4) again. Thus, we solve (P25) again, this time with an integer restriction on y. This will give our true answer of y = 2 and z = 4/3.

In this example, we solved four linear programs and one integer program whereas by Benders' algorithm we solved four integer programs. It is obvious that, due to the fact that on large problems integer solutions are much more time consuming to find than linear program solutions, this modification could possibly have a significant computational advantage over Benders' original algorithm.

GENERAL RESEARCH PROCEDURES AND RESULTS

Research Procedures:

The research done for this dissertation was limited to mixed integer programming problems with 0-1 variables since this type of problem is very common. The test problems (see Appendix D) were mostly manufactured ones and came from common, realistic models of different real-life situations. In addition, there were a few problems taken directly from the literature.

The linear subproblem was solved by means of a regular simplex tableau algorithm since time constraints and the necessity to change considerably this section of the program to effect the desired modifications precluded use of more intricate techniques. This procedure was quite adequate for the size problems considered in this research. Since the chief purpose of the running of test problems was a comparison of the different modifications to Benders' original algorithm, the efficiency of the subprograms was not of great importance. Indeed, the larger execution times allowed for better timings, reducing the percentage error from run to run.

Also, the linear subproblem was solved from a primal rather than a dual model as this results in more understandable intermediate results and eliminates the necessity for Step 3 of the algorithm. It also, unfortunately, results in the necessity for extra explanation and examination in one of the suggested modifications. The values of the U^p were, of course, found in the objective function row of the simplex tableau under the variables forming the original basis. The possibility of unboundedness in the dual was handled by the addition of another variable in the primal model with a large objective function coefficient. This variable was not allowed to enter the solution unless the primal proved infeasible.

The integer subproblem was handled by a modified Balas-type algorithm (see Appendix E). This modified algorithm seemed to be extremely fast in some problems and created some trouble in accurate timing comparisons in smaller problems. Also, as in most integer programming algorithms, problems of a particular structure resulted in extreme inefficiency in the algorithm and a correspondingly large increase in execution time can be noted. The modification of Balas' 0-1 algorithm was almost a complete disposal of his decision rules and a return to the basic concepts upon which that algorithm was based. This was necessitated by the fact that the value of the objective function in the integer subproblem is found directly from the constraints and also because the integer subproblem was not a truly pure one.

No provisions or tests were provided to detect constraints in the integer subproblem which became redundant during the procedure as this would have been of small consequence in the given test problems compared to the amount of time spent performing these tests.

Some difficulties were encountered due to the problem of real-number round-off in the computer so that all tests for equality are on a "small epsilon" basis, i.e. two numbers closer to one another than a given small value were considered to be the same number. Some possible ways to alleviate this problem would be double-precision numbers, holding all numbers as fractions or a modification of the simplex method. Thus, in some very selected cases, the program may fail, indicating a solution which is not the true solution. These cases, however, should be extremely rare.

Finally, no attempt was made to match or excel commercial mixed integer codes due to the limited scope of the study.

The problems considered in the testing of the suggested modifications were of various sizes as indicated by the table below:

Problem	Linear Variables	Integer Variables	Constraints
1a - 1f	12	15	12
2a - 2f	12	15	14
3a - 3f	36	24	14
4a - 4f	36	24	38
5a - 5f	12	25	12
6a	2	2	3
6b	16	4	20
6c	25	25	35
6d - 6e	24	24	32

Results of Testing of Modification A

Looking at the timing and iteration data in Appendix C, some points become obvious. First, in almost all cases where a solution was found, Benders' original algorithm and modification A ran for the same number of iterations. In fact, in a large portion of the iterations, it was necessary for the actual optimum to the integer subproblem to be found and proven to be optimal in modification A. Only in problems 2e - 2f, 4a - 4f and 6c - 6e was this not true. In that data set, quite often the integer subproblem was only partially solved before generation of a new constraint was possible. Thus, in general, the difference in timings lies in those few iterations in which the integer subproblem did not have to be solved completely in modification A.

In those limited cases where modification A did not have to solve the integer problem completely in many of the iterations, two possible occurrences are illustrated. In 2e, 2f, 4c and 6e, one can see that the solution of the integer subproblem to a feasible answer only has resulted in extra iterations, i.e. "extraneous" constraints have been generated. In 4a, 4b, 4e and 4f, 6c and 6d, the true relationship between Benders' original algorithm and modification A is clouded by the failure of Benders' algorithm to find a solution after almost five minutes of computer execution time. However, since modification A has found answers in reasonable amounts of time (with the exception of 6c), one can conclude that the solution of the integer subproblem to a suboptimal point proved extremely valuable as a modification.

Thus, in evaluation of modification A, it can be seen that to a large extent, the amount of efficiency involved is dependent upon problem structure. In general, on "small" problems, i.e. those requiring very little time for solution, the advantage of modification A is marginal at best. On larger problems, however, the efficiency involved in not solving the integer subproblem completely is extremely obvious in most problems. Modification A must be recommended in problems in which Benders' original algorithm fails due to excessive amounts of time being spent solving the integer subproblem.

Results of Testing of Modification B

As should be evident upon examination, modification B, in which two constraints are added each iteration, relies upon the reduction of the number of iterations to reduce time of execution. In addition, there can be small savings in execution time in some problems due to the fact that integer subproblems with more constraints are more tightly bound than those with fewer constraints. The question is, can the advantage due to reduced iterations and the time savings due to the reduction of feasible answers with extra constraints offset the increase in time of execution due to the larger size of the subproblems. In general, it was found that this occurred with very few exceptions; the most notable of which are, of course, problems 4c and 6e.

Once again, it is extremely hard to try to compare

Benders' original algorithm with modification B in several instances as the procedure did not go the completion. In the light of all the other examples, however, it would seem that modification B offers advantages in time of execution in the vast majority of the cases. Unfortunately, like Benders' original algorithm, modification B relies upon complete solution of the integer subproblem at each iteration and there are some problems whose particular structure causes extreme amounts of execution time to be necessary to solve certain of the integer programs.

Results of Testing of Modification C

Modification C relies chiefly upon the reduction of time per iteration to effect a reduction of overall execution time. Indeed, in almost all cases (and especially those in which a larger number of iterations are involved) modification C takes more iterations than Benders' original algorithm and thus relies strictly upon reduction of time per iteration.

This reduction of time per iteration occurs, of course, due to the fact that linear programs are executed instead of integer programs for the vast majority of iterations in each problem. As a matter of fact, every test problem completely solved resulted in only one integer subproblem being solved--a tremendous savings in time. Since the time of execution of an integer program generally goes up at a much faster rate dependent upon the size of the problem than does the time of execution of a linear program of the same size, the advantage inherent in the direct substitution of linear for integer programs is better illustrated in large problems. The addition of extra constraints over what are added in Benders' original algorithm will inevitably lead to larger times of execution on small problems. This is due to the fact that on small problems the time of execution of a problem as a linear program is larger than as an integer program.

The one extreme example of modification C's failure to far outdo Benders' original algorithm in a large problem was in the problem 6e. Upon close examination of the intermediate solution values for the integer subproblem and the constraints generated, it was found that the difficulty arose due to extreme degeneracy or near degeneracy about several of the intermediate solution points.

Thus, when the integer subproblem was solved as a linear program, there were an excessive number of iterations necessary in which the constraints generated were of little use in the advancement of the solution. In most cases the solution value for the integer subproblem changed only a small amount in many iterations (1/2 of one percent change in 8 iterations in one case). It was found, also, that the integer solution to the integer subproblem was far enough from the linear solution to avoid this problem.

Thus, on small problems, the modification is at best marginal. On large problems, however, the modification almost always proves to be the one which most reduced the time of execution from Benders' original algorithm (see 4a - 4d and 5a - 5c). Any problems in which large integer subproblems are expected should almost certainly be executed using modification C.

Overall Results

No attempt was made to study combinations, i.e. A and B or B and C, of modifications due to the desire to study the independent effects of the three modifications. Since the research has proven that modification B is questionable in advantage (due to its problem structure dependency for efficiency) over Benders' original algorithm, it would seem that combinations of modifications would also be questionable in advantage.

The inability of Benders' original algorithm and modification B to find solutions in certain of the problems was the result of a particularly structured integer subproblem. As in most integer programs, there are certain structured problems which create a situation in which excessive time is spent solving the problem. In those cases where an answer was not obtained in almost five minutes, the objective function value for a large percentage of the cases was very close to the true optimum.

It would appear that the two impressive modifications, A and C, can have great advantage over Benders' original algorithm in large problems--the larger the more advantage. In the cases where large integer subproblems are expected, one of the modifications (and preferably C) probably should be used.

APPENDIX A

In this appendix, it will be shown that (a) Benders' algorithm is finite, (b) it gives the optimum solution and (c) at any time in the process upper and lower bounds to the true optimum Z^* can be found.

If the set $UA \leq C$ is bounded, there are a finite number of vertices to the convex polytope of the dual space. If it is not bounded, the set $UA \leq C$, $\Sigma\, U_i \leq M$ will be bounded and will have a finite number of vertices. Therefore, if a different vertex U^p is generated by the linear program at each iteration, the algorithm must be finite. It would seem possible, however, that two non-optimal values of Y could bring about the same vertex U^p being selected since the value of Y affects only the objective function of the linear program and not the space itself. Fortunately, this cannot happen.

For proof, let the solution to Step 1 of the algorithm at iteration t be Z^t and Y^t, i.e.

$$Z^t = C'Y^t + U^r(B - A'Y^t)$$

$$\text{or} \quad Z^t - C'Y^t = U^r(B - A'Y^t) \qquad \text{(E15)}$$

where r represents one of the tight constraints at the t-th iteration. Since Z^t was obtained with a subset of the total number of constraints for the integer program, one knows:

$$Z^t \leq Z^* \qquad \text{(E16)}$$

where Z^* is, of course, the true optimal value for the mixed integer program.

From duality theory, one knows that Max $U(B - A'Y^t) =$ Min CX where U is constrained by $UA \leq C$ and X is constrained by $AX \geq B - A'Y^t$. Let the solution to Step 2 of the algorithm be U^{t+1}. Then:

$$U^{t+1}(B - A'Y^t) = CX^t. \qquad \text{(E17)}$$

It can be seen that X^t and Y^t provide a feasible solution to the mixed integer problem since:

$$AX^t + A'Y^t \geq B. \qquad \text{(E18)}$$

It is evident that:

$$CX^t + C'Y^t \geq Z^*$$

$$\text{or} \quad CX^t \geq Z^* - C'Y^t. \qquad \text{(E19)}$$

From (E15), (E16), (E17) and (E19):

$$U^{t+1}(B - A'Y^t) = CX^t \geq Z^* - C'Y^t \geq Z^t - C'Y^t = U^t(B - A'Y^t) \qquad \text{(E20)}$$

where the equality holds only if $Z^t = Z^*$. If:

$$U^{t+1}(B - A'Y^t) > U^t(B - A'Y^t) \qquad \text{(E21)}$$

then $U^{t+1} \neq U^t$, and a new vertex will be obtained on each iteration of Step 2 or else the optimum vertex will be obtained.

There are, therefore, two possibilities. Either the optimum will be found sometime during the execution of the algorithm or iterations will continue until all vertices are considered. In the former case, of course, the optimum is found. In the latter, one has generated the complete integer program and the solution to that program is by definition the solution which produces the optimum value of the objective function of the original mixed integer problem.

Note that for any modification which is considered to Benders' algorithm, (E21) must hold for finiteness. The key inequality in (E20) which controls (E21) is $Z^* - C'Y^t \geq Z^t - C'Y^t$ and this inequality must hold for any modification attempted.

To see that upper and lower bounds can be obtained at any point in the procedure, consider that Z^t in the integer problem is a lower bound to the true optimum Z^* (see (E16)). To set an upper bound on Z^*, it is necessary to add to Step 2 the solution of:

$$\begin{aligned} \text{Minimize:} \quad & CX \\ \text{Subject to:} \quad & AX \geq B - A'Y^t \\ & X \geq 0 \end{aligned} \tag{P27}$$

where Y^t is the value of Y used in Step 2. If (P27) has a feasible solution, say X^t, then (X^t, Y^t) is a feasible solution to the mixed integer problem and $CX^t + C'Y^t$ would then naturally be an upper bound on the mixed integer solution at each iteration. If (P27) has no feasible solution, then the dual of (P27) was either unbounded or infeasible. If the dual of (P27) was infeasible, it would have been detected in Step 2 of the algorithm that either the mixed integer problem was infeasible or unbounded. In either case, the procedure terminated at that point and determination of an upper bound is immaterial. If the dual of (P27) was unbounded, a constraint was added to prevent unboundedness in the dual of (P27). This would have the effect of increasing the dimensionality of (P27) and this altered (P27) would have a feasible solution which can be used to determine an upper bound.

APPENDIX B

The example presented here is the same one used by Hu (27) for illustration of Benders' algorithm. Consider the problem:

$$\begin{aligned} &\text{Minimize:} && 5X + 2Y + 2W \\ &\text{Subject to:} && X + 3Y + 2W \geq 5 \\ &&& 4X - Y + W \geq 7 \\ &&& 2X + Y - W \geq 4 \qquad\qquad (1) \\ &&& X,Y,W \geq 0 \quad Y \equiv W \equiv 0 \ (\text{mod } 1). \end{aligned}$$

Rewriting (1), one has:

$$\begin{aligned} &\text{Minimize:} && 5X \\ &\text{Subject to:} && X \geq 5 - 3Y - 2W \\ &&& 4X \geq 7 + Y - W \\ &&& 2X \geq 4 - Y + W \qquad\qquad (2) \\ &&& X \geq 0. \end{aligned}$$

The dual program of (2) is:

$$\begin{aligned} &\text{Maximize:} && (5 - 3Y - 2W)\ U_1 + (7 + Y - W)\ U_2 + \\ &&& (4 - Y + W)\ U_3 \\ &\text{Subject to:} && U_1 + 4U_2 + 2U_3 \leq 5 \qquad\qquad (3) \\ &&& U_1, U_2, U_3 \geq 0. \end{aligned}$$

Rewriting (1), it is possible to say:

$$\begin{aligned} &\text{Minimize:} && Z \\ &\text{Subject to:} && Z \geq 2Y + 2W + \text{Max } U^p(5 - 3Y - 2W, \\ &&& \quad 7 + Y - W,\ 4 - Y + W) \\ &&& Y,W \geq 0 \quad Y \equiv W \equiv 0 \pmod 1. \qquad (4) \end{aligned}$$

One feasible solution of (3) is $U_1^1 = 0$, $U_2^1 = 5/4$, $U_3^1 = 0$. Substituting this solution into (4), one gets:

$$\begin{aligned} &\text{Minimize:} && Z \\ &\text{Subject to:} && Z \geq 2Y + 2W + 5/4(7 + Y - W) \\ &&& Y,W \geq 0 \quad Y \equiv W \equiv 0 \pmod 1. \qquad (5) \end{aligned}$$

The solution of (5) is $Z^{1'} = 35/4$ and $Y^1 = W^1 = 0$. Substituting this solution into (3), one gets:

$$\begin{aligned} &\text{Maximize:} && 5U_1 + 7U_2 + 4U_3 \\ &\text{Subject to:} && U_1 + 4U_2 + 2U_3 \leq 5 \qquad (6) \\ &&& U_1, U_2, U_3 \geq 0. \end{aligned}$$

The solution to (6) is $(U_1^2, U_2^2, U_3^2) = (5,0,0)$ with an objective function value of 25.

Since $35/4 = 2Y - 2W = 35/4 < 25$, it is necessary to continue and one adds (5,0,0) into (4) in order to generate a completely new constraint.

$$\begin{array}{ll} \text{Minimize:} & Z \\ \text{Subject to:} & Z \geq 2Y + 2W + 5/4(7 + Y - W) \\ & Z \geq 2Y + 2W + 5(5 - 3Y - 2W) \\ & Y,W \geq 0 \quad Y \equiv W \equiv 0 \pmod 1. \end{array} \tag{7}$$

The solution if $Y^2 = 0$, $W^2 = 2$ and $Z^2 = 41/4$.

Substituting this solution into (3):

$$\begin{array}{ll} \text{Maximize:} & U_1 + 5U_2 + 6U_3 \\ \text{Subject to:} & U_1 + 4U_2 + 2U_3 \leq 5 \\ & U_1,U_2,U_3 \geq 0. \end{array} \tag{8}$$

The solution to (8) is $U_1^3 = 0$, $U_2^3 = 0$, $U_3^3 = 5/2$ with an objective function value of 15. Since $41/4 - 2Y - 2W = 41/4 - 4 = 25/4 < 15$, one continues by generating another constraint for (4):

$$\begin{array}{ll} \text{Minimize:} & Z \\ \text{Subject to:} & Z \geq 2Y + 2W + 5/4(7 + Y - W) \\ & Z \geq 2Y + 2W + 5\ (5 - 3Y - 2W) \\ & Z \geq 2Y + 2W + 5/2(4 - Y + W) \\ & Y,W \geq 0 \quad Y \equiv W \equiv 0 \pmod 1. \end{array} \tag{9}$$

The solution is $Y^3 = 1$, $W^3 = 0$ and $Z^3 = 12$.

Substituting this into (3):

$$\begin{array}{ll} \text{Maximize:} & 2U_1 + 8U_2 + 3U_3 \\ \text{Subject to:} & U_1 + 4U_2 + 2U_3 \leq 5 \\ & U_1,U_2,U_3 \geq 0. \end{array} \tag{10}$$

The solution is $U_1^4 = 5$, $U_2^4 = 0$ and $U_3^4 = 0$ with an objective function value of 10. Since $12 - 2Y - 2W = 12 - 2 = 10$, the solution is optimum. Substituting $Y = 1$ and $W = 0$ into (2), one has:

$$\begin{aligned} \text{Minimize:} \quad & 5X \\ \text{Subject to:} \quad & X \geq 2 \\ & 4X \geq 8 \\ & 2X \geq 3 \\ & X \geq 0. \end{aligned} \tag{11}$$

The solution is $X = 2$ with $5X = 10$ as expected. The optimum solution to (1) is therefore $X^* = 2$, $Y^* = 1$, $W^* = 0$ and $Z^* = 12$.

APPENDIX C

Below is a table showing the timing and iteration count results for Benders' algorithm and the three suggested modifications. The numbers in parentheses are the iterations required for solution.

Problem	Benders' Original	Modification A	Modification B	Modification C
1a	1.34(1)	1.23(1)	1.28(1)	1.31(1)
1b	2.69(2)	2.62(2)	2.59(2)	3.22(2)
1c	1.40(1)	1.59(1)	1.38(1)	1.35(1)
1d	2.76(2)	2.92(2)	2.54(2)	3.99(3)
1e	4.07(3)	3.94(3)	3.68(3)	4.39(3)
1f	4.44(4)	4.20(4)	4.22(4)	4.45(4)
2a	6.18(3)	6.94(3)	5.74(3)	6.94(3)
2b	7.71(4)	7.74(4)	5.98(3)	7.56(4)
2c	5.94(3)	6.82(3)	6.76(3)	7.01(3)
2d	6.70(4)	8.71(4)	5.11(3)	9.01(5)
2e	8.31(5)	22.44(7)	6.98(4)	10.01(6)
2f	7.12(4)	10.67(5)	7.08(4)	10.07(6)
3a	4.16(7)	3.26(7)	4.07(6)	5.53(9)
3b	5.23(9)	4.82(9)	4.17(6)	8.13(16)
3c	4.00(7)	3.29(7)	4.12(6)	5.20(9)
3d	5.23(11)	4.75(11)	4.58(8)	5.90(11)
3e	3.31(7)	3.13(7)	2.44(4)	4.57(9)
3f	5.15(7)	4.46(7)	4.71(8)	7.82(11)

Cont.

Problem	Benders' Original	Modification A	Modification B	Modification C
4a	*290.55(2+)	116.06(12)	*290.10(2+)	83.32(18)
4b	*290.25(3+)	158.98(14)	*288.90(3+)	89.57(19)
4c	272.82(10)	139.77(15)	*289.05(3+)	66.79(12)
4d	209.27(10)	142.31(9)	256.77(10)	97.76(15)
4e	*290.25(2+)	105.87(13)	*288.75(2+)	101.97(20)
4f	*290.85(5+)	40.72(9)	*290.55(4+)	85.78(18)
5a	126.00(5)	122.75(5)	88.01(5)	63.11(5)
5b	13.12(3)	12.45(3)	12.88(3)	11.79(4)
5c	98.47(4)	98.72(4)	76.99(4)	64.05(5)
5d	20.32(3)	19.80(3)	24.15(3)	19.05(3)
5e	29.73(4)	28.29(4)	41.78(4)	18.23(4)
5f	7.71(4)	9.88(4)	8.58(4)	9.31(6)
6a	0.12(2)	0.13(2)	0.08(1)	0.20(3)
6b	1.99(6)	1.73(6)	1.81(6)	4.19(10)
6c	*288.75(17+)	*288.75(33+)	276.81(13)	76.88(19)
6d	*288.00(20+)	251.87(33)	*287.55(17+)	74.62(18)
6e	191.56(15)	187.89(24)	*288.30(13+)	*289.20(19+)

* No solution in stated time.

All of the above problems were run on the UNIVAC 1108 located on the Madison Campus of the University of Wisconsin and all timings are in seconds. The timings were done so as to measure the actual time for solution, not including such things as input and output.

It has been discovered by the author that the modified Balas algorithm is so efficient that on small problems, i.e. 15 integer, 12 linear variables and 12 constraints, the suggested modifications' efficiency was at best marginal.

This is due to the fact that all of the modifications depend ultimately upon trading integer iterations for linear iterations. Since the modified Balas integer algorithm was so efficient, this was not usually a good trade.

As can be seen, however, on large problems, i.e. those requiring large amounts of time for the execution of the integer subproblem, the relative efficiency of the suggested modifications became marked.

APPENDIX D

Test Problems:

The test problems used to evaluate the modifications suggested by this paper fall into two categories. The first of these is composed of the limited number of mixed integer problems found in the literature search. These problems are frequently quite small but act as some sort of standard for the purpose of evaluation. The second category of test problems consists of problems generated by the author. They are developed by consideration of three classes of real-life situations, capital budgeting problems, fixed charge transportation problems and allocation-location problems, all of which may be advanced as mixed integer programs.

The test problems used are shown in the two sections below with a discussion of the mathematical model behind each of the type two problems.

Literature Problems:

There were only four problems taken from current literature to test the suggested modifications. There were two reasons why such a small number of literature problems were tested. First, almost all problems presented in detail in the literature were extremely small and therefore did not give a true test of the modifications. Second, those problems discussed but not presented in detail were too large (and therefore too costly) to be included in this limited study.

The first problem, 6a, was taken from Hu (27) and is the one presented as an example in Appendix B. The second problem, 6b, may be found in many references and the author found it first in the article by Balinski (4). It is a special form of the plant location problem and is presented in detail below:

$$\text{Minimize:} \quad \sum_{i=1}^{n} f_i \, y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} \, X_{ij}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} X_{ij} \geq 1 \text{ for } j = 1,\ldots,m$$

$$X_{ij} \leq Y_i \text{ for } i = 1,\ldots,n \text{ and } j = 1,\ldots,m$$

$$X_{ij} \geq 0 \quad Y_i = 0,1.$$

The particular problem presented in the literature and solved in the dissertation is one in which $m = n = 4$, $f_i = 7$ for $i = 1,\ldots,n$ and

$$(C_{ij}) = \begin{pmatrix} 0 & 12 & 20 & 18 \\ 12 & 0 & 8 & 6 \\ 20 & 8 & 0 & 6 \\ 18 & 6 & 6 & 0 \end{pmatrix}$$

Problems 6d and 6e were fixed charged transportation problems (as described below) and are found in a technical report by Gray (24). The particular problems used for test purposes are also presented in the next section.

Author Generated Problems:

Capital Budgeting --

Problems 1a - 1f were developed from a very simple capital budgeting model (33). In this model the 0-1 integer variables represent the decision on investment in each of a number of projects. Each of these projects requires investment of differing amounts of cash in each future time period considered. Investment in each project also results in some estimated future return on the investment. This return is then discounted at the business's required rate of return to obtain a present value for the project. This present value appears as a positive coefficient in the objective function.

The linear variables consist of the amount of money left uninvested at the end of each of the time periods.

Since money uninvested generally results in a loss of profit at a rate equal to the difference between the required rate of return and the bank rate of interest, the different amounts left uninvested must be discounted at this rate and the present value which is obtained will appear as a negative coefficient in the objective function.

The basic capital budgeting problem can be expressed as:

$$\text{Maximize:} \quad \sum_{i=1}^{n} c_i x_i + \sum_{j=1}^{m} c_j' y_j$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} a_{ji} x_i + y_j - y_{j-1} = b_j \quad j=1,2,\ldots,m$$

$$x_i = 0,1 \quad i=1,2,\ldots,n \qquad \text{(E1)}$$

$$y_o = 0, \; y_j \geq 0 \quad j=1,2,\ldots,m$$

where c_i represents the calculated present value of project i, c_j' represents the calculated present rate of loss on money left uninvested in period j, x_i the decision of investment in project i and y_j the amount of cash left uninvested at the end of period j. a_{ji} represents the amount of investment required in the time period j for project i and b_j represents the budgeted amount of cash for all projects in time period j.

(E1) can be expressed as:

$$\text{Maximize:} \quad CX - C'Y$$

$$\text{Subject to:} \quad AX + Y_j - Y_{j-1} = B \qquad \text{(E2)}$$

$$X = 0,1 \quad Y_{j-1}, Y_j \geq 0 \quad Y_o = 0.$$

where the use of Y_j and Y_{j-1} is merely a way to indicate the relationship between the two column vectors.

For test problems 1a - 1f, the following values were used:

$m = 12$

$n = 15$

C = (21.5, 32.0, 17.5, 40.0, 9.4, 3.5, 23.0, 9.0, 15.6, 11.0, 5.25, 17.0, 9.0, 60.0, 18.5)

C' = (.0175, .0180, .0186, .0192, .0198, .0204, .0210, .0217, .0223, .0230, .0237, .0245)

B =

1a) (58.45, 45.56, 29.25, 33.78, 19.84, 21.15, 37.36, 27.12, 13.88, 12.84, 13.70, 21.20)
1b) (30., 30., 30., 30., 30., 30., 30., 30., 30., 30., 30., 30.)
1c) (22.5, 22.5, 22.5, 22.5, 22.5, 22.5, 22.5, 22.5, 22.5, 22.5, 22.5, 22.5)
1d) (15., 15., 15., 15., 15., 15., 15., 15., 15., 15., 15., 15.)
1e) (7.5, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5)
1f) (50., 40., 30., 33., 21., 20., 35., 26., 15., 10., 25., 20.)

$$
A = \begin{bmatrix}
5.00 & 10.00 & 4.50 & 0.00 & 9.00 & 0.20 & 0.00 & 8.90 & 7.25 & 1.50 & 6.10 & 1.00 & 0.00 & 5.00 & 0.00 \\
5.00 & 6.00 & 3.50 & 0.00 & 0.00 & 0.21 & 0.00 & 0.00 & 7.10 & 1.20 & 6.25 & 1.00 & 0.00 & 4.50 & 0.00 \\
5.00 & 3.50 & 6.50 & 1.00 & 0.00 & 0.22 & 0.00 & 0.00 & 0.00 & 1.00 & 6.00 & 1.00 & 0.00 & 5.00 & 0.00 \\
5.00 & 1.00 & 1.25 & 1.00 & 0.00 & 0.23 & 0.00 & 0.00 & 0.00 & 1.00 & 5.80 & 13.00 & 0.00 & 5.50 & 0.00 \\
0.00 & 1.00 & 0.00 & 1.50 & 0.00 & 0.24 & 0.00 & 0.00 & 0.00 & 0.80 & 5.80 & 0.00 & 0.00 & 6.00 & 4.50 \\
0.00 & 1.00 & 0.00 & 1.25 & 0.00 & 0.25 & 0.00 & 0.00 & 0.00 & 0.80 & 6.00 & 0.00 & 0.00 & 7.50 & 4.35 \\
0.00 & 1.00 & 0.00 & 17.50 & 0.00 & 0.26 & 0.00 & 0.00 & 0.00 & 0.75 & 6.00 & 0.00 & 0.00 & 7.50 & 4.35 \\
0.00 & 1.00 & 0.00 & 11.50 & 0.00 & 0.27 & 3.50 & 0.00 & 0.00 & 0.50 & 4.50 & 0.00 & 0.00 & 1.60 & 4.25 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.28 & 4.50 & 0.00 & 0.00 & 0.00 & 5.00 & 0.00 & 2.00 & 1.60 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 4.25 & 0.00 & 0.00 & 0.00 & 5.00 & 0.00 & 2.20 & 1.10 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.30 & 4.10 & 0.00 & 0.00 & 0.00 & 6.00 & 0.00 & 2.30 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.40 & 4.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.80 & 14.00 & 0.00
\end{bmatrix}
$$

Quite often in capital budgeting problems there are additional constraints placed on the investment procedure. Two types of restrictions were added to problems 1a - 1f to produce problems 2a - 2f. The first additional type of constraint arises from the situation in which investment may be made in only one of a set of projects. For example, the set of projects may be different brand considerations of a new truck to be bought. If one lets S be the set of indices of such mutually exclusive projects, the constraint occurs thusly:

$$\sum_{i \in S} x_i = 1 \qquad \text{(C1)}$$

The second type of additional constraint occurs when the choosing of one project necessitates the selection of one of a set of other projects. An example might be the case where the decision to build a particular type plant necessitates choice among several types of equipment for some part of the plant.

Letting T represent the set of indices of linked projects, one gets:

$$\sum_{i \in T} x_i = X_T \qquad \text{(C2)}$$

where X_T is the project variable to which set T is linked.

Problems 2a - 2F were exactly the same as 1a - 1f except that these two additional constraints have been added. The sets S and T were chosen as:

$$S = (1, 3, 8, 9) \text{ and } T = (5, 6, 10)$$

and X_T was X_{11}.

A third set of capital budgeting problems was run to consider the efficiency of the different modifications in larger problems. Problems 1a - 1f were modified so that an additional ten projects were available, i.e. n = 25, for investment. The ten projects added were identical to the first ten projects already available and all numerical data pertaining to these additional projects is the same as given for projects X_1 through X_{10}. The labels 5a - 5f refer to these larger problems.

Production Allocation --

Problems 3a - 3f and 4a - 4f were developed from different mathematical modifications of a production allocation model. The basic production allocation problem is a linear program. With the addition of set-up costs or concave material costs, it becomes a mixed integer program with 0-1 integer variables. 3a - 3f consist of the basic problem with set-up costs, 4a - 4f with concave material costs.

In defining the complete production allocation model, the following definitions of variables and constants are necessary.

Variables defining the size of the problem:

n number of plants

m number of warehouses

l number of products

f number of materials necessary for production

s number of different types of products, each type requiring a different set-up

e_q number of concave cost segments for material q

Decision variables:

P_{kij} amount of production of product k at plant i for delivery to warehouse j

a_{ti} zero-one variables which are one if some product of type t is produced at plant i, zero otherwise

b_i^{qg} zero-one variables which are one if the amount of material q used at plant i is on the g-th cost segment, zero otherwise

o_i^{qg} amount of material q used in plant i as if all charged at the g-th segment cost

Cost variables:

c_{ki} production cost of product k at plant i, excluding set-up and material cost

w_{ki}^q cost of material q required to produce product k at plant i under linear material costs

x_{ij} cost of shipment of any product from plant i to warehouse j

v_{ti} set-up cost for product type t at plant i

y_i^{qg} base cost of the g-th segment for material q at plant i

z_i^{qg} incremental cost of the g-th segment for material q at plant i

Constraint variables:

d_{kj} demand at warehouse j for product k

r_i total available production time at plant i

t_{ki} amount of production time required to produce product k at plant i

h_{ti} amount of production time lost during set-up for product type t at plant i

u^q_{ki} amount of material q necessary to produce product k at plant i

s^{qg}_i upper bound in usage of material q at plant i at segment g's incremental price

In addition, it is useful to define the following summations to facilitate understanding of our final models.

$$C = \sum_{i=1}^{n} \sum_{k=1}^{l} \sum_{j=1}^{m} P_{kij}\ c_{ki}$$

$$W = \sum_{i=1}^{n} \sum_{k=1}^{l} \sum_{j=1}^{m} \sum_{q=1}^{f} P_{kij}\ w^q_{ki}$$

$$X = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} P_{kij}\ x_{ij}$$

$$A = \sum_{i=1}^{n} \sum_{t=1}^{x} v_{ti}\ a_{ti}$$

$$B = \sum_{q=1}^{f} \sum_{i=1}^{n} \sum_{g=1}^{e_q} y^{qg}_i\ b^{qg}_i$$

$$Z = \sum_{q=1}^{f} \sum_{i=1}^{n} \sum_{g=1}^{e_q} z^{qg}_i\ o^{qg}_i$$

Define M to be a very large number.

With the addition of set-up costs only, e.g. problems 3a - 3f, the form of problem to be solved is:

Minimize: $C + W + X + A$

Subject to:

$$\sum_{i=1}^{n} P_{kij} \geq d_{kj} \quad \text{for } k=1,\ldots,1 \text{ and } j=1,\ldots,m$$

$$\sum_{k=1}^{1} \sum_{j=1}^{m} t_{ki} P_{kij} + \sum_{t=1}^{s} h_{ti} a_{ti} \leq r_i \quad \text{for } i=1,\ldots,n$$

$$\sum_{k \in G_t} \sum_{j=1}^{m} P_{kij} \leq M a_{ti} \quad \text{for } i=1,\ldots,n \text{ and } t=1,\ldots,s$$

$$P_{kij} \geq 0 \text{ for all } i,\ j \text{ and } k$$

$$a_{ti} = 0,1 \text{ for all } i \text{ and } t$$

where G_t is the set of indexes of products of type t.

With the addition of concave material costs only, e.g. problems 4a - 4f, the form of the problem becomes:

Minimize: $C + X + B + Z$

Subject to:

$$\sum_{i=1}^{n} P_{kij} \geq d_{kj} \quad \text{for } k=1,\ldots,1 \text{ and } j=1,\ldots,m$$

$$\sum_{k=1}^{1} \sum_{j=1}^{m} t_{ki} P_{kij} \leq r_i \quad \text{for } i=1,\ldots,n$$

$$\sum_{k=1}^{1} \sum_{j=1}^{m} P_{kij} u_{ki}^{q} \leq \sum_{g=1}^{e_q} o_i^{qg}$$

$$\text{for } i=1,\ldots,n \text{ and } q=1,\ldots,f$$

$$\sum_{g=1}^{e_q} b_i^{qg} = 1 \text{ for } q=1,\ldots,f \text{ and } i=1,\ldots,n$$

$$o_i^{qg} \leq b_i^{qg} s_i^{qg} \text{ for } i=1,\ldots,n;\ l=1,\ldots,f \text{ and } g=1,\ldots,e_q$$

$$P_{kij} \geq 0 \text{ for all } i,\ j \text{ and } k$$

$$b_i^{qg} = 0,1 \text{ for all } i,\ q \text{ and } g.$$

In addition, certain assumptions about the models were made. First, for all practical purposes an infinite amount of each material is available at each plant; second, the cost of shipping any product from a plant to a warehouse is the same, regardless of the product; and third, the amount of material necessary to build a product at a plant was the same regardless of its eventual destination.

Test problems 3a and 4a were run with the following data:

$n = 2$

$m = 2$

$l = 4$

$f = 2$

$s = 2$ (products 1 and 2 were of type 1 and products 3 and 4 were of type 2)

$e_1 = 5$

$e_2 = 5$

$$c_{ki} = \begin{pmatrix} 5.54 & 6.71 \\ 9.72 & 14.46 \\ 18.50 & 15.03 \\ 5.32 & 16.54 \end{pmatrix}$$

$$w^{q}_{ki} = \begin{pmatrix} 0.80 & 0.95 \\ 4.80 & 3.80 \\ 1.60 & 1.52 \\ 2.20 & 2.28 \end{pmatrix} \qquad \begin{pmatrix} 4.06 & 3.90 \\ 3.77 & 3.90 \\ 2.90 & 3.00 \\ 3.48 & 3.30 \end{pmatrix}$$

$$x_{ij} = \begin{pmatrix} 3.90 & 2.10 \\ 1.40 & 2.80 \end{pmatrix}$$

$$v_{ti} = \begin{pmatrix} 34.44 & 19.74 \\ 30.66 & 48.88 \end{pmatrix}$$

$$y^{qg}_{i} = \begin{pmatrix} 0.0 & 0.0 \\ 3.05 & 1.26 \\ 6.175 & 8.3 \\ 12.675 & 10.9 \\ 16.215 & 11.565 \end{pmatrix} \qquad \begin{pmatrix} 0.0 & 0.0 \\ 2.0 & 1.8 \\ 4.925 & 8.565 \\ 12.605 & 11.74 \\ 13.655 & 13.69 \end{pmatrix}$$

$$z^{qg}_{i} = \begin{pmatrix} 2.75 & 2.75 \\ 2.50 & 2.65 \\ 2.25 & 2.10 \\ 1.75 & 1.90 \\ 1.55 & 1.85 \end{pmatrix} \qquad \begin{pmatrix} 2.60 & 2.80 \\ 2.45 & 2.65 \\ 2.20 & 2.10 \\ 1.60 & 1.85 \\ 1.55 & 1.70 \end{pmatrix}$$

$$d_{kj} = \begin{pmatrix} 1.40 & 2.80 \\ 2.40 & 4.20 \\ 3.90 & 6.00 \\ 1.10 & 3.20 \end{pmatrix}$$

$$r_{i} = (52.3,\ 85.1)$$

$$t_{ki} = \begin{pmatrix} 1.2 & 1.3 \\ 2.1 & 2.8 \\ 4.0 & 2.9 \\ 1.1 & 3.2 \end{pmatrix}$$

$$H_{ti} = \begin{pmatrix} 8.2 & 7.3 \\ 4.2 & 10.4 \end{pmatrix}$$

$$u^{q}_{ki} = \begin{pmatrix} 0.4 & 0.5 \\ 2.4 & 2.0 \\ 0.8 & 0.8 \\ 1.1 & 1.2 \end{pmatrix} \qquad \begin{pmatrix} 1.4 & 1.4 \\ 1.3 & 1.3 \\ 1.0 & 1.0 \\ 1.2 & 1.1 \end{pmatrix}$$

$$s_i^{qg} = \begin{pmatrix} 12.2 & 12.6 \\ 12.5 & 12.8 \\ 13.0 & 13.0 \\ 13.2 & 13.3 \\ 99.9 & 99.9 \end{pmatrix} \qquad \begin{pmatrix} 12.0 & 12.0 \\ 12.5 & 12.3 \\ 12.0 & 12.7 \\ 13.0 & 13.0 \\ 99.9 & 99.9 \end{pmatrix}$$

Problems 3b - 3f and 4b - 4f were the same as above except that demand at each of the warehouses was altered as was the total production time available at each plant. These changed values were:

3b and 4b $d_{kj} = \begin{pmatrix} 2.6 & 3.1 \\ 2.2 & 1.0 \\ 3.2 & 4.5 \\ 6.1 & 0.7 \end{pmatrix}$ $r_i = (48.7, 69.8)$

3c and 4c $d_{kj} = \begin{pmatrix} 9.4 & 1.2 \\ 6.2 & 0.2 \\ 0.0 & 7.1 \\ 0.0 & 2.2 \end{pmatrix}$ $r_i = (72.9, 48.8)$

3d and 4d $d_{kj} = \begin{pmatrix} 3.1 & 2.2 \\ 4.4 & 4.4 \\ 2.2 & 1.6 \\ 3.9 & 4.4 \end{pmatrix}$ $r_i = (55.0, 55.0)$

3e and 4e $d_{kj} = \begin{pmatrix} 3.9 & 3.9 \\ 4.0 & 4.0 \\ 2.0 & 2.0 \\ 0.9 & 0.9 \end{pmatrix}$ $r_i = (77.6, 44.4)$

3f and 4f $d_{kj} = \begin{pmatrix} 4.4 & 3.9 \\ 6.1 & 9.4 \\ 8.2 & 2.1 \\ 1.4 & 1.9 \end{pmatrix}$ $r_i = (28.7, 61.2)$

Fixed-Charge Transportation --

The fixed-charge transportation problem as discussed by Gray (25) is concerned with the transportation of goods from a set of n factories to a set of m warehouses with costs involving both a cost per unit measure for each possible transportation link and a fixed charge for the operation of the link. Thus, if <u>any</u> products are shipped from a particular factory to one of the warehouses, the fixed charge associated with that link must be added to the overall system cost.

Mathematically, one wishes to:

$$\text{Minimize:} \quad C = \sum_{i=1}^{n} \sum_{j=1}^{m} (c_{ij}\, x_{ij} + f_{ij}\, y_{ij})$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_{ij} \geq D_j \quad \text{for } j=1,2,\ldots,m$$

$$\sum_{j=1}^{m} x_{ij} \leq s_i \quad \text{for } i=1,2,\ldots,n$$

$$m_{ij}\, y_{ij} - x_{ij} \geq 0 \quad \text{for } i=1,2,\ldots,n \text{ and } j=1,2,\ldots,m$$

$$x_{ij} \geq 0 \quad y_{ij} = 0,1 \quad \text{for } i=1,2,\ldots,n \text{ and } j=1,2,\ldots,m$$

where x_{ij} is the amount to be shipped from factory i to warehouse j; y_{ij} = 1 if there is any shipment between factory i and warehouse j, y_{ij} = 0 otherwise; c_{ij} is the cost per unit measure from point i to point j; f_{ij} is the fixed charge for a shipment from i to j; D_j is the demand at warehouse j;

s_i is the supply at factory i; and $m_{ij} = \min(D_j, s_i)$, i.e. the maximum possible size of all of the shipments.

For the problems (6c, 6d and 6e) used in testing, the following values were used

6c: $D_j = (325, 610, 422, 291, 345)$

$s_i = (482, 799, 123, 385, 204)$

$$c_{ij} = \begin{pmatrix} 2.3 & 2.9 & 3.4 & 6.1 & 4.2 \\ 4.7 & 8.3 & 1.6 & 2.7 & 7.4 \\ 3.2 & 4.1 & 9.6 & 6.2 & 4.8 \\ 9.1 & 9.2 & 6.3 & 2.3 & 8.4 \\ 4.3 & 2.0 & 6.1 & 3.3 & 4.4 \end{pmatrix}$$

$$f_{ij} = \begin{pmatrix} 328 & 372 & 1462 & 1148 & 608 \\ 332 & 1934 & 778 & 632 & 1578 \\ 78 & 1488 & 180 & 844 & 322 \\ 428 & 430 & 1120 & 892 & 1826 \\ 1824 & 1000 & 1306 & 550 & 742 \end{pmatrix}$$

$$m_{ij} = \begin{pmatrix} 325 & 482 & 422 & 291 & 345 \\ 325 & 610 & 422 & 291 & 345 \\ 123 & 123 & 123 & 123 & 123 \\ 325 & 385 & 385 & 291 & 345 \\ 204 & 204 & 204 & 204 & 204 \end{pmatrix}$$

6d: $D_j = (35, 30, 25, 15, 5, 5)$

$s_i = (45, 35, 20, 15)$

$$c_{ij} = \begin{pmatrix} 0.69 & 0.64 & 0.71 & 0.79 & 1.70 & 2.83 \\ 1.01 & 0.75 & 0.88 & 0.59 & 1.50 & 2.63 \\ 1.05 & 1.06 & 1.08 & 0.64 & 1.22 & 2.37 \\ 1.94 & 1.50 & 1.56 & 1.22 & 1.98 & 1.98 \end{pmatrix}$$

$$f_{ij} = \begin{pmatrix} 11 & 16 & 18 & 17 & 10 & 20 \\ 14 & 17 & 17 & 13 & 15 & 13 \\ 12 & 13 & 20 & 17 & 13 & 15 \\ 16 & 19 & 16 & 11 & 15 & 12 \end{pmatrix}$$

$$m_{ij} = \begin{pmatrix} 35 & 30 & 25 & 15 & 5 & 5 \\ 35 & 30 & 25 & 15 & 5 & 5 \\ 20 & 20 & 20 & 15 & 5 & 5 \\ 15 & 15 & 15 & 15 & 5 & 5 \end{pmatrix}$$

6e: $D_j = (55, 54, 35, 22, 9, 8)$

$s_i = (23, 38, 56, 66)$

$$c_{ij} = \begin{pmatrix} 19 & 6 & 12 & 16 & 13 & 24 \\ 5 & 29 & 8 & 19 & 109 & 26 \\ 38 & 17 & 14 & 23 & 27 & 114 \\ 6 & 20 & 2 & 92 & 29 & 42 \end{pmatrix}$$

$$f_{ij} = \begin{pmatrix} 4 & 3 & 0 & 6 & 8 & 7 \\ 5 & 16 & 24 & 9 & 11 & 2 \\ 12 & 5 & 10 & 6 & 9 & 43 \\ 8 & 31 & 6 & 12 & 36 & 19 \end{pmatrix}$$

$$m_{ij} = \begin{pmatrix} 23 & 23 & 23 & 22 & 9 & 8 \\ 38 & 38 & 35 & 22 & 9 & 8 \\ 55 & 54 & 35 & 22 & 9 & 8 \\ 55 & 54 & 35 & 22 & 9 & 8 \end{pmatrix}$$

It is interesting to note that once a set of values for the 0-1 integer variables y_{ij} is determined that the problem is a simple transportation problem. The technique used by the author, however, was to continue to use a linear programming algorithm rather than a transportation one.

APPENDIX E

The 0-1 integer variable problems were solved using an algorithm based upon principles involved in Balas' 0-1 algorithm. Modification was necessitated by the unusual way in which the objective function values are determined in the "pure" integer problem in Benders' algorithm.

The first step in Balas' algorithm is the substitution of 1 - y' for any y in the objective function with a negative coefficient. This has the practical effect of setting y to one and therefore the changing of any value (from 0 to 1 for unsubstituted variables, from 1 to 0 for substituted ones) will result in an increase in the objective function value. As the value of the objective function in our case is determined directly from the constraints, we would like to do likewise, except that to accomplish the same end, i.e. get the problem to an optimal, though possibly nonfeasible state, the substitution must be made in each individual constraint for those variables with positive coefficients.

Then, in order to reach feasibility, some values of y and y' must be changed as it is necessary that $y = 1$ when $y' = 0$ and, obviously, that $y = 0$ when $y' = 1$.

Let us look at an example problem so that it will be possible to trace the procedure more clearly.

Minimize: z

Subject to:
$$z + 3x_1 - 2x_2 + 3x_3 \geq 4$$
$$z - x_1 + 3x_2 - 2x_3 \geq 2$$
$$z + 4x_1 - 3x_2 + 2x_3 \geq 3$$
$$x_1, x_2, x_3 = 0,1.$$

After substitution, one has:

Minimize: z

Subject to:
$$z - 3x_1' - 2x_2 - 3x_3' \geq -2$$
$$z - x_1 - 3x_2' - 2x_3 \geq -1$$
$$z - 4x_1' - 3x_2 - 2x_3' \geq -3$$
$$x_1, x_2, x_3, x_1', x_2', x_3' = 0,1.$$
$$x_1 + x_1' = 1,\ x_2 + x_2' = 1,\ x_3 + x_3' = 1.$$

Feasibility is now determined by assuring that when $x_i' = 0$, $x_i = 1$ and when $x_i' = 1$, $x_i = 0$ for $i = 1,2,3$. Note that the above solution is optimum but not feasible when all x_i' and x_i are zero.

A table can now be set up indicating the resulting increase in the right-hand side of each constraint if each of the six variables changes to a value of one.

		x_1	x_1'	x_2	x_2'	x_3	x_3'
Constraint:	1	0	3	2	0	0	3
	2	1	0	0	3	2	0
	3	0	4	3	0	0	2

By considering which of each pair of linked variables is to be set to one, it is possible to resolve the conflict between the linked variables to eventually develop feasibility.

It was found that in the example problems used in this dissertation that the use of complicated decision rules as to the proper order of entry of variables resulted in more calculations than a simple order of input scheduling scheme. Thus, consider x_1 and x_1'.

		x_1	x_1'	R.H.S.
Constraint:	1	-2	1*	-2
	2	0*	-1	-1
	3	-3	1*	-3

The columns headed by x_1 and x_1' were derived by adding the appropriate columns in the table above to the right-hand side values. The numbers in the two columns represent an optimal, though not necessarily feasible, value for the objective function if x_1 and x_1' respectively are set to one. Since all constraints are "greater than or equal to", one picks the largest value in each column (indicated by an asterisk) to indicate the limiting constraint.

Selecting the column with the smallest of the asterisked values (thus keeping the objective function value as low as possible), one then considers the next variable in line, using the selected column as the new right-hand side.

		x_2	x_2'	R.H.S.
Constraint:	1	0*	-2	-2
	2	0	3*	0
	3	0*	-3	-3

Continuing:

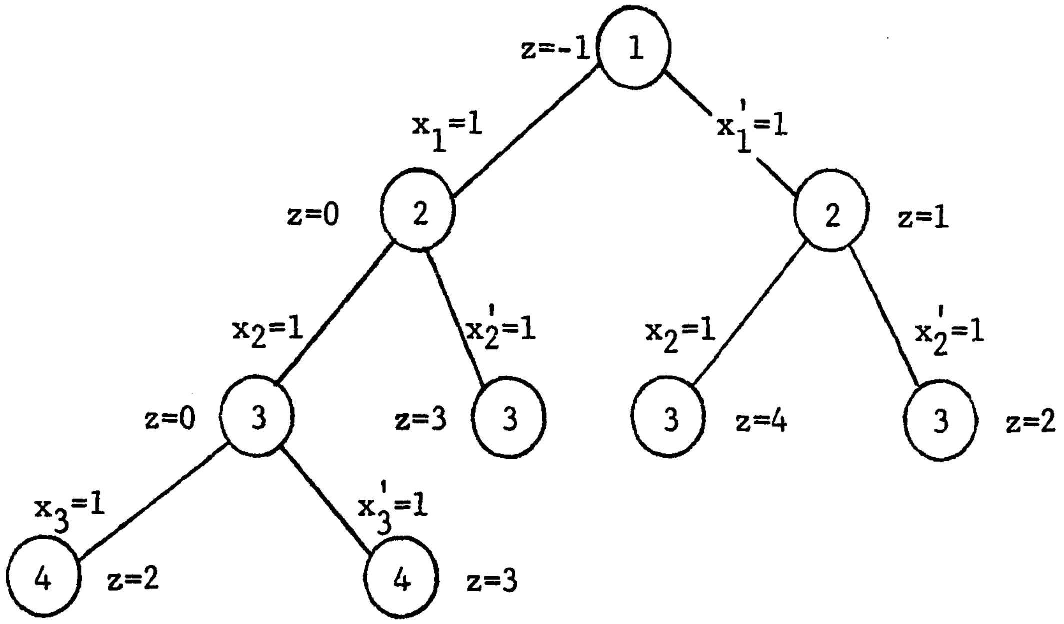
		x_3	x_3'	R.H.S.
Constraint:	1	0	3*	0
	2	2*	0	0
	3	0	2	0

Thus, one has an initial solution, namely $x_1 = 1$, $x_2 = 1$, and $x_3 = 1$ for a value of 2. It is now necessary to backtrack to see if any other branch of the solution tree can result in a lower value. If $x_3' = 1$, the best possible value is 3 so that one should backtrack further. If $x_2' = 1$, the best is 3 so backtrack even further. If $x_1' = 1$, the best is 1 so that it is conceivable that by setting $x_1' = 1$ and then applying the procedure as above that it will result in a lower value for the objective function than 2. Thus, continuing from this point:

		x_2	x_2'	R.H.S.
Constraint:	1	3	1	1
	2	-1	2*	-1
	3	4*	1	1

Since the best fs now 2 and we already have a solution with that value, this branch of the tree has been fathomed. Backtracking again now shows that the solution is complete. Checking back with our original problem, we find that the solution found does satisfy all constraints and produces a value of 2 for z.

Thus the solution tree generated in this problem would be:



A detailed description of the algorithm used to solve the integer subproblem follows. Given the problem:

$$\text{Minimize:} \quad Z$$

$$\text{Subject to:} \quad Z + \sum_{j=1}^{n} a_{ij}x_j \geq b_i \qquad \text{for } i = 1,\ldots,m$$

$$x_j = 0,1 \qquad \text{for } j = 1,\ldots,n.$$

In each constraint substitute $1 - x_j'$ for any x_j where $a_{ij} > 0$ and set $x_j + x_j' = 1$. Since one begins the procedure with all x_j and x_j' equal to zero, an obvious infeasibility exists and the procedure will determine whether x_j or x_j' will be equal to one. This determination will be made beginning with x_1 and x_1' and proceeding naturally to x_n and x_n'. After substitution, one has:

$$\text{Minimize:} \quad Z$$

$$\text{Subject to:} \quad Z - \sum_{j\varepsilon J_{i+}} a_{ij}x_j' - \sum_{j\varepsilon J_{i-}} a_{ij}x_j \geq$$

$$b_i - \sum_{j\varepsilon J_{i+}} a_{ij} \qquad \text{for } i = 1,\ldots,m$$

$$x_j + x_j' = 1 \qquad \text{for } j = 1,\ldots,n$$

$$x_j = 0,1 \quad x_j' = 0,1 \qquad \text{for } j = 1,\ldots,n,$$

where $J_{i+} = (j \mid a_{ij} > 0 \text{ in constraint } i)$ and $J_{i-} = (j \mid a_{ij} \leq 0 \text{ in constraint } i)$.

If $x_j = x_j^{'} = 0$,for all j, then one has an optimum but infeasible solution with:

$$z^* = \max_i \; (b_i - \sum_{j\varepsilon J_{i+}} a_{ij}).$$

Feasibility is satisfied by making $x_j + x_j^{'} = 1$ for all j.

Let t represent the level of the current node in the solution tree. P_t will represent the status of the branches from the t-th node with $P_t = 1$ indicating that the branch with $x_j = 1$ is being explored, P_t that the one with $x_j^{'} = 1$ is being explored and $P_t = 3$ that both branches have been explored.

Step 0: Set $t = 1$, $B = +\infty$ and $r_{i1} = b_i - \sum_{j\varepsilon J_{i+}} a_{ij}$.

Step 1: Calculate:

$$s_{it} = \begin{cases} r_{it} + a_{it} & \text{if } t \ \varepsilon \ J_{i-} \\ r_{it} & \text{if } t \ \varepsilon \ J_{i+} \end{cases}$$

$$s_{it}^{'} = \begin{cases} r_{it} & \text{if } t \ \varepsilon \ J_{i-} \\ r_{it} + a_{it} & \text{if } t \ \varepsilon \ J_{i+}. \end{cases}$$

<u>Step 2</u>: Let:

$$q_t = \max_i (s_{it}) \quad \text{and} \quad q_t' = \max_i (s_{it}').$$

If $\min(q_t, q_t') \geq B$, go to step 5. If $\min(q_t, q_t') < B$, and $t = n$ go to step 4, otherwise go to step 3.

<u>Step 3</u>: If $q_t' > q_t$, set $r_{i(t+1)} = s_{it}$ for $i=1,\ldots,m$ and set $P_t = 1$. If $q_t' \leq q_t$, set $r_{i(t+1)} = s_{it}'$ for $i=1,\ldots,m$ and $P_t = 2$. Let $t = t + 1$, go to step 1.

<u>Step 4</u>: One now has a feasible solution as indicated by the values of P_t for $t=1,\ldots,n$. Since $\min(q_t, q_t') < B$, the presently found feasible solution is better than the best found thus far. Set $B = \min(q_t, q_t')$ and save the currently found feasible solution. Go to step 5.

<u>Step 5</u>: Let $t = t - 1$. If $t = 0$, go to step 7, otherwise go to step 6.

<u>Step 6</u>: If (a) $P_t = 3$ or (b) $P_t = 1$ and $q_t' \geq B$ or (c) $P_t = 2$ and $q_t \geq B$, go to step 5. If $P_t = 1$ and $q_t' < B$, set $P_t = 3$ and $r_{i(t+1)} = s_{it}'$ for $i=1,\ldots,m$. If $P_t = 2$ and $q_t < B$, set $P_t = 3$ and $r_{i(t+1)} = s_{it}$ for $i=1,\ldots,m$. Set $t = t + 1$, go to step 1.

<u>Step 7</u>: The optimum solution is B with the corresponding derived values for x_j and x_j'.

BIBLIOGRAPHY

(1) Aldrich, D. W. "A Decomposition Approach to the Mixed Integer Problem," Dissertation, Purdue University, 1969.

(2) Balas, E. "Discrete Programming by the Filter Method," Operations Research, Vol. 15, No. 5, pp. 935-944 (September-October, 1967).

(3) Balinski, M. L. and Wolfe, P. "On Benders Decomposition and a Plant Location Problem," Mathematica Working Paper ARO-27, 1963.

(4) Balinski, M. L. "Integer Programming: Methods, Uses, Computation," Management Science, Vol. 12, No. 3, pp. 253-313 (November, 1965).

(5) Beale, E. M. L. "A Method of Solving Linear Programming Problems When Some But Not All of the Variables Must Have Integral Values," Statistical Tech Research Group, Princeton University (March, 1958).

(6) Beale, E. M. L. and Small, R. E. "Mixed Integer Programming by a Branch and Bound Technique," Proceedings, IFIP Congress, New York, Vol. 2 (May, 1965).

(7) Benders, J. F. "Partitioning Procedures for Solving Mixed-Variables Programming Problems," Numerische Mathematic, Vol. 4, pp. 238-252 (1962).

(8) Benichou, M., Gauthier, J. M., Girodet, P., Hentges, G., Ribiere, G.,and Vincent, O. "Experiments in Mixed-Integer Linear Programming," Mathematical Pro gramming, Vol. 1, pp. 76-94 (1971).

(9) Buzby, B. R., Stone, B. J. and Taylor, R. L. "Computational Experience with Nonlinear Distribution Problems," Privately communicated to M. L. Balinski, January 1965. See M. L. Balinski, "Integer Programming: Methods, Uses, Computations," Management Science, Vol. 12, No. 3, pp. 308 (November 1965).

(10) Dakin, R. J. "A Tree-Search Algorithm for Mixed-Integer Programming Problems," The Computer Journal, Vol. 8, No. 3, pp. 250-255 (1965).

(11) Dalton, R. E. and Llewellyn, R. W. "An Extension of the Gomory Mixed-Integer Algorithm to Mixed-Discrete Variables," Management Science, Vol. 12, No. 7, pp. 569-575 (March, 1966).

(12) Dantzig, G. B. "On the Significance of Solving Linear Programming Problems with Some Integer Variables," Econometrica, Vol. 28, No. 1, pp. 30-44 (January 1960).

(13) Davis, R. E. "A Simplex-Search Algorithm for Solving Zero-One Mixed Integer Programs," Technical Report No. 16, Department of Operations Research, Stanford University (October 1969).

(14) Driebeck, N. J. "An Algorithm for the Solution of Mixed Integer Programming Problems," Management Science, Vol. 12, pp. 576-587 (1966).

(15) Ellwein, L. B. and Gray. P. "Solving Fixed Charge Location-Allocation Problems with Capacity and Configuration Constraints," AIIE Transactions, Vol. 3, No. 4, pp. 290-298 (December 1971).

(16) Frair, L. "Benders' Partitioning Algorithm for Mixed-Variable Programming Problems," Graduate Paper, University of Oklahoma (Fall 1971).

(17) Geoffrion, A. "Optimal Distribution System Design and Operation," Feasibility Study for Hunt-Wesson Foods, Inc. (June 1970).

(18) Geoffrion, A. "Generalized Benders Decomposition," Proceeding of the Symposium on Nonlinear Programming, Mathematics Research Center, University of Wisconsin, Madison, May 4-6, 1970.

(19) Geoffrion, A. and Marsten, R. E. "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science, Vol. 18, No. 9, pp. 465-491 (May 1972).

(20) Glover, F. "Integer Programming," Working Paper, Chapter 5 (September 1969).

(21) Gomory, R. E. "An Algorithm for the Mixed Integer Problem," Rand Report P-1885 (February 1960).

(22) Gomory, R. E. "A Method for the Mixed Integer Problem," Rand Report RM-2597 (1960).

(23) Gorry, C. A., Shapiro, J. F. and Wolsey, L. A. "Relaxation Methods for Pure and Mixed Integer Programming Problems," Management Science, Vol. 18, No. 15, pp. 229-239 (January 1972).

(24) Gray, P. "Mixed Integer Programming Algorithm for Site Selection and Other Fixed Charge Problems Having Capacity Constraints," Technical Report No. 6, Department of Operations Research, Stanford University (November 1967).

(25) Gray, P. "Exact Solution of the Fixed-Charge Transportation Problem," Operations Research. Vol. 19, No. 4, pp. 1529-1538 (July-August 1971).

(26) Harris, P. M. J. "An Algorithm for Solving Mixed Integer Linear Programs," Operations Research Quarterly, Vol. 15, pp. 117-132 (1964).

(27) Herve, P. "Resolution Des Programmes Lineaires a Variables Mixtes Par la Procedure Sep," METRA, Vol. 16, pp. 1-67.

(28) Hu, T. C. "Integer Programming and Network Flows," Addison-Wesley Publishing Co., Inc., pp. 255-265 (1969).

(29) Land, A. H. and Doig, A. G. "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, Vol. 28, pp. 497-520 (1960).

(30) Lasdon, L. D. "Optimization Theory for Large Systems," Macmillan Co., pp. 135-142, 370-392 (1970).

(31) Lemke, C. E. and Spielberg, K. "Direct Search Algorithm for Zero-One and Mixed Integer Programming," *Operations Research*, Vol. 15, No. 5, pp. 892-915 (September-October 1967).

(32) Little, J. D. C. "The Synchronization of Traffic Signals by Mixed-Integer Linear Programming," *Operations Research*, Vol. 14, No. 4, pp. 568-594 (July-August 1966).

(33) Mount-Campbell, C. A. "Risk, Uncertainty, Money and Integer Programming," Unpublished Paper, University of Oklahoma (Spring 1972).

(34) Muckstadt, J. A. and Wilson, R. C. "An Application of Mixed-Integer Programming Duality to Scheduling Thermal Generating Systems," *IEEE Transactions*, Vol. PAS-87. No. 12, pp. 1968-1978 (December 1968).

(35) Rebelein, P. R. "An Extension of the Algorithm of Driebeck for Solving Mixed Integer programming Problems," *Operations Research*, Vol. 16, No. 1, pp. 193-198 (January-February 1968).

(36) Shareshian, R. and Spielberg, K. "On Integer and Mixed Integer Programming and Related Areas in Mathematical Programming," IBM New York Scientific Center (1966).

(37) White, C. H. "Production Allocation with Set-Up Penalties and Concave Material Costs," *Operations Researbh*, Vol. 17, No. 6, pp. 958-972 (November-December 1969).

(38) White, W. W. "On Gomory's Mixed Integer Algorithm," Senior These, Princeton University (May 1962).

(39) Woolsey, L. A. "Mixed Integer Programming Discretization and the Group Theoretic Approach," Tech. Report 42, Operations Research Center, M.I.T. (June 1969).

(40) Woolsey, L. A. "Group Theoretic Results in Mixed Integer Programming," *Operations Research*, Vol. 19, No. 7, pp. 1691-1697 (November-December 1971).

(41) Zionts, S. "On an Algorithm for the Solving of Mixed Integer Programming Problems," *Management Science*, Vol. 15, No. 1, pp. 113-116 (September 1968).

(42) Zoutendijk, G. "Enumeration Algorithms for the Pure and Mixed Integer Programming Problem," International Symposium on Mathematical Programming, Princeton University (August 1967).