129

AN INVESTIGATION OF THE USE OF THE LAWLER-BELL ZERO-ONE ALGORITHM IN SOLVING THE WEINGARTNER MODEL OF THE CAPITAL BUDGETING PROBLEM

by

WILLIAM JAMES MAURICE THOMAS

B.S., The University of Texas, 1955
B.A., The University of Texas, 1955
M.S., Kansas State Teachers College, 1965

9587

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1972

Approved by:

Major Professor

ACKNOWLEDGEMENT

The writer is indebted to Dr. Lynn E. Bussey for suggesting this research topic and for his advice and assistance in writing it, and to Dr. L. E. Grosh for his suggestions and help with the final draft.

WILLIAM JAMES MAURICE THOMAS

Contents

Figures

Tables

Chapter 1

INTRODUCTION

The purpose of one form of the capital budgeting problem is to find a subset of a given set of indivisible projects that will maximize some function of net present value while satisfying a set of budget and technical constraints. This is a special case of the knapsack problem.

In the knapsack problem, integral multiples of unit amounts of each variable must be chosen in such a manner as to optimize some objective function of the variables, such as value, while being constrained to hold some other function or functions of the variables, such as weight or volume, within a fixed set of limits.

Weingartner (19) showed that the capital budgeting problem in the deterministic form can be formulated as a zero-one integer programming problem.

Many methods are known for the solution of knapsack type problems but there is a specific need for an efficient method to solve large capital budgeting problems after they have been formulated in the form of the Weingartner model. An algorithm developed by Lawler and Bell (10) shows promise and this paper describes an investigation of its computational efficiency and size of problem it can handle. The use of the algorithm for sensitivity testing in the Weingartner model is also described.

Weingartner Model for the Capital Budgeting Problem.

Weingartner (19) showed that the capital budgeting problem can be formulated as a zero-one integer programming problem of the following form.

$$\left.\begin{aligned}
&\text{Maximize} \sum_{j=1}^{n} b_j\, x_j \\
&\text{Subject to} \\
&(1) \quad \sum_{j=1}^{n} c_{tj}\, x_j \le B_t \qquad (t = 1,2, \ldots, m;\ j = 1,2, \ldots, n) \\
&(2) \quad x_j = 0,1 \qquad \text{for all } j\ (j = 1,2, \ldots, n)
\end{aligned}\right\} \quad [1]$$

where b_j is the present value for project j (j = 1,2, ..., n);

c_{tj} is a constant coefficient for project j in year t (t = 1,2, ...,m);

(1) is a set of constraints of the following types:

Budget constraints: $\sum_{j=1}^{n} c_{tj}\, x_j \le B_t$, $(B_t > 0)$;

Mutual exclusivity: $x_1 + x_2 + x_3 + \cdots + x_k \le 1$ $(k = 2,3, \ldots,n)$;

Contingency between projects a and b: $- x_a + x_b \le 0$

$(a, b = 1,2, \ldots,n;\ a \neq b)$;

x_j is the decision variable;

and (2) is the indivisibility constraint.

Constraints other than budget and indivisibility constraints are collectively called "technical constraints" in this report.

The capital budgeting problem as formulated in Equations [1] is in zero-one integer form. That is, if project j (j = 1,2, ..., n) is selected for execution, the decision variable, x_j, takes the value 1; otherwise it takes the value 0.

To use this deterministic model requires at least two assumptions:

(1) There is perfect knowledge of all parameters of the problem being investigated.

(2) All alternatives are known.

Weingartner showed that the model can include more than one budget constraint and more than one of each of the technical constraints. He also showed that budget constraints need not be in terms of money; e.g. other scarce resources such as manpower or fuel can be budgeted.

The model can be enlarged slightly by adding another type of technical constraint:

$$x_1 + x_2 + x_3 + \dots + x_k \geq 1 \qquad [2]$$

This is a constraint that models the requirement that at least one of the constrained projects be accepted. It occurs, for example, in "make-or-buy" problems. The numerical examples in Chapter 2 taken from Mao (13,14) include this type of constraint.

It should be noted also that a constraint can be of the form of a budget constraint but with the inequality reversed, as follows:

$$\sum c_{tj} x_j \geq B_t, \quad t = 1,2, \dots, n. \qquad [3]$$

Such a constraint models the requirement that a minimum amount of some resource, B_t, be used. This type of constraint is no different mathematically from the one with the opposite inequality and can be handled in the same manner as the other constraints in the formulation of the problem.

Methods of Solving the Problem.

After the problem has been formulated in the Weingartner model it must be solved.

The problem is essentially a combinatorial problem. Theoretically at least, it can be solved by enumerating all possible combinations of the projects and eliminating all combinations that violate any constraint. The objective function is evaluated for each of the remaining combinations and those which maximize the objective are the solutions to the problem. Since there is a finite number of projects, there is a finite number of combinations, however, the number of combinations increases in powers of two as the number of projects increases. Thus for n projects, there are 2^n combinations possible. This number is used as the number of enumerations possible in evaluating the computational efficiency of an algorithm in Chapter 2.

Numerous methods of solution based on a partial enumeration of all possible combinations of the decision variables have been tried. Weingartner (20) presents a survey of attempts to solve the problem in the specific form of equations [1]. Integer methods based on the "cutting plane" approach are rejected as too inefficient and too unpredictable. For example, Weingartner cites a problem with ten projects and three constraints, for which an integer code failed to converge in 5000 iterations.

Linear programming was tried in which the decision variables are permitted to be continuous but again Weingartner rejects this approach as not being a solution to an integer problem.

Dynamic programming has been tried by Bellman (5) and is described by Weingartner. So far, according to Weingartner, the dynamic programming method has not been very efficient with very many projects or very many constraints.

Surveys not specifically in the capital budgeting context are given by Beale (4), Balinski (2,3) and Ashour and Char (1).

Beale (4) describes a number of integer programming methods and says that they are unpredictable in computing time and number of iterations. Beale also suggests that there is no single approach suitable for all programming problems in which the decision variables are required to be integer valued.

Balinski (2,3) gives a long survey of methods of solution with many examples and a long bibliography. No general conclusions were drawn from this work that are directly applicable to the capital budgeting problem.

Ashour and Char (1) present an outline of the different approaches for solving zero-one problems, with the capital budgeting problem exemplifying one of the areas of use. They divide the different algorithms into four classes, (1) algebraic, based on cutting plane methods, (2) combinatorial, (3) enumerative, and (4) heuristic. They then present an investigation of a pseudo-boolean algorithm from Hammer and Rudeanu (7) and an adaptive binary algorithm from Salkin and Spielberg (15). They apply the pseudo-boolean and adaptive binary algorithms to capital budgeting problems having ten projects and one constraint and found that the pseudo-boolean algorithm was more efficient in economizing computing time than the adaptive binary algorithm.

Lawler and Wood (11) present an extensive survey of branch-and-bound algorithms. In a branch-and-bound algorithm, the total set of possible combinations is partitioned into subsets or branches by a logical branching procedure for the selection of branches. A bounding procedure is used to determine if a selected branch is currently optimal and feasible.

Lawler and Bell (10) develop a branch-and-bound algorithm for minimization of zero-one problems. The branches are based on a vector partial ordering, and branch selection and bounding are accomplished by three rules for skipping based on the partial ordering. This algorithm together with the description of the partial ordering is described in detail in Chapter 2.

Lawler and Bell do not consider the capital budgeting problem specifically but do apply the algorithm to a variety of problems. They report that it appears more efficient than the other methods they studied. The largest problem they studied had 21 variables. Lawler and Bell also reported that the order in which the projects are taken in a problem affects the computation time but they give no conclusion regarding an optimal order.

Mao (13) and Mao and Wallingford (14) use the Lawler-Bell algorithm specifically for the capital budgeting problem. They give a linear transformation, described in Chapter 2 of this paper, which transforms the maximization problem of capital budgeting into the minimization form needed by the Lawler-Bell algorithm. Mao reports that the algorithm is efficient for problems with as many as 15 projects and 15 constraints.

Weingartner (20) reports on the extension of the original model to a quadratic form which can be used with the probabilistic form of the capital budgeting problem but does not give a method of solution.

Mao and Wallingford (14) give an extension of the Lawler-Bell algorithm to the probabilistic case by modifying the rules for skipping and adding another. They reported that the extension has been used with problems as large as 15 projects and 15 constraints. The extension is described in more fully in Chapter 2 of this report.

Chapter 2

RESEARCH

Research Objective

The objective of this research was to investigate the computational efficiency of the Lawler-Bell algorithm for the Weingartner model of the capital budgeting problem. Both the deterministic case and the extension to the probabilistic case were considered. The effect of problem size (number of projects and constraints) on computing time was investigated and a method of sensitivity testing was developed. A second objective was to make the algorithm available for academic courses in capital budgeting or related areas.

Lawler-Bell Algorithm.

The linear transformation used by Mao and Wallingford (14) and mentioned in Chapter 1 follows:

Substitute $x_j = 1 - x'_j$ into the objective function, which then becomes

$$\text{Minimize} \sum_{j=1}^{n} b_j x'_j - \sum_{j=1}^{n} b_j \qquad [4]$$

This function is monotonically nondecreasing.

The same substitution must be applied to the constraints which take the following general form.

$$\sum_{j=1}^{n} c_{tj} x'_j - \sum_{j=1}^{n} c_{tj} \leq B_t \qquad (t = 1,2, \ldots, m) \qquad [5]$$

Since some values of c_{tj} may be negative, the transformed constraints may not be monotonic. However, the constraints are linear and any linear function can be written as the difference between two monotonically nondecreasing functions.

The transformed indivisibility constraint is merely the complement of x_j, since x'_j is clearly zero when x_j is one and one when x_j is zero.

The algorithm due to Lawler and Bell and mentioned in Chapter 1 can now be presented. After using the linear transformation above and making some changes in notation to simplify writing, the minimization form of Equations [1] can be written as follows.

$$\left.\begin{array}{ll} \text{Minimize} & g_0(\underset{\sim}{x}) \\ \text{Subject to} & g_{11}(\underset{\sim}{x}) - g_{12}(\underset{\sim}{x}) \geq 0 \\ & g_{21}(\underset{\sim}{x}) - g_{22}(\underset{\sim}{x}) \geq 0 \\ & \vdots \\ & g_{m1}(\underset{\sim}{x}) - g_{m2}(\underset{\sim}{x}) \geq 0 \end{array}\right\} \quad [6]$$

where $\underset{\sim}{x} = (x_1, x_2, x_3, \ldots, x_n)$, a vector of project "complements"

$x_j = 1, 0 \quad (j = 1,2,3, \ldots, n)$

g_0 is a vector of coefficients of the objective function,

g_{j1} is a vector of positive coefficients and constants for constraint j,

g_{j2} is a vector of negative coefficients and constants for constraint j.

Each constraint is now the difference of two monotonic nondecreasing functions.

Equations [6] have a special mathematical structure which is exploited by the Lawler-Bell algorithm. Each set of projects can now be expressed as a vector. (Note that because of the transformation to the minimization space, we are now talking about the "complements" of the original projects.) An isomorphism exists between the transformed vectors of projects and the binary number system. For example, a vector representing non-selection of projects 2 and 3 may be written $\underset{\sim}{x} = (0,1,1,0,0)$ with project number indexing commencing from the left. This vector is isomorphic to the binary number 1100, which has a comparable base ten value of 12. Let $n(\underset{\sim}{x}) = 12$ denote the numerical value in base 10.

Let the symbol $\preceq$, called "under", be defined as follows:

$\underset{\sim}{x} \preceq \underset{\sim}{y}$ if and only if $x_j \leq y_j$ for all j, where $\leq$ means "less than or equal" and x_j, y_j refer to the jth components of $\underset{\sim}{x}$ and $\underset{\sim}{y}$.

It is well known that the relationship developed by $\underset{\sim}{x} \preceq \underset{\sim}{y}$ results in a partial ordering. That is, this relation is reflexive, antisymmetric and transitive.

If $\underset{\sim}{x} \preceq \underset{\sim}{y}$, then $n(\underset{\sim}{x}) \leq n(\underset{\sim}{y})$, but the converse is not necessarily true. For example, let $\underset{\sim}{x} = (0,0,0,1)$, $\underset{\sim}{y} = (0,1,1,1)$ and $\underset{\sim}{z} = (0,1,1,0)$. Then each component x_j of $\underset{\sim}{x}$ is less than or equal to its corresponding component of $\underset{\sim}{y}$, the relation $\underset{\sim}{x} \preceq \underset{\sim}{y}$ is satisfied and $[n(\underset{\sim}{x}) = 1] \leq [n(\underset{\sim}{y}) = 7]$. However, the rightmost component of $\underset{\sim}{x}$ is greater than the rightmost component of $\underset{\sim}{z}$ so $\underset{\sim}{x} \npreceq \underset{\sim}{z}$ even though $[n(\underset{\sim}{x}) = 1] \leq [n(\underset{\sim}{z}) = 6]$.

Two vectors related by $\preceq$, such as $\underset{\sim}{x}$ and $\underset{\sim}{y}$ above, are said to be <u>comparable</u>. Two vectors not related by $\preceq$ such as $\underset{\sim}{x}$ and $\underset{\sim}{z}$ above, are said to be <u>noncomparable</u>.

In this paper, the importance of the partial ordering lies in the fact that if g is a monotonic *nondecreasing* function and $\underset{\sim}{x} \preceq \underset{\sim}{y}$, then $g(\underset{\sim}{x}) \leq g(\underset{\sim}{y})$. This is well known and is shown in Johnson (9).

Now for any vector, $\underset{\sim}{x}$, any other vector with a greater numerical value must either be above $\underset{\sim}{x}$ in the partial ordering or noncomparable in the partial ordering. Denote by $\underset{\sim}{x}^*$ the first noncomparable vector with a higher numerical value than $\underset{\sim}{x}$ has, and let $\underset{\sim}{x}^* - 1$ be the vector just below $\underset{\sim}{x}^*$ in numerical value. Then, $\underset{\sim}{x}$ and $\underset{\sim}{x}^* - 1$, as well as all vectors between them, are comparable.

Some additional notation is now needed. g_0, with a single subscript, refers to the objective function; g_{j1}, with a double subscript, refers to a constraint j (j = 1,2, ..., m) and subscript 1 refers to the vector of terms in constraint j with positive coefficients and subscript 2 refers to the vector of the terms in the constraint j with negative coefficients. (Here, j now refers to the number of the subscript and not to the project number.) The vector $\underset{\sim}{x}$ is the one currently being considered by the algorithm and $\underset{\sim}{\hat{x}}$ is the current optimal vector, i.e., the best feasible vector already found. The subscript i on x refers to the "non-project" number (the component in vector notation). Note that since the transformation was made to the minimization space, these are "complements" of the original projects and the final solution must be inversely transformed to the original maximization space.

Lawler and Bell (10) give the following procedure for calculating $\underset{\sim}{x}^*$ for a given $\underset{\sim}{x}$. The vectors must be treated as binary numbers.

1) Calculate $(\underset{\sim}{x} - 1)$ by subtracting 1 from $\underset{\sim}{x}$.

2) Determine $(\underset{\sim}{x}^* - 1)$ by letting each element $(\underset{\sim}{x}^* - 1)_j$ equal zero if <u>both</u> elements x_j and $(x - 1)_j$ are equal to zero. Otherwise, let $(x^* - 1)_j$ be equal to 1.

3) Find $\underset{\sim}{x}^*$ by adding 1 to $(\underset{\sim}{x}^* - 1)$.

Since $\underset{\sim}{x}^*$ is the first noncomparable vector following $\underset{\sim}{x}$, all vectors between must be comparable and $\underset{\sim}{x} \preceq \underset{\sim}{x} + 1 \preceq \underset{\sim}{x}^* - 1$. If g is a monotonic nondecreasing function, then in this interval its smallest value is $g(\underset{\sim}{x})$ and its largest is $g(\underset{\sim}{x}^* - 1)$.

Lawler and Bell now give three rules for skipping through branches of vectors where each set of comparable vectors determines a branch.

Rule 1: If $g_0(\hat{\underset{\sim}{x}}) \leq g_0(\underset{\sim}{x})$, skip to $\underset{\sim}{x}^*$.

Explanation: Since $\underset{\sim}{x}$ minimizes the value of the objective function, g_0, in the range between $\underset{\sim}{x}$ and $\underset{\sim}{x}^*$, it is clear that no vector following $\underset{\sim}{x}$ but preceeding $\underset{\sim}{x}^*$ in the numerical order will be less costly than $\hat{\underset{\sim}{x}}$.

Rule 2: If $g_0(\hat{\underset{\sim}{x}}) \geq g_0(\underset{\sim}{x})$ and $\underset{\sim}{x}$ is feasible, set $\hat{\underset{\sim}{x}}$ equal $\underset{\sim}{x}$, and skip to $\underset{\sim}{x}^*$.

Explanation: If $\underset{\sim}{x}$ reduces the value of g_0, and moreover, is feasible, we know it is a possible solution. In fact, since $\underset{\sim}{x}$ minimizes g_0 in the range between $\underset{\sim}{x}$ and $\underset{\sim}{x}^*$, it is the best solution that can be found in this range.

Rule 3: If $g_0(\hat{\underset{\sim}{x}}) \geq g_0(\underset{\sim}{x})$ and $g_{j1}(\underset{\sim}{x}^* - 1) - g_{j2}(\underset{\sim}{x}) < 0$ for any $j(j = 1,2, \ldots, m)$, skip to $\underset{\sim}{x}^*$.

Explanation: If $\underset{\sim}{x}$ reduces the value of g_0, but is infeasible, then there are two possibilities. First, it is possible that all vectors between $\underset{\sim}{x}$ and $\underset{\sim}{x}^*$ are infeasible. This would be the case if $g_{j1}(\underset{\sim}{x}^* - 1) - g_{j2}(\underset{\sim}{x}) \not\geq 0$ for any j, since x* - 1 maximizes the value of a monotonically nondecreasing function in this range, and since $\underset{\sim}{x}$ minimizes the value of such a function (and therefore minimizes its negation). The use of both $\underset{\sim}{x}^* - 1$ and $\underset{\sim}{x}$ gives the largest possible value for the preceeding expression in the relevant range. If even this maximum value is not enough to satisfy the nonnegativity constraint, then no single vector between $\underset{\sim}{x}$ and $\underset{\sim}{x}^*$ will be feasible. Second, it is possible that some vectors between x and x* are feasible. In that case, no skipping is permitted.

If none of the rules apply, no skipping is permitted. Flow charts of the logic and computations are given in Figures 1 and 2.

<u>Extension of the Lawler-Bell Algorithm to the Probabilistic Case.</u>

Mao and Wallingford (14) give an extension of the Lawler-Bell algorithm to a probabilistic case. This case requires that the expected value of the project present value, $E(b)_j$, be used in the selection in lieu of the present value, b_j, in the deterministic algorithm. Next, suppose that the variances and covariances of the individual project present values, with each other pairwise, are known and that the constraints are still independent of one another.

Let A denote the risk aversion coefficient, which may be taken to be a value obtained from the decision-maker's strictly concave utility

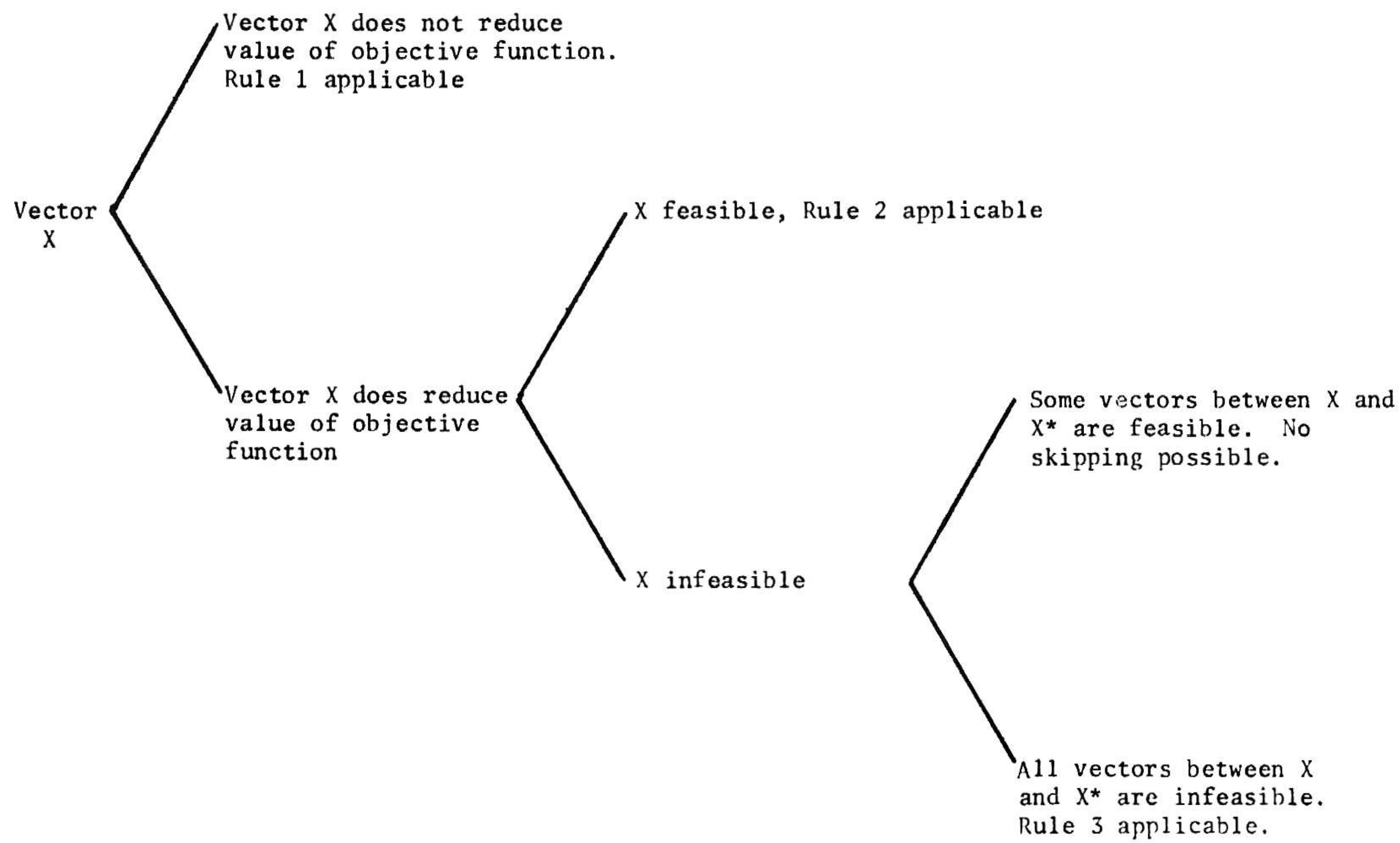

Figure 1

Logic Flow Chart of Lawler-Bell Algorithm

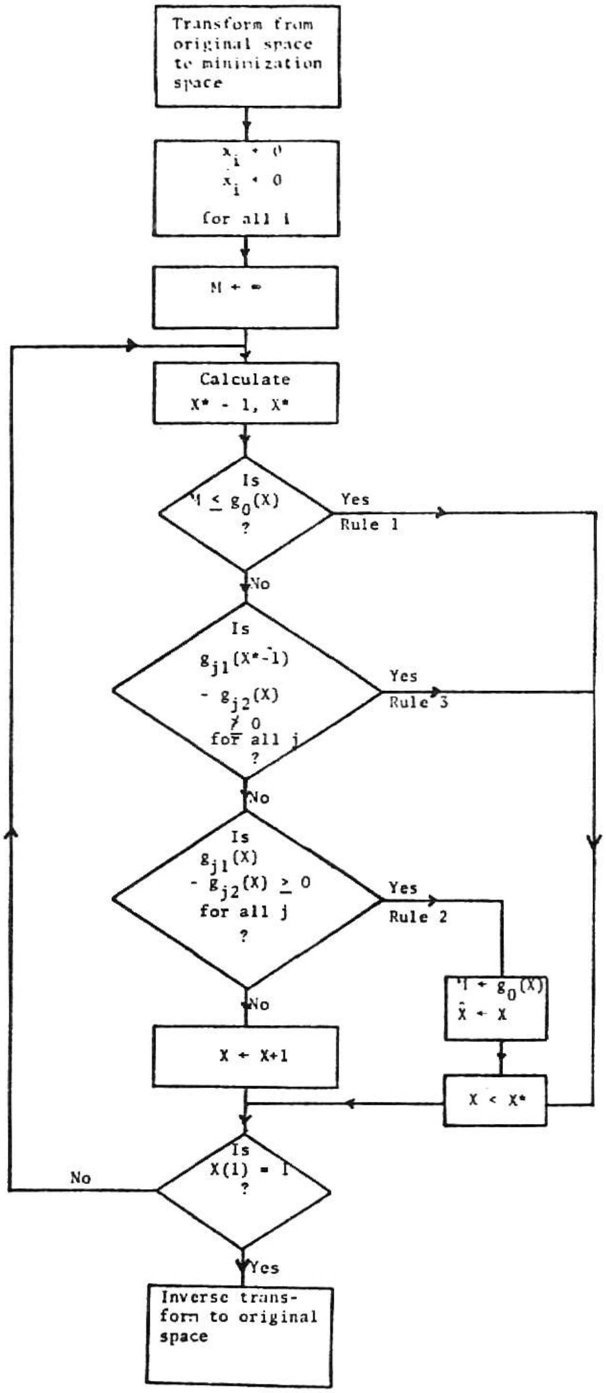

Figure 2

Computation Flow Chart of Lawler-Bell Algorithm function that expresses numerically his disinclination to assume risk, where risk is measured by project present value variance, $C(b)_j$.

Mao and Wallingford (14) write the objective function for the probabilistic case as

$$\text{Maximize} \quad \sum_{j=1}^{m} x_j E(b)_j - A \sum_{j=1}^{m} [x_j E(b)_j]^2 - A \left[\sum_{i=1}^{m} \sum_{j=1}^{m} x_i x_j C(b)_{ij} \right] \qquad [7]$$

where $E(b)_j$ is the expected value of present value for project j, (j = 1, 2, ..., m).

$C(b)_{ij}$ is the variance of the present value of project j, if i = j; and the covariance between present values of projects i and j (i,j = 1, 2, ..., m) if $i \neq j$ for all pairs i and j.

It appears that Mao and Wallingford assumed that the decision-maker's utility function is quadratic and thus the form of the second term in Equation [7].

In a later reference, Mao (13) apparently assumed that the decision-maker's utility function is of exponential form and the second term in Equation [7] disappears. The later form of the objective function is used here and is taken to be

Maximize

$$\sum_{j=1}^{m} x_j E(b)_j - A \left[\sum_{i=1} \sum_{j=1} x_i x_j C(b)_{ij} \right]. \qquad [8]$$

This objective function is no longer monotonic, since covariances may be negative, but since the cross product term $x_i x_j$ is 1 when x_i and

x_j are both 1 and 0 otherwise, it is still linear and thus can be written as the difference between two monotonic nondecreasing functions. After applying the linear transformation described in the description of the deterministic algorithm (which must now be applied to the variance/covariance matrix as well as to the objective function and constraints) to transform the probabilistic problem to the minimization space, and after separating positive and negative terms, Equation [8] becomes

Minimize $$g_0'(\underset{\sim}{x}) - g_0''(\underset{\sim}{x}) \quad [9]$$

where g' is a vector of positive coefficients and constants of the transformed Equation [8]

and g'' is a vector of negative coefficients and constants of the transformed Equation [8].

Now in the range between $\underset{\sim}{x}$ and $\underset{\sim}{x}*$ in the partial ordering given above, g_0' is minimized at $\underset{\sim}{x}$ and g_0'' is maximized at $\underset{\sim}{x}* - 1$. Therefore $g_0'(x) - g_0''(\underset{\sim}{x}* - 1)$ takes on its smallest possible value in the range between $\underset{\sim}{x}$ and $\underset{\sim}{x}*$. If this is still greater than $g_0'(\hat{\underset{\sim}{x}}) - g''(\hat{\underset{\sim}{x}})$, then no new minimum will be found in this range, and we can skip to $\underset{\sim}{x}*$. Therefore Mao modifies Rule 1 to read

Rule 1': If $g_0'(\hat{\underset{\sim}{x}}) - g_0''(\hat{\underset{\sim}{x}}) \leq g'(\underset{\sim}{x}) - g''(\underset{\sim}{x}* - 1)$ skip to $\underset{\sim}{x}*$.

If Rule 1' does not apply, then some vector in the interval between $\underset{\sim}{x}$ and $\underset{\sim}{x}* - 1$ may reduce the value of the objective function. In this case, use Rule 3 from the deterministic case, which Mao now calls the second rule for the probabilistic case:

Rule 2': Same as Rule 3 in the deterministic algorithm.

If Rule 2' does not apply, some vectors in the range between $\underset{\sim}{x}$

and $\underset{\sim}{x}$* may be feasible. Test this with:

<u>Rule 3'</u>: If $g_{j1}(x) - g_{j2}(x) \geq 0$ for any j (j = 1,2, ..., m) continue the enumeration with x + 1.

If a feasible vector $\underset{\sim}{x}$ is found, then Rule 4, which is an extension of Rule 2 in the deterministic algorithm must be used:

<u>Rule 4</u>: If $g_0'(\underset{\sim}{x}) - g_0''(\underset{\sim}{x}) \leq g_0'(\hat{\underset{\sim}{x}}) - g''(\hat{\underset{\sim}{x}})$, let $\hat{\underset{\sim}{x}}$ equal $\underset{\sim}{x}$ and continue with $\underset{\sim}{x}$ + 1.

Flow charts for the logic and computations of the probabilistic algorithm are given in Figures 3 and 4. The flow chart as shown in references (13,14) has a misprint in the block representing $M \leftarrow g'(x)-g''(x)$ which has been corrected in this paper in Figure 4. This follows from a literal reading of Rule 4, above.

<u>Procedure</u>.

Three computer programs were written in FORTRAN IV. The first, called the Strong Deterministic form in this report, was written to use the original Lawler-Bell algorithm. This algorithm finds only one optimal feasible vector of projects. A second program, called the Weak Deterministic form in this paper, was written with the rules of the algorithm weakened to find multiple optimal solutions if they exist. The third program, called the Probabilistic form in this report, was written to use the extension to the probabilistic case. The descriptions of the programs are given in the next section and the programs themselves are given in Appendix A.

After the programs were written and debugged, machine object decks were made to reduce compiling time. The object decks were used for the

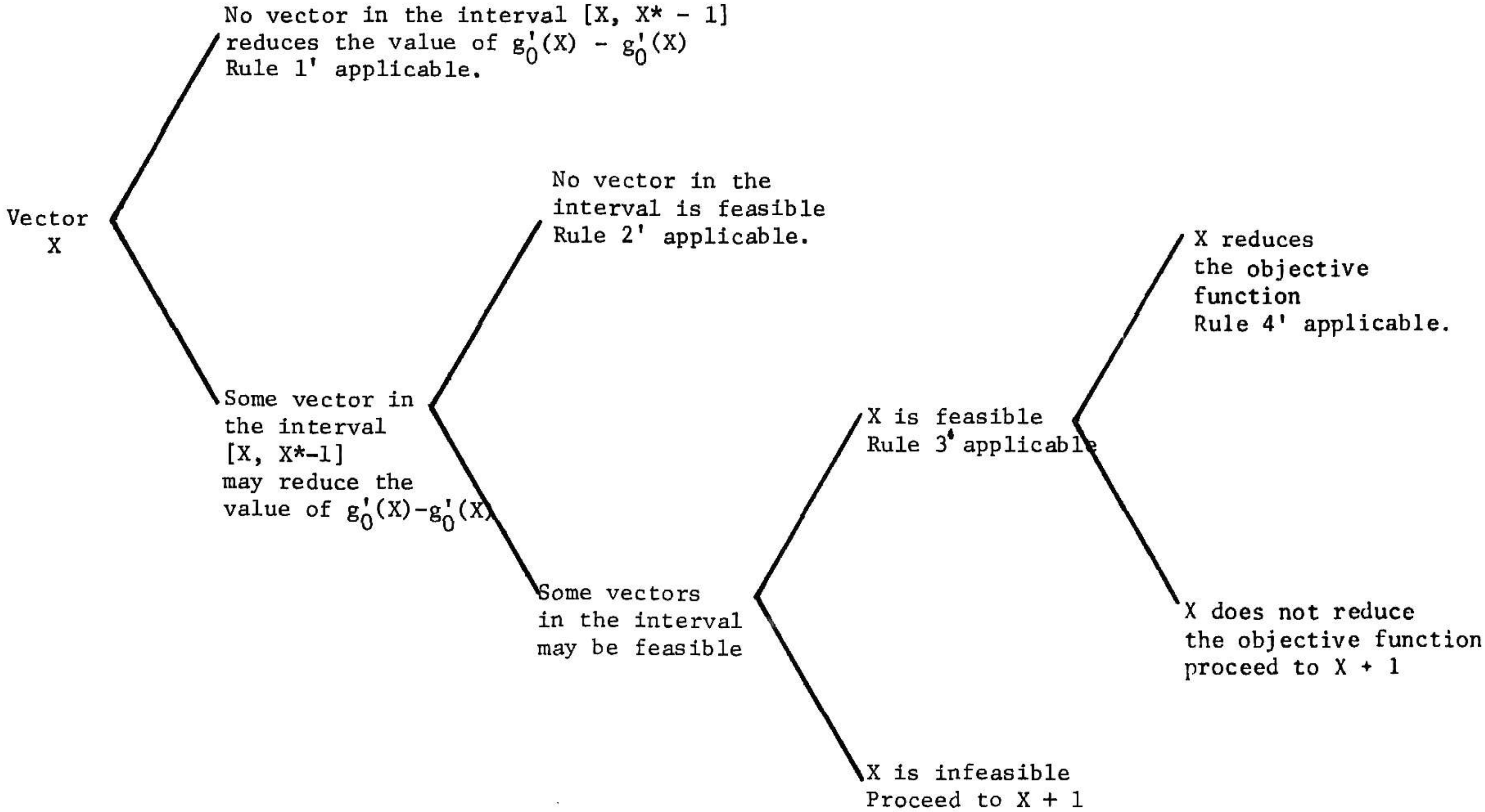

Figure 3

Logic Flow Chart for Probabilistic Case.

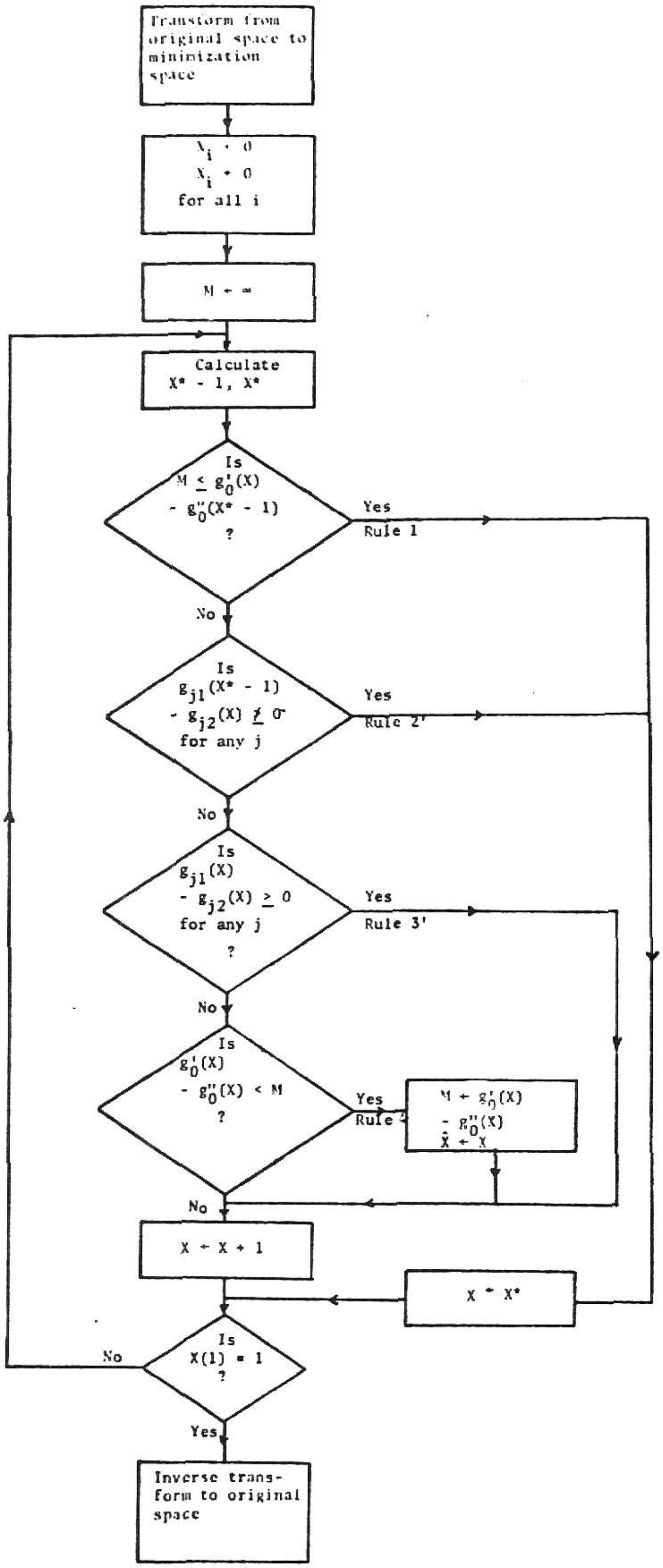

Figure 4

Computation Flow Chart for Probabilistic Case sensitivity tests which are described below. The problems given below were run with the object decks to determine the execution time and effectiveness of the skipping rules.

All programs were written to use with the capital budgeting problem and some generality in the use of the algorithm was sacrificed.

Since the source decks and object decks are to be made available for use in solving capital budgeting and related problems by other interested persons, an instruction booklet has been written for the preparation of data decks. It is presented as Appendix B.

Descriptions of the Programs.

A. Strong Deterministic Form.

This computer program is intended to converge as rapidly as possible and uses the Lawler-Bell algorithm in its original form. If there are multiple solutions, this form of the algorithm finds only the first one since skipping is done by the "less-than-or-equal" decisions (tests) in Rules 1 and 2.

The program is written to receive data conforming to the conventional Weingartner deterministic model, which is a maximization problem, and transforms the data internally within the program into the form needed for the Lawler-Bell algorithm, which solves a minimization problem. This is done by the linear transformation described earlier in this chapter and the transformed values are stored in arrays needed for the Lawler-Bell algorithm. Printouts are made of the transformed (minimization) arrays.

The Lawler-Bell algorithm itself follows the transform step in the program. Since the algorithm terminates when the leftmost digit (a signalling digit) in the solution vector equals one, all projects are displaced one position to the right with the leftmost digit representing a dummy project. An IF statement sends the computation to the end when this digit becomes one. The current vector is initially set to zero and the current objective value set very large. The vectors $\underset{\sim}{x}^*$ and $\underset{\sim}{x}^* - 1$ referred to in the description of the algorithm are designated XSTAR and XSTAR1, respectively, in the program.

The rules for skipping use the vectors XSTAR (the next noncomparable vector) and XSTAR1, so the next stage in the program is a routine to compute these vectors for a given current vector. The computation is based on the method given in the description of the algorithm.

Routines for the rules of skipping and feasibility follow. Most of these are included in the main program but those that are needed at more than one place are written as subroutines and called when needed.

The vectors under current consideration during execution are printed out so that the progress through the algorithm can be seen and solutions printed as they occur. When a solution is found, the current solution is updated.

A subroutine is provided to invert the minimization transformation so that when a solution is found it is printed in both the minimization space of the algorithm and in the original maximization space.

A subroutine is provided to print out the final result. Here, the projects comprising the optimal solution vector and the value of the objective, both in the original maximization space, are printed. In other words, the final answer to the problem is printed out with the answer stated in terms of the inputted maximization problem.

The vectors were dimensioned in this program for twenty-five projects, or twenty-four real projects plus one dummy project as mentioned above, and for twenty-five constraints. If a problem does not need all twenty-five values, the program accepts fewer, starting at the left for projects and from the top for constraints. For example, for a problem with six projects, the first seven positions from the left of the project vector are used and the remaining ones are zeroed internally. If there are five constraints, they are taken in order and the rest of the array is not used. The input/output routines are formatted accordingly. A different format is used for budget constraints than for technical constraints to simplify data input. The program will accept up to five budget constraints as part of the twenty-five constraints mentioned above.

B. Weak Deterministic Form.

This program is intended to be used after the optimal solution to a problem has been found with the Strong Deterministic form. It is identical to the Strong Deterministic form except as described below. The Lawler-Bell rules for skipping have been weakened so that skipping occurs only for a strict inequality. Thus if there are multiple solutions, the program can find them.

To reduce redundant computation, the initial vector and objective values in this program are set to the optimal solution already found by the Strong form program. This is done by means of a data card which is added to the data deck for the problem. This resetting of the current solution is analogous to the practice of resetting the initial conditions in simulation.

The inverse transform subroutine was modified to store up to five optimal solutions transformed to the original problem space (maximization). If there are more than five solutions, the skipping continues as in the Strong form.

The output subroutine was modified to print out the solutions stored by the inverse transform subroutine and print out a comment that there are more than five solutions when that fact exists.

If there are more than five solutions, they can be found by changing the card for the initial vector. After running the problem as described above, replace the initial vector on the data card with the fifth vector in the set of solutions and run it again. The iteration now begins at the new value and continues the search.

C. Probabilistic Form.

This program was written for the extension of the Lawler-Bell algorithm to the probabilistic case. The basic algorithm is the same but some modifications are needed to provide for the variance and covariance terms.

The input is modified to permit insertion of values of the risk aversion coefficient and to accept the variance/covariance matrix. After these values are read in, they are then transformed to the minimization space of the Lawler-Bell algorithm in a manner similar to that used in transforming the objective function and constraints. The routines for the rules are modified and Rule 4 is added as described in the description of the algorithm. A subroutine to find the variance term for use in the rules is also needed. The transformed variance/covariance matrix is printed out.

In order to investigate the behavior of the selection process for different values of the risk aversion coefficient, an array was set up to store the risk aversion coefficient values read in. The program selects the first and its value is printed and the computation is made. The results are printed as in the deterministic case. This is then repeated for the other values of the coefficient. For a risk aversion coefficient of zero, the Strong Deterministic or Weak Deterministic form should be used, to economize execution time in the computer.

Problems Used.

Three basic problems were used. Each was used in both the deterministic and the probabilistic form making six problems in all.

The first problem was taken from Mao (13), pages 253-255 and 295-296, and is given below.

$$
\begin{array}{lrl}
\text{Maximize} & z = & 10x_1 + 20x_2 + 5x_3 + 3x_4 + 2x_5 \\
\text{Subject to} & & 20x_1 + 30x_2 + 15x_3 + 10x_4 + 5x_5 \leq 65 \\
& & 20x_1 + 15x_2 + 5x_3 + 7x_4 + 4x_5 \leq 46 \\
& & 500x_1 + 1000x_2 + 100x_3 + 50x_4 + 20x_5 \geq 500 \\
& & 500x_1 + 1000x_2 + 100x_3 + 50x_4 + 20x_5 \leq 1100 \\
& & x_1 + x_2 \leq 1 \\
& & - x_2 + x_3 \leq 0 \\
& x_j = 0, 1 &
\end{array}
$$

This problem was used in the probabilistic case by adding the following variance/covariance matrix

Project No.	1	2	3	4	5
1	1.1	3.0	0.1	0	0.5
2	3.0	36.1	2.0	0	0
3	0.1	2.0	1.0	0	0.5
4	0	0	0	0	0
5	0.5	0	0.5	0	1.0

together with risk-aversion coefficients $A = 0, 0.1, 0.3, 1.1$.

The printouts of solutions of this problem are included with the programs in Appendix A.

The second problem was taken from Mao and Wallingford (14) and is given below.

Maximize

$$z = 757x_1 + 825x_2 + 987x_3 + 350x_4 + 596x_5 + 650x_6 + 1420x_7 + 1425x_8$$

Subject to:

$$
\begin{array}{rrrrrrrrl}
7x_1 + & 35x_2 + & 20x_3 + & 12x_4 + & 65x_5 + & 60x_6 + & 20x_7 + & 5x_8 & \le 100 \\
5x_1 + & 15x_2 + & 30x_3 + & 10x_4 + & 7x_5 + & 15x_6 + & 50x_7 + & 7x_8 & \le 70 \\
5x_1 + & 12x_2 + & 2x_3 + & 10x_4 + & 4x_5 + & 2x_6 + & 10x_7 + & 7x_8 & \le 30 \\
5x_1 + & 4x_2 + & & 10x_4 + & 4x_5 + & 2x_6 + & 5x_7 + & 7x_8 & \le 15 \\
5x_1 + & 4x_2 + & & 6x_4 + & 4x_5 + & 2x_6 + & & 7x_8 & \le 15 \\
2x_1 + & 4x_2 + & 8x_3 + & 3x_4 + & 4x_5 + & 2x_6 + & & 7x_8 & \le 15 \\
x_1 & & - & x_4 & & & & & \ge 0 \\
x_1 + & x_2 + & x_3 & & & & & & \le 1 \\
& & & x_4 + & x_5 + & x_6 & & & \le 1 \\
& & & & & & x_7 + & x_8 & \le 1 \\
x_1 + & x_2 + & x_3 & & & & & & \ge 1 \\
& & & x_4 + & x_5 + & x_6 & & & \ge 1 \\
& & & & & & x_7 + & x_8 & \ge 1
\end{array}
$$

$x_j = 0 \text{ or } 1$

The following variance/covariance matrix and values of risk aversion coefficients were used with this problem in the probabilistic program.

Project No.	1	2	3	4	5	6	7	8
1	2500	0	0	1800	-2100	-3300	-600	-990
2	0	6400	0	-2900	3400	4800	960	2000
3	0	0	12000	-3960	6500	10000	1200	3000
4	1800	-2800	-3960	3600	0	0	-800	-1000
5	-2100	3400	6500	0	4900	0	1000	1500
6	-3300	4800	10000	0	0	14000	1200	3000
7	-600	960	1200	-800	1000	1200	400	0
8	-990	4500	3000	-1000	1500	3000	0	1000

$A = 0,\ 1.5\ (10^{-4}),\ 2\ (10^{-4})$.

The third problem was constructed specially for this report:

$$\text{Maximize} \quad z = 255x_1 + 575x_2 + 80x_3 + 325x_4 + 560x_5 + 535x_6 + 115x_8$$
$$+ 335x_8 + 324x_9 + 358x_{10} + 125x_{11} + 440x_{12} + 550x_{13} + 560x_{14} + 45x_{15}$$

Subject to:

$$145x_1 + 250x_2 + 65x_3 + 130x_4 + 200x_5 + 310x_6 + 95x_7$$
$$+ 155x_8 + 140x_9 + 150x_{10} + 95x_{11} + 165x_{12} + 175x_{13} + 185x_{14} + 40x_{15} \leq 1000$$

$$100x_1 + 200x_2 + 120x_4 + 200x_5 + 220x_6$$
$$+ 125x_8 + 100x_9 + 125x_{10} + 125x_{12} + 145x_{13} + 150x_{14} \leq 670$$

$$100x_2 + 150x_5$$
$$+ 75x_9 + 25x_{10} - 115x_{12} + 125x_{13} + 140x_{14} \leq 250$$

$$100x_1 + 160x_2 + 60x_3 + 95x_4 + 165x_5 + 140x_6 + 75x_7$$
$$+ 121x_8 + 130x_9 + 124x_{10} + 80x_{11} + 150x_{12} + 135x_{13} + 135x_{14} + 55x_{15} \leq 800$$

$$x_2 + x_5 + x_6 \leq 1$$
$$-x_2 + x_3 \leq 0$$
$$x_8 + x_9 + x_{10} \leq 1$$
$$x_{12} + x_{13} + x_{14} \leq 1$$
$$-x_{12} + x_{15} \leq 0$$
$$x_j = 1,0$$

The variance/covariance matrix is given in Table 1.

The values for Risk Aversion Coefficient are: A = 0, 1, 10.

Table 1

Variance/Covariance Matrix for Problem 3

oject .	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	20.0	15.0	.1	-10.0	.5	2.5	1.0	.5	.5	-15.0	-2.0	7.5	.5	.5	.1
2	15.0	30.0	.1	12.5	.4	2.5	2.0	.4	.5	-20.0	-1.0	5.0	.1	.5	.1
3	.1	.1	.1	.1	0	0	0	0	0	0	0	0	0	.5	.1
4	-10.0	12.5	.1	25.1	.5	-2.5	2.0	.4	.1	7.5	1.0	2.0	0	.5	.1
5	.5	.4	0	.5	2.5	5.0	3.0	.5	.5	.1	.1	.1	0	0	.1
6	2.5	2.5	0	-2.5	5.0	7.5	-5.0	1.0	.5	0	0	.1	0	0	.1
7	1.0	2.0	0	2.0	3.0	-5.0	7.5	-2.0	7.5	.5	.5	7.5	0	0	.1
8	.5	.4	0	.4	.5	1.0	-2.0	5.0	5.0	0	.1	.1	0	0	.1
9	.5	.5	0	.1	.5	.5	2.5	5.0	7.5	0	-.5	.5	.1	.1	.1
10	-15.0	-20.0	0	7.5	.1	0	.5	0	0	30.0	15.0	-7.5	.1	.1	.1
11	-2.0	-1.0	0	1.0	.1	0	.5	.1	-.5	15.0	10.0	20.0	.4	-.5	.1
12	7.5	5.0	0	2.0	.1	.1	7.5	.1	.5	-7.5	20.0	35.0	.5	-.1	.1
13	.5	.1	0	0	0	0	0	0	.1	.1	.4	.5	1.0	.1	.1
14	.5	.5	.5	.5	0	0	0	0	.1	.1	-.5	-.1	.1	1.0	.1
15	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1

Sensitivity Tests

This part of the report is concerned with the effect of using different limits on the budget constraints. After an optimal solution has been found, it is sometimes desirable to know how much a somewhat higher or lower budget would affect the objective value. That is, would increasing the budget say 10%, produce 10% more present value in the objective, or more or less than that? The intent here is to make an incremental analysis of objective value sensitivity to budget changes.

In this report, the sensitivity tests were made using the third problem and varying the first three budget limits by ± 20% and ± 40% from the initially assumed value. The fourth budget constraint can be thought of as budgeting something besides money (such as manpower) and was held constant.

The budget limits for the sensitivity tests are shown below.

Percent Change	Budget Constraint No. 1	Budget Constraint No. 2	Budget Constraint No. 3	Budget Constraint No. 4
-40%	600	410	150	800
-20%	800	540	200	800
Initial value	1000	670	250	800
+20%	1200	800	300	800
+40%	1400	930	350	800

Chapter 3

RESULTS AND CONCLUSIONS

Iteration and Computation Time Results.

The problems described in Chapter 2 were solved with the programs for the Strong Deterministic Form and for the Probabilistic Form of the algorithm. The number of iterations necessary to solve the problem and the number of the iteration at which the optimal solution occured were compared to the total number of iterations possible (2^n) and the computer execution time was noted.

The optimal solution often occurs early in the iterations but to determine that it is optimal requires searching a larger number of vectors.

The number of iterations and computer execution time for the sensitivity tests were also used, making a larger set of problems for comparison.

The results of these observations are shown in Table 2. A separate set of results is given for the Weak Deterministic Form program, since this program does not process the iterations preceding the first optimal solution. For this program, the results are shown in Table 3 for total iterations and execution time. Multiple optimal solutions are shown when they exist. The third problem in Chapter 2 was not tested with this program in order to conserve computing time.

Probabilistic Results.

The results for the deterministic case were taken as the results

Table 2

Iteration and Computation Time Results, Strong and Probabilistic Forms

Problem Size and Type	Total Number of Iterations Possible	Number of Iterations Needed	Computer Execution Time, Minutes (IBM 360/50)	Iteration Giving Optimal Solution
5x6 Deterministic	$2^5 = 32$	14	.12	12
5x6 Probabilistic				
A = .1		18	.12	15
A = .3		18		8
A =1.1		16		9
8x13 Deterministic	$2^8 = 256$	75	.12	56
8x13 Probabilistic				
A = .01		77	.36*	62
A = .1		75		61
A = .5		75		57
15x9 Deterministic	$2^{15} = 32,768$	3416	2.1	187
15x9 Probabilistic				
A = 1.0		3338	8.04	1967
A =10.0		2610	6.12	1391
15x9 Determinisitc	$2^{15} = 32,768$			
+ 20% Bud Lim		3092	1.92	539
- 20% Bud Lim		3343	2.12	961
+ 40% Bud Lim		3150	1.98	539
- 40% Bud Lim		2261	1.39	1736

* The results for each of these entries were obtained on one run so a breakdown of execution time for each value of A is not available.

Table 3

Solution Sets Using Weak Deterministic Form.

Problem Size	Solution Project Set	Objective Value	Remarks
5x6 Deterministic	2,3	25	Optimal Feasible Solution No. 1
	2,4,5	25	Alternate Optimal Solution No. 2
8x13 Deterministic	2,6,8	2900	This is the only Optimal Feasible Solution.

for risk aversion coefficient A = 0. The results for the other values for A were obtained using the Probabilistic Form program and the change in objective value and shift in the selection of projects to those with smaller values of variance/covariance was observed. These results are shown in Table 4. It should be observed that the values for A were scaled to the values for the variance/covariance matrix and the two sets of values set at such a range that they would fit the formats already chosen for the program. This is permissible when the exponential assumption is made as mentioned in the discussion of results below.

Sensitivity Tests Results.

The effects of changing the budget limits were investigated. First, the limits as given in the third problem of Chapter 2 were taken as the base values. The optimal solution and objective value were observed along with computer execution time. The budget limits were changed + 20% and the optimal solution and objective value along with computer execution time were observed. This was repeated for - 20% and for ± 40%. The results are shown in Table 5. A graph of these results is given in Figure 5.

Discussion of Results.

From Table 2, it is seen that the number of iterations possible to solve the problem increases in powers of two as the number of projects increases but that the number of iterations necessary to find the optimal solution with the Lawler-Bell algorithm does not increase nearly so rapidly. The programs in this report took more execution time than

Table 4

Probabilistic Results

Problem	Risk Aversion Coefficient, A	Project Set Solution	Objective Value
5x6	0*	2,3	25.00
	.1	2,4,5	21.29
	.3	1,4,5	14.10
	1.1	1,4	11.90
8x13	0*	2,6,8	2900.00
	.01	2,6,8	2897.11
	.1	2,6,8	2871.00
	.5	2,6,8	2755.00
15x9	0*	1,2,3,4,10,11,12	2158.00
	1.0	2,3,4,7,10,11,13	1982.40
	10.0	1,6,7,10,14	1411.02

*The values for A = 0 were taken from the deterministic results.

Table 5

Sensitivity Tests Results

Budget Limits		Optimal Solution	Objective Value		Execution Time	
Dollars	Per Cent Change From Norm	Project Set	Dollars	Per Cent Change From Norm	Time, Min (IBM 360)	Per Cent Change From Norm
600,410,150	-40	4,10,11,13	1358	-37.1	1.39	-33.8
800,540,200	-20	1,4,7,10, 11,14	1738	-19.4	2.10	0
1000,670,250	0	1,2,3,4, 10,11,12	2158	0	2.10	0
1200,800,300	±20	1,2,4,7, 10,11,14	2313	±10.8	1.92	-9.05
1400,930,350	+40	1,2,4,7, 10,11,14	2313	+10.8	1.98	-5.72

ILLEGIBLE

THE FOLLOWING DOCUMENT IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

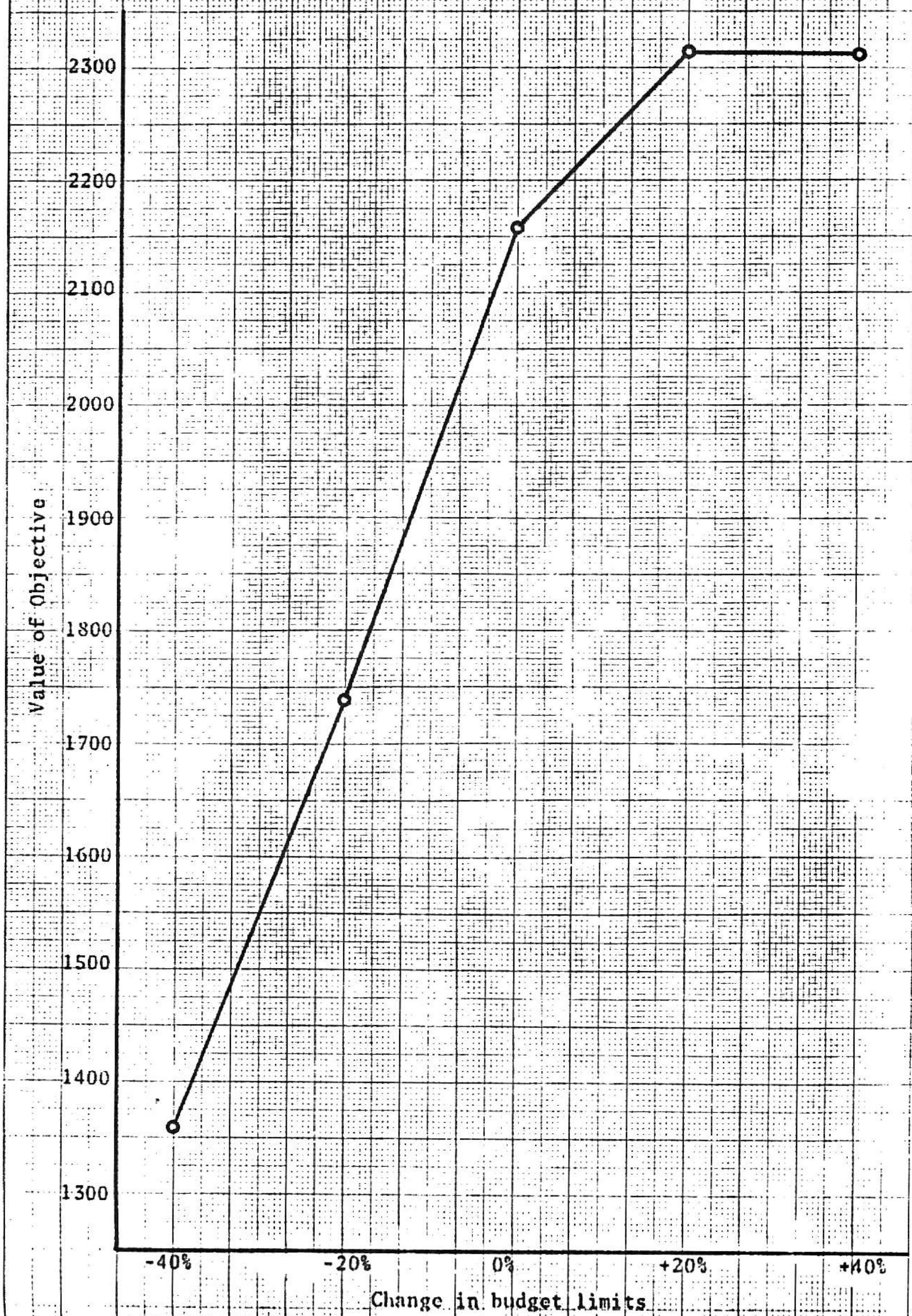

Figure 5

Sensitivity Tests Results

was reported in the literature, but it must be recognized that the transformation from a maximization to a minimization problem and the inverse transformation of the results were done by these programs, whereas the execution times reported in the literature did not include these two steps. The articles in the literature used a manual transformation for the sample problems and then used the algorithm.

The specific results for the first problem are identical with those reported by Mao (13) for both the deterministic and probabilistic cases. Mao does not report on the weak form of the algorithm and it is believed that this is the first use of it to discover alternate optima.

For the second problem, the results for the deterministic case are identical with those of Mao and Wallingford (14) but differ for the probabilistic case. The reason for this is that in reference (14), Mao and Wallingford apparently assume that the decision-maker's utility function is quadratic and hence, the risk aversion coefficient applies to both the variance/covariance matrix and to the square of the means of the projects, while in reference (13), Mao apparently assumes that the utility function is exponential and applies the risk aversion coefficient only to the variance/covariance matrix. The program reported here was written with the exponential assumption.

For the probabilistic case, the results were generally as expected, i.e., as the risk aversion coefficient is increased, the program shifts the optimal selection of projects to those having lower values of variance/covariance. For the first and third problems, the variance/covariance terms were large for the projects with high means and there was a noticable shift as A was increased. For the second problem, there was no shift. This was perhaps due to the terms of the variance/covariance matrix being too small to produce a significant change in the solution. If the program had been written with the quadratic assumption mentioned above, it is thought that there would have been a shift in the solution.

For the sensitivity tests, it was found that as the limits for the budget constraints were increased, the value of the objective increased and the selection of projects shifted. For the smaller values of the limits, the increase in the value of the objective was almost proportional but as the limits became larger, the value of the objective leveled out. Further increase in budget limits gave no increase in objective value and no further shifting of projects in the optimal solution vector, thereby indicating a marginal effect on the objective value and a loosening of the budget constraint effectiveness. This procedure makes it possible to investigate a number of similar problems without reformulation and specifically makes it possible to investigate the effects of different budgets.

In this paper it was found that the computing time was small for small problems but increases sharply at about fifteen projects. The number of constraints and types of constraints seem to have an effect, with the fastest convergence being for three or four budget constraints and five or six other constraints. Attempts to use twenty projects or fewer than five or six constraints exceeded the time limits used on the control cards, with the last vector printed out still being far from the end of the process. The time for many projects is not surprising as there is an increase in the number of iterations as the number of projects increases even though the algorithm has been seen to reduce the total quite effectively. The increase in iterations due to few constraints is because the skipping rules of the algorithm are designed to use the constraints in the skipping process. It appears that if the number of constraints becomes too large, the saving in iterations is lost in additional computations.

The programs were dimensioned for twenty-four projects and twenty-five constraints and although they were not tested with a problem that large, there is no reason to expect that they won't run with problems that large. However it is thought that the execution time would be great.

It is concluded that this is an efficient algorithm for problems of moderate size. It is much better than manual solution procedures but the increase in the amount of computing time required for problems greater than fifteen or twenty projects indicates that it probably is not too useful for larger problems. In any integer programming problem the number of possible solutions increases by powers of two for zero-one problems, but the exact increase in the number of iterations and in execution time depends on the algorithm used. In general, each program becomes too large for economical computation eventually. The question is, when does this happen. It appears that this algorithm is limited to the size mentioned above. If one had plenty of computing time, he might use the Lawler-Bell algorithm with slightly larger problems but the increase in iterations would prevent a great increase in problem size.

Suggestions for Use of Programs.

There are two main areas where these programs could be used. The first in in capital budgeting analysis where they could be used for project selection. If more than twenty-four projects or twenty-five constraints were to be considered, the dimensions and formats could be changed accordingly. The WRITE statements could be removed except for those in SUBROUTINE OTPT, to reduce the input/output time and printing since all that is needed in a management application is the final answer. If this is done, new object decks should be made as computing with the object deck takes less time than with the source deck.

The other main area of use is in academic courses in capital budgeting or management. The programs can handle problems of sufficient size to be useful for instructional purposes, and the instructions for preparation of data decks (Appendix B) should be adequate. Problems such as those used in this paper, already formulated or in word-problem form for practice in formulation, could be assigned to the students. Either object decks or source decks could be used. The WRITE statements should be left in the programs for class use as the printout of each of the iterations is also instructive.

Suggestions for Further Research.

Other algorithms for zero-one programming could be coded for this type of problem and then the computer execution times compared. Conversely, the input transformations and inverse transformations could be removed from these programs and then problems already in minimizing form could be solved with this algorithm and with programs for other algorithms. The execution times for the algorithms could then be compared.

The dimensions could be increased and the formats changed, which would make possible a systematic study of the number of iterations and execution time for larger problems.

The order of projects or constraints could be varied to determine if there is some most efficient arrangement in problem formulation. Lawler and Bell (10) report that the order does affect the efficiency but give no specific conclusions. A study of the effects of order of projects to find if the effects are systematic or random would be interesting and perhaps useful. Similarly, a study of the effects of order of constraints might be useful.

Although it is felt that these are good programs, they can be coded more efficiently to conserve computer time. Also, it is possible that there are other routines which would require less execution time than these do. No attempt has been made in this research to provide compacted efficient codes.

Appendix A

PROGRAMS AND PRINTOUTS OF SOLUTION TO THE FIRST PROBLEM OF CHAPTER 2

This appendix is subdivided into six parts as follows:

Appendix A - 1

PROGRAM FOR THE STRONG DETERMINISTIC FORM OF THE ALGORITHM

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

```
              C    DETERMINISTIC FORM OF THE ALGORITHM.
              C    STRONG FORM OF THE ALGORITHM.
 0001                INTEGER XHAT(25),X(25),X1(25),XSTAR(25),XSTAR1(25),X11(25)
 0002                INTEGER G(25),G1(25,25),G2(25,25),LG(25),GL(25),GOD(6),KK(25)
 0003                INTEGER XIN(25),JAY(25),XINB(25),XIND(25)
 0004                MMM=999999999
 0005                IBBB=0
 0006                DO3 J=1,25
 0007                JAY(J)=0
 0008                XIN(J)=0
 0009                KK(J)=0
 0010                XHAT(J)=0
 0011                X(J)=0
 0012                X1(J)=0
 0013                XSTAR1(J)=0
 0014                XSTAR(J)=0
 0015                X11(J)=0
 0016                XINB(J)=0
 0017                XIND(J)=0
 0018                G(J)=0
 0019                GL(J)=0
 0020                LG(J)=0
 0021                DO3 I=1,25
 0022                G1(I,J)=0
 0023              3 G2(I,J)=0
 0024              2 FORMAT (7I6)
 0025            700 FORMAT(1H1,'   THE OBJECTIVE FUNCTION, G(I), FOLLOWS.'//10(I6,1X))
 0026            701 FORMAT(1H0,'    J= ',I2,'    G1(J,I) IS '/10(I6,1X))
 0027            703 FORMAT (I6,I2/5(I2,I6))
 0028            704 FORMAT(1H ,'    J= ',I2,'    G2(J,I) IS '/10(I6,1X))
 0029            705 FORMAT(//'   THE LIMIT VECTOR, LG(I), OF THE CONSTRAINTS FOLLOWS.'
                    X//10(I6,1X))
 0030            706 FORMAT(//'   THE TRANSFORMED LIMIT, JG, OF THE OBJECTIVE IS ',I6,'
                    X.')
 0031            707 FORMAT(//'   THE TRANSFORMED CONSTRAINT MATRIX FOLLOWS.'//)
 0032            505 FORMAT(///T9,'TRANSFORMED VECTOR X',T35,'REASON FOR SKIP      ',T54,
                    X'SKIP TO',T66,'OBJECT FUNCT VALUE',T88,'VALUE OF CURRENT OPT')
 0033            506 FORMAT(5X,25I1,T35,'RULE 1',T54,'X STAR',T66,I6,T88,I9)
 0034            507 FORMAT(5X,25I1,T35,'RULE 3',T42,'CONST ',I2,T54,'X STAR',T66,I6,T8
                    X8,I9)
 0035            508 FORMAT(5X,25I1,T35,'RULE 2',T42,'FEASIBLE ',T54,'X+1',T66,I6,T88,I
                    X9)
 0036            509 FORMAT(5X,25I1,T35,'INFEAS',T42,'CONST ',I2,T54,'X+1',T66,I6,T88,I
                    X9)
 0037                READ(5,2)NP,NP,NOC
 0038                K=NP+1
 0039                NC=NP+NOC
 0040                NB1=NB+1
 0041                READ(5,2)(G(I),(G1(J,I),J=1,6),I=2,K)
 0042                JG=0
 0043                DO40 I=2,K
 0044             40 JG=JG+G(I)
 0045                IF (NOC.EQ.0) GO TO 47
 0046                DO 41 I=NB1,NC
 0047                READ(5,703)LG(I),KJ,(JAY(L),KK(L),L=1,KJ)
 0048                DO 42 L=1,KJ
 0049                J=JAY(L)
 0050             42 G1(I,J)=KK(L)
```

```
      DO43 J=1,K
      IF(G1(I,J).GE.0)GO TO 43
      G2(I,J)=(-1)*G1(I,J)
      G1(I,J)=0
   43 CONTINUE
      NNG=0
      DO44 J=1,K
   44 NNG=NNG+G1(I,J)-G2(I,J)
      LG(I)=LGG1-NNG
   41 CONTINUE
   47 READ(5,2)(BUD(J),J=1,NB)
      DO45 I=1,NB
      NNNG=0
      DO46 J=1,K
   46 NNNG=NNNG+G1(I,J)
   45 LG(I)=BUD(I)-NNNG
      WRITE(6,700)(G(I),I=1,K)
      WRITE(6,707)
      DO 702 J=1,NC
      WRITE(6,701)J,(G1(J,I),I=1,K)
  702 WRITE(6,704) J,(G2(J,I),I=1,K)
      WRITE(6,705) (LG(I),I=1,NC)
      WRITE(6,706) JG
      WRITE(6,505)
      GO TO 7
    5 CONTINUE
      DO6 J=1,K
    6 X(J)=XSTAR(J)
    7 CONTINUE
      IF(X(1).EQ.1)GO TO 100
C SUBPROGRAM STAR
C CALCULATE X-1
C BINARY SUBTRACTION
      J=K
      IF(X(J).NE.0)GO TO 10
   11 J=J-1
      IF(J.EQ.1)GO TO 25
      IF(X(J).EQ.0)GO TO 11
      M=J+1
      DO13 I=M,K
   13 X1(I)=1
   10 X1(J)=0
      L=J-1
      DO14 I=1,L
   14 X1(I)=X(I)
C CALCULATE XSTAR1
C BOOLEAN ADDITION
      DO15 J=1,K
      IF(X(J).NE.0)GO TO 16
      IF(X1(J).NE.0)GO TO 16
      XSTAR1(J)=0
      GO TO 15
   16 XSTAR1(J)=1
   15 CONTINUE
C CALCULATE XSTAR
C BINARY ADDITION
      CALL BIADD(K,XSTAR1,XSTAR)
      GO TO 75
```

```
      25 XSTAR(K)=1
C END OF SUBPROGRAM STAR
      75 CONTINUE
C RULE 1
         NG=0
         DO30 J=1,K
      30 NG=NG+G(J)*X(J)
         IF(NG.LT.MMM)GO TO 31
         WRITE(6,506) (X(L),L=1,25),NG,MMM
         GO TO 5
      31 CONTINUE
C END OF RULE 1
C RULE 3
         DO35 I=1,NC
         NG1=0
         NG2=0
         DO33 J=1,K
         NG1=NG1+G1(I,J)*XSTAR1(J)
      33 NG2=NG2+G2(I,J)*X(J)
         NN=NG1-NG2+LG(I)
         IF(0.LE.NN)GO TO 35
         WRITE(6,507) (X(L),L=1,25),I,NG,MMM
         GO TO 5
      35 GL(I)=NG2
C END OF RULE 3
C TO DETERMINE IF INFEASIBLE
         DO36I=1,NC
         NG1=0
         DO37 J=1,K
      37 NG1=NG1+G1(I,J)*X(J)
         NN=NG1-GL(I)+LG(I)
         IF(NN.LT.0)GO TO 50
      36 CONTINUE
C RULE 2
         MMM=NG
         DO38 J=1,K
      38 XHAT(J)=X(J)
         WRITE(6,508) (X(L),L=1,25),NG,MMM
         CALL INVT(X,XIN,XINB,NMG,JG,MMM,IBBB,K)
         GO TO 5
C INFEASIBLE
      50 CONTINUE
         WRITE(6,509) (X(L),L=1,25),I,NG,MMM
         CALL BIADD(K,X,X11)
         DO49 I=1,K
      49 X(I)=X11(I)
         GO TO 7
C END OF FEASIBILITY
C END OF RULE 2
     100 CONTINUE
         CALL OTPT(XINF,XIND,IBBB)
         STOP
         END
```

```
FORTRAN IV G LEVEL  18                BIADD              DATE = 72014         22/25/35          [illegible]

              SUBROUTINE BIADD(K,X,X11)
              INTEGER X(25),X11(25)
              J=K
              IF(X(J).EQ.0)GO TO 44
           45 J=J-1
              IF(X(J).NE.0)GO TO 45
              M=J+1
              DO46 I=M,K
           46 X11(I)=0
           44 X11(J)=1
              IF(J.EQ.1)GO TO 48
              L=J-1
              DO47 I=1,L
           47 X11(I)=X(I)
           48 CONTINUE
              RETURN
              END
```

FORTRAN IV G LEVEL 18 INVT DATE = 72014 22/25/33

```
      SUBROUTINE INVT(X,XIN,XINB,NMG,JG,MMM,IBBB,K)
      INTEGER X(25),XIN(25),XINB(25)
      NMG=JG-MMM
      DO 72 J=2,K
      IF(X(J).EQ.1)GO TO 71
      XIN(J)=1
      GO TO 72
   71 XIN(J)=0
   72 CONTINUE
   73 FORMAT(1H0,'    A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE
     X IS ',25I1,' .'/'     THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPA
     XCE IS ',I6,' .'/'     ITERATION CONTINUES.'/)
      WRITE(6,73)(XIN(J),J=1,25),NMG
      IF(NMG.LE.IBBB)GO TO 54
      IBBB=NMG
      DO 53 J=1,25
   53 XINB(J)=XIN(J)
   54 CONTINUE
      RETURN
      END
```

```
FORTRAN IV G LEVEL  18                OTPT                DATE = 72014           22/25/33

          SUBROUTINE OTPT (XINB,XIND,IBBB)
          INTEGER XINB(25),XIND(25)
        1 FORMAT (//'    THE OPTIMAL VALUE OF THE OBJECTIVE IS ',I6,' ,'///'
         X  THE OPTIMAL FEASIBLE VECTOR OF PROJECTS FOLLOWS.'//T4,25(I2,', '
         X))
          K=0
          DO 3 I=2,25
          IF (XINB(I).EQ.0) GO TO 3
          K=K+1
          XIND(K)=I-1
        3 CONTINUE
          WRITE(6,1)IBBP,(XIND(J),J=1,K)
          RETURN
          END
```

Appendix A - 1 - 1

PRINTOUT OF THE SOLUTION TO THE FIRST PROBLEM OF CHAPTER 2 USING THE STRONG DETERMINISTIC FORM OF THE ALGORITHM

THE OBJECTIVE FUNCTION, G(I), FOLLOWS.

0	10	20	5	3	2

THE TRANSFORMED CONSTRAINT MATRIX FOLLOWS.

```
J=  1   G1(J,I) IS
 0    20     30    15    10     5
J=  1   G2(J,I) IS
 0     0      0     0     0     0

J=  2   G1(J,I) IS
 0    20     15     5     7     4
J=  2   G2(J,I) IS
 0     0      0     0     0     0

J=  3   G1(J,I) IS
 0     0      0     0     0     0
J=  3   G2(J,I) IS
 0   500   1000   100    50    20

J=  4   G1(J,I) IS
 0   500   1000   100    50    20
J=  4   G2(J,I) IS
 0     0      0     0     0     0

J=  5   G1(J,I) IS
 0     1      1     0     0     0
J=  5   G2(J,I) IS
 0     0      0     0     0     0

J=  6   G1(J,I) IS
 0     0      0     1     0     0
J=  6   G2(J,I) IS
 0     0      1     0     0     0
```

THE LIMIT VECTOR, LG(I), OF THE CONSTRAINTS FOLLOWS.

-15	-5	1170	-570	-1	0

THE TRANSFORMED LIMIT, JG, OF THE OBJECTIVE IS 40.

TRANSFORMED VECTOR X	REASON FOR SKIP	SKIP TO	OBJECT FUNCT VALUE	VALUE OF CURRENT JP
0000000000000000000000000	RULE 3 CONST 1	X STAR	0	999999999
0000010000000000000000000	RULE 3 CONST 1	X STAR	2	999999999
0000100000000000000000000	RULE 3 CONST 4	X STAR	3	999999999
0001000000000000000000000	RULE 3 CONST 4	X STAR	5	999999999
0010000000000000000000000	INFEAS CONST 6	X+1	20	999999999
0010010000000000000000000	RULE 3 CONST 6	X STAR	22	999999999
0010100000000000000000000	RULE 3 CONST 6	X STAR	23	999999999
0011000000000000000000000	RULE 2 FEASIBLE	X+1	25	25

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0100110000000000000000000 .

THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 15 .
ITERATION CONTINUES.

0100000000000000000000000	INFEAS CONST 4	X+1	10	25
0100010000000000000000000	RULE 3 CONST 4	X STAR	12	15
0100100000000000000000000	INFEAS CONST 4	X+1	13	25
0100110000000000000000000	RULE 2 FEASIBLE	X+1	15	15

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0011000000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 25 .
ITERATION CONTINUES.

0101000000000000000000000	RULE 1	X STAR	15	15
0110000000000000000000000	RULE 1	X STAR	30	15

THE OPTIMAL VALUE OF THE OBJECTIVE IS 25 .

THE OPTIMAL FEASIBLE VECTOR OF PROJECTS FOLLOWS.

2, 3,

Appendix A - 2

PROGRAM FOR WEAK DETERMINISTIC FORM OF THE ALGORITHM

```
C     WEAK FORM OF THE ALGORITHM.
C    DETERMINISTIC FORM OF THE ALGORITHM.
C     DATA CARDS FOR NEAR OPTIMAL MMM AND X(I) MUST BE INCLUDED.
      INTEGER XHAT(25),X(25),X1(25),XSTAR(25),XSTAR1(25),X11(25)
      INTEGER G(25),G1(25,25),G2(25,25),LG(25),GL(25),JUP(5),FF(25)
      INTEGER XIN(25),JAY(25),XIND(25),XINB(5,25)
      REAL(5,1)MMM
      INMB=0
      KLM=1
      DO4 J=1,25
      JAY(J)=0
      XIN(J)=0
      KK(J)=0
      XHAT(J)=0
      X(J)=0
      X1(J)=0
      XSTAR1(J)=0
      XSTAR(J)=0
      X11(J)=0
      XIND(J)=0
      G(J)=0
      GL(J)=0
      LG(J)=0
      DO3 I=1,25
      G1(I,J)=0
    3 G2(I,J)=0
      DO 4 I=1,5
    4 XINB(I,J)=0
      READ(5,300)(X(I),I=2,25)
  300 FORMAT(25I1)
    1 FORMAT(I9)
    2 FORMAT (7I6)
  505 FORMAT(///T9,'TRANSFORMED VECTOR X',T35,'REASON FOR SKIP     ',T54,
     X'SKIP TO',T66,'OBJECT FUNCT VALUE',T88,'VALUE OF CURRENT OPT')
  506 FORMAT(5X,25I1,T35,'RULE 1',T54,'X STAR',T66,I6,T88,I9)
  507 FORMAT(5X,25I1,T35,'RULE 3',T42,'CONST ',I2,T54,'X STAR',T66,I6,T8
     X8,I9)
  508 FORMAT(5X,25I1,T35,'RULE 2',T42,'FEASIBLE ',T54,'X+1',T66,I6,T88,I
     X9)
  509 FORMAT(5X,25I1,T35,'INFEAS',T42,'CONST ',I2,T54,'X+1',T66,I6,T88,I
     X9)
  700 FORMAT(1H1,'    THE OBJECTIVE FUNCTION, G(I), FOLLOWS.'//10(I6,1X))
  701 FORMAT(1H0,'    J= ',I2,'    G1(J,I) IS '/10(I6,1X))
  703 FORMAT (I6,I2/5(I2,I6))
  704 FORMAT(1H ,'    J= ',I2,'    G2(J,I) IS '/10(I6,1X))
  705 FORMAT(//'    THE LIMIT VECTOR, LG(I), OF THE CONSTRAINTS FOLLOWS.'
     X//10(I6,1X))
  706 FORMAT(//'    THE TRANSFORMED LIMIT, JG, OF THE OBJECTIVE IS ',I6,'
     X.')
  707 FORMAT(//'    THE TRANSFORMED CONSTRAINT MATRIX FOLLOWS.'//)
      READ(5,2)NP,NP,NOC
      K=NP+1
      NC=NB+NOC
      NB1=NB+1
      READ(5,2)(G(I),(G1(J,I),J=1,6),I=2,K)
      JG=0
      DO40 I=2,K
   40 JG=JG+G(I)
```

```
      IF (NUC.EQ.0) GO TO 47
      DO 41 I=NB1,NC
      READ(5,703)LGG1,KJ,(JAY(L),KK(L),L=1,KJ)
      DO 42 L=1,KJ
      J=JAY(L)
   42 G1(I,J)=KK(L)
      DO43 J=1,K
      IF(G1(I,J).GE.0)GO TO 43
      G2(I,J)=(-1)*G1(I,J)
      G1(I,J)=0
   43 CONTINUE
      NNG=0
      DO44 J=1,K
   44 NNG=NNG+G1(I,J)-G2(I,J)
      LG(I)=LGG1-NNG
   41 CONTINUE
   47 READ(5,2)(BUD(J),J=1,NB)
      DO45 I=1,NB
      NNNG=0
      DO46 J=1,K
   46 NNNG=NNNG+G1(I,J)
   45 LG(I)=BUD(I)-NNNG
      WRITE(6,700)(C(I),I=1,K)
      WRITE(6,707)
      DO 702 J=1,NC
      WRITE(6,701)J,(G1(J,I),I=1,K)
  702 WRITE(6,704) J,(G2(J,I),I=1,K)
      WRITE(6,705) (LG(I),I=1,NC)
      WRITE(6,706) JG
      WRITE(6,505)
      GO TO 7
    5 CONTINUE
      DO6 J=1,K
    6 X(J)=XSTAR(J)
    7 CONTINUE
      IF(X(1).EQ.1)GO TO 100
C SUBPROGRAM STAR
C CALCULATE X-1
C BINARY SUBTRACTION
      J=K
      IF(X(J).NE.0)GO TO 10
   11 J=J-1
      IF(J.EQ.1)GO TO 25
      IF(X(J).EQ.0)GO TO 11
      M=J+1
      DO13 I=M,K
   13 X1(I)=1
   10 X1(J)=0
      L=J-1
      DO14 I=1,L
   14 X1(I)=X(I)
C CALCULATE XSTAR1
C BOOLEAN ADDITION
      DO15 J=1,K
      IF(X(J).NE.0)GO TO 16
      IF(X1(J).NE.0)GO TO 16
      XSTAR1(J)=0
      GO TO 15
```

```
   16 XSTAR1(J)=1
   15 CONTINUE
C CALCULATE XSTAR
C BINARY ADDITION
      CALL BIADD(K,XSTAR1,XSTAR)
      GO TO 75
   25 XSTAR(K)=1
C END OF SUBPROGRAM STAR
   75 CONTINUE
C RULE 1
      NG=0
      DO30 J=1,K
   30 NG=NG+G(J)*X(J)
      IF(NG.LE.MMM)GO TO 31
      WRITE(6,506) (X(L),L=1,25),NG,MMM
      GO TO 5
   31 CONTINUE
C END OF RULE 1
C RULE 3
      DO35 I=1,NC
      NG1=0
      NG2=0
      DO33 J=1,K
      NG1=NG1+G1(I,J)*XSTAR1(J)
   33 NG2=NG2+G2(I,J)*X(J)
      NN=NG1-NG2+LG(I)
      IF(0.LE.NN)GO TO 35
      WRITE(6,507) (X(L),L=1,25),I,NG,MMM
      GO TO 5
   35 GL(I)=NG2
C END OF RULE 3
C TO DETERMINE IF INFEASIBLE
      DO36I=1,NC
      NG1=0
      DO37 J=1,K
   37 NG1=NG1+G1(I,J)*X(J)
      NN=NG1-GL(I)+IG(I)
      IF(NN.LT.0)GO TO 50
   36 CONTINUE
C RULE 2
      MMM=NG
      DO38 J=1,K
   38 XHAT(J)=X(J)
      WRITE(6,508) (X(L),L=1,25),NG,MMM
      CALL INVT(X,XIN,XINB,NMG,JG,MMM,IBBB,K,KLM)
      GO TO 51
C INFEASIBLE
   50 CONTINUE
      WRITE(6,509) (X(L),L=1,25),I,NG,MMM
C BINARY ADDITION
      CALL BIADD(K,Y,X11)
      DO49 I=1,K
   49 X(I)=X11(I)
      GO TO 7
   51 CONTINUE
      CALL BIADD(K,X,X11)
      DO91 L=1,K
      IF(X11(L).NE.XSTAR(L))GO TO 92
```

```
      91 CONTINUE
         GO TO 5
      92 CONTINUE
         DO89 I=1,K
      89 X(I)=X11(I)
         IF(X(1).EQ.1)GO TO 100
         GO TO 75
C END OF FEASIBILITY
C END OF RULE 2
     100 CONTINUE
         CALL OTPT (XINB,XIND,IBBB,KLM)
         STOP
         END
```

```
      C SUBROUTINE BIADD
            SUBROUTINE BIADD(K,X,X11)
            INTEGER X(25),X11(25)
            J=K
            IF(X(J).EQ.0)GO TO 44
         45 J=J-1
            IF(X(J).NE.0)GO TO 45
            M=J+1
            DO46 I=M,K
         46 X11(I)=0
         44 X11(J)=1
            IF(J.EQ.1)GO TO 48
            L=J-1
            DO47 I=1,L
         47 X11(I)=X(I)
         48 CONTINUE
            RETURN
            END
```

```
FORTRAN IV G LEVEL  19                INVT                DATE = 72014          22/27/52          [illegible]

C001               SUBROUTINE INVT(X,XIN,XINB,NMG,JG,MMM,IBBB,K,KLM)
C002               INTEGER X(25),XIN(25),XINB(5,25)
C003               NMG=JG-MMM
C004               DO 72 J=2,K
C005               IF(X(J).EQ.1)GO TO 71
C006               XIN(J)=1
C007               GO TO 72
C008            71 XIN(J)=0
C009            72 CONTINUE
C010            73 FORMAT(1H0,'    A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE
                  X IS ',25I1,' .'/'     THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPA
                  XCE IS ',I6,' .'/'     ITERATION CONTINUES.'/)
C011               WRITE(6,73)(XIN(J),J=1,25),NMG
C012               IF(NMG.EQ.IBBB)GO TO 54
0013               IBBB=NMG
C014               KLM=1
C015               DO 50 I=1,5
0016               DO 50 J=1,25
C017            50 XINB(I,J)=0
C018               GO TO 55
C019            54 KLM=KLM+1
0020               IF(KLM.EQ.6)GO TO 60
C021            55 DO 56 J=1,25
C022            56 XINB(KLM,J)=XIN(J)
0023            60 CONTINUE
C024               RETURN
C025               END
```

```
FORTRAN IV G LEVEL  18             OTPT              DATE = 72014          22/27/52

      SUBROUTINE OTPT(XINB,XIND,IBBB,KLM)
      INTEGER XINB(5,25),XIND(25)
      WRITE(6,1)IBBB
      IF(KLM.LT.6)GO TO 12
      WRITE(6,2)
      KLM=5
      GO TO 10
   12 IF(KLM.EQ.1)GO TO 14
      WRITE(6,4)KLM
      GO TO 10
   14 WRITE(6,5)
   10 DO 13 J=1,KLM
      K=0
      DO 15 I=2,25
      IF(XINB(J,I).EQ.0)GO TO 15
      K=K+1
      XIND(K)=I-1
   15 CONTINUE
      IF(KLM.EQ.1)GO TO 16
      WRITE(6,3) J,(XIND(N),N=1,K)
      GO TO 13
   16 WRITE(6,7)(XIND(N),N=1,K)
   13 CONTINUE
    1 FORMAT(//'   THE OPTIMAL VALUE OF THE OBJECTIVE IS ',I6,' .')
    2 FORMAT (/'   THERE ARE MORE THAN FIVE OPTIMAL FEASIBLE VECTORS. TH
     XE FIRST FIVE FOLLOW.'/)
    3 FORMAT('   OPTIMAL FEASIBLE VECTOR NUMBER ',I2,' FOLLOWS.'/T4,25(I
     X2,', '))
    4 FORMAT (/'   THERE ARE ',I2,' OPTIMAL FEASIBLE VECTORS.'/)
    5 FORMAT(/'   THERE IS ONE OPTIMAL FEASIBLE VECTOR.'/)
    7 FORMAT('   THE OPTIMAL FEASIBLE VECTOR FOLLOWS.'/T4,25(I2,', '))
      RETURN
      END
```

Appendix A - 2 - 1

PRINTOUT OF THE SOLUTION TO THE FIRST PROBLEM OF CHAPTER 2

USING THE WEAK DETERMINISTIC FORM OF THE ALGORITHM

```
THE OBJECTIVE FUNCTION, G(I), FOLLOWS.

 0      10      20      5       3       2


THE TRANSFORMED CONSTRAINT MATRIX FOLLOWS.


J=  1   G1(J,I) IS
 0     20      30     15      10       5
J=  1   G2(J,I) IS
 0      0       0      0       0       0

J=  2   G1(J,I) IS
 0     20      15      5       7       4
J=  2   G2(J,I) IS
 0      C       0      0       0       0

J=  3   G1(J,I) IS
 0      0       0      0       0       0
J=  3   G2(J,I) IS
 0    500    1000    100      50      20

J=  4   G1(J,I) IS
 0    500    1000    100      50      20
J=  4   G2(J,I) IS
 0      0       0      0       0       0

J=  5   G1(J,I) IS
 0      1       1      0       0       0
J=  5   G2(J,I) IS
 0      0       0      0       0       0

J=  6   G1(J,I) IS
 0      0       0      1       0       0
J=  6   G2(J,I) IS
 0      0       1      0       0       0

THE LIMIT VECTOR, LG(I), OF THE CONSTRAINTS FOLLOWS.

-15     -5   1170   -570      -1       0


THE TRANSFORMED LIMIT, JG, OF THE OBJECTIVE IS     40.


    TRANSFORMED VECTOR X       REASON FOR SKIP    SKIP TO     OBJECT FUNCT VALUE    VALUE OF [illegible]
 0100100C000000000000000000    INFEAS CONST  4    X+1              13                  [illegible]
 010011CC000000000CC0000000    RULE 2 FEASIBLE    X+1              15                  15

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0011000000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS      25 .
ITERATION CONTINUES.

 010100CC0000000CCC0000000     RULE 2 FEASIBLE    X+1              15                  15

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0010110000000000000000000 .
```

```
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS      25 .
ITERATION CONTINUES.

 0101010CC000000000CC0000000     RULE 1          X STAR        17                  15
 011C000C0000000000CC0000000     RULE 1          X STAR        30                  15

THE OPTIMAL VALUE OF THE OBJECTIVE IS      25 .

THERE ARE  2 OPTIMAL FEASIBLE VECTORS.

OPTIMAL FEASIBLE VECTOR NUMBER  1 FOLLOWS.
 2,  3,
OPTIMAL FEASIBLE VECTOR NUMBER  2 FOLLOWS.
 2,  4,  5,
```

Appendix A - 3

PROGRAM FOR PROBABILISTIC FORM OF THE ALGORITHM

```
FORTRAN IV G LEVEL  18                MAIN              DATE = 72014          22/22/29

         C   PROBABILISTIC FORM OF THE ALGORITHM.
C001          INTEGER XHAT(25),X(25),X1(25),XSTAR(25),XSTAR1(25),X11(25)
C002          INTEGER G(25),G1(25,25),G2(25,25),LG(25),GL(25),BJD(6),KK(25)
C003          INTEGER XIN(25),JAY(25),XINB(25),XIND(25)
C004          DIMENSION GVAR(25,25),GBAR(25),AB(10)
C005          BMM=99999999.
C006        2 FORMAT (7I6)
C007      505 FORMAT(///T9,'TRANSFORMED VECTOR X',T35,'REASON FOR SKIP     ',T54,
             X'SKIP TO',T66,'OBJECT FUNCT VALUE',T88,'VALUE OF CURRENT OPT')
0008      506 FORMAT(5X,25I1,T35,'RULE 1',T54,'K STAR',T66,F12.2,T88,F12.2)
0009      507 FORMAT(5X,25I1,T35,'RULE 3',T42,'CONST ',I2,T54,'K STAR',T66,F12.2
             X,T88,F12.2)
C010      508 FORMAT(5X,25I1,T35,'RULE 2',T42,'FEASIBLE ',T54,'X+1',T66,F12.2,T8
             X8,F12.2)
C011      509 FORMAT(5X,25I1,T35,'INFEAS',T42,'CONST ',I2,T54,'X+1',T66,F12.2,T8
             X8,F12.2)
C012      510 FORMAT(5X,25I1,T35,'RULE 4',T42,'FEASIBLE ',T54,'X+1',T66,F12.2,T8
             X8,F12.2)
0013      700 FORMAT(1H1,'    THE OBJECTIVE FUNCTION, G(I), FOLLOWS.'//10(I6,1X))
C014      701 FORMAT(1H0,'    J= ',I2,'    G1(J,I) IS '/10(I6,1X))
C015      703 FORMAT (I6,I2/5(I2,I6))
C016      704 FORMAT(1H ,'    J= ',I2,'    G2(J,I) IS '/10(I6,1X))
C017      705 FORMAT(//'    THE LIMIT VECTOR, LG(I), OF THE CONSTRAINTS FOLLOWS.'
             X//10(I6,1X))
0018      706 FORMAT(//'    THE TRANSFORMED LIMIT, JG, OF THE OBJECTIVE IS ',I6,'
             X.')
C019      707 FORMAT(//'    THE TRANSFORMED CONSTRAINT MATRIX FOLLOWS.'//)
C020      901 FORMAT(5F6.2)
0021      902 FORMAT(10(F6.2,1X))
0022      903 FORMAT(1H1,'    THE FOLLOWING SOLUTIONS ARE FOR RISK AVERSION COEFF
             XICIENT A= ',F6.2,' .')
0023      904 FORMAT(//'    THE INPUT COVARIANCE MATRIX, GVAR, IS'//)
C024      910 FORMAT(1H0,'    GBAR IS'/10(F6.2,1X))
C025      911 FORMAT(1H0,'    THE TRANSFORMED LIMIT OF THE OBJECTIVE, ADJUSTED FO
             XR COVARIANCE, BMG, IS ',F6.2,'.')
C026          DO4 J=1,25
C027          JAY(J)=0
C028          KK(J)=0
C029          G(J)=0
C030          GL(J)=0
C031          LG(J)=0
C032          DO 4 I=1,25
C033          GVAR(I,J)=0
C034          G1(I,J)=0
C035        4 G2(I,J)=0
C036          READ(5,2)NP,NF,NOC,KAM
C037          K=NP+1
C038          NC=NB+NOC
C039          NP1=NP+1
C040          READ(5,2)(G(I),(G1(J,I),J=1,6),I=2,K)
C041          JG=0
C042          DO40 I=2,K
C043       40 JG=JG+G(I)
C044          IF (NOC.EQ.0) GO TO 47
C045          DO 41 I=NB1,NC
C046          READ(5,703)LG(I),KJ,(JAY(L),KK(L),L=1,KJ).
C047          DO 42 L=1,KJ
C048          J=JAY(L)
```

```
      42 G1(I,J)=KK(L)
         DO43 J=1,K
         IF(G1(I,J).GE.0)GO TO 43
         G2(I,J)=(-1)*G1(I,J)
         G1(I,J)=0
      43 CONTINUE
         NNG=0
         DO44 J=1,K
      44 NNG=NNG+G1(I,J)-G2(I,J)
         LG(I)=LGS1-NNG
      41 CONTINUE
      47 READ(5,2)(BUD(J),J=1,NB)
         DO45 I=1,NB
         NNNG=0
         DO46 J=1,K
      46 NNNG=NNNG+G1(I,J)
      45 LG(I)=BUD(I)-NNNG
         DO 152 I=2,K
     152 READ(5,901)(GVAR(I,J),J=2,K)
         READ(5,901)(AB(IJK),IJK=1,KAM)
         WRITE(6,700)(G(I),I=1,K)
         WRITE(6,707)
         DO 702 J=1,NC
         WRITE(6,701)J,(G1(J,I),I=1,K)
     702 WRITE(6,704) J,(G2(J,I),I=1,K)
         WRITE(6,705) (LG(I),I=1,NC)
         WRITE(6,706) JG
         WRITE(6,904)
         DO 150 J=1,K
     150 WRITE(6,902)(GVAR(J,I),I=1,K)
         DO 102 IJK=1,KAM
         DO 151 J=1,25
         XIMD(J)=0
         XIMB(J)=0
         X(J)=0
         X1(J)=0
         X11(J)=0
         XSTAR1(J)=0
         XSTAR(J)=0
         XHAT(J)=0
         XIM(J)=0
     151 GBAR(J)=0
         A=AB(IJK)
         BMMB=0
C  TRANSFORM GVAR AND FIND BMG
         BAR = 0
         DO 101 I=2,K
         DO 101 J=2,K
         GBAR(I)=GBAR(I)+GVAR(I,J)
         IF(I.NE.J)GBAR(I)=GBAR(I)+GVAR(I,J)
     101 BAR=BAR+A*GVAR(I,J)
         BMG=JG-BAR
C  END OF TRANSFORM
         WRITE(6,903)A
         WRITE(6,910)(GBAR(J),J=1,K)
         WRITE(6,911)BMG
         WRITE(6,505)
         GO TO 7
```

```
0105        5 CONTINUE
0106          DO6 J=1,K
0107        6 X(J)=XSTAR(J)
0108        7 CONTINUE
0109          IF(X(1).EQ.1)GO TO 100
      C SUBPROGRAM STAR
      C CALCULATE X-1
0110          J=K
0111          IF(X(J).NE.0)GO TO 10
0112       11 J=J-1
0113          IF(J.EQ.1)GO TO 25
0114          IF(X(J).EQ.0)GO TO 11
0115          M=J+1
0116          DO13 I=M,K
0117       13 X1(I)=1
0118       10 X1(J)=0
0119          L=J-1
0120          DO14 I=1,L
0121       14 X1(I)=X(I)
      C CALCULATE XSTAR1
0122          DO15 J=1,K
0123          IF(X(J).NE.0)GO TO 16
0124          IF(X1(J).NE.0)GO TO 16
0125          XSTAR1(J)=0
0126          GO TO 15
0127       16 XSTAR1(J)=1
0128       15 CONTINUE
      C CALCULATE XSTAR
0129          CALL BIADD(K,XSTAR1,XSTAR)
0130          GO TO 75
0131       25 XSTAR(K)=1
      C END OF SUBPROGRAM STAR
0132       75 CONTINUE
      C RULE 1
0133          NG=0
0134          DO30 J=1,K
0135       30 NG=NG+G(J)*X(J)
0136          SVAR=0.
0137          TVAR=0.
0138          CALL VAR(GVAR,X,GBAR,SVAR,TVAR,K,A)
0139          BVG=NG+TVAR
0140          BVGG=BVG-SVAR
0141          SVAR=0.
0142          TVAR=0.
0143          CALL VAR(GVAR,XSTAR1,GBAR,SVAR,TVAR,K,A)
0144          BPG=BVG-SVAR
0145          IF(BPG.LE.BMM)GO TO 31
0146          WRITE(6,506)(X(L),L=1,25),BVGG,BMM
0147          GO TO 5
0148       31 CONTINUE
      C END OF RULE 1
      C RULE 3
0149          DO35 I=1,NC
0150          NG1=0
0151          NG2=0
0152          DO33 J=1,K
0153          NG1=NG1+G1(I,J)*XSTAR1(J)
0154       33 NG2=NG2+G2(I,J)*X(J)
```

```
                NN=NG1-NG2+LG(I)
                IF(O.LE.NN)GO TO 35
                WRITE(6,507)(X(L),L=1,25),I,BVGG,BMM
                GO TO 5
           35 GL(I)=NG2
      C END OF RULE 3
      C TO DETERMINE IF INFEASIBLE
                DO36I=1,NC
                NG1=0
                DO37 J=1,K
           37 NG1=NG1+G1(I,J)*X(J)
                NN=NG1-GL(I)+LG(I)
                IF(NN.LT.0)GO TO 50
           36 CONTINUE
      C RULE 2
      C   RULE 4
                IF(BVGG.GT.BMM)GO TO 52
                BMM=BVGG
                DO38 J=1,K
           38 XHAT(J)=X(J)
                WRITE(6,508)(X(L),L=1,25),BVGG,BMM
                CALL INVT (X,XIN,XINB,BBMG,BMG,BVGG,BBBB,K)
                GO TO 51
      C INFEASIBLE
           50 CONTINUE
                WRITE(6,509)(X(L),L=1,25),I,BVGG,BMM
           51 CONTINUE
                CALL BIADD(K,X,X11)
                DO49 I=1,K
           49 X(I)=X11(I)
                GO TO 7
           52 WRITE(6,510)(X(L),L=1,25),BVGG,BMM
                CALL INVT (X,XIN,XINB,BBMG,BMG,BVGG,BBBB,K)
                GO TO 51
      C END OF RULE 2
          100 CONTINUE
                CALL OTPT(XINP,XIND,BBBB)
          102 CONTINUE
                STOP
                END
```

```
FORTRAN IV G LEVEL  18                MAIN                DATE = 72014        22/22/2?

      C SUBROUTINE BIADD
            SUBROUTINE BIADD(K,X,X11)
            INTEGER X(25),X11(25)
            J=K
            IF(X(J).EQ.0)GO TO 44
         45 J=J-1
            IF(X(J).NE.0)GO TO 45
            M=J+1
            DO46 I=M,K
         46 X11(I)=0
         44 X11(J)=1
            IF(J.EQ.1)GO TO 48
            L=J-1
            DO47 I=1,L
         47 X11(I)=X(I)
         48 CONTINUE
            RETURN
            END
```

FORTRAN IV G LEVEL 18 MAIN DATE = 72014 22/22/2[illegible]

```
C   SUBROUTINE VAR
        SUBROUTINE VAR(GVAR,X,GBAR,SVAR,TVAR,K,A)
        INTEGER X(25)
        DIMENSION GVAR(25,25),GBAR(25)
        DO 1 I=2,K
        DO 1 J=2,K
        IF(I.EQ.J)GO TO 3
        IF(GVAR(I,J).GT.0) GO TO 2
        SVAR=SVAR+A*GVAR(I,J)*X(I)*X(J)
        GO TO 1
      2 TVAR=TVAR+A*GVAR(I,J)*X(I)*X(J)
        GO TO 1
      3 IF(GBAR(I).GT.0)GO TO 4
        TVAR=TVAR+A*GBAR(I)*X(I)
        GO TO 1
      4 SVAR=SVAR+A*GBAR(I)*X(I)
      1 CONTINUE
        RETURN
        END
```

```
FORTRAN IV G LEVEL  18                INVT               DATE = 72014          22/22/29

0001              SUBROUTINE INVT (X,XIN,XINB,BBMG,BMG,BVGG,BBDB,K)
0002              INTEGER X(25),XIN(25),XINB(25)
0003               BBMG=BMG-BVGG
0004              DO 72 J=2,K
0005              IF(X(J).EQ.1)GO TO 71
0006              XIN(J)=1
0007              GO TO 72
0008           71 XIN(J)=0
0009           72 CONTINUE
0010              WRITE(6,73)(XIN(J),J=1,25),BBMG
0011           73 FORMAT(1HO,'    A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE
                 X IS ',25I1,' .'/'     THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPA
                 XCE IS ',F12.2,' .'/'     ITERATION CONTINUES.'/)
0012              IF(BBMG.LE.BBDB) GO TO 54
0013              BBDB=BBMG
0014              DO 53 J=1,25
0015           53 XINB(J)=XIN(J)
0016           54 CONTINUE
0017              RETURN
0018              END
```

FORTRAN IV G LEVEL 18 OTPT DATE = 72014 22/22/29

```
      SUBROUTINE OTPT (XINB,XIND,BBBB)
      INTEGER XINB(25),XIND(25)
    1 FORMAT (//'    THE OPTIMAL VALUE OF THE OBJECTIVE IS ',F12.2,' ,'//
     X/'    THE OPTIMAL FEASIBLE VECTOR OF PROJECTS FOLLOWS.'//T4,25(I2,'
     X, '))
      K=0
      DO 3 I=2,25
      IF (XINB(I).EQ.0) GO TO 3
      K=K+1
      XIND(K)=I-1
    3 CONTINUE
      WRITE(6,1)BBBB,(XIND(J),J=1,K)
      RETURN
      END
```

Appendix A - 3 - 1

PRINTOUT OF THE SOLUTION TO THE FIRST PROBLEM OF CHAPTER 2
USING THE PROBABILISTIC FORM OF THE ALGORITHM

```
THE OBJECTIVE FUNCTION, G(I), FOLLOWS.

 0     10     20      5      3      2

THE TRANSFORMED CONSTRAINT MATRIX FOLLOWS.


J=  1   G1(J,I) IS
 0    20     30     15     10      5
J=  1   G2(J,I) IS
 0     0      0      0      0      0

J=  2   G1(J,I) IS
 0    20     15      5      7      4
J=  2   G2(J,I) IS
 0     0      0      0      0      0

J=  3   G1(J,I) IS
 0     0      0      0      0      0
J=  3   G2(J,I) IS
 0   500   1000    100     50     20

J=  4   G1(J,I) IS
 0   500   1000    100     50     20
J=  4   G2(J,I) IS
 0     0      0      0      0      0

J=  5   G1(J,I) IS
 0     1      1      0      0      0
J=  5   G2(J,I) IS
 0     0      0      0      0      0

J=  6   G1(J,I) IS
 0     0      0      1      0      0
J=  6   G2(J,I) IS
 0     0      1      0      0      0

THE LIMIT VECTOR, LG(I), OF THE CONSTRAINTS FOLLOWS.

-15    -5   1170   -570     -1      0

THE TRANSFORMED LIMIT, JG, OF THE OBJECTIVE IS      40.

THE INPUT COVARIANCE MATRIX, GVAR, IS

0.0    0.0    0.0    0.0    0.0    0.0
0.0    1.00   3.00   0.10   0.0    0.50
0.0    3.00  36.10   2.00   0.0    0.0
0.0    0.10   2.00   1.00   0.0    0.50
0.0    0.0    0.0    0.0    0.0    0.0
0.0    0.50   0.0    0.50   0.0    1.00
```

THE FOLLOWING SOLUTIONS ARE FOR RISK AVERSION COEFFICIENT A= 0.10 .

CBAR IS
0.0 8.20 46.10 6.20 0.0 3.00

THE TRANSFORMED LIMIT OF THE OBJECTIVE, ADJUSTED FOR COVARIANCE, BHG, IS 34.87.

TRANSFORMED VECTOR X	REASON FOR SKIP	SKIP TO	OBJECT FUNCT VALUE	VALUE OF OBJ. AT OPT
0000000000000000000000000	RULE 3 CONST 1	X STAR	0.0	999999744.00
0000010000000000000000000	RULE 3 CONST 1	X STAR	1.70	999999744.00
0000100000000000000000000	RULE 3 CONST 4	X STAR	3.00	999999744.00
0001000000000000000000000	RULE 3 CONST 4	X STAR	4.38	999999744.00
0010000000000000000000000	INFEAS CONST 6	X+1	13.39	999999744.00
0010010000000000000000000	RULE 3 CONST 6	X STAR	17.09	999999744.00
0010100000000000000000000	RULE 3 CONST 6	X STAR	18.39	999999744.00
0011000000000000000000000	RULE 2 FEASIBLE	X+1	20.17	20.17

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS C100110000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 14.70 .
ITERATION CONTINUES.

0011010000000000000000000	RULE 1	X STAR	21.97	20.17
0011100000000000000000000	RULE 1	X STAR	23.17	20.17
0100000000000000000000000	INFEAS CONST 4	X+1	9.18	20.17
0100010000000000000000000	RULE 3 CONST 4	X STAR	10.98	20.17
0100100000000000000000000	INFEAS CONST 4	X+1	12.18	20.17
0100110000000000000000000	RULE 2 FEASIBLE	X+1	13.98	13.98

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0011000000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 20.89 .
ITERATION CONTINUES.

0101000000000000000000000	RULE 2 FEASIBLE	X+1	13.58	13.58

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0010110000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 21.29 .
ITERATION CONTINUES.

0101010000000000000000000	RULE 1	X STAR	15.48	13.58
0101100000000000000000000	RULE 1	X STAR	16.58	13.58
0110000000000000000000000	RULE 1	X STAR	25.17	13.58

THE OPTIMAL VALUE OF THE OBJECTIVE IS 21.29 .

THE OPTIMAL FEASIBLE VECTOR OF PROJECTS FOLLOWS.

2, 4, 5,

THE FOLLOWING SOLUTIONS ARE FOR RISK AVERSION COEFFICIENT A=   0.30 .

GBAR IS
0.0   8.20 46.10   6.20   0.0   3.00

THE TRANSFORMED LIMIT OF THE OBJECTIVE, ADJUSTED FOR COVARIANCE, BNG, IS 24.61.

TRANSFORMED VECTOR X	REASON FOR SKIP	SKIP TO	OBJECT FUNCT VALUE	VALUE OF CURRENT OPT
0000000000000000000000000	RULE 3 CONST 1	X STAR	0.0	14.58
0000010000000000000000000	RULE 3 CONST 1	X STAR	1.10	13.58
0000100000000000000000000	RULE 3 CONST 4	X STAR	3.00	13.58
0001000000000000000000000	RULE 3 CONST 4	X STAR	3.14	13.58
0010000000000000000000000	INFEAS CONST 6	X+1	6.17	13.58
0010010000000000000000000	RULE 3 CONST 6	X STAR	7.27	13.58
0010100000000000000000000	RULE 3 CONST 6	X STAR	9.17	13.58
0011000000000000000000000	RULE 2 FEASIBLE	X+1	10.51	10.51

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0100110000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS   14.10 .
ITERATION CONTINUES.

0011010000000000000000000	RULE 1	X STAR	11.91	10.51
0011100000000000000000000	RULE 1	X STAR	13.51	10.51
0100000000000000000000000	INFEAS CONST 4	X+1	7.54	10.51
0100010000000000000000000	RULE 3 CONST 4	X STAR	8.94	10.51
0100100000000000000000000	INFEAS CONST 4	X+1	10.54	10.51
0100110000000000000000000	RULE 1	X STAR	11.94	10.51
0101000000000000000000000	RULE 4 FEASIBLE	X+1	10.74	10.51

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0010110000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS   13.87 .
ITERATION CONTINUES.

0101010000000000000000000	RULE 1	X STAR	12.44	10.51
0101100000000000000000000	RULE 1	X STAR	13.74	10.51
0110000000000000000000000	RULE 1	X STAR	15.51	10.51

THE OPTIMAL VALUE OF THE OBJECTIVE IS   14.10 .

THE OPTIMAL FEASIBLE VECTOR OF PROJECTS FOLLOWS.

1, 4, 5,

THE FOLLOWING SOLUTIONS ARE FOR RISK AVERSION COEFFICIENT A= 1.10 .

GBAR IS
0 8.20 46.10 6.20 0.0 3.00

THE TRANSFORMED LIMIT OF THE OBJECTIVE, ADJUSTED FOR COVARIANCE, BNG, IS -16.43.

TRANSFORMED VECTOR X	REASON FOR SKIP	SKIP TO	OBJECT FUNCT VALUE	VALUE OF CURRENT OPT
000000000000000000000000	RULE 3 CONST 1	X STAR	0.0	10.51
000001000000000000000000	RULE 3 CONST 1	X STAR	-1.30	10.51
000010000000000000000000	RULE 3 CONST 4	X STAR	3.00	10.51
000100000000000000000000	RULE 3 CONST 4	X STAR	-1.82	10.51
001000000000000000000000	INFEAS CONST 6	X+1	-30.71	10.51
001001000000000000000000	RULE 3 CONST 6	X STAR	-32.01	10.51
001010000000000000000000	RULE 3 CONST 6	X STAR	-27.71	10.51
001100000000000000000000	RULE 2 FEASIBLE	X+1	-28.13	-28.13

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0100110000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 11.70 .
ITERATION CONTINUES.

001101000000000000000000	RULE 2 FEASIBLE	X+1	-28.33	-28.33

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0100100000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 11.90 .
ITERATION CONTINUES.

001110000000000000000000	RULE 4 FEASIBLE	X+1	-25.13	-28.33

A FEASIBLE VECTOR TRANSFORMED TO THE ORIGINAL SPACE IS 0100010000000000000000000 .
THE OBJECTIVE TRANSFORMED TO THE ORIGINAL SPACE IS 8.70 .
ITERATION CONTINUES.

001111000000000000000000	RULE 1	X STAR	-25.33	-28.33
010000000000000000000000	INFEAS CONST 4	X+1	0.98	-28.33
010001000000000000000000	RULE 1	X STAR	0.78	-28.33
010010000000000000000000	RULE 1	X STAR	3.98	-28.33
010100000000000000000000	RULE 1	X STAR	-0.62	-28.33
011000000000000000000000	RULE 3 CONST 3	X STAR	-23.13	-28.33

THE OPTIMAL VALUE OF THE OBJECTIVE IS 11.90 .

THE OPTIMAL FEASIBLE VECTOR OF PROJECTS FOLLOWS.

1, 4,

Appendix B

Problem Formulation and Preparation of Data Cards.

First, define the notation used as follows:

NP = Number of projects.

NB = Number of budget constraints.

NOC = Number of other constraints.

BUD m = Limit for budget constraint m.

LGGI k = Limit for other constraint k.

In this report, the dimensions and formats are such that n must not be greater than 24, NB not greater than 6 and NOC + NB not greater than 25.

To formulate a problem for solution with these programs, write it in the following form.

$$
\begin{array}{llllllll}
\text{Maximize} & z = a_1x_1 & + a_2x_2 & + a_3x_3 & + \dots & + a_nx_n & & \\
\text{Such that} & b_{11}x_1 & + b_{12}x_2 & + b_{13}x_3 & + \dots & + b_{1n}x_n & \leq & \text{BUD } 1 \\
 & \vdots & \vdots & \vdots & & \vdots & & \vdots \\
 & b_{NB1}x_1 & + b_{NB2}x_2 & + b_{NB3}x_3 & + \dots & + b_{NB}x_n & \leq & \text{BUD NB} \\
 & c_{11}x_1 & + c_{12}x_1 & + c_{13}x_3 & + \dots & + c_{1n}x_n & \leq & \text{LGGI } 1 \\
 & \vdots & \vdots & \vdots & & \vdots & & \vdots \\
 & c_{NOC1}x_1 & + c_{NOC2}x_2 & + c_{NOC3}x_3 & + \dots & + c_{NOCn}x_n & \leq & \text{LGG1 NOC} \\
 & x = 0,1 & & & & & &
\end{array}
$$

If any constraint is $\geq$ in the problem as formulated on paper, multiply it by -1 to reverse the inequality as the program was written with the constraints all $\leq$.

Since there are three programs, there are three forms for the data deck. Each data deck has several types of cards. A description of each data deck and its card types along with specific instructions for preparation follows.

Data Deck for the Strong Deterministic Form.

Description.

The data deck for this program contains five types of cards.

Type 1 This type gives the number of projects NP, number of budget constraints NB, and number of other constraints NOC. There is only one card of this type in the data deck.

Type 2 This type gives the objective and budget constraint coefficients, a_i, b_{1i}, ..., b_{NBi} for each project. There is one card of this type for each project.

Type 3 This type gives the limit and number of non-zero coefficients for each other constraint.

Type 3A This type gives the project number and value of each non-zero coefficient. Each type 3A card contains up to five projects and coefficients. Additional cards of this type are used as needed for constraints with more than five non-zero coefficients.

One type 3 card followed by one or more type 3A cards are used for each other constraint.

Type 4 This type gives the budget limits. There is only one card of this type in the data deck.

Preparation.

All entries are to be right justified in their respective fields and written as integers. If an entry is zero, it may be entered as zero or left blank but if it is left blank, the next non-zero entry must be correctly placed.

Type 1 Write number of projects, NP, in columns 1 - 6.
Write number of budget constraints, NB, in columns 7 - 12.
Write number of other constraints, NOC, in columns 13 - 18.

Type 2 Write a_i in column 1 - 6.
Write b_{1i} in column 7 - 12.
Write b_{2i} in column 13 - 18.
⋮
Write b_{6i} in column 36 - 42.
There are as many entries as there are budget constraints.
Repeat for each project in order, each on a separate card.

Type 3 Increase the index of each project by 1.
Write LGGI in column 1 - 6.
Write number of non-zero coefficients in column 7 - 8.

Type 3A For the first non-zero coefficient, write its project number (increased by 1 above) in column 1 - 2, and its coefficient in column 3 - 8. Write the project number for the next non-zero coefficient in column 9 - 10 and the coefficient in column 11 - 16. Repeat for up to five values. If there are more than five, continue the procedure on the next card for as many cards as are needed.

Type 4 Write the first budget limit BUD 1 in column 1 - 6.

Write the second budget limit BUD 2 in column 7 - 12.

Continue, six columns per budget for each budget constraint.

Data Deck for the Weak Deterministic Form.

Description.

The data deck for the Weak Form is identical to that for the Strong Form except for the addition of two cards. After running a problem with the program for the Strong Form, if a search for additional optimal solutions is desired, put the type 5 and type 6 cards ahead of the cards already used in the Strong Form and use the deck with the Weak Form program.

Type 5 This type gives a known optimal value of the objective. There is only one card of this type.

Type 6 This type gives a known optimal vector. There is only one card of this type.

Preparation.

Type 5 Write the known optimal value (the value found in the program for the Strong Form) in column 1 - 9.

Type 6 Write the known optimal vector of projects in column 1 - 24.

Data Deck for the Probabilistic Form.

Description.

The data deck for the probabilistic form contains six types of data cards.

Type 1A This type is similar to type 1 for the Deterministic form except that there is a fourth entry giving the number of values for A, the risk aversion coefficient. There is only one card of this type in the data deck.

Type 2 Identical to type 2 for the deterministic programs.

Type 3,3A Identical to type 3 and type 3A for the deterministic programs.

Type 4 Identical to type 4 of the deterministic programs.

Type 7 This type gives the entries in the variance/covariance matrix. There is one card or one set of cards for each constraint.

Type 8 This type gives the values for the Risk Aversion Coefficient.

Preparation.

Type 1A Prepare this card exactly as for the deterministic programs but write the number of values for the risk aversion coefficient in columns 19 - 24.

Prepare the cards or types 2, 3, 3A, and 4 exactly as for the deterministic programs.

Type 7 The entries for this type are in F6.2.

Write the value for variance/covariance for project one in column 1 - 6 with the decimal in column 4, for project two in column 7 - 12 with the decimal in column 10, continuing in this manner for up to five values. If there are more than five projects, continue on as many cards as needed. Repeat on new cards for each row of the matrix.

Type 8 The entries for this type are in F6.2 also.

Write the first value for A in columns 1 - 6, with the decimal in column 4. Continue using six columns per entry for up to five values per card.

Bibliography

(1) Ashour, S. and Char, A. R., "Computational Experience on Zero-One Programming Approach to Various Combinatorial Problems," J. Operations Research Society of Japan, Volume 13, Number 2, Oct. 1970.

(2) Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Management Science, Volume 12, Number 3, 1965.

(3) Balinski, M. L., "On Recent Developments in Integer Programming," Proceedings of the International Symposium in Mathematical Programming, Princeton: Princeton University Press, 1970.

(4) Beale, E. M. L., "Survey of Integer Programming," Operational Research Quarterly, Volume 16, Number 2, 1965.

(5) Bellman, R., Dynamic Programming. Princeton: Princeton University Press, 1957.

(6) Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs, Recent Advances in Mathematical Programming, New York: McGraw-Hill, 1963.

(7) Hammer, P. L., and Rudeaunu, S., "Pseudo-Boolean Programming," Operations Research, Volume 17, Number 2, 1969.

(8) Hillier, F. S. and Lieberman, G. F., Introduction to Operations Research, San Francisco: Holden-Day, Inc., 1967.

(9) Johnson, J. E., University Algebra, Englewood Cliffs, N. J., Prentice-Hall, Inc., 1966.

(10) Lawler, E. L. and Bell, M. D., "A Method for Solving Discrete Optimization Problems," Operations Research, Volume 14, 1966.

(11) Lawler, E. L. and Wood, D. E., "Branch and Bound Methods, A Survey," Operations Research, Volume 14, 1966.

(12) Lemke, C. E. and Spielberg, K., "Direct Search Algorithms for Zero-One and Mixed Integer Programming," Operations Research, Volume 15, Number 5, 1967.

(13) Mao, J. C. T., Quantative Analysis of Financial Decisions. London: The Macmillan Company, Collier-Macmillan Limited, 1967.

(14) Mao, J. C. T. and Wallingford, B. A., "An Extension of Lawler and Bell's Method of Discrete Optimization with Examples from Capital Budgeting," Management Science, Volume 15, Number 2, Oct. 1968.

(15) Salkin, H. and Spielberg, K., "Adaptive Binary Programming," IBM New York Scientific Center Report No. 320-2951, 1968.

(16) Tillman, F. A., "Optimization by Integer Programming of Constrained Reliability Problems with Several Failure Modes," I.E.E.E. Transactions On Reliability, R-18, 2, 1969.

(17) Tillman, F. A. and Liittschwager, "Integer Programming Formulations of Constrained Reliability Problems," Management Science, Volume 13, Number 11, 1967.

(18) Wagner, H. M., Principles of Operations Research. Englewood Cliffs, N. J.: Prentice-Hall Inc., 1969.

(19) Weingartner, H. M., Mathematical Programming and the Analysis of Capital Budgeting Problems. Englewood Cliffs: Prentice-Hall, Inc., 1963.

(20) Weingartner, H. M., "Capital Budgeting of Interrelated Projects: Survey and Synthesis," Management Science, Volume 12, Number 7, March 1966.

AN INVESTIGATION OF THE USE OF THE LAWLER-BELL ZERO-ONE ALGORITHM IN SOLVING THE WEINGARTNER MODEL OF THE CAPITAL BUDGETING PROBLEM

by

WILLIAM JAMES MAURICE THOMAS

B.S., The University of Texas, 1955
B.A., The University of Texas, 1955
M.S., Kansas State Teachers College, 1965

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1972

The Lawler-Bell algorithm for zero-one integer programming was coded in FORTRAN IV and its efficiency, as measured by computing time and number of vectors enumerated, was observed in solving Weingartner-type capital budgeting problems. The program was written so that it could accept data from typical problems in which some function of net present value is to be maximized. To do this, it was necessary to transform the capital budgeting problem, which is a maximization problem, into a minimization problem solvable by the Lawler-Bell algorithm. This was done with a linear transformation of coefficients in the objective function and constraints.

A second program was written with the decision statements in the algorithm weakened in order to find all alternate optimal solutions. It is believed that this is a new form of the algorithm.

A third program was written for the extension of the algorithm to the probabilistic capital budgeting case. Results using different values for the risk aversion coefficient were compared.

A method of incremental sensitivity testing was developed and investigated in which the resource limits for the budget constraints were varied, giving corresponding incremental changes in the objective value and optimal project vectors. It is believed that this is a new application of the algorithm.

The algorithms were found to be very efficient in reducing the number of vectors to be enumerated, as compared with the number required for complete enumeration. For fewer than about twenty projects the algorithms are also efficient ineconomizing computing time but for larger problems the computing time becomes excessive from a practical standpoint.

It is concluded that these algorithms are useful for moderately large problems ($n \leq 20$ projects, $n' \leq 10$ constraints) but not for larger ones because of the greater computing time required.