In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

[illegible]

7/25/68

A QUADRATIC PROGRAMMING APPROACH TO THE SOLUTION OF THE 0-1 LINEAR INTEGER PROGRAMMING PROBLEM

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

by

Jeffery Lynn Kennington

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Industrial and Systems Engineering

Georgia Institute of Technology

August, 1970

A QUADRATIC PROGRAMMING APPROACH TO THE SOLUTION OF THE 0-1 LINEAR INTEGER PROGRAMMING PROBLEM

Approved:

[illegible]
Chairman

[illegible]

[illegible]

[illegible]

Date Approved by Chairman: 8/31/70

ACKNOWLEDGMENTS

I gratefully acknowledge the assistance, council, and encouragement of Dr. D. E. Fyffe. Without his help and encouragement, this research would not have reached a successful completion. The numerous hours Dr. Fyffe spent helping the author are certainly appreciated.

Special thanks are also due to Drs. C. M. Shetty and J. J. Jarvis who served on the reading committee. Both made valuable suggestions concerning the proofs of many of the theorems which greatly improved the original manuscript.

Finally, I would like to express my thanks to Mrs. Peggy Weldon for an excellent typing job.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SUMMARY

This thesis presents a new algorithm for the solution of the 0-1 linear integer programming problem. The algorithm is a specialization and modification of a quadratic programming algorithm by Klaus Ritter for the maximization of a convex quadratic function subject to linear constraints. The algorithm as presented also incorporates the branch-and-bound algorithm of Egon Balas for the solution of the 0-1 linear integer programming problem. Finiteness and optimality of the algorithm are proven, and the computational experience developed at this writing is reported.

CHAPTER I

INTRODUCTION

Since the development of the simplex method for solving linear programming problems by George Dantzig in 1947, researchers in all fields of applied mathematics have been drawn to the interesting field of mathematical programming. One particular area of mathematical programming which is, at present, or prime interest to operations researchers, industrial engineers, and applied mathematicians, is that of discrete or integer programming. The best explanation for this interest is that numerous practical problems can be formulated as integer linear programming problems. Scheduling, capital budgeting, resource allocation, and distribution problems are but a few examples of where integer linear programming problems arise.

Before 1959, problems which required integer solutions were usually solved by rounding optimal linear programming variables to the nearest integer. Although this is often adequate, especially when the variables are relatively large, it is difficult in some cases to obtain near optimal solutions to the discrete problem. It is often difficult if not impossible to see in which way the rounding should be done to maintain feasibility. After a solution has been rounded, it may even become necessary to change one or more of the variables by one or more units to regain feasibility. A good example demonstrating the difficulty which can be encountered when rounding optimal linear programming solutions is found in Hillier and Lieberman (18). Because of this difficulty, researchers have turned to other means of solving linear optimization problems which have discrete requirements. The new methodology developed over the last twelve years for the solution of these types of problems can be grouped under the single title, linear integer programming. It is this area of mathematical programming of concern for this research.

In the field of linear integer programming, there are four problems of interest. The first and most restrictive is the pure 0-1 integer programming problem. Problem (1.1) is this type.

(1.1)
$$\begin{aligned} \text{Max:}\quad & f(\underline{x}) \\ \text{Subj:}\quad & A\underline{x} \leq \underline{b} \\ & x_i = 0 \text{ or } 1 \text{ for all } i. \end{aligned}$$

The second is the mixed 0-1 integer programming problem. Problem (1.2) is of this type.

(1.2)
$$\begin{aligned} \text{Max:}\quad & f(\underline{x}) + g(\underline{y}) \\ \text{Subj:}\quad & A\underline{x} + B\underline{y} \leq \underline{b} \\ & \underline{y} \geq \underline{0} \\ & x_i = 0 \text{ or } 1 \text{ for all } i \end{aligned}$$

The third is the pure integer programming problem. Problem (1.3) takes this form.

(1.3)
$$\begin{aligned} \text{Max:}\quad & f(\underline{x}) \\ \text{Subj:}\quad & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \\ & x_i \text{ is integer for all } i \end{aligned}$$

The last and most general type of problem is the mixed integer programming problem. Problem (1.4) is this type problem.

(1.4)
$$\begin{aligned} \text{Max:} \quad & f(\underline{x}) + g(\underline{y}) \\ \text{Subj:} \quad & A\underline{x} + B\underline{y} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \\ & \underline{y} \geq \underline{0} \\ & x_i \text{ is integer for all } i \end{aligned}$$

For problems (1.1) through (1.4)

$\underline{x}$ is a n-component column vector

$\underline{y}$ is a p-component column vector

$\underline{b}$ is a m-component column vector

$\underline{0}$ is a n-component column vector of zeroes

A is a m x n matrix

B is a m x p matrix

f is a mapping from E^n into E^1

and g is a mapping from E^p into E^1.

Each of the four linear integer programming problems is important in its own right and practical applications for each frequently appear in the literature. These practical applications often require the solution to very large integer programming problems and consequently a good integer programming algorithm must be able to solve large problems. A comparison between two algorithms is usually made by comparing the computational times of each algorithm on a set of test problem. Since the success of an algorithm is dependent upon the speed of solution, integer programming algorithms are typically designed for one and only one of the four integer programming problems. This enables one to take advantage of any special characteristics of the problem. The special structure of the pure 0-1 integer programming problem enabled the author to reformulate this problem in terms of a continuous programming problem which could be solved. Therefore, this research is specifically concerned with the pure 0-1 integer programming problem.

The objectives of this research are as follows:

(1) to develop an algorithm for the pure 0-1 linear integer programming problem via a quadratic programming approach,

(b) to develop a computer code for this algorithm,

and (c) to solve several published test problems and report the computational times for the new algorithm.

The algorithm is based on a quadratic programming approach in which the integer problem is solved by solving a related quadratic programming problem. It is shown in Chapter III, that the optimal solution to the related quadratic problem is, in a special case, the optimal solution to the integer problem. The algorithm to solve this special quadratic programming problem is based on a partitioning procedure. At each stage a quadratic programming problem is partitioned into two quadratic problems by the introduction of a constraint which divides the feasible region into two nonempty sets. One of the regions and the original objective function are solved as a quadratic programming problem and the remaining region is transformed into a new space. The branch-and-bound machinery of Balas (1) is used to obtain a feasible integer point about which the transformation occurs. Since the transformed problem has the identical form of the original problem, the partitioning procedure may be reapplied. The algorithm terminates when the branch-and-bound machinery gives the signal that no integer point exists in the remaining feasible region. The algorithm is developed in Chapter IV and the computational experience is reported in Chapter V.

The notation and conventions used in this study are now presented. Matrices are denoted by upper case Latin letters and the elements of a matrix by the corresponding lower case Latin letters with two subscripts. Lower case Latin letters underlined denote column vectors. Lower case Latin letters with a single subscript denote an element of the vector with the same name. Sets are denoted by upper case Greek and Latin letters and scalars by both lower case Greek and Latin letters.

The symbols $\underline{0}$ and $\underline{1}$ denote the zero and one column vectors respectively. The notation $\Gamma \subset \Delta$ implies that Γ is a subset of Δ and $\Gamma \neq \Delta$. Non-negative is expressed by $\underline{c} \geq \underline{0}$. A' denotes the transposition of matrix A. The terms, integer programming and discrete programming are used interchangeably throughout this text and both refer to the problems of types (1.1) through (1.4). All integer problems considered in this text shall be composed of a linear objective function and linear constraints. Any special notation not given here is defined as needed.

CHAPTER II

SURVEY OF INTEGER PROGRAMMING

This chapter attempts to survey the current state of theory and methodology available for the solution of integer programming problems. Another survey was recently given by Balinsky and Spielberg (4) with over 200 integer programming articles referenced. Many of the articles presented there are not discussed here, however; it is felt that the most important work covered by Balinsky and Spielberg is discussed here as well as some new work which is not given in their survey, but has recently come to the attention of the author. The work of Graves, Whinston, Hammer, and the most recent work of Balas fall into this category.

To facilitate discussion of the progress in this field, the approaches have been separated into five categories as follows:

(a) Cutting Plane Methods

(b) Branch-and-Bound Methods

(c) Stochastic Methods

(d) Boolean Methods

and (e) Duality Theory.

Dynamic programming approach has been proposed for the solution of integer programming problems by Bellman (5). At the present time this approach does not appear promising as a feasible approach for discrete programming. Since there has been great progress in other areas, this approach has been omitted from this survey.

The algorithm developed in this research for the solution of the pure 0-1 integer programming problem falls into a separate category which we will call the quadratic programming approach. The term, "quadratic programming method," refers to the algorithm presented in Chapter IV of this text. Even though the quadratic programming method is a unique approach for the solution to the 0-1 programming problem, it is similar in many respects to other integer programming algorithms. These differences and similarities are presented at logical places in this survey.

Cutting Plane Techniques

The idea of introducing cutting planes to eliminate unwanted feasible solutions from a special structured linear programming problem was first advanced by Dantzig, Fulkerson and Johnson (8) in 1954. This work resulted in an algorithm for the solution of the traveling salesman problem.[†] Their approach requires reformulation of the original problem into a slightly different problem with the characteristic that any optimal solution to the revised problem is also optimal for the original problem. However, feasible solutions to the revised problem were not necessarily feasible for the original problem.

The revised problem was solved and its solution inspected to determine if it lay within the feasible region for the original problem. If this solution was feasible, the optimal solution had been found. If this was not the case, a cutting plane (constraint) reduced

[†]Simply stated the traveling salesman problem is as follows: given a set of cities, find the minimum distance route which begins at one city, passes through each of the other cities exactly once, and returns to the original city.

the feasible region of the revised problem so that the current solution was no longer feasible for either problem. The new revised problem was again solved and the process repeated.

Markowitz and Manne (22) in 1957 applied this approach to obtain the solution of general integer programming problems. Their work resulted in a general approach rather than an automatic algorithm and can be summarized as follows: first, solve the integer problem as a linear programming problem by ignoring the integer restrictions. If the solution is not in integers, judgment and ingenuity are used to formulate a new constraint that can be shown to be satisfied by the still unknown integer solution, but not by the noninteger solution already found. This additional constraint is added to the original ones and the simplex technique is again applied. The previous noninteger solution will be infeasible and a new solution will be generated. If this solution is noninteger, the process is repeated until the first integer solution is found. Since there was no systematic method for generating the new constraints, this method is a general approach rather than an algorithm.

Markowitz and Manne solved the dual rather than the primal at each iteration. Each new problem begins with a super optimal infeasible solution and proceeds toward feasibility. This is precisely the requirements of the dual simplex approach.

The following year, Ralph Gomory (12) developed a systematic method for new constraint generation which turned the cutting plane approach into an automatic algorithm. Gomory also proved that these cutting planes guarantee that an integer solution (if one exists) can be found in a finite number of steps.

These cutting planes are derived from the coefficients of the simplex tableau at the completion of each cycle. Recall that any tableau represents a set of equations of the form

$$x_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ij}x_j = b_i \tag{2.1}$$

where x_i is a basic variable, y_{ij} is the $(i,j)^{th}$ element of the tableau and x_j is the j^{th} variable. If x_j is a basic variable, then y_{ij} will equal zero. Therefore (2.1) reduces to

$$x_i + \sum_{j\varepsilon R} y_{ij}x_j = b_i \tag{2.2}$$

where R is the set of nonbasic variables. The equation (2.2) can be rewritten into the form

$$x_i = [b_i]_I + [b_i]_F - \sum_{j\varepsilon R} ([y_{ij}]_I x_j + [y_{ij}]_F x_j) \tag{2.3}$$

where $[z]_I$ denotes the largest integer less than z and $[z]_F$ denotes the fraction such that $[z]_I + [z]_F = z$, and $0 \leq [z]_F < 1$. Then,

$$x_i - [b_i]_I + \sum_{j\varepsilon R} [y_{ij}]_I x_j = [b_i]_F - \sum_{j\varepsilon R} [y_{ij}]_F x_j \tag{2.4}$$

is obtained by rewriting (2.3). It follows that any integer solution to x_i implies that the left side of (2.4) is integer. Therefore for any feasible integer solution

(2.5) $$[b_i]_F - \sum_{j\varepsilon R} [y_{ij}]_F x_j$$

must be integer. Since $\sum_{j\varepsilon R} [y_{ij}]_F x_j$ can not be negative and $0 < [b_i]_F < 1$, (2.5) can not be a positive integer. Therefore,

(2.6) $$- \sum_{j\varepsilon R} [y_{ij}]_F x_j \leq - [b_i]_F \ .$$

It follows that if x_i is to be integer, then the constraint (2.6) must also be satisfied. After adding a slack variable s_p, one obtains the cutting plane

(2.7) $$s_p - \sum_{j\varepsilon R} [y_{ij}]_F x_j = -[b_i]_F$$

which is known as a Gomory cut.

Gomory's algorithm for the solution of integer programming problems can be summarized as follows: first solve the integer programming problem as a standard linear programming problem by ignoring the integer restrictions. If the optimal solution is integer, the integer problem has been solved. If this is not the case, then a Gomory cut is developed from some row of the final tableau which gave the linear programming solution. Since it is desirable to make the largest cut possible, the usual rule for selecting the row from which to derive the Gomory cut is to choose that row which yields the largest $[b_i]_F$. This new constraint is appended to the current tableau and the dual simplex algorithm is applied to obtain another optimal feasible solution. If this new solution is integer, the problem has been solved. If this is not the case, a new cut is generated and the process is repeated until such time as an integer solution is obtained.

Later, Gomory (13) developed an all-integer integer programming algorithm that began with a problem statement in which all coefficients were integer and maintained this property throughout the solution process. Maintaining the integer property prevents the round-off errors which often occur when solving large problems on a computer. This algorithm also uses the idea of constraint generation, however; the new constraint is appended to the tableau immediately at each iteration. This new constraint is so constructed that it will be the row chosen for the leaving variable and the pivot element will always be minus one. This insures that if the coefficients in the original matrix are integer, that they remain integer from tableau to tableau. The computational experience thus far has shown little consistency in the time required to solve integer problems using this method.

A primal analogue to Gomory's all-integer algorithm was developed by Young (26) and is known as the simplified primal (all-integer) integer programming algorithm. Young's algorithm is built on Dantzig's simplex technique with the addition of a special row at each iteration. This special row is a Gomory cut and is appended to the tableau after the pivot column is chosen at each cycle. This cut is selected so that it will have a unit coefficient in the pivot column and will qualify as the pivot row. To show finiteness, Young imposed certain restrictions on the row used to generate the Gomory cut. At present there are no results available on the computational efficiency of Young's algorithm.

Balinski (4) predicted that the algorithm would not prove to be efficient. This prediction is based on the fact that the algorithm is made to work by imposing decision rules which insure finiteness, but are not necessarily geared to natural measures of progress toward optimality.

As noted from the discussion of the previous algorithms, an optimal feasible solution to a pure integer programming problem must exhibit three properties:

(a) it must be optimal, i.e. $\underline{c}'\underline{x}^* \geq \underline{c}'\underline{x}$ for all $\underline{x} \varepsilon \Gamma = (\underline{x} : g(\underline{x}) \leq \underline{b}, \ \underline{x} \geq \underline{0})$

(b) it must be feasible, i.e. $\underline{x}^* \varepsilon \Gamma$, and

(c) all variables must assume integer values.

Each of the cutting plane methods presented above have maintained two of the three properties while achieving tableau-to-tableau progress toward satisfaction of the remaining property. The quadratic programming method presented in Chapter IV maintains only one property while moving toward satisfaction of the other two. A comparison of the methods is as shown in Table 1.

Table 1. Comparison of Cutting Plane Methods and the Quadratic Method

Method	Property maintained at each cycle	Property improved at each cycle
1. Gomory's method of integer forms	a and b	c
2. Gomory's all integer method	a and c	b
3. Young's primal integer method	b and c	a
4. Quadratic programming method	b	a and c

The Gomory method differ from the quadratic approach primarily due to the fact that Gomory's methods are dual in nature whereas the quadratic approach resembles a primal method. The dual approach has the disadvantage that no feasible solution is obtained until the optimal solution is found. If this requires more iterations than can be afforded, the method yields no useful information. If, however, a primal method is employed, the best suboptimal feasible solution can be kept as the iterations progress so that at the end of some given time period the best suboptimal feasible solution found up to that time is available.

The last difference between the cutting plane methods and the quadratic approach is the special way in which additional constraints are derived for future iterations. For cutting plane methods, constraints are derived from another constraint, whereas in the quadratic approach, the constraints are derived from the objective function.

Branch-and-Bound Methods of Enumeration

The best known methods for obtaining solutions to integer programming problems fall under the general heading of branch-and-bound methods. The terms branch-and-bound, tree search, and implicit enumeration are used interchangeably in the literature, however; they refer to the same general technique when used for solving discrete programming problems. A good description of the enumerative substructure is given by Glover (11) and is essentially this. The solution space of a discrete programming problem can be readily represented by a tree diagram. The general procedure involves tracing some path of this tree until either a feasible solution is obtained, or a node is reached which yields information that all solutions in which that node is included may be eliminated from further consideration. When one of the above two conditions are met, the process backtracks to the node preceding the one just eliminated and traces out another path, if one exists. If none exists at this node, the process backtracks to the next node and attempts to find another path not yet eliminated. The process continues until it has backtracked to the starting node, and information is obtained that eliminates the necessity to trace any more paths. The enumeration procedure terminates at this point.

The first automatic branch-and-bound method for the solution of integer programming problems was advanced by Land and Doig (19) in 1960. With their general approach, all branching is accomplished by adding constraints to the continuous problem and obtaining the general linear programming solution. All bounding is set by the value of the objective function at each branch.

Their algorithm initially solves the integer problem as a continuous linear programming problem. If this solution happens to be integer, the solution to the integer problem is as given. If this is not the case, the algorithm chooses a variable restricted to integer solutions which is noninteger at the optimal of the unrestricted problem, and creates two new problems which do require this variable to be integer.

For example, suppose x_k is a variable restricted to discrete values for some integer programming problem. Further suppose that $\bar{x}_k$ is the value of x_k at the optimal solution of the unrestricted problem and that $\bar{x}_k$ is noninteger. Let $[\bar{x}_k]_I$ denote the largest integer less than $\bar{x}_k$. Then it follows that two new problems (branches) can be created by adding the constraint $x_k = [\bar{x}_k]_I$ to the original constraint set to form problem (1) and by adding $x_k = [\bar{x}_k]_I + 1$ to the constraint set to form problem (2). Suppose that the problem of interest is a maximization problem, and that the value of the objective function at the optimal solutions of (1) and (2) are θ_1 and θ_2 respectively. If θ_o represents the value of the objective function of the unrestricted problem at optimality, then it follows that $\theta_o \geq \theta_1$ and $\theta_o \geq \theta_2$. The objective functions of all branches, have the property that at optimality, the value is less than or equal to the value of the objective function on any less restricted problem from which this branch originated. It is this property which allows Land and Doig to abandon branches without completely enumerating the entire branch.

Suppose that $\theta_1 > \theta_2$. Then a third problem is solved with the additional constraint $x_k = [\bar{x}_k]_I - 1$ appended to the original constraint set. Each noninteger node of the tree will have three leaving branches, if three logical constraints can be constructed as accomplished above. If $0 \leq \bar{x}_k \leq 1$, then the node may have only two leaving branches, but in general each node will give three new branches. These branches are always determined by the most desirable integer for some particular variable, and the integer on either side.

For the procedure of Land and Doig, the θ_i's are saved along with the necessary information to pursue any of the branches currently under consideration. The node with the largest θ (for a maximization problem) is examined to determine if a leaving branch will yield a feasible solution for the integer problem. If an integer solution is found, the solution is recorded and the corresponding θ becomes the lower bound. Once a lower bound is established, all branches which yield θ's less than the lower bound can be abandoned. If the node under consideration does not represent a feasible solution (in the integer sense), new branches are determined as above and the process is repeated. If some new integer solution is found which is preferred to one previously obtained, it is recorded as the current best solution and the corresponding θ becomes the new lower bound. If the process is continued, all branches will eventually be completely enumerated or abandoned, and the optimal feasible solution, if one exists, will have been found.

The Land and Doig branch-and-bound approach differs from that of the quadratic programming approach in several respects. The Land and Doig approach is applicable for any discrete problem whereas the quadratic approach presented in Chapter IV is only applicable for the 0-1 integer programming problem. The Land and Doig approach has characteristics of both dual and primal methods. Their method begins with a superoptimal solution and proceeds toward a feasible solution, which is exactly a dual approach. When the first feasible solution (in the integer sense) is found, one records this solution and attempts to locate another feasible solution more desirable than the one currently known, which resembles the primal approach. The quadratic method of this research is basically a primal approach. When the Land and Doig approach eliminates a set of solutions from further consideration, all abandoned solutions lie on a single branch of the solution tree. The quadratic approach can eliminate several branches with a single constraint or parts of several branches with a single constraint. Finally the Land and Doig approach requires that a linear programming problem be solved at each iteration.

In 1965 Egon Balas (1) presented a different approach based on the general branch-and-bound technique to obtain the solution of 0-1 pure integer programming problems. His additive algorithm gave a method for systematically enumerating part of the solutions of the 0-1 problem, and examining them in such a way as to ensure that by explicitly enumerating a relatively small number of solutions, it had implicitly examined all elements of the solution set.

The additive algorithm begins at some starting node on the tree of solutions and applies two tests. The first test determines if the best completion of this node is feasible. If this completion is feasible, this node and all of its descendants can be fathomed (i.e. discarded from further consideration). If this is not the case, one applies the second test which attempts to determine that no feasible completion at the node under consideration is better than any previously found. If the second test succeeds, then the node under consideration may be fathomed along with its descendants. When a node is fathomed, the process backtracks to the last node visited and reapplies the above two tests. The rules for the choosing of successive nodes are such that no descendant of any abandoned node will ever be reconsidered. If the node under consideration cannot be fathomed, then the leaving arc chosen is the one which leads to a new node which most reduces the total infeasibility of the solution. The tests are applied at the new node and the cycle is repeated. The procedure terminates when it backtracks to the starting node and information is obtained that eliminates the necessity to trace out of any more branches of the tree.

If the examination of a particular node is defined as an iteration, then the efficiency of the additive algorithm is dependent on the number of iterations required before an optimal feasible solution is found or the absence of a feasible solution is established. One way of improving the efficiency of the additive process is to increase the strength of the tests applied and thereby eliminate larger portions of the tree than would be eliminated under the rules suggested by Balas. The multiphase-dual-algorithm of Glover (11) was the first attempt to strengthen these tests of Balas. Glover introduced the idea of the <u>surrogate constraint</u> which was used in the test mechanism to establish restrictions on the problem which could not be determined from any individual constraint in isolation. Since it was computationally practical to apply the tests to only one constraint at a time, the surrogate constraint enabled Glover to sharpen the tests of Balas. This surrogate constraint of Glover is defined as a nonnegative linear combination of the original constraints in which at least one of the constraints is given a positive weight. The computational experience published by Glover is meager and he suggested that there may be better methods which could be used to compute this surrogate constraint which could significantly improve the multiphase-dual-algorithm. He reported that in some cases, the time required to compute the s-constraint (surrogate constraint) exceeded what savings it could produce.

A second contribution was made by Glover with the introduction of a new and efficient (in the sense of computer storage required) method of bookkeeping for the tree search. Geoffrion (9) then reformulated the algorithm of Balas using this general bookkeeping system presented by Glover. The new formulation required considerably less computer storage than the original version.

Shortly thereafter, Balas (2) presented a new algorithm (the filter method) which was a continuation of the additive algorithm. The filter method incorporated a filter mechanism to sharpen the tests of the basic algorithm and thereby reduce the size of the solution tree. The filter mechanism required the solution of a special o-1 programming problem with a single constraint. This new constraint is a special case of the surrogate constraint as defined by Glover. As of this writing, computational results for the filter method are not available.

The current best o-1 pure integer programming algorithm is the improved implicit enumeration approach of Geoffrion (10). The method applies the tree search bookkeeping system described in an earlier paper, Geoffrion (9), it incorporates the use of surrogate constraints defined slightly different from those of Glover, and relies on an imbedded linear programming problem to calculate the strongest possible surrogate constraints. The linear programming problem is so constructed, that the dual variables give information as to the existence of a binary infeasible surrogate constraint as well as feasible integer solutions better than any previously found. A computer code for this method was written and tested extensively on an IBM 7044. Geoffrion reports that the use of the imbedded linear program reduced solution times by a factor of about one hundred. He further reports that his computer code dramatically reduced the solution times of virtually every published test problem attempted, and sufficed to render the tested algorithm superior to the five other implicit enumeration algorithms for which comparable published experience was available.

The implicit enumeration method of Geoffrion is generally accepted to be the best of the methods available for solving 0-1 integer programming problems. This method differs from that of the quadratic approach in three main respects. First, the method of Geoffrion is based on a tree search, whereas the quadratic approach is based on a partitioning procedure. Second, the method of Geoffrion introduces surrogate constraints to the original constraint set which are redundant in the usual sense, whereas the constraints introduced by the quadratic approach tend to make the original constraints redundant. Lastly, the quadratic approach works in a transformed region whereas the implicit enumeration method works only in the original region. The similarities between the two methods are as follows: (a) both algorithms stop, only when all integer points have been either explicitly or implicitly considered, (b) both methods keep available a current best integer solution, and (c) both algorithms make full use of the additive algorithm of Balas.

A Stochastic Approach to Discrete Programming

Graves and Whinston (14) introduced a new approach for integer programming which incorporates the implicit enumeration bookkeeping scheme, but relies on the use of population statistics to eliminate large portions of the tree of solutions from further consideration. They consider the 0-1 programming problem as if it were an n-stage decision problem. Identical to the Glover approach, it attempts to fathom a node along with its descendants by either feasibility or optimality considerations. Graves and Whinston suggest an additional mechanism which can be used to fathom nodes based on the probability of the existence of the global optimum being a descendant of the node under consideration.

Recall that in the tree search algorithm, the nodes, other than the last node of a branch, represent partial solutions in which some of the variables are specified and the remainder are free to assume either of the values zero or one. Then if node k is some node of the solution tree, and j elements of $\underline{x}$ have been specified, then n-j elements are free. Then it follows that at the $(k + 1)^{st}$ node (a descendant of the k^{th} node) one of the n-j free variables will be set at either zero or one. There are two considerations which may be taken into account when selecting this variable to be fixed as well as determining the value to which it is assigned. First there is the local or immediate effect. By selecting $x_r = 1$, the objective function of the new node is increased by c_r and each constraint $(i = 1,2,\ldots,m)$ is altered by a_{ir}, where a_{ir} is the $(i,r)^{th}$ element of the matrix which determines the constraint set. Secondly, alternate choices for the remaining unspecified variables are differently restricted because of this choice. The first consideration can easily be taken into account and is used to guide the progress of the additive and multiphase-dual algorithms. The second consideration is difficult to isolate and is discovered eventually by local considerations. If one were able to evaluate the second factors exactly, it would be possible to assign at each step the correct value and solve the problem exactly in n steps. Since this is not possible, the next best thing to knowing exactly the values of the completions is to know them almost surely. Graves and Whinston suggest the use of probability theory to obtain good information at a fraction of the computational cost required to obtain exact information.

They have successfully derived the probability of the existence of a feasible completion of any particular node which yields a more desirable local optimum than one previously known. They use these probabilities in their confidence level implicit enumeration scheme. This algorithm fathoms a node along with its descendents once it is established that there is less than an α percent chance that this branch contains a better feasible solution than one already known. The constant α may be set at any level and is left to the discretion of the user.

It appears that the general approach of Graves and Whinston has merit especially for very large practical applications. As of this writing, only limited computational experience is available for the approach. It should be noted that this approach can not guarantee that the optimal solution will be found.

A Boolean Approach to Discrete Programming

A fourth approach which is a variation on the branch-and-bound techniques was recently presented by Hammer (17). His algorithm is known as the boolean branch-and-bound method for the solution of 0-1 integer programming problems. The boolean approach constructs a solution tree quite different from that considered by Glover and Geoffrion. The boolean solution tree may have numerous branches leaving each node, whereas the implicit enumeration tree has but two departing branches at any particular node. Branches of the implicit enumeration tree are determined by allowing one free variable to assume the values of zero or one. The boolean tree uses the concept of a directrix to determine the branches at each node.

A directrix is determined by consideration of each constraint taken in the form

(2.8) $$a_{i1}x_1 + \dots + a_{in}x_n \geq b_i \quad \text{where}$$
$$a_{i1} \geq a_{i2} \geq \dots \geq a_{in} \geq 0 .$$

Let the partial sums of the constraint coefficients be denoted by

(2.9) $$\begin{aligned} S_n &= a_{in} \\ S_{n-1} &= a_{in-1} + a_{in} \\ &- - - - - - - - - - \\ S_1 &= a_{i1} + \dots + a_{in} . \end{aligned}$$

For a particular constraint the following cases can be distinguished.

(1) $b_i \leq 0$

(2) $b_i > 0,\ S_1 < b_i$

(3) $b_i > 0,\ S_1 = b_i$

(4) $b_i > 0,\ S_1 > b_i$

The directrix (D) will be defined equal to one if the inequality (2.8) is satisfied, otherwise it will equal zero.

In case 1, the inequality (2.8) always holds and $D = 1$. In case 2, the inequality is never satisfied and $D = 0$. In case 3, the inequality holds if and only if all $x_j = 1$; $j = 1,2,\ldots,n$. For equalities of case 4 type, the sums developed in (2.9) must be considered. Let r be the greatest index for which $S_r \geq b_i$.

$$S_1 \geq \cdots \geq S_r \geq b_i \geq S_{r+1} \geq \cdots \geq S_n .$$

This implies that if $x_1 = \ldots = x_r = 0$, the inequality (2.8) is not satisfied. It follows that a necessary condition for (2.8) to hold is that $\max (x_1,\ldots,x_r) = 1$. Therefore, for case 4 let $D = \max (x_1,\ldots,x_r)$. The D as defined in each of the four cases is called the <u>directrix of an inequality</u>.

Hammer defines the directrix (Δ) of a system of inequalities as the product of the directrices of the inequalities of the system. A <u>necessary condition</u> for a binary vector to be feasible for a set of inequality constraints is that its system directrix be one. Note that this condition is necessary but not sufficient.
For example the constraint set

$$12\bar{x}_2 + 8x_4 + 7\bar{x}_1 + 5\bar{x}_3 + 3x_5 \geq 17$$

$$7x_2 + 6\bar{x}_3 + 6x_1 + 4\bar{x}_6 + 4x_4 \geq 12$$

$$8\bar{x}_2 + 4x_6 + 3\bar{x}_5 + 2x_1 \geq 7$$

where $\bar{x}_i = 1 - x_i$ gives the following inequality directrices

$$D_1 = \max(\bar{x}_2,\ x_4)$$

$$D_2 = \max(x_2, \bar{x}_3, x_1)$$

$$D_3 = \max(\bar{x}_2, x_6)$$

and the system directrix reduces to

$$\Delta = \max(x_1\bar{x}_2,\ x_1x_4x_6,\ \bar{x}_2\bar{x}_3,\ \bar{x}_3x_4x_6,\ x_2x_4x_6)\ .$$

Each of the above partial solutions represent a branch extending from the starting node of the boolean solution tree. The branching is accomplished by tracing some branch chosen from the above set, and bounding is set, as usual by the objective function value of the best solution currently known.

Unfortunately, there are no published results on the computational efficiency of the boolean branch-and-bound technique. It appears to the author, that this branching mechanism coupled with the fathoming mechanism of Geoffrion could result in an algorithm potentially more efficient than either of the algorithms taken separately.

Duality Theory in Discrete Programming

The first duality formulation for linear integer programming problems was advanced by Balas (3) in 1967. This formulation is relatively new and at present no algorithm based on integer programming duality theory has appeared in the literature.

Balas shows that the linear mixed-integer programming problem is a special case of a certain minimax problem which has a Lagrangian type objective function, linear constraints, and some variables constrained to belong to an arbitrary set of real numbers. This minimax problem represents the primal in the duality formulation. The dual of this minimax problem is shown to be a problem of the same type, such that the dual of the dual is the primal. It is shown that the optimal solutions of both problems are identical and that a certain type of complimentary slackness holds.

If the linear mixed-integer problem of interest is denoted by (2.10), then the equivalent minimax problem given by Balas is as shown in (2.11). The problem (2.11) represents Balas' primal and (2.12) gives the corresponding dual.

(2.10)

$$\begin{aligned} \text{Max:}\quad & \underline{c}'\underline{x} \\ \text{Subj:}\quad & A\underline{x} + \underline{y} = \underline{b} \\ & \underline{x}, \underline{y} \geq \underline{0} \\ & x_j \ \text{ integer } j \in N_1 \end{aligned}$$

where $N = 1,\ldots,n$, $N_1 = 1,\ldots,n_1$

(2.11)

$$\begin{aligned} \min_{\underline{u}} \ \max_{\underline{x}} :\ & \underline{c}'\underline{x} + \underline{u}_1'\underline{y}_1 + \underline{u}_1'A_{11}\underline{x}_1 \\ \text{Subj:}\quad & A\underline{x} + \underline{y} = \underline{b} \\ & \underline{x}, \underline{u} \geq 0 \\ & x_j \ \text{ integer, } \ j \in N_1 \\ & u_j \ \text{ integer, } \ j \in M_1 \\ & y_j \ \text{ unrestricted, } \ j \in M \\ & \quad \geq 0, \ \ j \in M - M_1 \end{aligned}$$

(Primal)

where $M = 1,\ldots,m$, $M_1 = 1,\ldots,m_1$

$\underline{y}' = (\underline{y}_1', \underline{y}_2')$

$\underline{u}' = (\underline{u}_1', \underline{u}_2')$

and A_{11} is the first m_1 rows and first n_1 columns of A.

(2.12) $$\max_{\underline{x}} \min_{\underline{u}} : \underline{b}'y - \underline{x}_1'\underline{v}_1 + \underline{x}_1' A_{11}y_1$$

$$\begin{aligned} \text{Subj:} \quad & \underline{u}'A - \underline{v} = \underline{c} \\ & \underline{u}, \underline{x} \geq \underline{0} \\ & u_j \text{ integer}, \quad j \in M_1 \\ & x_j \text{ integer}, \quad j \in N_1 \\ & v_j \text{ unrestricted}, \quad j \in N_1 \\ & \quad \geq 0, \; j \in N - N_1 \end{aligned}$$

(Dual)

where $\underline{x}' = (\underline{x}_1', \underline{x}_2')$

$\underline{v}' = (\underline{v}_1', \underline{v}_2')$.

The partitioning of the vectors is made such that u_j is a component of $\underline{u}_1$ and y_j is a component of $\underline{y}_1$ if $j \in M_1$ and x_i is a component of $\underline{x}_1$ and v_i is a component of $\underline{v}_1$ if $i \in N_1$.

Even though a major contribution has been made by Balas in developing the above duality theory, efficient means must be determined for solving problems of the form (2.11) before this theory results in an efficient solution procedure.

CHAPTER III

JUSTIFICATION FOR THE QUADRATIC PROGRAMMING APPROACH TO THE SOLUTION OF THE 0-1 PURE INTEGER PROGRAMMING PROBLEM

The quadratic programming approach to the solution of the 0-1 integer programming problem is not a new idea. This basic approach was suggested most recently by Raghavachari (24); however, full development of the approach resulting in an algorithm has not appeared in the literature at this writing. It is a relatively straightforward task to construct a continuous quadratic programming problem such that the solution to the quadratic problem is also the solution to the integer problem. Finding an efficient solution procedure for this quadratic problem is not straightforward. A complete discussion of a solution technique for the nonlinear problem is deferred until Chapter IV. Chapter III is devoted to developing a justification for the quadratic approach.

Let us begin with a definition of the problem of interest for this research. The 0-1 pure integer programming problem is stated as follows:

$$\begin{aligned} &\text{Max:} \quad \underline{\bar{c}}'\,\underline{x} \\ &\text{Subj:} \quad A\,\underline{x} \leq \underline{b} \\ &\qquad\qquad x_j = 0 \text{ or } 1, \quad j = 1,2,\ldots,n; \end{aligned} \tag{3.1}$$

where $\underline{\bar{c}}$ is a n x 1 column vector

$\underline{b}$ is a m x 1 column vector

A is a m x n matrix

and $\underline{x}$ is a n x 1 column vector.

The development of this chapter assumes that $\bar{c}_i \geq 0$ for i = 1,2,...,n. If $\bar{c}_i$ is initially less than zero, one can set $\bar{x}_i = 1 - x_i$ to obtain a positive form of $\underline{\bar{c}}$.

The quadratic programming problem of interest is stated as follows:

(3.2)
$$\begin{aligned} \text{Max:}\quad & Q(\underline{x}) = \underline{\bar{c}}'\underline{x} - a(\underline{1}'\underline{x} - \underline{x}'\underline{x}) = \underline{c}'\underline{x} - \frac{1}{2}\underline{x}'C\underline{x} \\ \text{Subj:}\quad & A\underline{x} \leq \underline{b} \\ & \underline{x} \leq \underline{1} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

where "a" is some arbitrarily large positive constant

$\underline{c} = \underline{\bar{c}} - a\underline{1}$

$C = -2aI$.

Relationship Between the Discrete Linear Problem and the Continuous Quadratic Problem

Two observations about problems (3.1) and (3.2) are immediate.

(a) if $\underline{x}$ is feasible for (3.2) and $\underline{x}$ is integer, then $\underline{x}$ is feasible for (3.1),

and (b) if $\underline{x}$ is feasible for (3.1), then $\underline{x}$ is feasible for (3.2).

These observations simplify the proofs of the following three theorems.

Theorem 1

If $\underline{x}^*$ is an optimal solution to (3.2) and $\underline{x}^*$ is integer[1], then $\underline{x}^*$ is an optimal solution to (3.1).

[1] $\underline{x}^*$ integer if and only if every component of $\underline{x}^*$ is integer.

Proof. Let $\underline{x}^*$ be integer and an optimal solution to (3.2). Then $\underline{x}^*$ is feasible for (3.1). Let $\underline{z}$ be any other feasible solution to (3.1). Then $\underline{z}$ is feasible for (3.2). Hence $\underline{\bar{c}}'\underline{x}^* - a(\underline{1}'\underline{x}^* - \underline{x}^{*\prime}\underline{x}^*) \geq \underline{\bar{c}}'\underline{z} - a(\underline{1}'\underline{z} - \underline{z}'\underline{z})$. But $(\underline{1}'\underline{x}^* - \underline{x}^{*\prime}\underline{x}^*) = (\underline{1}'\underline{z} - \underline{z}'\underline{z}) = 0$ since $\underline{x}^*$ and $\underline{z}$ are integer. Therefore $\underline{\bar{c}}'\underline{x}^* \geq \underline{\bar{c}}'\underline{z}$, i.e. $\underline{x}^*$ is the solution to (3.1). This completes the proof of Theorem 1.

Theorem 2

If $\underline{x}^*$ is an optimal solution to (3.2) and $\underline{x}^*$ is noninteger,[1] then (3.1) has no solution.

Proof. Let $\underline{x}^*$ noninteger be the optimal solution to (3.2). Then $(\underline{1}'\underline{x}^* - \underline{x}^{*\prime}\underline{x}^*) < 0$. Hence $Q(\underline{x}^*) = \underline{\bar{c}}'\underline{x}^* - a(\underline{1}'\underline{x}^* - \underline{x}^{*\prime}\underline{x}^*) < 0$ for sufficiently large a. Suppose (3.1) has a feasible solution $\underline{z}$. Then $\underline{z}$ is feasible for (3.2) and $Q(\underline{z}) > 0$. Hence $Q(\underline{z}) > Q(\underline{x}^*)$ which contradicts the assumption that $\underline{x}^*$ is optimal for (3.2). Therefore (3.1) has no feasible solution. This completes the proof of Theorem 2.

Theorem 3

If (3.2) has no solution, then (3.1) has no solution.

Proof. Since the feasible region of (3.1) is a subset of the feasible region of (3.2) and since (3.2) has no feasible solution, then (3.1) has no feasible solution.

Remark 1 Summary of Theorems 1, 2, and 3.

a. If $\underline{x}^*$ is integer and is the optimal solution to (3.2), $\underline{x}^*$ is the optimal solution of (3.1).

b. If $\underline{x}^*$ is noninteger and is the optimal solution to (3.2), (3.1) has no solution.

[1] $\underline{x}^*$ noninteger if and only if for some i, $x_i^* \neq (0 \text{ or } 1)$.

c. If (3.2) has no solution, (3.1) has no solution.

Determination of the Penalty Cost Coefficient

For solving problem (3.2) either on a digital computer or by hand calculations, it is more convenient to set the constant "a" at some finite value. This section gives a method for determining a sufficiently large "a" so that the solution of (3.2) is forced sufficiently close to an integer solution, if one exists.

Problem (3.2) gives $Q(\underline{x})$ as follows:

$$Q(\underline{x}) = (\bar{c}_1 x_1 - ax_1 + ax_1^2) + (\bar{c}_2 - ax_2 + ax_2^2) + \dots + (\bar{c}_n - ax_n + ax_n^2) .$$

Considering only the x_1 terms, one obtains the function

$$F_1 = \bar{c}_1 x_1 + a(x_1^2 - x_1) .$$

The constant "a" is to be chosen so that $F_1(x_1 = 0) > F_1(d < x_1 < 1 - d)$ where d is the allowable deviation from 0 or 1 which will be tolerated. It follows that d and a must be chosen so that $a(d - d^2) > \bar{c}_1 d$ and $a[(1 - d) - (1 - d)^2] > \bar{c}_1(1 - d)$. Therefore, for any allowable deviation d, an a which satisfies

(a) $$a > \frac{\bar{c}_1 d}{d - d^2}$$

and (b) $$a > \frac{\bar{c}_1(1 - d)}{(1 - d) - (1 - d)^2}$$

will insure that the x_i variable will be forced to either 0 or $1 \pm d$ if one of these integer values yields a feasible solution.

The expressions can be simplified to the following forms

(c) $$a > \frac{\bar{c}_1}{1 - d}$$

and (d) $$a > \frac{\bar{c}_1}{d} \quad .$$

Since $\underline{\bar{c}} > 0$ and d is small, the expression (d) always dominates (c). Furthermore, the constant a must satisfy an expression of the form (d) for each variable $x_1, x_2, \ldots, x_n$. Therefore a must be chosen so that

$$a > \max_i (\bar{c}_i/d), \quad i = 1,2,\ldots,n.$$

For all example problems reported in Chapter V, a was chosen as shown below.

$$a = \max (\bar{c}_i/d) + 1, \; i = 1,2,\ldots,n.$$

This completes the justification for the quadratic programming approach to the solution of the 0-1 integer programming problem. The theorems of section 1 give the relationship between the integer and continuous problem and the development of section 2 gives the process whereby numerical problems of the form (3.2) may be formulated.

CHAPTER IV

AN ALGORITHM FOR THE SOLUTION OF THE PURE 0-1 INTEGER PROGRAMMING PROBLEM

It was shown in Chapter III that any pure 0-1 integer programming problem can be solved by solving a related quadratic programming problem. The following chapter presents the development which results in an algorithm for this related quadratic programming problem which is, in turn, an algorithm for the integer programming problem.

A definition of the quadratic problem to be solved is as follows:

$$\text{(4.1)} \qquad \text{Max:} \quad Q(\underline{x}) = \bar{\underline{c}}'\underline{x} - a(\underline{1}'\underline{x} - \underline{x}'\underline{x}) = \underline{c}'\underline{x} - \frac{1}{2}\underline{x}'C\underline{x}$$

$$\text{Subj:} \quad A\underline{x} \leq \underline{b}$$

$$\underline{x} \leq \underline{1}$$

$$\underline{x} \geq \underline{0}$$

where $\bar{\underline{c}} \geq 0$

a is an arbitrarily large positive constant

$\underline{c} = \bar{\underline{c}} - a\underline{1}$

$C = -2aI$

and I is a n x n identity matrix.

Since C is negative definite, the function $Q(\underline{x})$ is strictly convex. It is well known that the maximization of a convex quadratic function over a bounded and closed convex set is attained at one of its finitely many extreme points (15).

Furthermore problems of this type may have local optima at extreme points which are not global optima. This possibility of numerous feasible local optimum renders the problem (4.1) unsolvable by the well known method of Wolfe (28).

Klaus Ritter (25) has developed an algorithm to solve quadratic programming problems in which numerous feasible local optima may be present. His method is applicable to the maximization of any nonconcave quadratic function subject to linear constraints whereas the problem (4.1) has special characteristics which may be used to aid in increasing the efficiency of a general solution procedure. The two important characteristics of (4.1) not necessarily present in the problem considered by Ritter are as follows:

(a) $Q(\underline{x})$ is strictly convex

and (b) every feasible integer point is a local optimum.

Ritter's algorithm requires a feasible local optimum at each iteration. The method he uses to obtain this local optimum is very time consuming. Property (b) above allowed the author to draw from other resources to obtain this local optimum in a more efficient manner than that suggested by Ritter. The work presented in this chapter is Ritter's algorithm with modifications to take advantage of these special properties.

His approach requires partitioning of the feasible region of (4.1) into two regions such that each contains one or more of the finitely many feasible integer extreme points. The cutting plane which performs this partitioning is constructed so that the global optimum in one of the regions can be found. The objective function $Q(x)$, and the remaining region then become a new problem denoted $(4.1)_1$ which has the identical form of the original problem (4.1). The new problem $(4.1)_1$ differs from the problem (4.1) in that the feasible region of the former contains fewer of the finitely many integer extreme points than does the feasible region of the latter. Then the new problem $(4.1)_1$ is partitioned such that each of the resulting regions contains at least one of the remaining feasible integer extreme points. Again the cutting plane which performs this partitioning is constructed so that the global optimum in one of the new regions can be found. The objective function Q(x), and the remaining region then become a new problem $(4.1)_2$ which again contains fewer integer extreme points, than either of the problems $(4.1)_1$ or (4.1). Since there are a finite number of feasible integer extreme points and since each successive problem (4.1), $(4.1)_1$, $(4.1)_2$,..., contains fewer of these extreme points than all preceding problems, eventually a problem will result which contains none of the integer extreme points. When this condition occurs, the algorithm stops and the global optimum of the problem (4.1) is the best global optimum of each of the partitioned regions. Figure 1 illustrates how this partitioning reduces the feasible region at each step. The points 1,2, and 3 represent the feasible integer points of the original problem, and the dotted lines represent the cutting planes which perform the partitioning. Note that (4.1) has three feasible integer points, $(4.1)_1$ has one, and $(4.1)_2$ has none. A complete description of this approach along with a proof of finiteness is presented in the next section.

Solution Procedure for the Quadratic Problem

The solution of problem (4.1) can be obtained by solving each of the problems (4.2) and (4.3) and selecting the better of the two solutions,

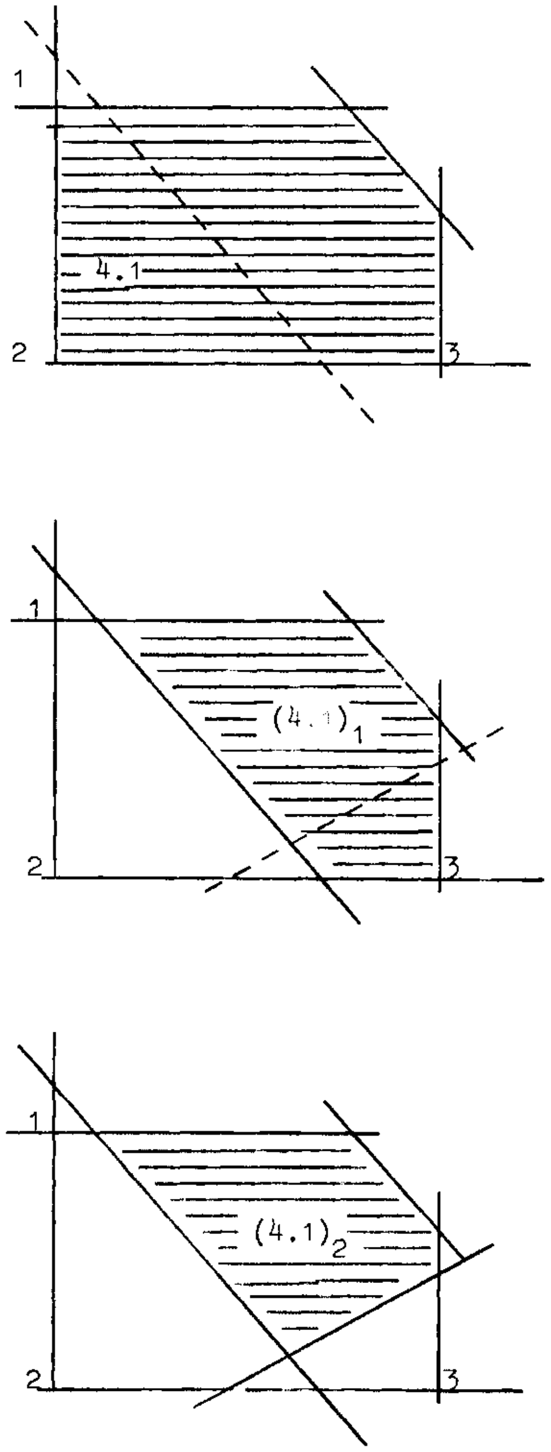


Figure 1. Illustration of Ritter's Partitioning Method.

since the sum of the feasible regions of (4.2) and (4.3) is precisely the feasible region of (4.1) and the objective functions are identical. Problem (4.2) and (4.3) are defined as follows:

(4.2)
$$\begin{aligned} \text{Max:} \quad & Q(\underline{x}) = \underline{c}'\underline{x} - \frac{1}{2}\,\underline{x}'C\underline{x} \\ \text{Subj:} \quad & A\underline{x} \leq \underline{b} \\ & -\underline{c}'\underline{x} \leq t \\ & \underline{x} \leq \underline{1} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

(4.3)
$$\begin{aligned} \text{Max:} \quad & Q(x) = \underline{c}'\underline{x} - \frac{1}{2}\,\underline{x}'C\underline{x} \\ \text{Subj:} \quad & A\underline{x} \leq \underline{b} \\ & -\underline{c}'\underline{x} \geq t \\ & \underline{x} \leq \underline{1} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

where t is some scalar.

For the development of the algorithm, assume that the following conditions are met by problem (4.1).

(a) $\underline{0}$ is a feasible extreme point of (4.1)

(b) $Q(\underline{x})$ is a strictly convex quadratic function

and (c) $\underline{c} < \underline{0}$.

Suppose condition (a) is initially met by the problem (4.1), then conditions (b) and (c) are also met by definition of problem (4.1). If, however, $\underline{0}$ is not feasible for the problem (4.1), then a new problem (4.1)* can be obtained which does satisfy property (a). The new problem (4.1)* is obtained by transforming some feasible extreme point of

(4.1) to the origin of the new problem while preserving all extreme points of the original problem as well as the value of the objective function at corresponding points. In order to insure that the objective function of the new problem $(4.1)^*$ is strictly convex, a feasible integer extreme extreme point is transformed to the origin.

Suppose $\underline{z}$ is one such feasible integer extreme point of the region ($A\underline{x} \leq \underline{b}$, $\underline{x} \leq \underline{1}$, $\underline{x} \geq \underline{0}$). Then since $\underline{z}$ is integer, exactly n of the 2n inequalities $\underline{x} \leq \underline{1}$, $\underline{x} \geq \underline{0}$ are met as equalities at $\underline{x} = \underline{z}$. If these n constraints which are met as equalities at $\underline{z}$ are denoted by $A_1\underline{x} \leq \underline{b}_1$, then $A_1\underline{z} = \underline{b}_1$ and $A\underline{z} \leq \underline{b}$. The matrix A_1 is constructed to be a diagonal matrix whose diagonal elements are either one or minus one. According to Ritter (25), the feasible region can be transformed by

$$A_1\underline{x} = \underline{b}_1 - \underline{v}; \quad \underline{v} \geq \underline{0}$$

into a $\underline{v}$-region where the origin is feasible. By introducing $\underline{x} = A_1^{-1}(\underline{b}_1 - \underline{v})$ into $A\underline{x} \leq \underline{b}$, $\underline{x} \leq \underline{1}$, $\underline{x} \geq \underline{0}$, and $Q(\underline{x}) = \underline{c}'\underline{x} - \frac{1}{2}\underline{x}'C\underline{x}$ a new problem denoted $(4.1)^*$ is obtained.

$$(4.1)^* \qquad \begin{aligned} \text{Max:} \quad & Q^*(\underline{v}) = Q(\underline{z}) + \underline{c}^{*\prime}\underline{v} - \frac{1}{2}\underline{v}'C^*\underline{v} \\ \text{Subj:} \quad & A^*\underline{v} \leq \underline{b}^* \\ & \underline{v} \leq \underline{1} \\ & \underline{v} \geq \underline{0} \end{aligned}$$

where $A_1^{-1} = A_1$, since A_1 is a diagonal matrix with unit coefficients

$$C^* = A_1'^{-1}CA_1 = A_1'^{-1}CA_1^{-1} = C$$

$$\underline{c}^{*\prime} = \underline{b}_1'C^* - \underline{c}'A_1^{-1} = \underline{b}_1'C - \underline{c}'A_1$$

$$A^* = -AA_1^{-1} = -AA_1$$

$$\underline{b}^* = \underline{b} + A^*\underline{b}_1$$

Thus (4.1)* is an equivalent problem to (4.1) and condition (a) is met by the new problem. Condition (b) is also met since $C^* = C$ and C is negative definite. It will now be shown that condition (c) also holds for $\underline{c}^*$.

$\underline{c}^* = \underline{b}_1 C - \underline{c}' A_1$ and $\underline{c}' < \underline{0}$.

Then $c_i^* = 2ab_{1_i} - c_i a_{ii}$.

<u>Case 1</u> $b_{1_i} = 1$ and $a_{ii} = 1$

Then $c_i^* = -2a - c_i < 0$, since a dominates c_i.

<u>Case 2</u> $b_{1_i} = 0$ and $a_{ii} = -1$

Then $c_i^* = 0 + c_i < 0$, since $c_i < 0$.

Therefore condition (c) is also met by the problem $(4.1)^*$.

A basic algorithm for the solution of problem (4.1) is now presented along with a proof of finiteness. First, let us introduce the following notation.

Let T_{w-y} denote the linear transformation from the w-space into the y-space (i.e. $\underline{y} = T_{w-y}(\underline{w})$ and $\underline{w} = T_{y-w}(\underline{y})$ for all $\underline{y},\underline{w}$).

$F(4.1) = (\underline{x} \mid \underline{x}$ integer, $Ax \leq \underline{b}$, $\underline{x} \leq \underline{1}$, $\underline{x} \geq \underline{0})$

$F(4.2) = (\underline{x} \mid \underline{x} \varepsilon F(4.1),\ -\underline{c}'\underline{x} \leq t)$

$F(4.3) = (\underline{x} \mid \underline{x} \varepsilon F(4.1),\ -\underline{c}'\underline{x} \geq t)$

For $t > 0$, $F(4.3) \subset F(4.1)$ since $\underline{0} \notin F(4.3)$. The problem (4.3) can be transformed into a new problem in v-space such that each $\underline{x}$ maps onto a unique $\underline{v}$. The new problem is so constructed that $\underline{v} = \underline{0}$ is feasible. Denote the transformed (4.3) as $(4.1)_1$.

Let $F(4.1)_1 = (\underline{v} \mid \underline{v} = T_{x-v}(\underline{x}),\ \underline{x} \varepsilon F(4.3))$

$T(4.1)_1 = (\underline{x} \mid \underline{x} = T_{v-x}(\underline{v}),\ \underline{v} \varepsilon F(4.1)_1)$

Then $T(4.1)_1 = F(4.3) \subset F(4.1)$. Now we partition $(4.1)_1$ into $(4.2)_1$ and $(4.3)_1$ with the hyperplanes $-\underline{c}^{*\prime}\underline{v} \leq t$ and $-\underline{c}^{*\prime}\underline{v} \geq t$.

Let $F(4.2)_1 = (\underline{v} \mid \underline{v} \varepsilon F(4.1)_1,\ -\underline{c}^{*\prime}\underline{v} \leq t)$

$F(4.3)_1 = (\underline{v} \mid \underline{v} \varepsilon F(4.1)_1,\ -\underline{c}^{*\prime}\underline{v} \geq t)$

$T(4.2)_1 = (\underline{x} \mid \underline{x} = T_{v-x}(\underline{v}),\ \underline{v} \varepsilon F(4.2)_1)$

and $T(4.3)_1 = (\underline{x} \mid \underline{x} = T_{v-x}(\underline{v}),\ \underline{v} \varepsilon F(4.3)_1)$.

It follows that the optimal solution to (4.1) is the best optima of (4.2), $(4.2)_1$, and $(4.3)_1$. Furthermore, for $t > 0$, $F(4.3)_1 \subset F(4.1)_1$ since $0 \notin F(4.3)_1$, and $T(4.3)_1 \subset T(4.1)_1 \subset F(4.1)$. We now transform the problem $(4.3)_1$ into a new problem in the u-space such that $\underline{u} = \underline{0}$ is feasible. Denote the transformed $(4.3)_1$ as $(4.1)_2$.

Let $F(4.1)_2 = (\underline{u} \mid \underline{u} = T_{v-u}(\underline{v}),\ \underline{v} \varepsilon F(4.3)_1)$

$T(4.1)_2 = (\underline{x} \mid \underline{x} = T_{u-x}(\underline{u}),\ \underline{u} \varepsilon F(4.1)_2)$

Then $T(4.1)_2 \subset T(4.1)_1 \subset F(4.1)$, since at least one integer extreme point has been deleted for each of the successive problems. Reapplication of the above process will eventually result in a $(4.1)_n$ in which $F(4.1)_n = \varphi$. When this point is reached, the solution of (4.1) is given by the best solution of (4.2), $(4.2)_1$, $(4.2)_2, \ldots, (4.2)_{n-1}$. Figure 2 shows the basic flow chart of the Ritter algorithm as modified by the author. The problem $(4.1)_0$ refers to the original problem. Finiteness of this algorithm is immediate, however; for completeness and formalism Theorem 4 provides this proof.

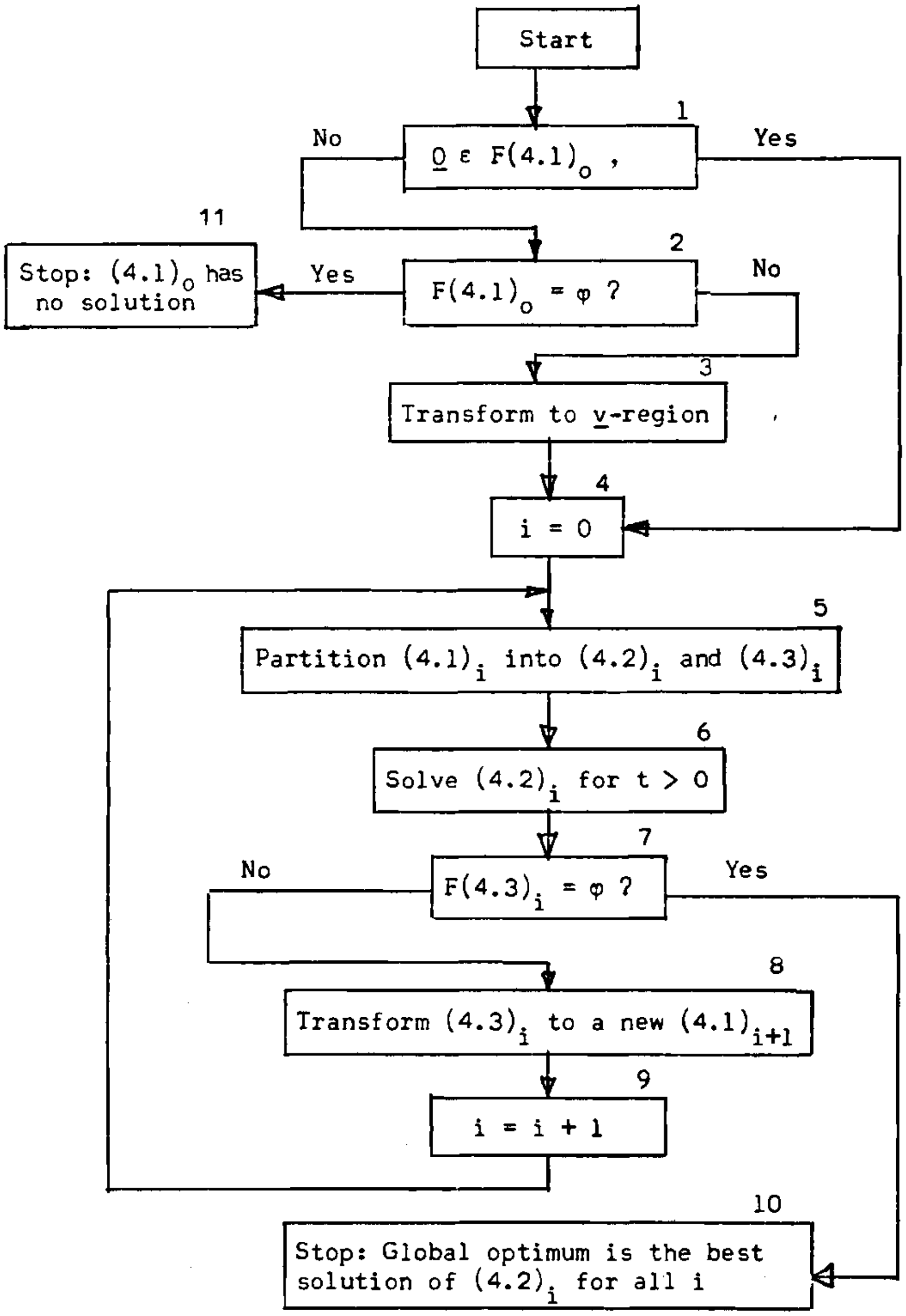

Start
1
$\underline{0} \varepsilon F(4.1)_o$,
No
Yes
11
Stop: $(4.1)_o$ has no solution
Yes
2
$F(4.1)_o = \varphi$?
No
3
Transform to $\underline{v}$-region
4
i = 0
5
Partition $(4.1)_i$ into $(4.2)_i$ and $(4.3)_i$
6
Solve $(4.2)_i$ for t > 0
7
No
$F(4.3)_i = \varphi$?
Yes
8
Transform $(4.3)_i$ to a new $(4.1)_{i+1}$
9
i = i + 1
10
Stop: Global optimum is the best solution of $(4.2)_i$ for all i

Figure 2. Flow Chart for 0-1 Programming Algorithm.

Theorem 4 (Finiteness)

The algorithm of Figure 2 terminates in a finite number of iterations.

Proof.

Case I. Problem $(4.1)_o$ has a solution. There are a finite number of integer extreme points in the feasible region of problem $(4.1)_o$. Each partitioning of $(4.1)_i$ occurs such that $\underline{0} \notin$ of $F(4.3)_i$. Therefore, each problem $(4.3)_i$ and consequently each new problem $(4.1)_{i+1}$ contains at least one less integer extreme point than the preceding problem. Since there are a finite number of integer extreme points there are a finite number of partitionings which can occur such that an original extreme point remains in the problem $(4.3)_i$. When a $(4.3)_i$ is developed which contains no feasible integer extreme point $(F(4.3)_i = \varphi)$, the algorithm terminates. Since this will occur in a finite number of steps the algorithm terminates in a finite number of iterations.

Case II. Problem $(4.1)_o$ has no solution. If problem $(4.1)_o$ has no solution, the algorithm terminates immediately at block 11. This completes the proof of Theorem 4.

A Solution Procedure for the Reduced Quadratic Problem

Finiteness of the algorithm is dependent upon a means of finding the solution of problem (4.2) for some $t > 0$. The required mechanism for obtaining this solution is developed through consideration of the following problem,

$$(4.4) \qquad \begin{aligned} \text{Max:} \quad & Q(\underline{x}) = \underline{c}'\underline{x} - \frac{1}{2}\,\underline{x}'C\underline{x} \\ \text{Subj:} \quad & -\underline{c}'\underline{x} = t \\ & \underline{x} \geq \underline{0}\,. \end{aligned}$$

Now it will be shown that for t greater than some lower bound, the solution of (4.4) is an upper bound on the solution of problem (4.2).

Theorem 5

Let $\underline{x}_4^*(\bar{t})$ be the optimal solution to problem (4.4) for $t = \bar{t}$. Let $\underline{x}_2^*(\bar{t})$ be the optimal solution to problem (4.2) for $t = \bar{t}$.

Let $t_i = \dfrac{-2c_i^2}{c_{ii}}$ and $t_r = \min\limits_i t_i$.

Then for all $\bar{t} \geq t_r$, $Q[\underline{x}_4^*(\bar{t})] \geq Q[\underline{x}_2^*(\bar{t})]$.

Proof. Let

(4.5) $$\underline{x}^i(t) = (0,\dots,0, -t/c_i, 0,\dots,0).$$

It follows that $\underline{x}^i(t)$, $i = 1,2,\dots,n$, correspond to the extreme points of problem (4.4) and the corresponding value of the solution is

$$\begin{aligned} Q[\underline{x}^i(t)] &= c_i x_i - \tfrac{1}{2} c_{ii} x_i^2 \\ &= -t - \tfrac{1}{2} c_{ii} t^2/c_i^2 . \end{aligned}$$

Then for $t > t_i = -2c_i^2/c_{ii}$, $Q[\underline{x}^i(t)] > 0$.

Since $t_r = \min\limits_i t_i = -2c_r^2/c_{rr}$, then it follows that

(i) $Q[\underline{x}_4^*(t)] = \max\limits_i Q[\underline{x}^i(t)] = Q[\underline{x}^r(t)]$

and (ii) smallest value of t for which $Q[\underline{x}_4^*(t)] \geq 0$ is $t = t_r$.

Now consider $\underline{x}^r(t) = (0,\dots,0, -t/c_r, 0,\dots,0)$. Note that $Q[\underline{x}^r(t)]$ is a monotone increasing function of t for $t \geq t_r$. Also the optimal of the problem

(4.6) $$\text{Max:} \quad Q(\underline{x}) = \underline{c}'\underline{x} - \tfrac{1}{2}\underline{x}'C\underline{x}$$

$$\text{Subj:} \quad -\underline{c}'\underline{x} \leq t$$
$$\underline{x} \geq \underline{0}$$

for $t \geq t_r$ occurs at an extreme point of the form (4.5) [see Hadley (15)]. Hence $\underline{x}^*(\bar{t})$ is the solution to both (4.4) and (4.6) for $\bar{t} \geq t_r$. Both (4.6) and (4.2) have the same objective function and (4.2) is a more restricted problem than (4.6). Hence $Q[\underline{x}_4^*(\bar{t})] = Q[\underline{x}^*(\bar{t})] \geq Q[\underline{x}_2^*(\bar{t})]$ for $\bar{t} \geq t_r$. This completes the proof of Theorem 5.

Remark 2. Special application of Theorem 5 provides the basic machinery upon which the algorithm is built. All other development in this text provides the details for applying this basic theorem.

The above theorem is applied at every iteration of the algorithm. The particular $\bar{t}$ used at each application is chosen such that either the solution of (4.2) can be found or there is no solution to (4.2) for $t = \bar{t}$ better than some current best known solution. The solution to (4.4) is the solution to (4.2) for all $t \geq t_r$. The solution to (4.4) for any $t \geq t_r$ is simply $\underline{x}^r$ where $r = (j \mid c_j^2 \leq c_i^2,\ i = 1,\ldots,n)$. If $\underline{x}^*$ denotes the best local optima of (4.4) for $t = 1$ and $\underline{x}^* \neq \underline{0}$, then $\bar{t}\underline{x}^*$ denotes the solution of (4.4) $t = \bar{t}$. Beginning with some $\underline{x}^*$, and some current best solution, $\underline{x}^{**}$, Theorem 5 is applied in two ways. First, determine the t denoted t_0 which is the largest t such that $t\underline{x}^*$ is the optimal solution of (4.2) for $t = t_0$. Second, determine t_1 such that $Q(t_1\underline{x}^*) = Q(\underline{x}^{**})$.

$$t_1 = \frac{-1 - [1 - (2)(c_{ii})(x_i^*)^2(Q(\underline{x}^{**}) - Q(\underline{0}))]^{1/2}}{c_{ii}(x_i^*)^2}$$

Since $c_{ii} < 0$ for all i, t_1 will always have one positive solution greater than or equal to t_r.

$$t_1 \geq t_r = \frac{-2c_r^2}{c_{rr}} .$$

To insure that $\bar{t} \geq t_r$ choose $\bar{t} = \max(t_0, t_1)$. Since $t_1 \geq t_r$, $\bar{t} \geq t_r$, and Theorem 5 can always be applied for some $\bar{t} \geq t_r$.

Remark 3. The following is a summary of the rules used in the application of Theorem 5.

(a) Find $t_0 = \max(T\colon \; At\underline{x}^* \leq \underline{b}, t\underline{x}^* \leq \underline{1})$

(b) Find $t_1 : Q(t_1\underline{x}^*) = Q(\underline{x}^{**})$

(c) $\bar{t} = \max(t_0, t_1)$

Remark 4. The efficiency of the proposed algorithm (Figure 2) is dependent on its ability to solve (4.2) for large t. If (4.2) can only be solved for small t, the method reduces to complete enumeration since only one integer extreme point is excluded from further consideration at each iteration. This can be shown by observing that for $t < c_i$ $-\underline{c}'\underline{x} \leq t$ excludes all extreme points from problem (4.2) with $x_i = 1$. If $t = \min(c_i)$, no nonzero integer point is feasible for (4.2) and only $x = \underline{0}$ can be eliminated at each iteration. Therefore for sufficiently small t at each iteration, the algorithm reduces to complete enumeration. The decision rules presented in Remark 3 yield the largest $\bar{t}$ (largest cut) which insures that we have not eliminated any integer point better than $\underline{x}^{**}$.

A second cutting plane which can be used to eliminate part of the feasible region will now be introduced. This plane is a parallel shift of the original objective function hyperplane (i.e. $\bar{\underline{c}}'\underline{x} \geq k$) and can only be inserted when $t_0 \geq t_1$ (i.e. a new integer extreme point has been found to replace the current best solution). Consider now two new problems formed by partitioning problem (4.3) as follows:

$$\text{(4.9)} \qquad \begin{aligned} \text{Max:} \quad & Q(\underline{x}) = \underline{c}'\underline{x} - \tfrac{1}{2}\,\underline{x}'C\underline{x} \\ \text{Subj:} \quad & A\underline{x} \leq \underline{b} \\ & -\underline{c}'\underline{x} \geq t_0 \\ & \bar{\underline{c}}'\underline{x} \leq \bar{\underline{c}}'t_0\underline{x}^* \\ & \underline{x} \leq \underline{1} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

$$\text{(4.10)} \qquad \begin{aligned} \text{Max:} \quad & Q(\underline{x}) = \underline{c}'\underline{x} - \tfrac{1}{2}\,\underline{x}'C\underline{x} \\ \text{Subj:} \quad & A\underline{x} \leq \underline{b} \\ & -\underline{c}'\underline{x} \geq t_0 \\ & \bar{\underline{c}}'\underline{x} \geq \bar{\underline{c}}'t_0\underline{x}^* \\ & \underline{x} \leq \underline{1} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

When $t_0\underline{x}^*$ is integer, it follows that the global optimum of (4.1) is that $\underline{x}$ which is the largest global optima of (4.2), (4.9), and (4.10). Theorem 6 states that the solution of (4.2) is an upper bound on the solution of (4.9). With Theorem 6 one need only solve (4.2) and (4.10) to obtain the solution of (4.1) when $t_0 \geq t_1$.

Theorem 6

If $\underline{z}$ is an optimal integer solution to (4.9) and $t_0\underline{x}^*$ is an optimal integer solution to (4.2), then $Q(\underline{z}) \leq Q(t_0\underline{x}^*)$.

Proof. For any integer solution $\underline{z}$, $Q(\underline{z}) = \bar{\underline{c}}'\underline{z}$. But the constraint $\bar{\underline{c}}'\underline{z} \leq \bar{\underline{c}}'t_0\underline{x}^*$ implies $Q(\underline{z}) = \bar{\underline{c}}'\underline{z} \leq \bar{\underline{c}}'t_0\underline{x}^* = Q(t_0\underline{x}^*)$ or simply

$Q(\underline{z}) \leq Q(t_0\underline{x}^*)$. This completes the proof of Theorem 6.

Recall that in each step of the proposed algorithm (Figure 2), the problem (4.1) and hence the problem (4.2) are no longer in terms of the original region. Each problem (4.1) begins with the origin feasible and each iteration eliminates at least the origin from further consideration in all successive iterations. Therefore, it becomes necessary to transform some other feasible point to the origin to make possible the solution of the next problem. It is also implied by the algorithm of Figure 2, that the best local optimum currently known is available for reference at any iteration. When a new local optimum is discovered which is better than any previously found, it must be saved in place of its predecessor. Since the new local optimum is given in some transformed region, a means must be developed to relate this point to the variables of the original region. Therefore it is necessary that one have a reverse transformation by which points in some $\underline{v}$-region can be related to original points in the $\underline{x}$-region. The following transformation gives the desired result.

(4.11) $$T^*\underline{v} + \underline{tt}^* = \underline{x}$$

where $$T^* = -TA_1^{-1}$$

$$\underline{tt}^* = \underline{tt} - T^*\underline{b}_1 \ .$$

Initially T and $\underline{tt}$ are an identity matrix and a zero vector of the appropriate dimensions. For successive transformations, T^* and $\underline{tt}^*$ become the T and $\underline{tt}$ of the next transformation so that the reverse transformation through numerous forward transformations can be accomplished by application of (4.11) only once.

Furthermore, the objective function of the original integer programming problem can be expressed in terms of the new region by the following transformation

$$\underline{\bar{c}}^{*\prime} = -\underline{\bar{c}}A_1^{-1} .$$

It follows that if the transformed variables are substituted for the original variables of Theorem 6 and the condition that $Tt_0\underline{x}^* + \underline{tt}$ be integer instead of $t_0\underline{x}^*$, then Theorem 6 holds for any transformed region.

Location of a Feasible Integer Point

The basic algorithm of Figure 2 requires that the problem $(4.3)_i$ be transformed to a new region at each iteration. The transformed problem denoted $(4.1)_{i+1}$ must satisfy the following conditions, $\underline{0}$ must be feasible, $Q^*(\underline{x})$ must be convex, and $\underline{c}^*$ must be negative. To make this transformation so that the above conditions are satisfied, the algorithm must transform a feasible integer point of the problem $(4.3)_i$ to the origin for the problem $(4.1)_{i+1}$. The following section describes how these integer points are obtained.

The general approach followed is that of a tree search in which each node of the tree represents an integer point. The search begins with an infeasible solution and moves successively toward feasibility. Once a feasible solution is found, the search is ended. When it becomes necessary to locate another feasible integer point, the search resumes from the stopping point of the previous search. Since each integer point found becomes the origin of the new $(4.1)_{i+1}$, and since the origin is eliminated from further consideration at each stage, each search begins with an infeasible solution.

The tree search scheme is based on the work of Glover (11) as revised by Geoffrion (9). The latter defines a partial solution S as a binary assignment of n or fewer of the integer variables. All variables assigned a value by S are called fixed variables and variables not assigned a value by S are known as free variables. The notation $j \in S$ and $-j \in S$ denotes $x_j = 0$ and $x_j = 1$ respectively. Hence if $n = 5$ and $S = (3,5,-2)$, then $x_3 = 0$, $x_5 = 0$, $x_2 = 1$, and x_1 and x_4 are free variables. For any solution S, all free variables will be assigned the value one. Therefore the solution $S = (3,5,-2)$ denotes the solution $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$. It follows that any partial solution S is different from another partial solution if at least one element of S is different in the two solutions. With this notation, it follows that the scheme of Figure 3 terminates only after a feasible solution has been found or all 2^n unique solutions have been explicitly enumerated.

The scheme of Figure 3 could be used to find a feasible integer point; however, it would be inefficient for even small problems. The efficiency of the scheme can be greatly increased if some of the solutions can be implicitly enumerated. This can be accomplished if one or more tests can be devised which indicate the futility of further examination along the branch in question. Assuming that such tests are available, the explicit enumeration tree search could be converted to an implicit enumeration tree search as shown in Figure 4.

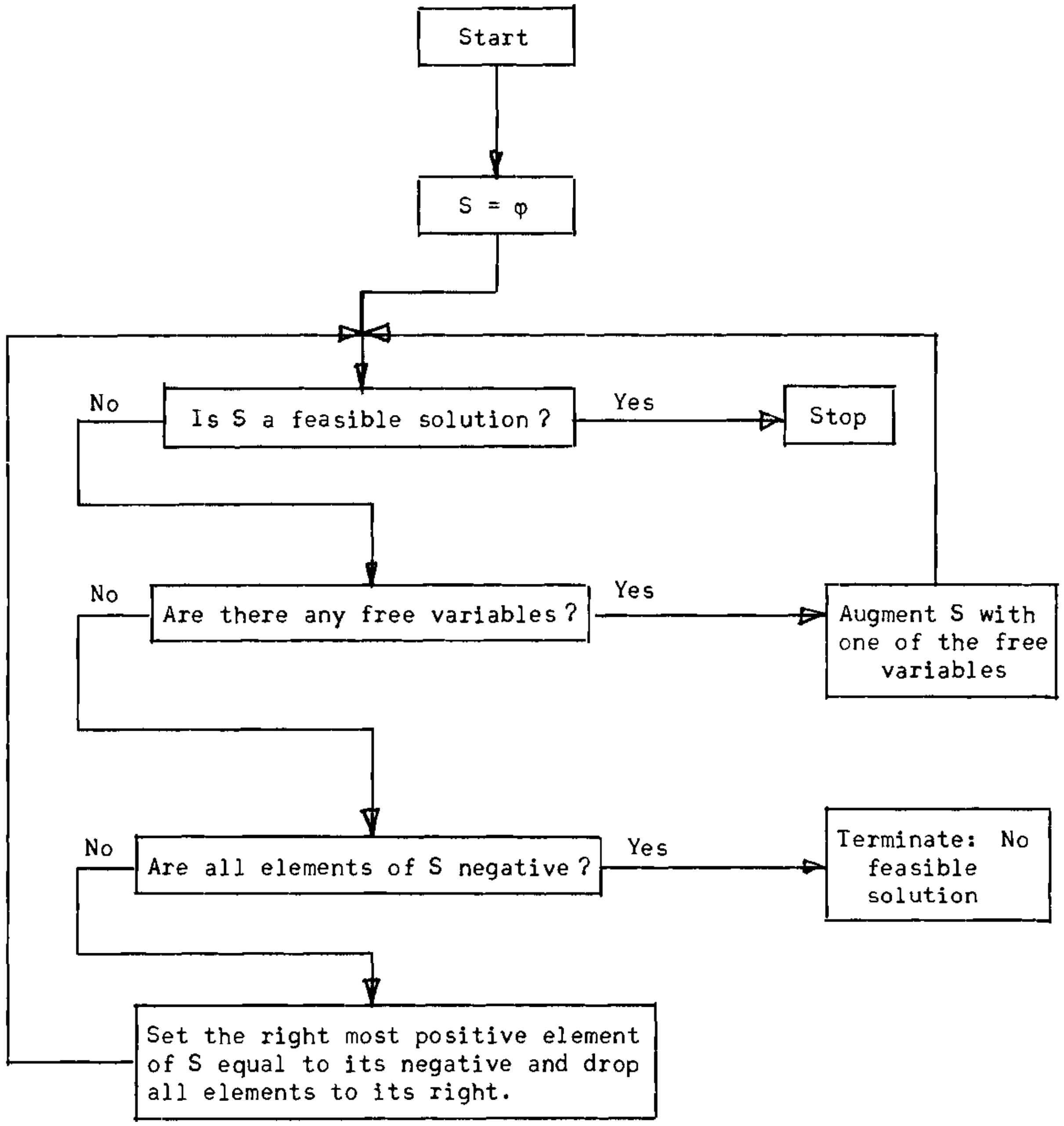

Start
S = φ
Is S a feasible solution ?
Yes
Stop
No
Are there any free variables ?
Yes
Augment S with one of the free variables
No
Are all elements of S negative ?
Yes
Terminate: No feasible solution
No
Set the right most positive element of S equal to its negative and drop all elements to its right.

Figure 3. Explicit Enumeration Tree Search.

Theorem 7

The implicit enumeration scheme of Figure 4 leads to a non-redundant sequence of partial trial solutions which does not terminate before a feasible solution is found, or all 2^n solutions have been implicitly enumerated.

A proof of Theorem 7 can be found in Geoffrion (9) and is not repeated here.

Attention is now turned to the details of the method used to accomplish the task of block 7 Figure 4 in which the partial solution S is augmented with one of the free variables. The method used to select this free variable is due to Geoffrion (9) and is this, fix that free variable in the next solution which most reduces the infeasibility of the present solution. With this augmentation criterion, it follows that the only free variables which can decrease infeasibility without allowing the value of the objective function to fall below some lower limit are elements of Γ where

$$\Gamma = (j : j \text{ free}, \ \bar{c}_j < \underline{\bar{c}}'\underline{x}^S - \bar{z}, \ a_{ij} > 0 \ \text{ for some } i \text{ such that } y_i < 0) ,$$

where $\underline{x}^S$ is the solution determined by S with $A\underline{x} + \underline{y} = \underline{b}$ and $\bar{z}$ is some lower limit on the objective function.

Remark 5. Notice that the assumption $\underline{\bar{c}} > \underline{0}$ is used here. Also notice that if Γ is the null set, there is no feasible completion of S that is better than some known solution and S can be fathomed. Therefore, indirectly a fathoming mechanism has been developed.

If Γ is not null, then it can be seen that the free variable

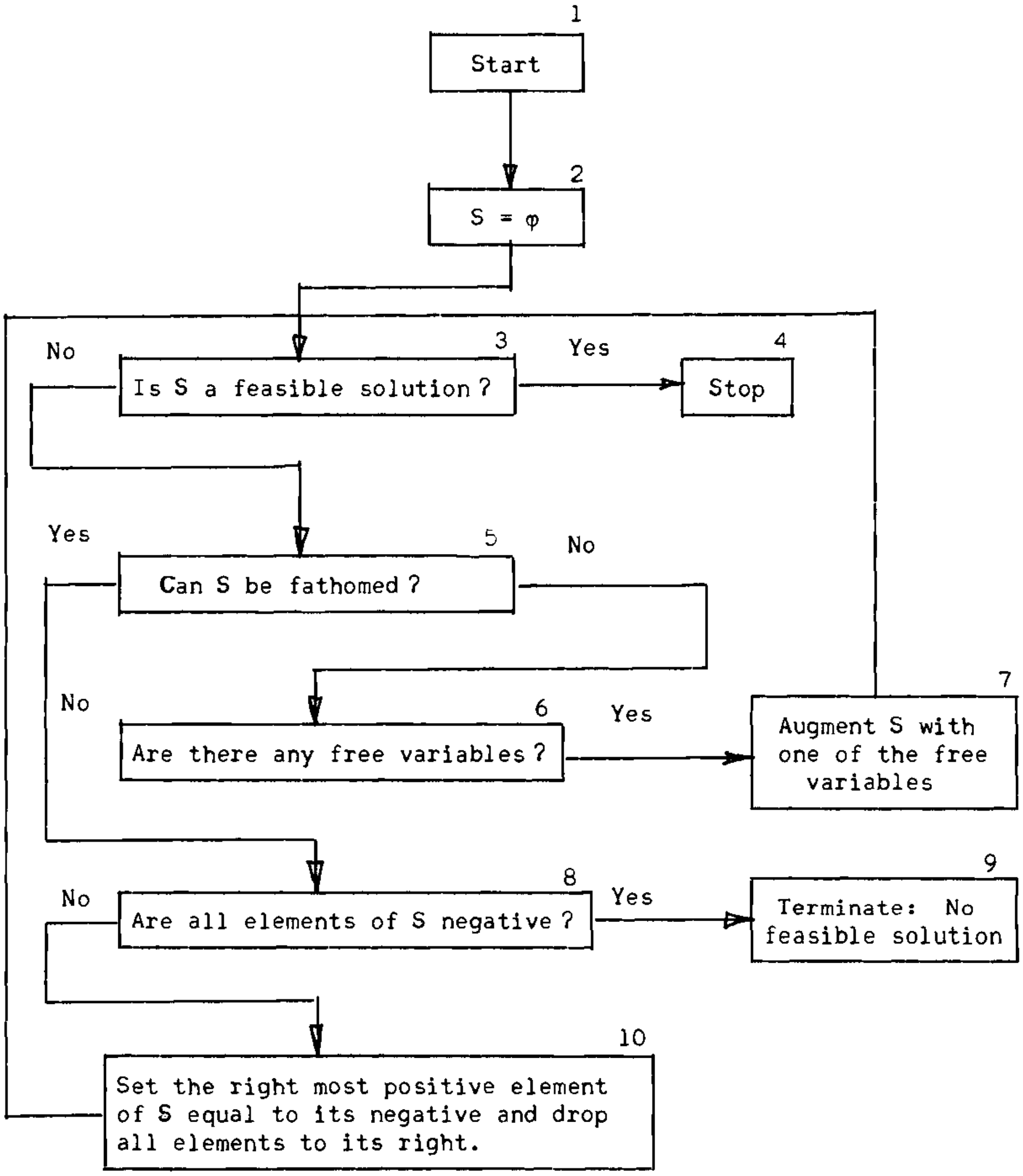

1
Start
2
S = φ
3
Is S a feasible solution ?
Yes
4
Stop
No
5
Can S be fathomed ?
Yes
No
6
Are there any free variables ?
Yes
No
7
Augment S with one of the free variables
8
Are all elements of S negative ?
Yes
No
9
Terminate: No feasible solution
10
Set the right most positive element of S equal to its negative and drop all elements to its right.

Figure 4. Implicit Enumeration Tree Search.

which when fixed would most reduce the infeasibility is that k where

$$k = j \mid \sum_{i=1}^{m} \min(y_i + a_{ij}, 0) \geq \sum_{i=1}^{m} \min(y_i + a_{i\ell}, 0),$$
$$\ell = 1,\ldots,n.$$

The original notion that all free variables take on the value of one is used in the development of the above expression.

Using this augmentation mechanism attention is now turned to block 5 of Figure 4 which attempts to fathom the current partial solution denoted by S. Balas (1) presented three tests to be used in an attempt to fathom partial solutions. The first requires the determination of the set Γ. If Γ is the null set, the branch may be fathomed. The second test attempts to demonstrate that the best completion of S, regardless of feasibility considerations, is not preferable to some other known feasible solution. If z^s denotes the value of $Q(\underline{x})$ at S, then $z^s - \bar{z} < 0$ implies that no solution preferable to the one already known exists along the branch in question. Note that $\underline{\bar{c}} > 0$ is used in the second test. The final test attempts to show that at least one of the m constraints will be violated by any completion of the partial solution. Mathematically the third test reduces to computing the quantities

$\alpha_i = y_i + \sum_{j\varepsilon\Gamma} \max(a_{ij}, 0)$ for all i such that $y_i < 0$. If

$\alpha_i < 0$ for some i, then the branch in question may be fathomed.

<u>Remark 6</u>. A summary of Balas' tests follows:

(a) Test 1. Compute $\Gamma = (j : j \text{ free}, \bar{c}_j < \underline{c}'\underline{x}^s - \bar{z}, a_{ij} > 0$ for some i such that $y_i < 0)$.

If $\Gamma = \varphi$, S may be fathomed.

(b) Test 2. Compute $\beta = z^s - \bar{z}$.

If $\beta < 0$, S may be fathomed.

(c) Test 3. Compute $\alpha_i = y_i + \sum_{j\varepsilon\Gamma} \max(a_{ij}, 0)$ for all i

such that $y_{ij} < 0$.

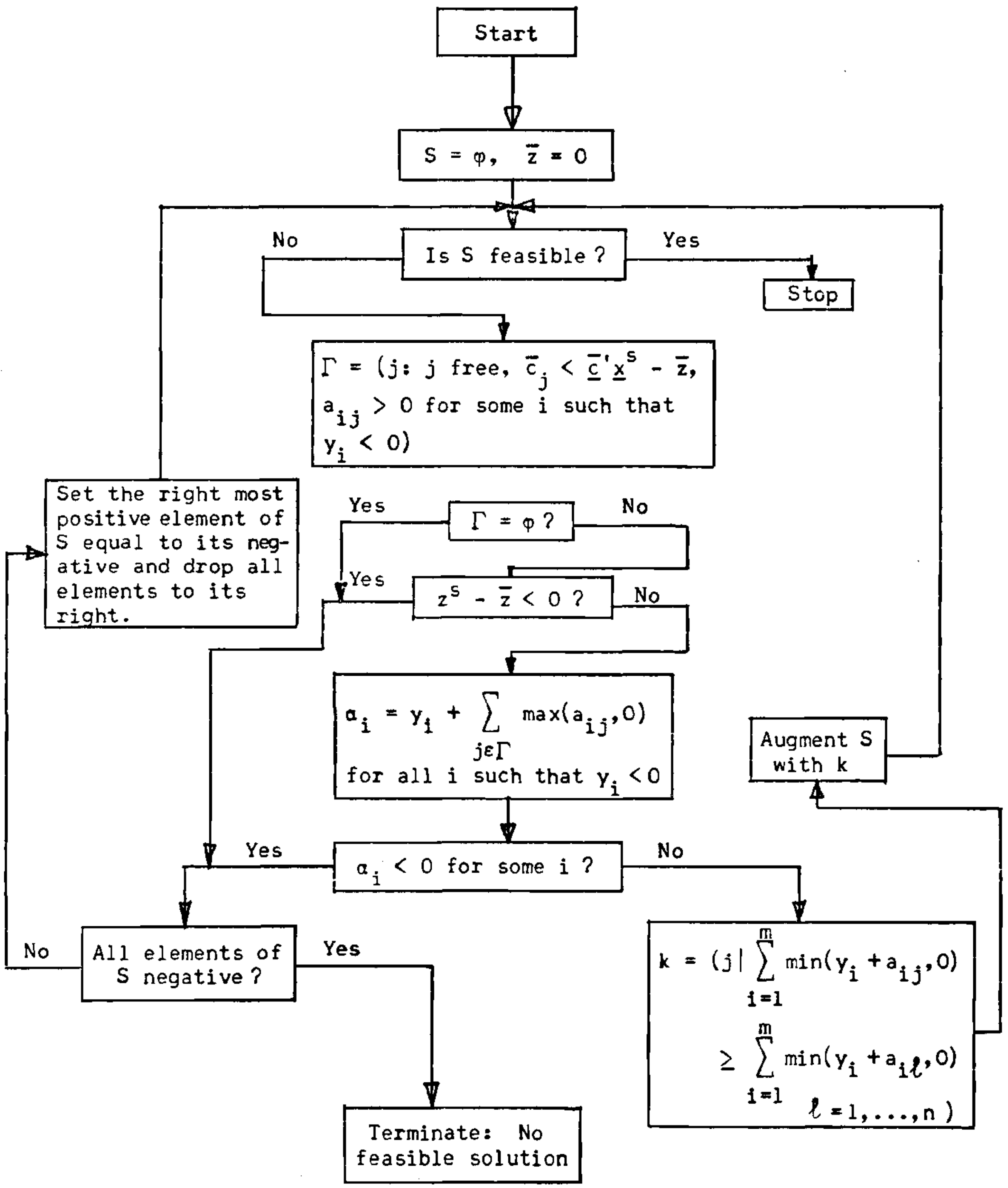
If $\alpha_i < 0$ for some i, S may be fathomed.

A complete flow chart of the Balas algorithm as modified by the author to locate feasible integer points is as shown in Figure 5.

The above algorithm provides not only a means for determining feasible local optimum (integer points) required at each iteration, but also a stopping mechanism for the integer programming algorithm of Figure 2. If at the pth iteration, the search routine fails to find a feasible integer point, the global optimal is the local optimal found at the p-1st iteration. Failure to find a feasible point on the first iteration implies that $\underline{0}$ is the optimal solution if $\underline{0}$ is feasible and that there is no solution if it is infeasible.

Determination of Redundant Constraints

Recall that at each iteration of the algorithm of Figure 2, one or more constraints or cutting planes are added to the constraint set. These constraints cut away or eliminate from further consideration one or more integer extreme points. It follows that as more of these constraints are added to the constraint set, that some of the original constraints as well as some of the additional constraints no longer



Start
$S = \varphi, \quad \bar{z} = 0$
Is S feasible ?
No
Yes
Stop
$\Gamma = (j: j \text{ free}, \bar{c}_j < \underline{\bar{c}}'\underline{x}^s - \bar{z}, a_{ij} > 0 \text{ for some } i \text{ such that } y_i < 0)$
Set the right most positive element of S equal to its negative and drop all elements to its right.
$\Gamma = \varphi$?
Yes
No
$z^s - \bar{z} < 0$?
Yes
No
$\alpha_i = y_i + \sum_{j\varepsilon\Gamma} \max(a_{ij}, 0)$ for all i such that $y_i < 0$
Augment S with k
$\alpha_i < 0$ for some i ?
Yes
No
All elements of S negative ?
No
Yes
$k = (j \mid \sum_{i=1}^{m} \min(y_i + a_{ij}, 0) \geq \sum_{i=1}^{m} \min(y_i + a_{i\ell}, 0) \quad \ell = 1, \ldots, n)$
Terminate: No feasible solution

Figure 5. Implicit Enumeration Tree Search.

serve to restrict the feasible region. These ineffective constraints shall be referred to as geometrically redundant constraints. It can be easily seen that the constraint set would become quite unwieldy if several hundred iterations were required to solve the integer problem. Therefore, it is imperative that the algorithm be able to locate and eliminate these redundant constraints. A full explanation of these constraints and a means for locating them follows.

Definition 1: Type A Geometrical Redundancy

Let $\underline{a}_j'$ be the j^{th} row of the A matrix. Then $\underline{a}_j'\underline{x} \leq b_j$ is a type A geometrically redundant constraint if and only if $\underline{a}_j'\underline{z} \neq b_j$ for all $\underline{z} \in \Gamma = (\underline{z} \mid A\underline{z} \leq \underline{b})$ when $\Gamma = \varphi$. When $\Gamma = \varphi$, there are no feasible points in the constraint set, and the definition has no meaning.

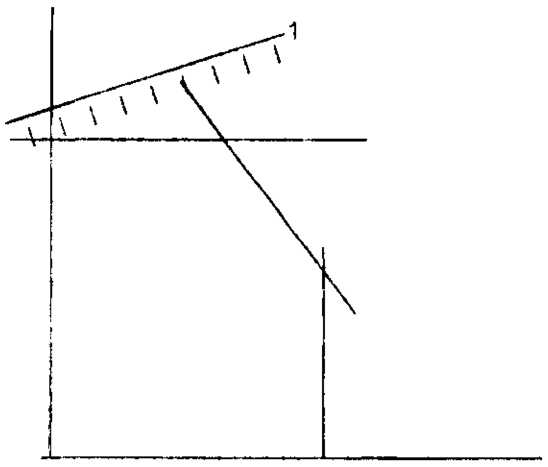


Figure 6. Illustration of a Type A Geometrically Redundant Constraint.

Constraint 1, Figure 6 is a type A geometrically redundant constraint.

Definition 2: Type B Geometrical Redundancy

Let $\underline{a}_j'\underline{x} \leq b_j$ be a row of $A\underline{x} \leq \underline{b}$. Then $\underline{a}_j'\underline{x} \leq b_j$ is a type B geometrically redundant constraint if and only if there exists an $i \neq j$ such that $J \subset I$ where

$$J = (\underline{z} \mid \underline{a}_j'\underline{z} = b_j, \quad A\underline{z} \leq \underline{b}) \quad \text{and}$$

$$I = (\underline{x} \mid \underline{a}_i'\underline{x} = b_i, \quad A\underline{x} \leq \underline{b}).$$

Simply stated, a type B geometrically redundant constraint is one which is met as an equality in the feasible region at only a single point or at a set of points which lie along the face of the convex set which is determined by another constraint.

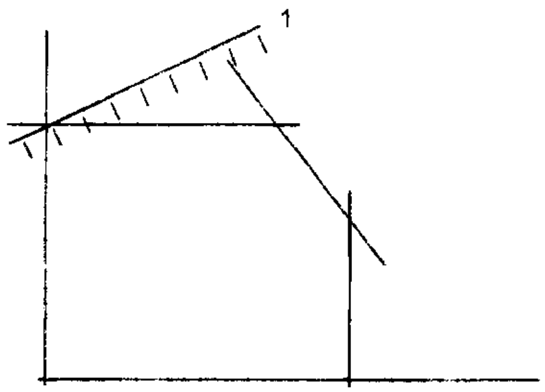


Figure 7. Illustration of a Type B Geometrically Redundant Constraint.

Constraint 1, Figure 7 is a type B geometrically redundant constraint.

Remark 7. Observe that a type B redundancy becomes a type A redundancy when the redundant constraint is perturbed to form $\underline{a}_j'\underline{x} \leq b_j + e$ where e is some scalar greater than zero.

Remark 8. Also notice that a type B redundancy implies that there exists a degenerate basic feasible solution.

Klaus Ritter (25) developed a method to locate type A geometrical redundancies. The author has extended this basic approach by perturbing certain elements of $\underline{b}$ so that type B redundancies can also be detected. After the addition of slack variables the constraints take the form $\underline{a}_j'\underline{x} + s_j = b_j$. Positively perturbed constraints take the form $\underline{a}_j'\underline{x} + s_j = b_j + e$, $e > 0$. Positively perturbed type B geometrically redundant constraints become type A geometrically redundant. Positively perturbed essential constraints remain essential. The method presented below is based on the idea of perturbing constraints and then applying the basic approach of Ritter to locate type B as well as type A redundancies.

The method is based upon the solution of the following linear programming problem.

$$\begin{aligned} &\text{Max:} \quad -s_j \\ &\text{Subj:} \quad A\underline{x} + \underline{s} = \underline{b} \\ &\qquad\qquad \underline{x}, \underline{s} \geq \underline{0} \end{aligned} \tag{4.12}$$

with the jth element of $\underline{b}$ perturbed by e where $e > 0$. To accomplish this perturbation without actually changing the b_j, the following decision rule was incorporated for determining the leaving variable. <u>If there is a tie for the leaving variable which involves b_j, allow the</u>

j^{th} variable to remain in the solution. For example if k is the entering variable for some tableau and there is a tie for the leaving variable such that $\frac{b_j}{y_{jk}} = \frac{b_r}{y_{rk}} \geq 0$ then the perturbed b_j would produce $\frac{b_j + e}{y_{jk}} > \frac{b_r}{y_{rk}}$ and would indicate that r should be the leaving variable. The above decision rule accomplishes precisely what is needed to solve the perturbed problem.

If problem (4.12) is solved using the above decision rule, one of three cases will occur at the optimal solution.

Case I: s_j is greater than zero. This implies that the j^{th} constraint is type A geometrically redundant.

Case II. s_j is a basic variable at the zero level. This implies that the j^{th} constraint is type B geometrically redundant.

Case III: s_j is not a basic variable. This implies that the j^{th} constraint is essential.

With the above information, a procedure for locating redundant constraints could be developed. If a constraint set contained m constraints, one could determine if each is essential by solving the perturbed problem (4.12) for $j = 1,2,\ldots,m$. This would involve solving m linear programming problems which could be a very lengthy process. This can be significantly reduced by observing that at any feasible tableau, if all elements of the ℓ^{th} row are less than or equal to zero except the $y_\ell = 1$ element, then the problem max $- s_\ell$ subject to the constraints of that tableau will have the solution $s_\ell = b_\ell$. If b_ℓ is positive, then the ℓ^{th} constraint is type A geometrically redundant. If b_ℓ is equal to zero, then the ℓ^{th} constraint is type B geometrically redundant. The number of linear programming problems can be reduced by observing that at each tableau the columns which are not in the basis which have unique pivot elements could enter the basis and remove the variable corresponding to the row of this pivot element. If this variable is a slack, then the constraint corresponding to this slack variable is essential. The arguments presented above form the basis for the following method for determining geometrically redundant constraints.

Let E = the set of essential constraints

M = the set of constraints about which no decision has been made

$A\underline{x} + \underline{s} = \underline{b}$ is the constraint set

y_{ij} = the $(i,j)^{th}$ element of any tableau

Initially E is the null set and M contains all real numbers $(1,2,\ldots,m)$.

Step 1. Let $j = 1$

Step 2. Solve

$$\begin{aligned} \text{Max:} \quad & -s_j \\ \text{Subj:} \quad & A\underline{x} + \underline{s} = \underline{b} \\ & \underline{x}, \underline{s} \geq \underline{0} \end{aligned}$$

by the perturbed method.

Step 3. Is s_j greater than zero?

Yes - The j^{th} constraint is type A geometrically redundant. Delete j from M and proceed to step 5.

No - Proceed to step 4.

Step 4. Is s_j a basic variable?

Yes - The j^{th} constraint is type B geometrically redundant. Delete j from M and proceed to step 5.

No - The j^{th} constraint is an essential constraint. Delete j from M, place j in E, and proceed to step 5.

Step 5. Are there any rows in the current tableau which have only one positive component?

Yes - If the row is r, the r^{th} constraint is redundant. Delete r from M and check for all other such rows. After all rows have been checked, proceed to step 6.

No - Proceed to step 6.

Step 6. Calculate the pivot element in each column not in the basis. If the pivot element in a column is unique, then examine the variable corresponding to the row containing this pivot element. If this variable is a slack, then the constraint associated with this slack variable is essential. Delete the appropriate entry from M and make an entry in E.

Step 7. Is M the null set?

Yes - The procedure is complete, the essential constraints are in E. All other constraints are redundant.

No - Let j equal some element of M, and proceed to step 2.

An Algorithm for the Maximization of the Special Quadratic Programming Problem

Section 1 presented the basic algorithm for the solution of the special quadratic programming problem. The approach involves successively partitioning the feasible region in such a way that the global optimum can be found in one of these regions. Section 2 presents the methods which are used to solve each of the special sub-problems. Section 3 gives a means for finding a local optimum which is required to solve the partitioned problem and section 4 presents a method for eliminating geometrically redundant constraints. A complete expansion of the flow chart of Figure 2 is given in Figure 8. The flow chart of Figure 11 was used to write a computer program from which computational experience with the proposed algorithm was obtained. These results are given in Chapter V.

<u>Example</u>

$$\begin{aligned} \text{Max:}\quad & 3x_1 + 2x_2 + x_3 \\ \text{Subj:}\quad & x_1 + x_2 + x_3 \leq 2.75 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

Following the procedure of Section 2, Chapter III

$$a = \max\,(30,\ 20,\ 10) + 1 = 31$$

Therefore the quadratic programming problem to be solved is as follows:

$$\begin{aligned} \text{Max:}\quad & Q(\underline{x}) = -28x_1 = -29x_2 - 30x_3 + 31x_1^2 + 31x_3^2 \\ \text{Subj:}\quad & x_1 + x_2 + x_3 \leq 2.75 \\ & \underline{x} \leq \underline{1} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

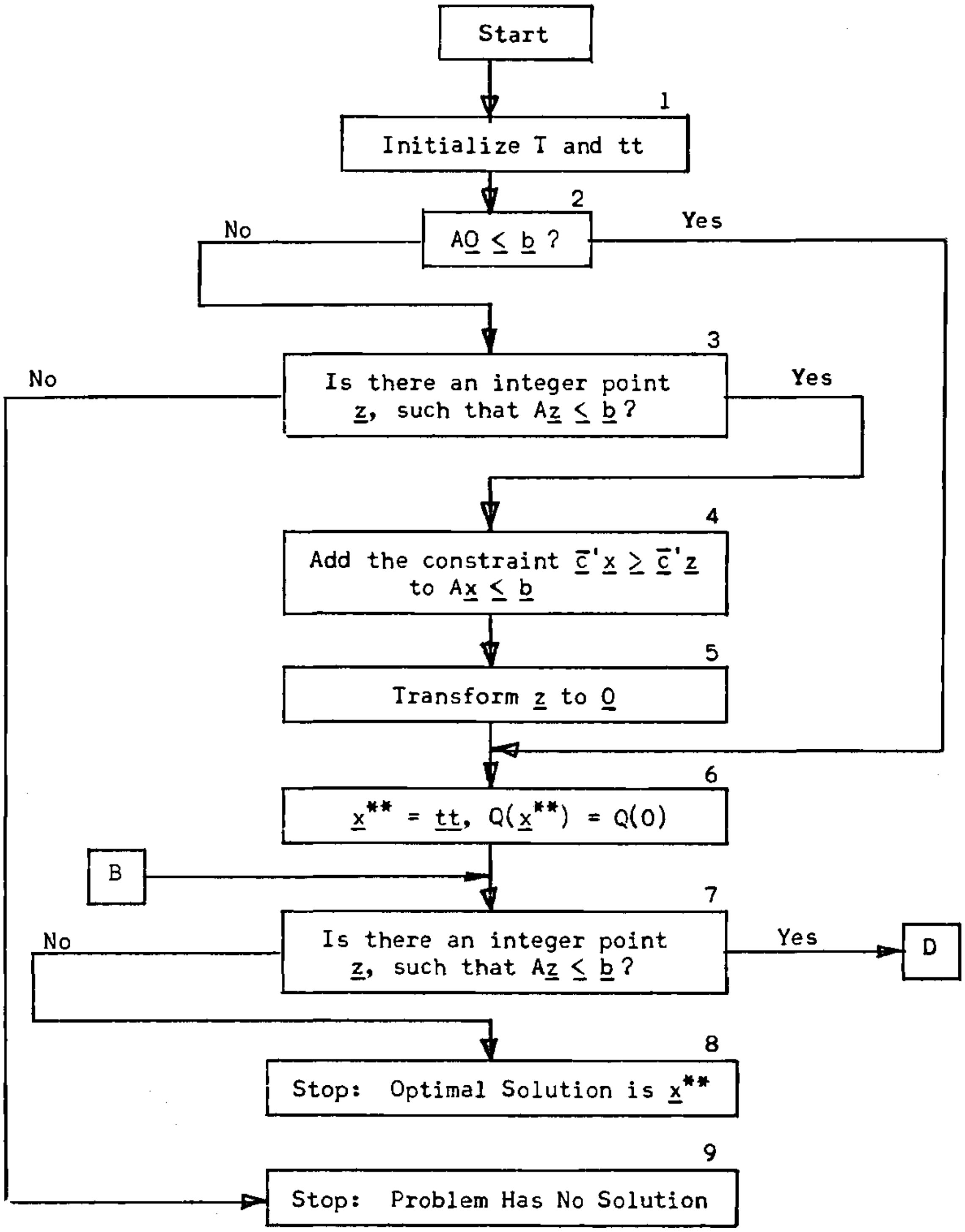
In terms of the notation used in the algorithm Figure 11

$$A = [\,1 \quad 1 \quad 1\,] \qquad \underline{b} = 2.75$$

$$C = \begin{matrix} -62 & 0 & 0 \\ 0 & -62 & 0 \\ 0 & 0 & -62 \end{matrix} \qquad \begin{aligned} \underline{\bar{c}}' &= [\,3 \quad 2 \quad 1\,] \\ \underline{c}' &= [-28 \quad -29 \quad -30]\,. \end{aligned}$$

The numbers at each step of the solution procedure reference the blocks in the flow chart of Section 5.

Flow Chart for the 0-1 Programming Algorithm



Start
1
Initialize T and tt
2
$A\underline{0} \leq \underline{b}$?
No
Yes
3
Is there an integer point $\underline{z}$, such that $A\underline{z} \leq \underline{b}$?
No
Yes
4
Add the constraint $\underline{\bar{c}}'\underline{x} \geq \underline{\bar{c}}'\underline{z}$ to $A\underline{x} \leq \underline{b}$
5
Transform $\underline{z}$ to $\underline{0}$
6
$\underline{x}^{**} = \underline{tt}$, $Q(\underline{x}^{**}) = Q(0)$
B
7
Is there an integer point $\underline{z}$, such that $A\underline{z} \leq \underline{b}$?
No
Yes
D
8
Stop: Optimal Solution is $\underline{x}^{**}$
9
Stop: Problem Has No Solution

Figure 8. Complete Flow Chart of the Integer Programming Algorithm.

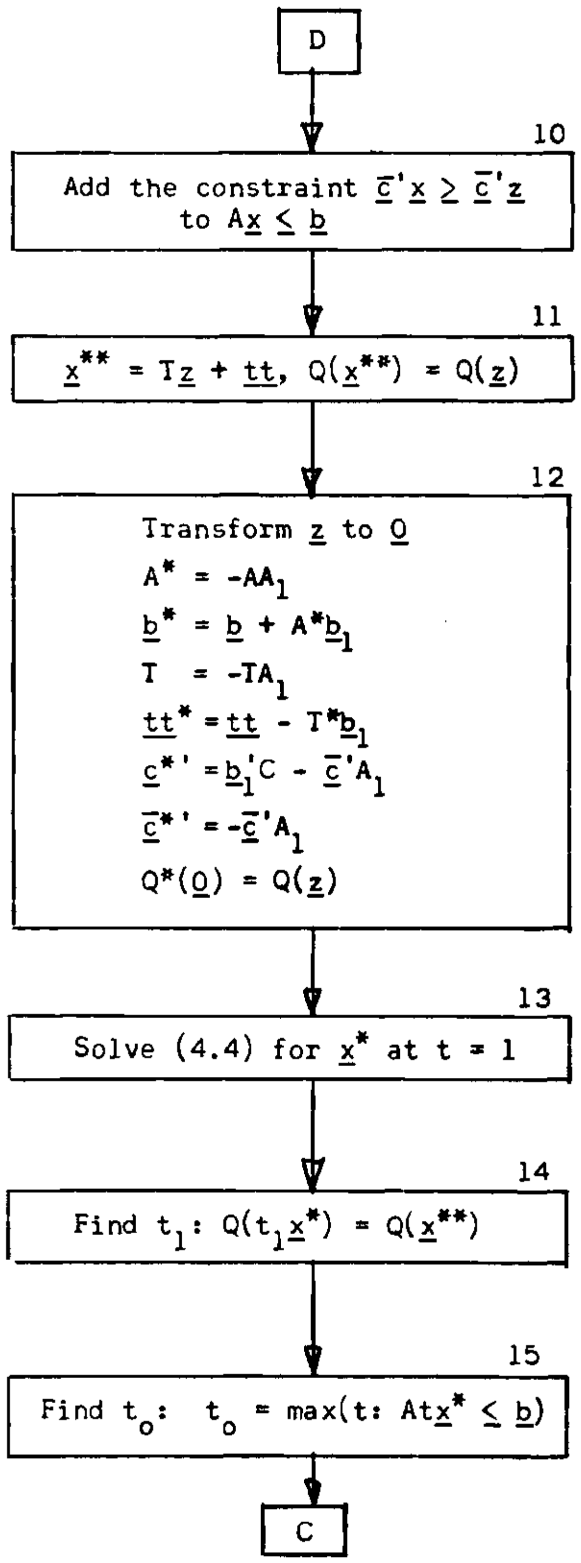
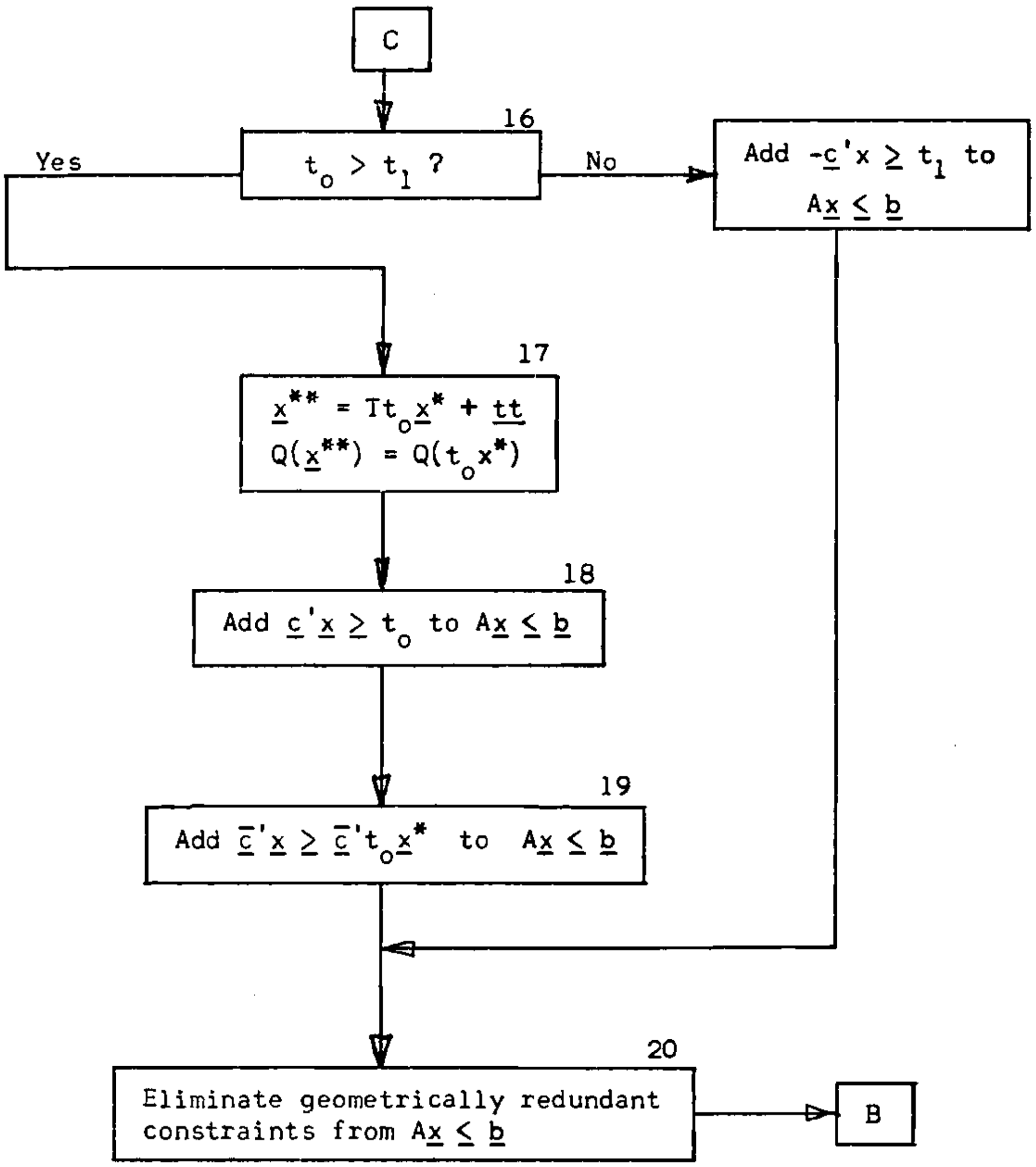

D
10
Add the constraint $\underline{\bar{c}}'\underline{x} \geq \underline{\bar{c}}'\underline{z}$ to $A\underline{x} \leq \underline{b}$
11
$\underline{x}^{**} = T\underline{z} + \underline{tt}$, $Q(\underline{x}^{**}) = Q(\underline{z})$
12
Transform $\underline{z}$ to $\underline{0}$
$A^{*} = -AA_1$
$\underline{b}^{*} = \underline{b} + A^{*}\underline{b}_1$
$T = -TA_1$
$\underline{tt}^{*} = \underline{tt} - T^{*}\underline{b}_1$
$\underline{c}^{*\prime} = \underline{b}_1'C - \underline{\bar{c}}'A_1$
$\underline{\bar{c}}^{*\prime} = -\underline{\bar{c}}'A_1$
$Q^{*}(\underline{0}) = Q(\underline{z})$
13
Solve (4.4) for $\underline{x}^{*}$ at $t = 1$
14
Find t_1: $Q(t_1\underline{x}^{*}) = Q(\underline{x}^{**})$
15
Find t_o: $t_o = \max(t: At\underline{x}^{*} \leq \underline{b})$
C

Figure 8. (Continued)



C
16
Yes
$t_o > t_1$?
No
Add $-\underline{c}'x \geq t_1$ to $A\underline{x} \leq \underline{b}$
17
$\underline{x}^{**} = Tt_o\underline{x}^* + \underline{tt}$
$Q(\underline{x}^{**}) = Q(t_o x^*)$
18
Add $\underline{c}'\underline{x} \geq t_o$ to $A\underline{x} \leq \underline{b}$
19
Add $\underline{\bar{c}}'\underline{x} \geq \underline{\bar{c}}'t_o\underline{x}^*$ to $A\underline{x} \leq \underline{b}$
20
Eliminate geometrically redundant constraints from $A\underline{x} \leq \underline{b}$
B

Figure 8. (Continued)

1. Initialize T and $\underline{tt}$

$$R = T = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \underline{tt}' = [0 \quad 0 \quad 0]$$

Note: R is a matrix which provides a transformation of a constraint in some $\underline{v}$-region to the identical constraint in the $\underline{x}$-region. If $\underline{a}_j'$ represents the coefficients of some transformed constraint, the $\underline{a}_j'R$ gives the coefficients of this constraint on the x-region. The primary purpose of R for the example problem is to allow the author to determine the cutting planes in terms of the original feasible region. Transformed regions often become difficult to draw, therefore all sketches for this example are in terms of the original feasible region although it should be remembered that transformed problems are actually being solved at each iteration. $R^* = -A_1R$ gives the transformation required for R.

2. $A\underline{0} \leq \underline{b}$? Yes

6. $x^{**} = \underline{0}$, $Q(\underline{x}^{**}) = Q(\underline{0}) = 0$

Iteration 1

7. Is there an integer point $\underline{z}$, such that $A\underline{z} < \underline{b}$? Yes, $\underline{z} = (0,1,1)$

10. Add the constraint $3x_1 + 2x_2 + x_3 \geq 3$ to $A\underline{x} \leq \underline{b}$. The original feasible region is shown in Figure 9, and the feasible region reduced by the above constraint is shown in Figure 10. Note that in Figure 13, points 1, 2, and 3 are no longer candidates for consideration.

11. $\underline{x}^{**} = T\underline{z} + \underline{tt} = (0, 1, 1)$. $Q(\underline{x}^{**}) = Q(\underline{z}) = 3$

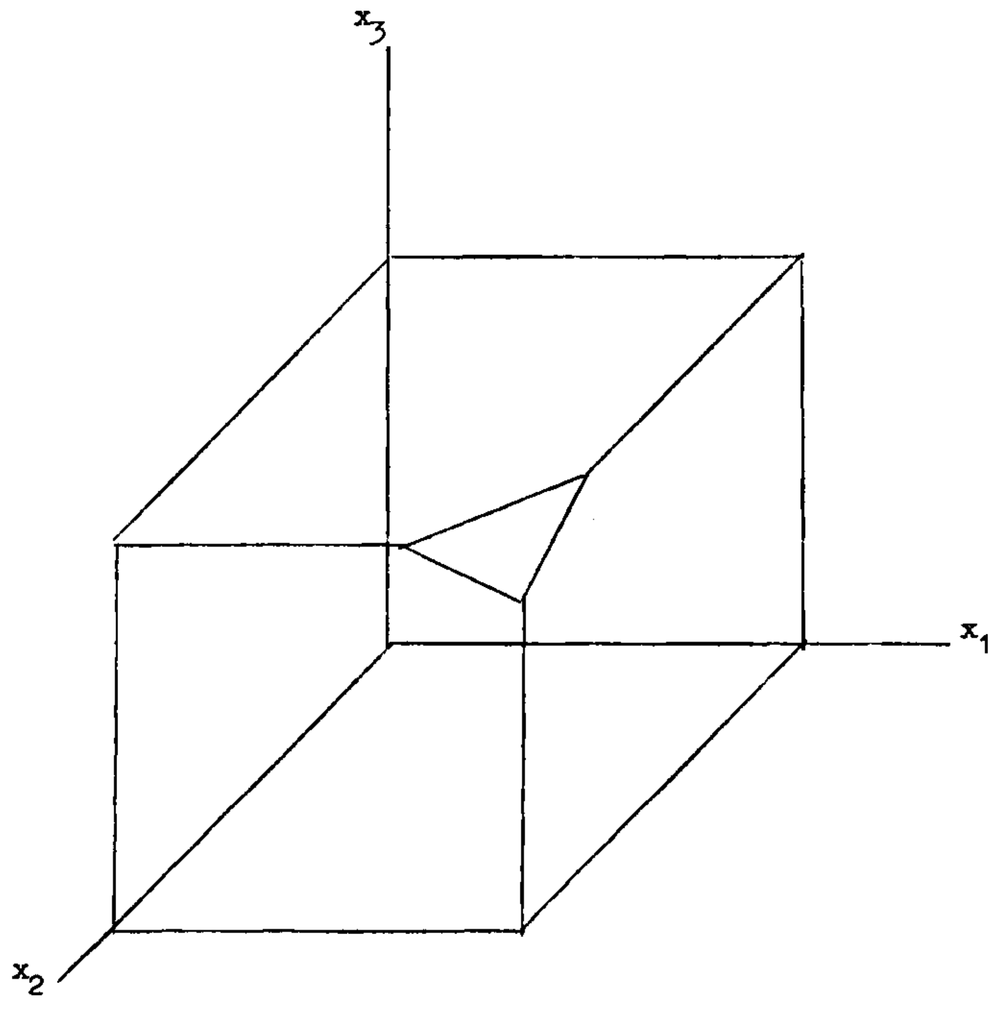


Figure 9. Feasible Region of Example Problem

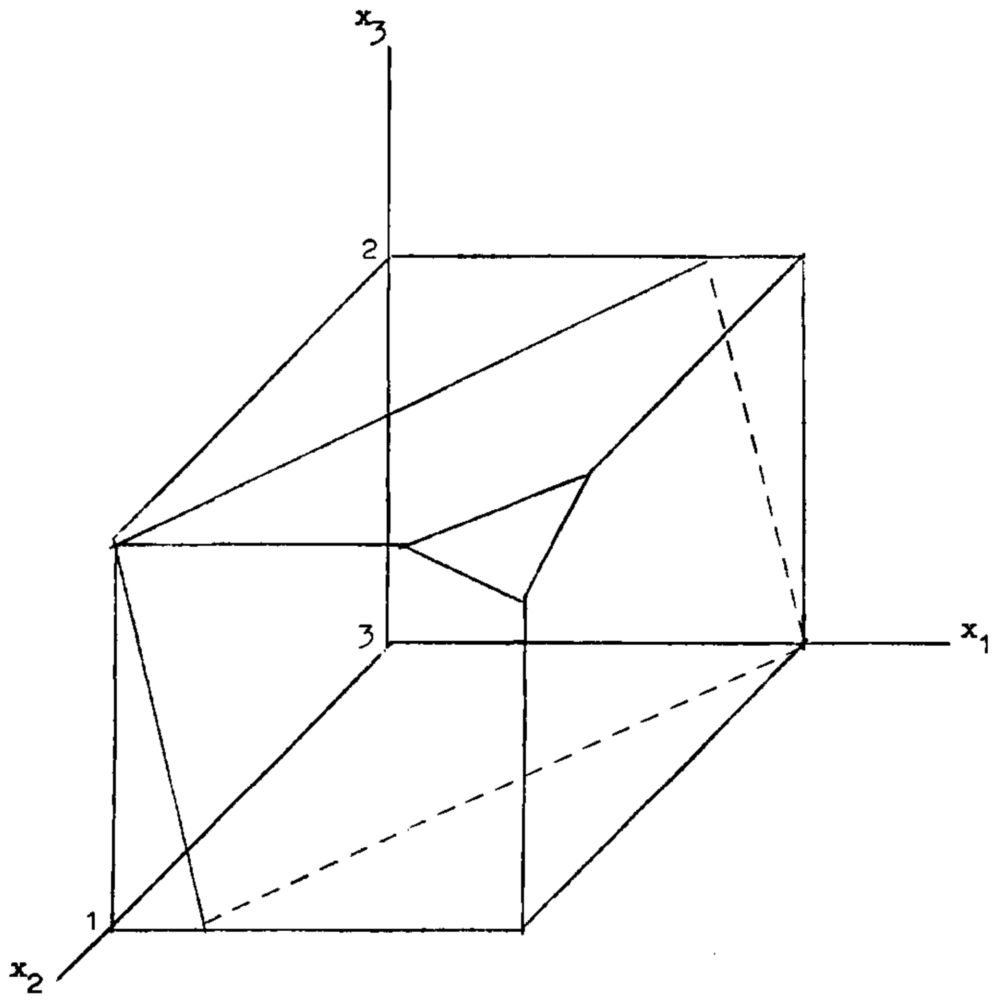


Figure 10. Illustration of First Cutting Plane (Iteration 1).

12. Transform $\underline{z}$ to $\underline{0}$

The transformed problem is as follows:

$$A = \begin{matrix} 1 & -1 & -1 \\ -3 & 2 & 1 \\ -62 & 0 & 0 \end{matrix} \qquad \underline{b} = \begin{matrix} 0.75 \\ 0 \end{matrix}$$

$$\underline{\bar{c}}' = [3 \quad -2 \quad -1]$$

$$C = \begin{matrix} 0 & -62 & 0 \\ 0 & 0 & -62 \end{matrix} \qquad \underline{c}' = [-28 \quad -33 \quad -32]$$

$$R = T = \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{matrix} \qquad \underline{tt} = \begin{matrix} 0 \\ 1 \\ 1 \end{matrix}$$

13. Solve (4.4) for $\underline{x}^*$ at $t = 1$

$\underline{x}^* = (1/28,\ 0,\ 0)$

14. Find t_1: $A(t_1\underline{x}^*) = Q(\underline{x}^{**})$

$t_1 = 25.29$

15. Find t_o: $t_o = \max(t\colon\ At\underline{x}^* < \underline{b})$

$t_o = 21.00$

16. $t_o > t_1$?

$21.00 > 25.29$? No.

21. Add $28x_1 + 33x_2 + 32x_3 > 25.29$ to $A\underline{x} \leq \underline{b}$. The reduced constraint set is as shown in Figure 11.

22. Eliminate geometrically redundant constraints from $A\underline{x} \leq \underline{b}$.

<u>Iteration 2</u>

7. Is there an integer point $\underline{z}$, such that $A\underline{z} \leq \underline{b}$,

Yes, $\underline{z} = (1,\ 1,\ 0)$.

10. Add $3x_1 - 2x_2 - x_1 \geq 1$ to $A\underline{x} \leq \underline{b}$.

The above cutting plane is as shown in Figure 12. Note that point 5 has also been eliminated from further consideration.

11. $\underline{x}^{**} = (1, 0, 1)$, $Q(\underline{x}^{**}) = 4$

12. Transform (1, 1, 0) to $\underline{0}$.

The transformed problem is as follows:

$$A = \begin{matrix} 28 & 33 & -32 \\ 3 & -2 & 1 \\ -1 & 1 & -1 \\ 3 & -2 & 1 \end{matrix} \qquad \underline{b} = \begin{matrix} 35.71 \\ 1 \\ 0.75 \\ 0 \end{matrix}$$

$$C = \text{same} \qquad \underline{\bar{c}}' = [-3 \quad 2 \quad -1]$$

$$\underline{c}' = [-34 \quad -29 \quad -32]$$

$$T = \begin{matrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{matrix} \qquad \underline{tt} = \begin{matrix} 1 \\ 0 \\ 1 \end{matrix}$$

$Q(\underline{0}) = 4$

13. Solve (4.4) for $\underline{x}^*$ at $t = 1$

$\underline{x}^* = (0, 1/29, 0)$

14. Find t_1: $Q(t_1\underline{x}^*) = Q(\underline{x}^{**})$

$t_1 = 27.13$

15. Find t_o: $t_o = \max(t\text{: } At\underline{x}^* \leq \underline{b})$

$t_o = 21.75$

16. $t_o > t_1$?

$21.75 > 27.13$ No

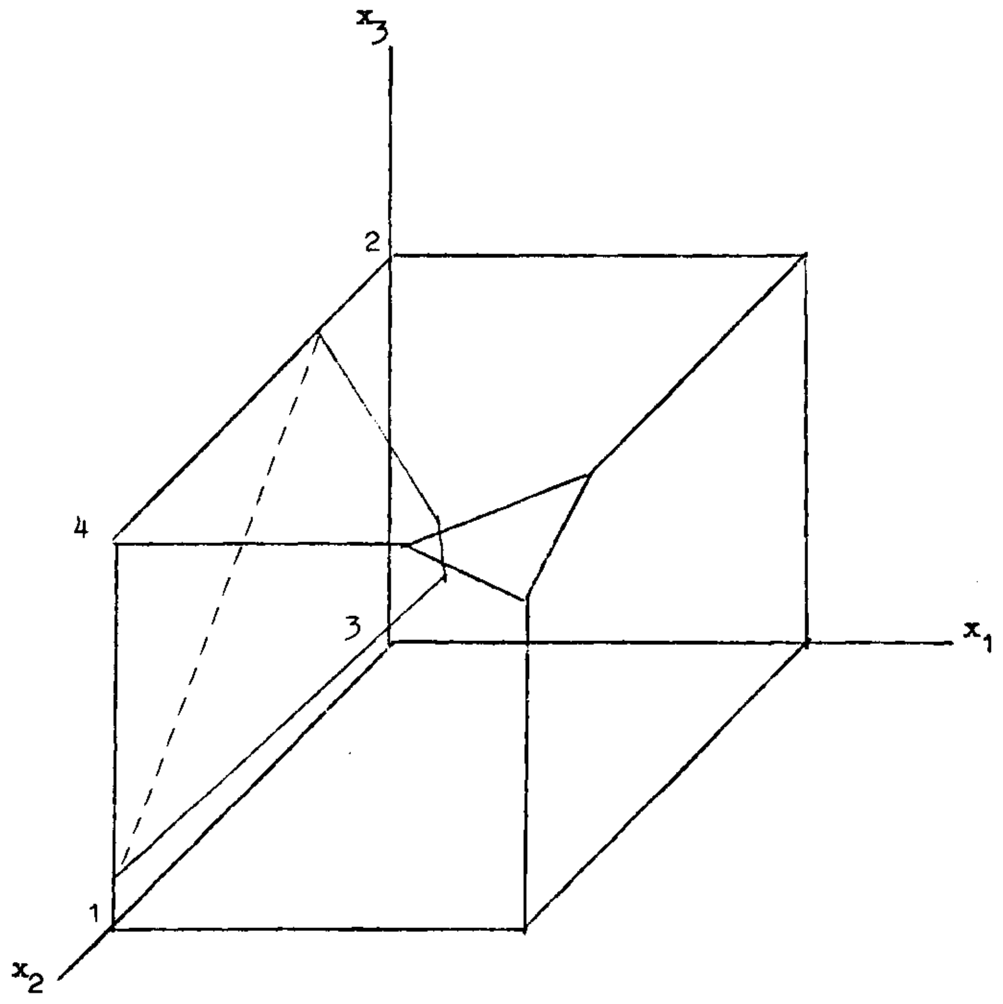


Figure 11. Illustration of Second Cutting Plane (iteration 1)

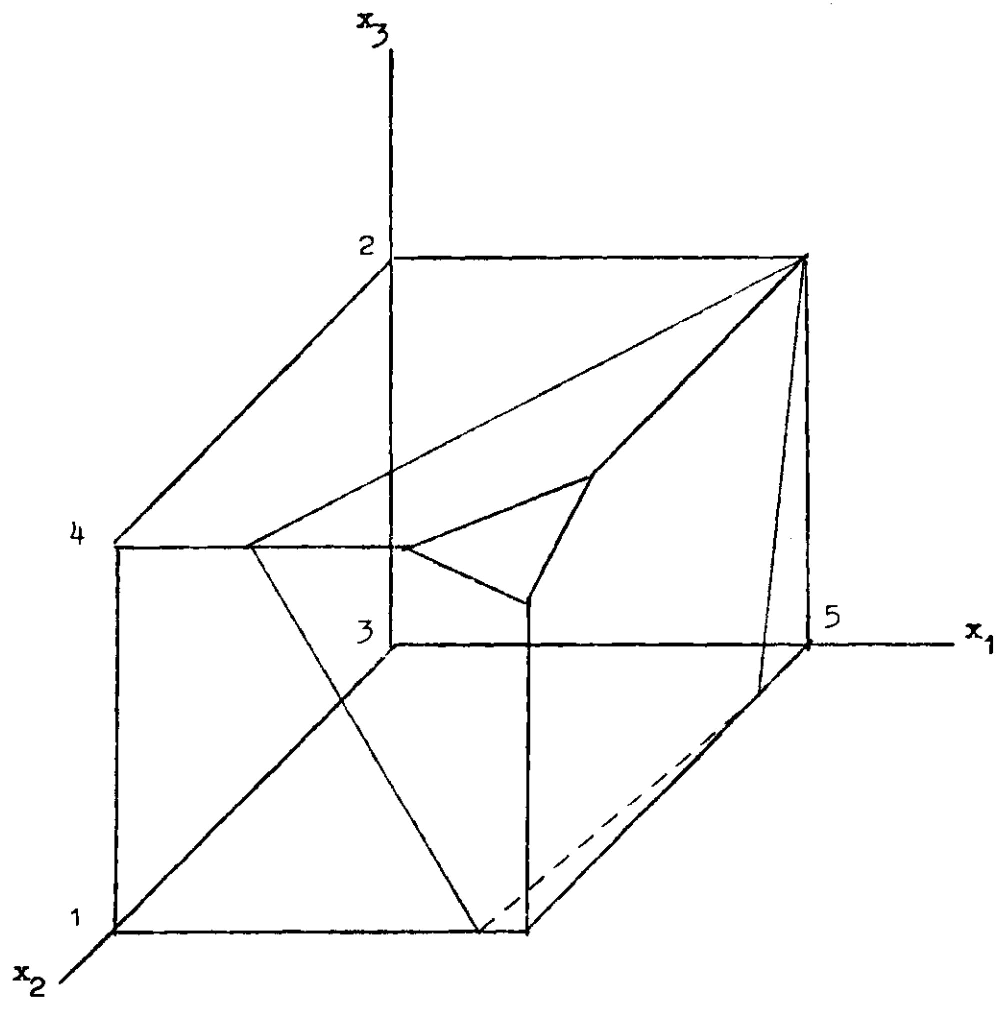


Figure 12. Illustration of Third Cutting Plane (Iteration 2)

21. Add $34x_1 + 29x_2 + 32x_3 \geq 27.13$ to $A\underline{x} \leq \underline{b}$.

Figure 13 shows the above cutting plane. Note that point 6 has now been eliminated.

22. Eliminate geometrically redundant constraints from $A\underline{x} \leq \underline{b}$

Iteration 3

7. Is there an integer point $\underline{z}$, such that $A\underline{z} \leq b$?

Yes, let $\underline{z} = (0, 1, 1)$

10. Add $-3x_1 + 2x_2 - x_3 \geq 1$ to $A\underline{x} \leq \underline{b}$.

Figure 14 shows the new cutting plane.

11. $\underline{x}^{**} = (1, 1, 0)$, $Q(\underline{x}^{**}) = 5$

12. Transform $\underline{z}$ to $\underline{0}$

The transformed problem is as follows:

$$A = \begin{matrix} -1 & -1 & 1 \\ 28 & -33 & 32 \\ -34 & 29 & 32 \\ 3 & 2 & -1 \\ 3 & 2 & -1 \end{matrix} \qquad \underline{b} = \begin{matrix} 0.75 \\ 34.71 \\ 33.89 \\ 1.00 \\ 0.000 \end{matrix}$$

$C =$ same $\qquad \underline{\bar{c}}' = [-3 \quad -2 \quad 1]$

$\qquad \underline{c}' = [-34 \quad -33 \quad -30]$

$$T = \begin{matrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \underline{tt} = \begin{matrix} \underline{1} \\ \underline{1} \\ 0 \end{matrix}$$

13. Solve (4.4) for $\underline{x}^*$ at $t = 1$

$\underline{x}^* = (0, 0, 1/30)$

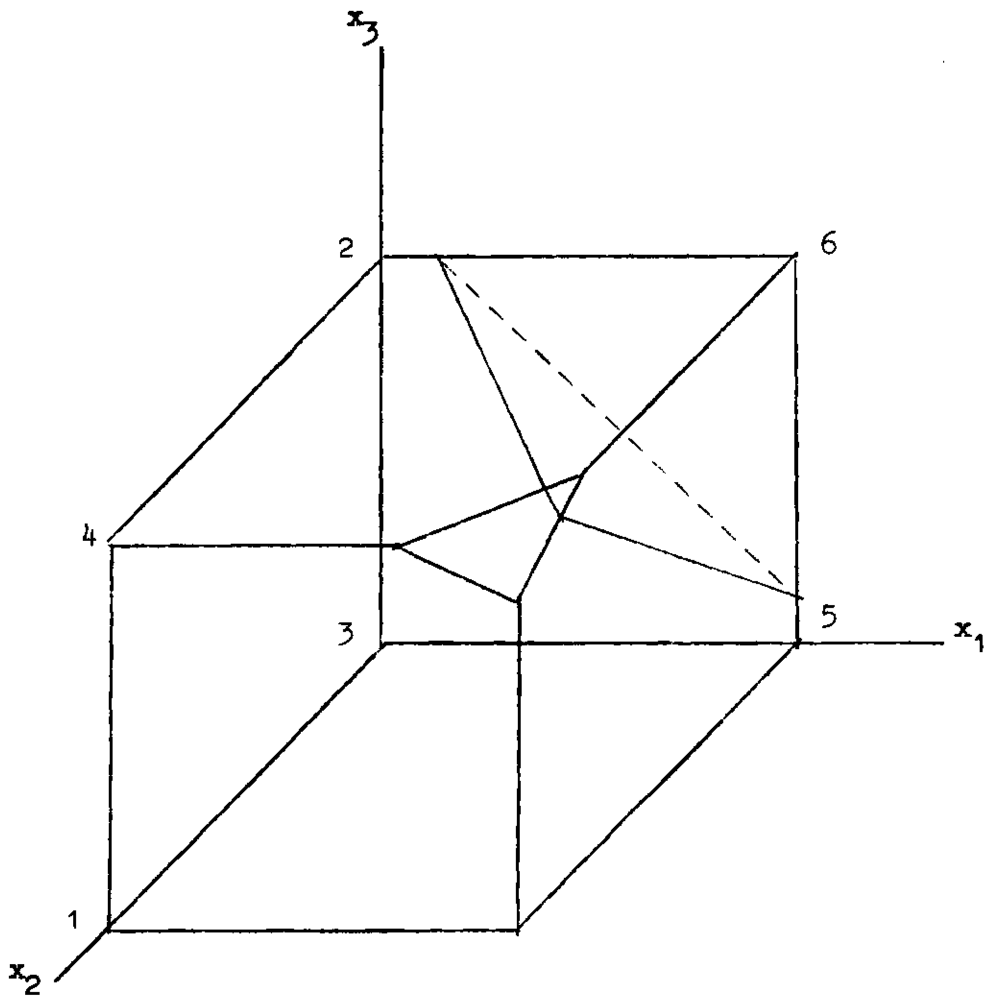


Figure 13. Illustration of Forth Cutting Plane
(Iteration 2)

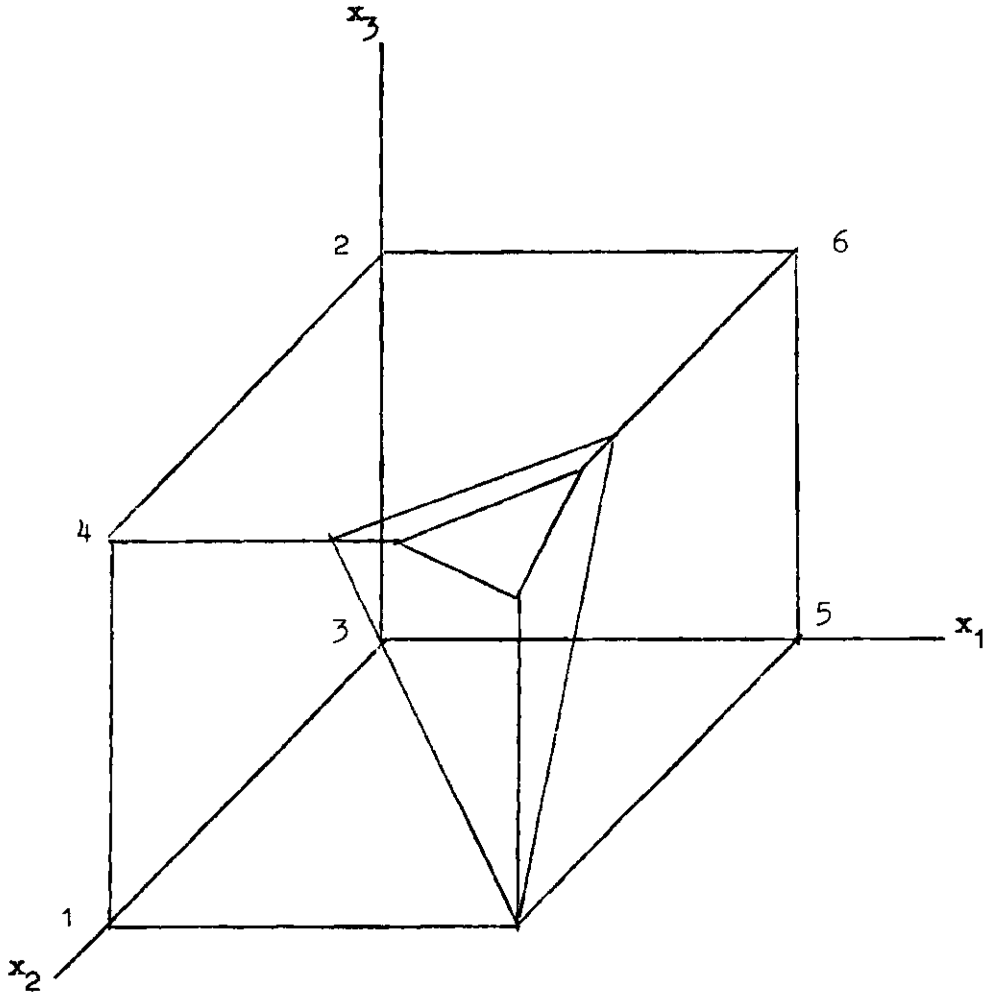


Figure 14. Illustration of Fifth Cutting Plane
(Iteration 3)

14. Find t_1: $Q(\underline{x}^{**})$

$t_1 = 29.03$

15. Find t_o : $t_o = \max(t\colon\ At\underline{x}^{*} \leq \underline{b})$

$t_o = 22.50$

16. $t_o > t_1$?

$22.50 > 29.03$? No

21. Add $34x_1 + 33x_2 + 30x_3 \geq 29.03$ to $A\underline{x} \leq \underline{b}$.

The cutting plane is as shown in Figure 15. Note that all seven feasible integer extreme points have been eliminated from further consideration.

22. Eliminate geometrically redundant constraints from $A\underline{x} \leq \underline{b}$.

<u>Iteration 4</u>

7. Is there an integer point $\underline{z}$, such that $A\underline{z} \leq \underline{b}$. No

8. Stop: Optimal Solution is $\underline{x}^{**} = (1, 1, 0)$

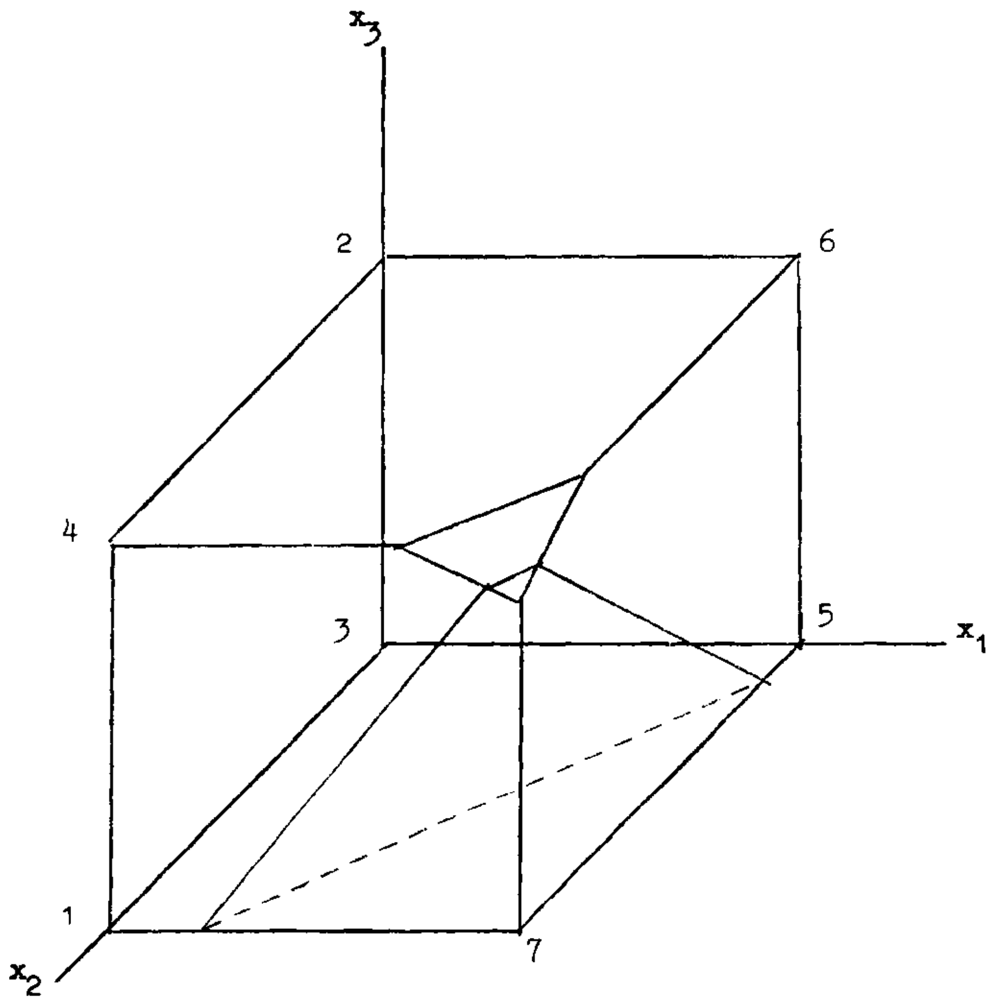


Figure 15. Illustration of Sixth Cutting Plane (Iteration 3)

CHAPTER V

COMPUTATIONAL EXPERIENCE

A computer code was developed from the algorithm of Figure 11 and used to obtain the computational results reported in Table 2. The computer code is written entirely in Fortran V for the Univac 1108. The object program and data are all held in core with all program data in floating point. Total number of 36 fixed bit storage words required for this program is $2mn + 10m + 4n^2 + 11n + 2305$ where m is the number of constraints and n is the number of 0-1 variables.

The program was written in four segments as follows:

a) MAIN

b) FIND

c) TRANS

and d) SUPER.

MAIN is the control program which directs the flow of the algorithm as well as generating the cutting planes required to reduce the feasible region. The subroutine FIND is used to locate a feasible local optimum required at each iteration. This subroutine is essentially the implicit enumeration algorithm of Balas (1) with revisions to transfer control back to MAIN after each new feasible local optimum has been found. The TRANS subroutine transforms this local optimum to the origin while storing reverse transformations to enable the algorithm to reference extreme points in the original region. The last subroutine, SUPER, is used to eliminate geometrically redundant constraints after each transformation. The linear programming routine required is basically a revised simplex algorithm which has been modified to take advantage of the special structure of the problem.

The test problems were taken from the literature and are referenced in Table 2. The computational times for other algorithms were reported by Geoffrion (10) and are repeated for purposes of comparison.

The limited computational experience reported here indicates that the present form of the computer code used to develop the computation experience is inferior to the code used by Geoffrion (10). The program developed in this research has not been refined at this writing and consequently there are several opportunities for improvement which have not been explored. The main objective of the programming effort was to obtain a code that worked so that, at least, some computational experience could be reported. This objective has been accomplished. Further refinement of the computer code is left as a future area of research.

Table 2. Comparative Computational Experience

Problem	Problem Size 0-1 Var. x Const.	Balas(10) (7044) min.	Geoffrion(10) (7044) min.	Other Algorithms min.	Other Algorithms Ref.	Quadratic Approach (U1108) min.
Bouvier and Messoumian(7)						
15	20 x 20	0.48	0.09	0.47	7	0.42
16	20 x 20	1.69	0.62	2.07	7	1.30
17	20 x 23	1.06	0.64	0.85	7	0.44
23	27 x 20	8.08	1.18	7.10	7	7.21
Petersen(23)						
4	20 x 10	0.46	0.04	0.06	23	0.89
Haldi(16)						
II-7	20 x 4	0.10	0.03	0.02	20	3.50
II-8	20 x 4	0.10	0.05	0.40	20	5.07
II-10	30 x 10	0.41	0.06	-	-	1.81
IBM(16)						
1	21 x 7	0.14	0.01	0.13	27	0.26
2	21 x 7	0.11	0.02	0.17	27	0.23
3	20 x 3	0.04	0.01	0.04	27	0.03

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

The objective of this research was to develop a 0-1 integer programming algorithm via a quadratic programming approach. An algorithm based on this approach has been developed, finiteness of the algorithm has been proved, and several published test problems have been solved by a computer code developed from this algorithm. The singular result of this research is the 0-1 integer programming algorithm presented in Chapter IV.

Even though the practicality of the quadratic programming approach to the solution of the 0-1 integer programming problem has been established, there still remain many unanswered questions concerning this approach. The following is a brief outline of recommendations for further research in the area of integer programming via quadratic programming approach.

a) Can the computer code written by the author be refined to increase its efficiency? There are, at least, two avenues which may be pursued in this endeavor. First, the present code uses the SUPER subroutine at each iteration. At some computational cost, this routine locates and eliminates any geometrically redundant constraints present in the constraint set to thereby reduce the computation time, in other phases of the program. The computational relationship between carrying say k redundant constraints and eliminating them is not known. It may be possible that more time is expended in locating and eliminating a few redundant constraints than would have been incurred had they been carried along through the other phases of the program. If this relationship were known, it may be advantageous to apply routine SUPER less frequently than at every iteration. The second avenue available would be to revise the program to work only in the original region rather than the transformed regions. This would eliminate the necessity to make two (one forward and one backward) transformations at each iteration.

b) Does the algorithm perform better on a specific type of problem? It is impossible to determine this with the computational experience presently available.

c) How does the computational time increase with respect to the number of variables and number of constraints? Again this can only be determined after much more computational experience is available.

d) Can the integer programming algorithm developed in Chapter IV be extended to solve the mixed 0-1 integer programming problem? The logical approach to follow here is to use Benders (6) partitioning method. This method requires that one be able to solve a problem of the form

$$(6.1) \qquad \begin{aligned} \text{Max:} \quad & z \\ \text{Subj:} \quad & z \leq \underline{\bar{c}}_1'\underline{x} + (\underline{b} - A_1\underline{x})'\underline{u}^1 \\ & \vdots \\ & z \leq \underline{\bar{c}}_1'\underline{x} + (\underline{b} - A_1\underline{x})'\underline{u}^P \\ & x_i = 0 \text{ or } 1 \text{ for all } i \end{aligned}$$

where z and $\underline{x}$ are variables, and $\underline{\bar{c}}_1$, $\underline{b}$, A_1, and $\underline{u}^p$ $(p = 1,\ldots,P)$ are constants. The problem (6.1) is a pure 0-1 integer programming problem with an additional continuous variable z. If one attempts to use the quadratic approach presented in Chapter IV, a problem of the form (6.2) will be obtained.

$$(6.2) \qquad \begin{aligned} \text{Max:} \quad & Q(\underline{y}) = \underline{c}'\underline{y} - \frac{1}{2}\underline{y}'C\underline{y} \\ \text{Subj:} \quad & A\underline{y} \le \underline{b} \\ & \underline{y} \ge \underline{0} \end{aligned}$$

where $\underline{y}' = (\underline{x}', z_1, z_2)$, $\quad z = z_1 - z_2$

$\underline{c}' = (a\underline{1}', 1, -1)$

$$C = \left[\begin{array}{c|c|c} -2aI & \underline{0} & \underline{0} \\ \hline \underline{0}' & 0 & 0 \\ \hline \underline{0}' & 0 & 0 \end{array}\right]$$

Note that the C matrix contains two diagonal elements which are zero. Recall that from Chapter IV section 4.2, a t_1 greater than zero is only guaranteed if $c_{ii} < 0$ for all i. Since (6.2) does not initially meet this criteria, Benders partitioning method can not be used without extensive modifications to the present algorithm. This problem is a very important area for future work.

LITERATURE CITED

1. Balas, Egon. 1965, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," Operations Research, 13, 5-7-546.

2. Balas, E. 1967b. "Discrete Programming by the Filter Method," Opns. Res., 15, 915-157.

3. Balas, E., Duality in Discrete Programming, Technical Report No. 67-5, July 1967, Department of Operations Research, Stanford University, Revised December 1967.

4. Balinski, M. and K. Spielberg, "Methods for Integer Programming: Algebraic, Combinatorial, and Enumerative," Progress in Operations Research, V. III, Relationship between Operations Research and the Computer, (edited by Julius Arnofsky) John Wiley and Sons, 195-292, 1969.

5. Bellman, Richard. 1956. "Maximization over Discrete Sets," Naval Research Logistics Quarterly, 3, 67-70.

6. Benders, J. F. 1962. "Partitioning Procedures for Solving Mixed-Variables Programming Problems," Numerische Mathematik, 4, 238-252.

7. Bouvier, B., and G. Messoumian. 1965. "Programmes Lineaires en Variables Bivalentes-Algorithme de E. Balas," Faculte des Sciences de Grenoble.

8. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson. 1954 "Solution of a Large Scale Travelling Salesman Problem," Opns. Res., 2, 393-410.

9. Geoffrion, A. M., Integer Programming by Implicit Enumeration and Balas' Method, Rand Corporation Memorandum, RM-4783-PR, February 1966.

10. Geoffrion, A. M., An Improved Implicit Enumeration Approach for Integer Programming, Rand Corporation Memorandum, RM-5644-PR, June 1968.

11. Glover, F. 1965c. "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Opns. Res., 13, 879-919.

12. Gomory, R. E. 1958a. "An Algorithm for Integer Solutions to Linear Programs," pp. 269-302 in Graves and Wolfe, (1963). First Issued as Princeton-IBM Mathematics Research Project, Tech. Rept. 1, Nov. 17.

13. Gomory, R. E. 1960a. "All-Integer Integer Programming Algorithm," pp. 193-206 in Muth and Thompson (1963). First Issued as IBM Research Center, Res. Rept. RC-189, Jan. 29.

14. Graves G., and Whinston A., "A New Approach to Discrete Mathematical Programming," Management Science, Vol. 15, No. 3, November 1968.

15. Hadley, G., Nonlinear and Dynamic Programming, 1964, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

16. Haldi, J. 1964. "25 Integer Programming Test Problems," Stanford University, Working Paper No. 43, Dec.

17. Hammer, P. L., "A Boolean-Branch-and-Bound Method for Linear and Nonlinear Bivalent Programming," Working Paper Weizmann Institute of Science, Rehovoth, 1968.

18. Hillier, F. S., and Lieberman, G. J., Introduction to Operations Research, 1968, Holden-Day, Inc., San Francisco, California.

19. Land, A. H., and A. G. Doig, 1969. "An Automatic Method for Solving Discrete Programming Problems," Econometrica, 28, 497-520.

20. Lemke, C. and K. Spielberg, "Direct Search Algorithms for Zero-One and Mixed Integer Programming," Operations Research, V. 15, No. 5, 892-914, October 1967.

21. Mangasarian, Olvi L., Nonlinear Programming, 1969, McGraw-Hill Inc., New York, New York.

22. Markowitz, Harry M., and A. S. Manne. 1957. "On the Solution of Discrete Programming Problems," Econometrica, 25, 84-110.

23. Petersen, Clifford C. 1967. "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R and D Projects," Mgmt. Sci., 13, 736-750.

24. Raghavachari, M., "On Connections Between Zero-One Integer Programming and Concave Programming Under Linear Constraints," Journal of Operations Research, Vol. 17, No. 4, pp. 680-684 (1969).

25. Ritter, K., "A Method for Solving Maximum Problems with a Non-concave Quadratic Objective Function," Z. Wahrscheinlichkeitstheorie 4, 340-351 (1966).

26. Young, R. D., "A Simplified Primal (All-Integer) Integer Programming Algorithm," Journal of Operations Research, Vol. 16, No. 4, (1968).

27. Woiler, S., "Implicit Enumeration Algorithms for Discrete Optimization Problems," Ph.D. Dissertation, Department of Industrial Engineering, Stanford University, May 1967.

28. Wolfe, P., The Simplex Method for Quadratic Programming. Econometrica, 27, 1959, No. 3.

OTHER REFERENCES

Balinski, M. L. 1965. "Integer Programming: Methods, Uses, Computations," Mgmt. Sci., 12, 253-313.

Cabot, A. V. and Francis, R. L., "Solving Certain Nonconvex Quadratic Minimization Problems by Ranking the Extreme Points," Journal of Operations Research, Vol. 18, No. 1, p. 82 (1970).

Climescu, Al., "La Programmation Algebrique," Buletinul Institutului Politechnic, Din Iasi, Serie Novā, Tomul VIII (XII), Fasc. 1-2, 1962.

Dantzig, G. B. 1960. "On the Significance of Solving Linear Programming Problems with Some Integer Variables," Econometrica, 28, 30-44.

Glover, F., and S. Zionts. 1965. "A Note on the Additive Algorithm of Balas," Opns. Res., 13, 546-549.

Gomory, R. E. 1958b. "Outline of an Algorithm for Integer Solutions to Linear Programs," Bulletin of the American Mathematical Society, 64, 275.

Hadley, G., Linear Programming, 1962, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Hu, T. C., Integer Programming and Network Flows, 1969, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Lawler, E. L., and D. E. Wood. 1966. "Branch-and-Bound Methods: A Survey," Opns. Res., 14, 699-719.

Mueller, R. K., "A Method for Solving the Indefinite Quadratic Programming Problem," Management Science, Vol. 16, No. 5, January, 1970.

Murty, Katta G., "Solving the Fixed Charge Problem by Ranking the Extreme Points," Journal of Operations Research, Vol. 16, No. 2, pp. 268-279 (1968).

Ritter, K., "Stationary Points of Quadratic Maximum-Problems," Z. Wahrscheinlichkeitstheorie, verw. Geb 4, 149-158 (1965).

Tui, Hoang, "Concave Programming Under Linear Constraints," Soviet-Math., 1937-1940 (1964).