

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.

[Handwritten signature]
_____ 10

7/25/68

AN IMPLICIT ENUMERATION ALGORITHM
FOR INTEGER PROGRAMMING

A THESIS

Presented to

The Faculty of the Graduate Division

by

Leslie Earl Trotter, Jr.

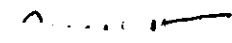
In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in the School of Industrial Engineering

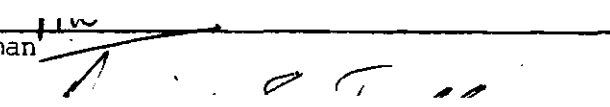
Georgia Institute of Technology

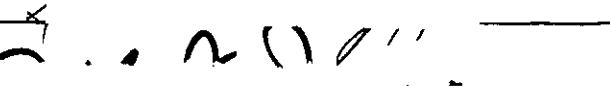
September, 1970


AN IMPLICIT ENUMERATION ALGORITHM
FOR INTEGER PROGRAMMING

Approved:



Chairman 





Date approved by Chairman: Dec. 8, 1970

ACKNOWLEDGMENTS

The author expresses his deepest appreciation and thanks to Dr. C. M. Shetty whose perfect combination of instruction and inspiration made this work possible; his keen insight serves as a continuing inspiration to the author.

In addition, I would like to thank Dr. D. E. Fyffe and Dr. J. J. Jarvis for their helpful comments and suggestions concerning all aspects of the research presented herein.

Special thanks are also due to G. E. Moore of Control Data Corporation who granted the CDC 6600 computer time required for the computational investigation pursued in this thesis. Such an in-depth computational study would certainly have been impossible without this assistance.

I would also like to express my gratitude to my wife, Jomi, for her continued insistence upon the excellence of this work despite the many long hours of my attention which it required.

Lastly, I wish to thank Betty Sims for her excellent typing job.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS.	ii
LIST OF TABLES	v
LIST OF ILLUSTRATIONS.	vi
SUMMARY.	vii
Chapter	
I. INTRODUCTION	1
Cutting-Plane Algorithms	
Branch and Bound Methods	
Geoffrion's Algorithm	
Scope of this Study	
II. BASIC ALGORITHM.	8
Problem Statement	
Definitions	
Statement of the Algorithm	
Non-Redundance and Exhaustiveness	
III. MATHEMATICAL IMPLEMENTATION.	22
Initial Solution	
Simple Tests for Fathoming	
Augmentation	
Fathoming Several Partial Solutions Simultaneously	
IV. SURROGATE CONSTRAINTS.	34
Introduction	
Strongest Surrogate Constraints	
Sequential Solutions to the Imbedded Linear Program	
Fathoming by Optimality Considerations	
Relative Bound Redefinition	
Near Optimal Solutions	
An Alternative View of the Imbedded Linear Program	
Geometric Interpretation of Surrogate Constraints	

Chapter	Page
V. COMPUTATIONAL RESULTS.	57
Basic Considerations	
Results	
VI. CONCLUSIONS AND RECOMMENDATIONS.	74
APPENDIX	
A.	81
B.	87
C.	105
BIBLIOGRAPHY	118

LIST OF TABLES

Table		Page
1.	Problem Characteristics	63
2.	Results Using Minimum-Branch Augmentation and No Bound Redefinition	69
3.	Results Using Balas' Augmentation Rule and No Bound Redefinition	70
4.	Results Using Minimum-Branch Augmentation and Bound Redefinition.	71
5.	Results Using Balas' Augmentation Rule and Bound Redefinition.	72
6.	A Comparison Between Binary and Direct Implicit Enumeration	73

LIST OF ILLUSTRATIONS

Figure		Page
1.	Flow Diagram of Implicit Enumeration Algorithm.	58
2.	Flow Diagram of Imbedded Linear Program Fathoming Process	59

SUMMARY

This study deals mainly with the bounded variable pure integer programming problem of the form $\text{Min } cx$ subject to linear constraints where the variables must assume bounded integer values. The algorithm proposed for solving such a problem is an implicit enumeration procedure closely related to that advanced by Geoffrion for binary programming problems. However, the proposed algorithm deals with the bounds on program variables directly. Following Geoffrion, we develop various tests for implicitly enumerating "completions" to partial solutions; these tests are then extended to permit implicit enumeration of "completions" to several partial solutions simultaneously. The use of surrogate constraints in such implicit enumerations is examined in depth and two new interpretations of such surrogate constraints are given, including a geometric interpretation formulated in terms of determining a separating hyperplane between two subsets of the space of lattice points being enumerated. Also, means of obtaining near optimum solutions through a slight modification of the algorithm are discussed.

The theoretical considerations are supplemented by a computational study which deals not only generally with the efficiency of the algorithm developed as a whole, but also specifically with an evaluation of some of the more important decision steps of the algorithm. The results of this computational study indicate not only that the proposed algorithm is far more effective than existing methods for solving

integer programming problems but also that its ability to dynamically redefine bounds on program variables relative to specific partial solutions yields significant improvement in the efficiency of the enumeration process. The computational investigation also indicates that the proposed algorithm will allow the efficient solution of integer programming problems subject only to core memory restrictions of present-day computers.

CHAPTER I

INTRODUCTION

In recent years the field of Operations Research has witnessed a sizable increase in the methodology concerned with discrete variable programming problems; a formal statement of such a problem is given by Equations (1)-(4) in Chapter II. The literature now reflects many varied approaches for solving the general integer programming problem. The computational success, however, of virtually all existing algorithms is limited, thus leaving the issue of an economical means for determining the solution to a given integer program largely unsettled. Both the size of the current research effort in the field of integer programming [3,4]^{*} and the large number of problems which can be stated in discrete variables attest to the importance of this area. In fact, given computationally efficient means of solving integer programs, many industrial problems currently stated as linear programming models, would be more realistically phrased in the context of integer programming. This thesis deals mainly with an extension of a particular integer programming algorithm [10]; consequently only a broad view of existing methods of integer programming will be discussed here. In this regard, the works of Balinski [3,4] provide an excellent survey of all aspects of integer programming.

* Bracketed numbers refer to references cited in the Bibliography.

Cutting-Plane Algorithms

Despite the variety of available algorithms for integer programming, the majority of those proposed to date belong to either of two* broad categories: cutting-plane methods and branch and bound methods. Historically, cutting-plane techniques were developed first, with the bulk of the initial contribution in this area due to R. E. Gomory [15,16,17]. The basic cutting-plane approach requires that one first remove the integrality restrictions on all program variables and solve the resulting linear program; then by alternating the introduction of additional constraints (cuts) with re-optimization, one continually restricts the feasible region of solution until an optimum solution thus obtained is integer-valued. Of course, the character of an algorithm of this type requires that one take appropriate steps to insure that no feasible integral-valued solutions are passed over during the process; this demonstrates the dual nature of most cutting-plane algorithms in the sense that intermediate results are infeasible until one simultaneously achieves optimality and feasibility of the original problem. Primal cutting-plane algorithms have also been developed [13,27] which attain optimality by passing through a continually improving sequence of feasible solutions. More recently, promising theoretical results have been produced [18] for cutting-plane algorithms by using a group theoretic approach to develop cuts associated with the faces of the convex

*A more detailed categorization is, of course, possible. For example, [3,4] consider additional methods such as dynamic programming and heuristic techniques.

hull of feasible integer solutions to the original integer programming problem.

To a large extent, computational experience with cutting-plane algorithms has proven disappointing. Some problems are solved quite easily while others, differing only slightly, yield results only after an inordinate amount of computation. A comprehensive explanation of the extreme sensitivity of cutting-plane algorithms to problem structure has not yet been proposed.

Branch and Bound Methods

Recent developments in integer linear programming show an increasing amount of attention being focused on branch and bound methods of solution. The idea central to branch and bound techniques is that of conducting a highly specialized search of the entire solution space in such a manner that a large number of possible solutions will be eliminated from consideration by exclusion tests based on the character of the algorithm. These methods may be conceptualized as tree-search algorithms which implicitly or explicitly consider each possible solution in its turn in a systematic effort to determine an optimal solution. Intuitively, an effective algorithm of this variety must possess an efficient means of remembering which solutions have been considered; further, the strength of a branch and bound algorithm will lie in its ability to implicitly enumerate a large proportion of the possible solutions. These two ingredients, an efficient history-remembering scheme and a high rate of implicit exclusion, are essential in evaluating the computational effectiveness of a branch and bound algorithm.

Initial branch and bound approaches were advanced by Eastman [6], Land and Doig [20], Little, et al. [22], and Balas [1]. The survey article by Lawler and Wood [21] gives an excellent account of branch and bound history and methodology. In addition, the work of Mitten [24] provides a good discussion on the structure of this methodology. Balas has extended his work in [1] to produce his Filter Algorithm [2] for binary (0-1) integer linear programs. This algorithm, dual in nature, uses the solution to a linear programming problem to develop a "filter constraint" which speeds the convergence of the algorithm to an optimal solution. This concept of using a surrogate constraint (so-called because of its direct derivation from the "parent" constraints of the original problem) to accelerate convergence was first proposed by Glover [12] and later modified and extended by others [2,8,10,14].

Geoffrion's Algorithm

The research of Geoffrion [7,8,9,10,11] in this area appears to be especially promising in terms of computational results. Geoffrion's original work [7] was a refinement of Balas' additive algorithm [1] which was very amenable to ease of computation. His later extensions [8,9,10,11] include an imbedded linear program which is solved at successive iterations of the algorithm in order to produce surrogate constraints which are highly effective in relation to the exclusion tests of the algorithm. The computational results presented by Geoffrion in [10] illustrating the improvement gained by using this type of surrogate constraint are very convincing; this evidence seems to indicate that, for his test problems, the increase in solution time as a function of

the number of binary variables in the original problem is linear or low-order monomial.

Scope of this Study

Geoffrion's algorithm is stated within the context of the binary programming problem; since any integer linear program whose variables are bounded can be represented in binary form, his algorithm is also applicable to the general bounded variable pure integer programming problem. We note that any bounded integer variable $0 \leq x \leq n$ may be represented as the sum of binary variables; i.e., $x = 2^0 x_0 + 2^1 x_1 + \dots + 2^k x_k$ where k is the smallest integer such that $n \leq 2^{k+1} - 1$ and x_j , for $j = 0, 1, \dots, k$ are binary variables. Clearly, in this formulation we are presented with 2^{k+1} possible values for x (one value for each of the 2^{k+1} combinations of the binary variables x_0, x_1, \dots, x_k). Since $0 \leq x \leq n$ and we have defined k to be the smallest integer such that $n \leq 2^{k+1} - 1$, it is clear that $2^{k+1} \geq n + 1$. Hence one might suspect that an enumeration algorithm dealing only with the $n + 1$ permissible values of x directly should prove more efficient than an approach allowing $2^{k+1} \geq n + 1$ permissible values of x . In a computational context, an additional practical advantage of an algorithm which avoids binary expansions of variables would be its ability to handle problems with large bounds on the variables more effectively since an increase in the number of variables of a problem also implies an increase in the amount of core storage required for handling the problem. The extension of Geoffrion's basic algorithm [7,10] to allow direct treatment of bounded integer variables is straightforward; in the following chapters we turn

to the development, refinement, and computational examination of such an algorithm. Since the approach taken is closely related to that advanced by Geoffrion for binary programming problems, the assumption is made that the reader is familiar with references [7] and [10] although the presentation here is reasonably self-contained.

In the chapter which follows we will present the structure of a basic algorithm designed for solving bounded variable pure integer programs; bounded variables will be treated directly rather than by conversion to sums of binary variables as described above. In addition we will show that the algorithm is both non-redundant and exhaustive. Chapter III will give the mathematical means for implementing the basic algorithm presented in Chapter II; this implementation will include the development of methods of implicit enumeration capable of accelerating the enumeration process by the simultaneous consideration of several solutions. Chapter IV extends these means of implementation through examination of an imbedded linear program similar to that of Geoffrion in [10]. Also given in Chapter IV are two alternative interpretations of the type of surrogate constraints generated by this imbedded linear program; Chapter IV also presents convenient means for the development of an algorithm which is designed to find near optimal (rather than exactly optimal) solutions. Many of the techniques indicated in Chapters III and IV take advantage of the specific structure of an algorithm which treats bounded integer variables directly. These improvements are simply unavailable to an algorithm which relies on the binary conversion of such variables. These considerations are then implemented in Chapter

V in the form of a computational study which deals with a general evaluation of the efficiency of the algorithm as well as with a specific investigation of two important aspects of the algorithm. The computer timing results presented here indicate that the algorithm developed in Chapters II, III, and IV is certainly among the most efficient, if not the most efficient, means available for solving integer programming problems. Furthermore, these results also indicate that such a direct algorithm as that presented in Chapters II, III, and IV is far superior in virtually all cases considered to algorithms which rely on binary conversion of bounded integer variables. Finally we indicate in Chapter VI the conclusions implied by this thesis and those avenues open to further investigation.

CHAPTER II

BASIC ALGORITHM

Problem Statement

The bounded variable pure integer programming problem may be stated in the following form. Determine x such that:

$$\text{Min } cx \quad (1)$$

$$\text{subject to } b + Ax \geq 0 \quad (2)$$

$$0 \leq x \leq d \quad (3)$$

$$x \text{ is integer-valued} \quad (4)$$

where c , x , and d are n -vectors, b is an m -vector, and A is an $m \times n$ matrix. Note that "a priori" knowledge of an upper bound d_j on the value of each element x_j of the solution vector x is assumed. In addition, complete flexibility on bounding a given variable from below is provided by (3) since if one requires that $l_j \leq x_j \leq d_j$ then the simple change of variables $x'_j = x_j - l_j$ provides the equivalent constraints $0 \leq x'_j \leq d_j - l_j$. In most practical problems such bounds as l_j and d_j are readily available; if not, a lower bound l_j for the variable x_j can be obtained as the solution to the linear program $\text{Min } x_j$ subject to $b + Ax \geq 0$ and

$x \geq 0$ while an upper bound d_j can be obtained by solving $\text{Max } x_j$ subject to $b + Ax \geq 0$ and $x \geq 0$. Further assumptions about (1)-(4), without loss of generality, are that d is an integer-valued vector* and that each element of c is non-negative.** For each x_j , those integer values of x_j such that $0 \leq x_j \leq d_j$ will be called the *admissible values* for x_j . Each x -vector which satisfies constraints (3) and (4) will be called an *admissible solution*; if (2) is also satisfied, the solution will be termed *feasible*; any solution which verifies (1)-(4) will be designated as an *optimal solution*. Inadmissible values, inadmissible solutions, and infeasible solutions are all defined by reference to the obvious negation of the conditions for definition given above.

Since each element of the solution vector is bounded, it is apparent that the number of admissible solutions to (3) and (4) is finite and, in fact, equal to $\prod_{j=1}^n (d_j + 1)$. The particular technique to be developed for solving (1)-(4) will proceed through a systematic enumeration of all admissible solutions to (3) and (4) while retaining candidates for the optimal solution to the problem as they are discovered. Since very few practical problems are amenable to total enumeration of admissible solutions, effective means of implicitly considering portions

* If some d_j is not integral then we can replace d_j by $[d_j]$, where the square brackets denote the greatest integer less than or equal to d_j , to obtain an equivalent statement of (1)-(4) in which d is an integer-valued vector.

** For each j such that $c_j < 0$, the transformation $x_j' = d_j - x_j$ will result in an equivalent formulation of (1)-(4) in which $c_j \geq 0$.

of the admissible solution space must be considered; these techniques for implicit enumeration will be based on those considerations which characterize the feasible solutions to (2)-(4) and the optimal solutions to (1)-(4).

Definitions

The approach for enumerating admissible solutions which follows is very closely related to that given by Geoffrion in [7] and [10]; although the definitions stated in the above study will be reviewed in our context for the sake of completeness, familiarity with these references is essential. One quite natural approach for examining admissible solutions is to attempt to reduce the number of variables involved by considering the solutions which result when certain variables are held constant at admissible values. We refer to such variables which have been assigned one of their admissible values as *fixed* and, similarly, we denote the remaining variables as *free*. This dichotomy of the problem variables implies that we will be dealing with admissible solutions which result from assigning admissible values to those variables which are free. Hence we define a collection of fixed variables as a *partial solution*; any assignment of admissible values to those variables not in a given partial solution will be called a *completion* to that partial solution. Note the implication that there may be many different completions to a given partial solution. A completion to a partial solution resulting in a feasible (infeasible) solution to the problem under consideration will be called a *feasible (infeasible) completion*. It is clear that a given partial solution may possess completions of either variety.

Any enumeration procedure which is computationally efficient must possess the following properties: The procedure must be **exhaustive**,* in that it must consider (whether explicitly or implicitly) all admissible solutions to the problem being considered. Secondly, the process must be **non-redundant**;* i.e., it should never return (whether explicitly or implicitly) to a solution previously enumerated. Consequently, any effective enumeration process must be equipped with means for "remembering" at each stage which solutions have previously been examined. In this regard, we will use the following notation: The ordered index set $S = \{j_1, j_2, \dots\}$ will contain the indices of those variables which are currently fixed in the order in which they became fixed; the partial solution vector $X_S = (x_{j_1}, x_{j_2}, \dots)$ will be used to record the values assumed by those variables in the current partial solution. The correspondence between the elements of X_S and those of S will be that the order of appearance of the elements of S (from left to right) determines the same ordering of the elements of X_S . For example, if $S = \{5, 4, 1, 3\}$ and $X_S = (3, 8, 9, 0)$ then the partial solution currently under consideration is: $x_5 = 3$, $x_4 = 8$, $x_1 = 9$, $x_3 = 0$; the remaining variables are currently free.

To insure the implicit character of the algorithm, means of determining information pertinent to optimality and feasibility from a specific partial solution must be considered. Since we are solving a minimization problem, the following question is of interest: Does a

*More precise definitions of exhaustive and non-redundant are given in a later section of this chapter.

given partial solution admit to a feasible completion with objective value lower than the best feasible solution discovered thus far? This question may be answered in one of three ways:

(1) By demonstrating the non-existence of a completion with the required properties.

(2) By exhibiting the best feasible completion available.

(3) By resorting to additional explicit enumeration of the completions to the partial solution under consideration.

In any event, if, at any given stage of the enumeration procedure either (1) or (2) can be shown, we shall say that the partial solution being considered has been *fathomed*. The fathoming process indicates that all relevant information has been extracted from a partial solution; hence, after fathoming a particular partial solution, the enumeration process should proceed to consider a different partial solution. This process of directing the enumeration to another partial solution after one has been fathomed is called *backtracking*.

As previously indicated, effective means must be provided which will not only guide the enumeration process to the next partial solution for consideration after fathoming but will also maintain records of which partial solutions have been previously investigated. At each stage of the algorithm we attempt to fathom the current partial solution X_S . If this attempt is successful, we backtrack to a new partial solution as follows: we simply change the value of the rightmost element of X_S , requiring only that this new value has not been previously assigned to this variable (i.e., the variable whose index is the rightmost element

of S) since its index last entered S . Furthermore, if the value assumed by a fixed variable under such an operation is its last admissible value, we *underline* the rightmost element of S to indicate that the current partial solution has been fathomed for all values, except the current value, of that variable. If, on the other hand, an attempt to fathom X_S is unsuccessful, we append the subscript of some free variable as the new rightmost element of S and enter one of the admissible values of this variable as the new rightmost element of X_S . This augmentation process is then followed by an attempt to fathom the new partial solution, and so the process continues.

These ideas are illustrated in the following example. Suppose $S = \{1,4,2\}$ and $X_S = (3,7,2)$, indicating that the current partial solution is $x_1 = 3$, $x_4 = 7$ and $x_2 = 2$. First we attempt to fathom X_S ; if successful, we then proceed to a new partial solution by altering the value of x_2 . Suppose then that $0 \leq x_2 \leq 5$, of which values x_2 has assumed 5, 4, 3 and 2 since the index 2 last entered S . Assume success on this attempt to fathom X_S ; we might now choose to set $x_2 = 1$, proceeding to the partial solution with $S = \{1,4,2\}$ and $X_S = (3,7,1)$. Continuing, we now attempt to fathom this new partial solution; again assume this attempt is successful. The next partial solution to be considered must be $S = \{1,4,\underline{2}\}$ and $X_S = (3,7,0)$. Note the significance of the underline; we have now fathomed X_S for each admissible value of x_2 except for its current value of 0. Once more an attempt is made to fathom the new partial solution. If again successful, then all possible completions of X_S will have been enumerated, indicating that the partial

solution given by $S = \{1,4\}$, $X_S = (3,7)$ has been fathomed. Hence, allowing the process to telescope, the next partial solution would be obtained by dropping the index 2 from S and altering x_4 . On the other hand, suppose that the attempt to fathom the partial solution with $S = \{1,4,\underline{2}\}$ and $X_S = (3,7,0)$ is unsuccessful. In this event, some free variable would then be fixed at one of its admissible values, $x_7 = 4$ for instance, and the next partial solution considered would be given by $S = \{1,4,\underline{2},7\}$ and $X_S = (3,7,0,4)$. The process would then continue as above.

Thus we have characterized the simplest form of the enumeration algorithm. Each iteration of the process is begun with an attempt to fathom the current partial solution. If this is successful the enumeration process backtracks to a different partial solution; fathoming is attempted for the new partial solution, etc. If unsuccessful, however, the enumeration proceeds to a new partial solution obtained by fixing some free variable at one of its admissible values; fathoming is attempted for the new partial solution, etc. This process is continued until the partial solution with $S = \phi$ (or, equivalently, a partial solution in which all elements of S are underlined) has been fathomed, indicating that all admissible solutions to the original problem have been examined, and the process terminates.

Statement of the Algorithm

The following steps demonstrate the sequence of operations followed by the enumeration process:

- (1) Choose an initial partial solution. Proceed to step (2).

(2) Attempt to fathom the current partial solution. If unsuccessful, go to step (6). If successful, proceed to step (3).

(3) If the current partial solution admits to a feasible completion with objective value lower than that of the current best feasible solution (*incumbent*), replace the incumbent by this improved solution. Proceed to step (4).

(4) If all elements of S are underlined, stop; the current incumbent is optimal. Otherwise, proceed to step (5).

(5) Alter the value of the fixed variable whose index is the rightmost non-underlined element of S to a new (not yet considered) admissible value and drop all elements of S to the right of this index. Underline the new rightmost element of S if its corresponding variable has now assumed each of its admissible values since its index last entered S . Proceed to step (2).

(6) Fix some free variable to one of its admissible values, update S and X_S accordingly, and go to step (2).

We next focus attention on means by which the enumeration process described above may be proven non-redundant and exhaustive, after which some mathematical tests sufficient for fathoming partial solutions are presented in Chapter III.

Non-Redundance and Exhaustiveness

The two essentialities which are central to the success of an enumerative algorithm are non-redundance and exhaustiveness. It seems reasonable that one should require that the nature of an enumerative solution process be such that neither does it repeat itself nor does it

omit any admissible candidates for solution. These heuristic concepts are specified more precisely when stated in terms of the sequence of partial solutions to (1)-(4) examined by the enumeration algorithm. A sequence of partial solutions which share the following property is called *non-redundant*: No partial solution admits to a completion which duplicates a completion of a partial solution previously fathomed. Further, if an enumerative algorithm ceases only after the implicit or explicit generation of all admissible solutions to a problem, it will be called *exhaustive*.

We shall refer to any one of the $(d_j+1)!$ possible complete orderings of the admissible values of the variable x_j as an *arrangement* of those values. A specific arrangement for x_j will be used to determine the order in which successive admissible values of the fixed variable x_j are considered. It is natural, for reasons relating to computational simplicity, when implementing the algorithm presented in the earlier sections of this chapter, to restrict attention to the following two arrangements for the values of x_j : $0, 1, \dots, d_j$ and $d_j, d_j-1, \dots, 1, 0$ (i.e., those arrangements which define a monotone sequence of the admissible values of x_j). It will become clear that the following proofs for non-redundance and exhaustiveness apply in complete generality with respect to any particular arrangement chosen for the values of a given fixed variable. However, for convenience we shall assume that the arrangement for x_j is given by $0, 1, \dots, d_j$ in the proofs which follow.

The proof of non-redundance presented here is very similar to that given by Geoffrion in [7]; several lemmas are first provided in

order to facilitate the final proof of non-redundance. The following lemma establishes a means of perpetuating the non-redundant property throughout a sequence of partial solutions. For notational convenience we suppress the reference to the associated index sets and denote a sequence of partial solutions as X^1, \dots, X^k where it is understood that the index sets S^1, \dots, S^k which prescribe the variables in these partial solutions are not necessarily identical (and also not necessarily different).

Lemma 1: Given a non-redundant sequence of partial solutions, X^1, \dots, X^k and a partial solution X^{k+1} which contains at least one element different from each fathomed partial solution in the sequence X^1, \dots, X^k , the sequence X^1, \dots, X^{k+1} is also non-redundant.

Proof. Since X^{k+1} contains at least one element different from each fathomed partial solution of the sequence X^1, \dots, X^k , it is clear that no completion of X^{k+1} can duplicate a completion of any fathomed partial solution in that sequence. Hence X^1, \dots, X^{k+1} is non-redundant.

Next suppose j is the rightmost element of S at some stage of the enumeration process. We can insure that the property of non-redundance is maintained throughout a sequence of partial solutions obtained by allowing x_j to assume each of its admissible values in turn, according to its arrangement. To this end, let X^{k-1} be a partial solution with respect to a specific index set S of fixed variables. Further, let $S' : S \cup \{j\}$ for some j such that x_j is free relative to X_S with admissible values $0, 1, \dots, d_j$. In the following lemma we consider the

$d_j + 1$ partial solutions X^k, \dots, X^{k+d_j} (recall notational simplification) with respect to the index set S' , where x_j assumes the values $0, 1, \dots, d_j$ in the partial solutions $X^k, X^{k+1}, \dots, X^{k+d_j}$, respectively.

Lemma 2: Assume the sequence of partial solutions X^1, \dots, X^{k-1} is non-redundant, and that the partial solutions $X^k, X^{k+1}, \dots, X^{k+d_j}$ are formed in the following manner: X^{k+i} is obtained by including x_j at the value i in X^{k-1} . Further assume that the partial solution X^{k-1} has not been fathomed. Then the entire sequence X^1, \dots, X^{k+d_j} is non-redundant.

Proof. Assume, on the contrary, that X^{k_2} has a completion which duplicates a completion of X^{k_1} where X^{k_1} was fathomed, $1 \leq k_1 < k_2 \leq k + d_j$. If $k_2 \leq k-1$ we violate the non-redundancy of X^1, \dots, X^{k-1} since $k_1 < k_2$. Hence $k_2 \geq k$ and we consider the following mutually exclusive and collectively exhaustive cases:

(1) $k_2 \geq k$ and $k_1 > k-1$. In this case it is impossible for a completion of X^{k_2} to duplicate one of X^{k_1} since x_j assumes different values in X^{k_1} and X^{k_2} .

(2) $k_2 \geq k$ and $k_1 = k-1$. This contradicts the assumption that X^{k_1} has been fathomed whereas X^{k-1} has not been fathomed.

(3) $k_2 \geq k$ and $k_1 < k-1$. We have assumed that X^{k_2} has a completion which duplicates a completion of X^{k_1} . But, every completion of X^{k_2} is also a completion of X^{k-1} . Hence X^{k-1} has a completion which duplicates a completion of X^{k_1} , which contradicts the non-redundancy of the sequence X^1, \dots, X^{k-1} . The desired conclusion follows.

The following theorem establishes the property of non-redundance for the enumerative algorithm presented in the previous section of this chapter.

Theorem 1: The sequence of partial solutions examined by the basic enumeration process described previously (under the section "Statement of the Algorithm") is non-redundant.

Proof: Proceeding by induction, it is clear that the initial partial solution alone is non-redundant; hence we assume that X^1, \dots, X^k is a non-redundant sequence of partial solutions and non-redundance will now be shown for the sequence X^1, \dots, X^k, X^{k+1} where X^{k+1} is the partial solution generated by the algorithm after attempting to fathom X^k . In the context of the algorithm described there are only three means by which X^{k+1} can be generated from X^k :

(1) Step (6) of the basic algorithm: After an unsuccessful attempt to fathom X^k , X^{k+1} is formed by augmenting a free variable into the partial solution X^k at value 0 (recall our restriction on the arrangements of the values of $x_j : 0, 1, \dots, d_j$).

(2) Step (5) of the basic algorithm: After successfully fathoming X^k , X^{k+1} is formed by a unit increase in the rightmost element of X^k .

(3) Step (5) of the basic algorithm: After successfully fathoming X^k , X^{k+1} is formed by dropping a string of underlined elements from the right of X^k , and then increasing the new rightmost element of the partial solution thus obtained by 1.

In cases (1) and (2) the desired result follows from Lemma 2. In case (3) suppose that the partial solution X^{k_0} , $k_0 < k$ corresponds to the most recent introduction (at the value of 0) of the variable which was increased by unity to obtain X^{k+1} from X^k . Then it is clear that the

variable whose value is the last element of X^{k+1} now assumes a different value than it assumed in the sequence of partial solutions X^{k_0}, \dots, X^k . By Lemma 1 we conclude that X^{k+1} is non-redundant with respect to the sequence X^{k_0}, \dots, X^k . As in case (2) it is also clear from Lemma 2 that X^{k+1} is non-redundant with respect to the sequence X^1, \dots, X^{k_0} . Consequently the entire sequence X^1, \dots, X^{k+1} is non-redundant, as required.

It remains to verify that the sequence of partial solutions examined by the enumeration process is exhaustive.

Theorem 2: The sequence of partial solutions examined by the basic enumeration process described previously (under the section "Statement of the Algorithm") is exhaustive.

Proof: The proof relies heavily on the manner in which the algorithm accomplishes its "history remembering." Termination of the algorithm occurs only if a partial solution with an index set S of all underlined elements is fathomed. After Geoffrion ([7], page 190), it is clear that a sufficient condition for exhaustiveness is that the underlining of a certain element of S connotes the following: that we have successfully fathomed (implicitly or explicitly) all possible completions up to and including *all* other admissible values for the variable indicated by the underlined index. This property is evident from the description of the algorithm, since we underline an element of S if and only if its corresponding variable is currently assumed fixed at the final admissible value in its arrangement. Hence, when a partial solution consisting of all underlined elements is fathomed, the fathoming

process telescopes, indicating the complete examination of all admissible solutions.

Hence the enumeration algorithm under investigation does indeed possess the requisites for computability: non-redundance and exhaustiveness. Note that for the most part the description of the algorithm thus far has dealt in generalities, thus permitting a maximum of flexibility in its implementation. Simultaneously, however, we have precluded the possibility of making quantitative statements concerning the efficiency of the algorithm. This characteristic will depend on the ability of the algorithm to function implicitly rather than explicitly--a feature which can only be evaluated within the framework of a specific implementation of the algorithm. Thus we next consider the development of such mathematical tests as will be used in implementing the algorithm.

CHAPTER III

MATHEMATICAL IMPLEMENTATION

The implicit enumeration algorithm described in Chapter II has several decision steps for which mathematical rules must be provided. Three specific areas of the algorithm will be considered in detail:

- (1) means for choosing a partial solution for initializing computation,
- (2) tests for fathoming a partial solution, and
- (3) criteria for augmentation which specify not only which variable should be fixed, but also at which value it should be fixed.

We will also investigate how certain fathoming tests can be easily extended in order to fathom several partial solutions simultaneously.

Initial Solution

The inductive nature of the proof for non-redundance assures us of the preservation of this property, regardless of the initial partial solution. However, the dependence of the exhaustive property on the method of remembering which solutions have been considered restricts the initial set S to be of the following form: Any subset of the problem variables may be fixed, but each fixed variable must be initialized at the first value of its respective arrangement of admissible values. This is intuitively clear; it is closely related to the connotation

given to an underlined element of S . Within these considerations, some permissible starting conventions would be:

(1) Let S_0 denote the initial index set of fixed variables.

Then $S_0 = \phi$ represents a partial solution with which the algorithm can be initialized.

(2) Computation may begin with every variable in the initial partial solution. To be consistent with the convention stated in Chapter II of considering only the arrangements $0, 1, \dots, d_j$ and $d_j, d_j - 1, \dots, 1, 0$ for the variable x_j , any such initial partial solution (i.e., with $S_0 = \{1, 2, \dots, n\}$) must have that element of X_{S_0} corresponding to x_j fixed at either d_j or 0 . Two possible X_{S_0} vectors corresponding to $S_0 = \{1, 2, \dots, n\}$ are $X_{S_0} = (0, 0, \dots, 0)$ and $X_{S_0} = (d_1, d_2, \dots, d_n)$.

(3) S_0 could be initialized to contain only those indices corresponding to variables which are integer-valued in the continuous correspondent of the integer program under consideration. In this case a suitable arrangement should be chosen for each x_j , $j \in S_0$ so that x_j would be initially fixed at some value near that taken by x_j in the optimal solution to the continuous problem.

(4) If a feasible solution to the initial problem is known, it can be used as a source for several different initial partial solutions. For instance, one might have $S_0 = \{j | x_j > 0 \text{ in the given feasible solution}\}$ with X_{S_0} chosen as described in (3). An alternative would be to have $S_0 = \{j | x_j = 0 \text{ in the given feasible solution}\}$ with $X_{S_0} = (0, \dots, 0)$.

Heuristic schemes for initialization which reflect the structure of certain classes of problems could also be used. In fact,

initialization can be a very important factor in speed of solution; e.g., a good initial starting point may lead quickly to a good feasible solution, thus allowing optimality considerations (see the following section on fathoming) to exclude large numbers of solutions from explicit consideration.

Simple Tests for Fathoming

To a large extent, the strength of any implicit enumeration algorithm lies in its fathoming process, since it is by fathoming that solutions are evaluated implicitly rather than explicitly. To fathom a given partial solution it is sufficient either to demonstrate a best feasible completion to that partial solution or to determine that there are no feasible completions to that partial solution or finally to show that no feasible completion to the partial solution improves on the objective value of the incumbent. We shall consider fathoming tests based on each of these three considerations.

Fathoming by Optimality

Recalling that the problem formulation (refer to Chapter II) requires that $c_j \geq 0$, $j = 1, \dots, n$, a trivial test for fathoming by optimality is available. Let \bar{z} be the current incumbent objective value and $z^S = \sum_{j \in S} c_j x_j$ be the objective value of the current partial solution. Then, clearly, if $\bar{z} \leq z^S$ the partial solution X_S has been fathomed by optimality. As another possibility, note that at any specific iteration of the algorithm, the problem of determining a best feasible completion to the current partial solution is again an integer programming problem. In certain instances the optimal solution to this

subproblem is easily obtained. The following simple test provides sufficient means for establishing a best feasible completion to the partial solution being examined: Form the completion corresponding to an assignment of 0 value to all free variables. If this particular completion is feasible, it is certainly a *best feasible completion* since to assign a positive value to any free variable would increase the value of the objective function. This particular completion is also a best possible completion from the current partial solution in the sense that it determines a lower bound on the values of the objective function which can be obtained by any completion of the current partial solution. We are thus provided with sufficient means for fathoming a partial solution via determination of its best feasible completion; if this test proves successful, the new incumbent solution is recorded and the enumeration process backtracks to a new partial solution (as discussed in Chapter II). If this test is unsuccessful, however, we next attempt to fathom the current partial solution by demonstrating either that it can possess no feasible completions or that no feasible completion to the current partial solution has objective value improving on that of the incumbent.

Fathoming by Infeasibility

Further means of fathoming a partial solution can be developed based on feasibility considerations alone. Note that at any specific iteration, the problem under consideration relative to the current partial solution X_S may be expressed as:

$$\text{Min } z^S + \sum_{j \notin S} c_j x_j \quad (5)$$

$$\text{subject to } b_i^S + \sum_{j \notin S} a_{ij} x_j \geq 0, \quad i=1, \dots, m$$

$$d_j \geq x_j \geq 0 \quad \text{and } x_j \text{ integer, } j \notin S$$

where we define $z^S = \sum_{j \in S} c_j x_j$ and $b_i^S = b_i + \sum_{j \in S} a_{ij} x_j$.

A specific constraint of (5) will be termed *infeasible* if no assignment of admissible values to the free variables exists which satisfies that constraint. More precisely, a necessary and sufficient condition that the i th constraint of (5) be infeasible is that $b_i^S + \text{Max}_{x_j} \sum_{j \notin S} a_{ij} x_j < 0$. Note that the maximization in this expression is accomplished by assigning a value of 0 to those x_j for which $a_{ij} < 0$ and a value of d_j to those x_j for which $a_{ij} \geq 0$. Thus if any constraint of (5) satisfies the inequality $b_i^S + \sum_{j \notin S} \text{Max}(0, a_{ij} d_j) < 0$, then the partial solution given by X_S has no feasible completion and hence has been fathomed. We next consider a means for fathoming a partial solution by showing that any feasible completion to that partial solution cannot improve on the objective value of the incumbent solution.

Fathoming by Conditions Related to Both Optimality and Feasibility

Consider again the problem (5) which is encountered at any stage of the enumerative procedure. If we denote the incumbent objective value as \bar{z} , then it is clear that no x_j , $j \notin S$ for which $z^S + c_j \geq \bar{z}$ can be non-zero in any completion to X_S which improves (strictly) the objective value of the incumbent solution. Similarly, in the event that augmentation is necessary we would like to assign positive value only to those free variables which can lead to a reduction in the infeasibility

of (5) (i.e., to those $x_j, j \notin S$ for which $a_{ij} > 0$ for some i such that $b_i^S < 0$). In this context we define the set T^S as follows:

$$T^S = \{j \mid j \notin S; z^S + c_j < \bar{z} \text{ and } a_{ij} > 0 \\ \text{for some } i \text{ such that } b_i^S < 0\}.$$

T^S thus contains indices of those free variables which are candidates for becoming fixed; i.e., augmentation of one or several of those variables indicated by the elements of T^S into the set S may lead to an improved feasible solution (i.e., a new incumbent). Note in particular that if $T^S = \phi$ then the partial solution given by X_S has been fathomed by demonstration that no feasible completion of X_S results in an improved incumbent objective value.

These various simple tests for fathoming are easily implemented computationally to provide several sufficient means for fathoming a given partial solution. Note, in particular, that the test for constraint infeasibility is somewhat limited in that in its present form it is only applicable to individual constraints. Many powerful means of fathoming, based on considering non-negative linear combinations of the constraints of (5) will be developed in Chapter IV. We now turn to consideration of the augmentation step of the algorithm.

Augmentation

The augmentation process is the explicit enumeration portion of the algorithm. When attempts to enumerate implicitly (i.e., via fathoming) fail, we resort to explicit enumeration until fathoming success

occurs. Given that there is considerable latitude in determining the specifics of augmentation, it seems reasonable to consider an augmentation process which might increase the efficiency of the algorithm (see results in Chapter V) by prescribing the augmentation of variables which lead to partial solutions which are amenable to the fathoming tests used. Consequently several possible augmentation disciplines will be described; each is based on heuristic considerations of the enumerative procedure.

(1) In an attempt to proceed quickly to a feasible solution, Balas' augmentation rule is available: Fix at its upper bound that variable x_j which maximizes the expression $\sum_{i=1}^m \min(0, b_i^S + a_{ij} d_j)$ over all $j \in T^S$. In a sense the noted expression gives a measure of the total system infeasibility remaining after x_j is fixed at value d_j , so that maximization of this non-positive expression minimizes system infeasibility for the next iteration.

(2) In direct contrast to (1), we might fix at its upper bound that variable which maximizes total system infeasibility. This approach might prove useful (during for instance, the phase in which the algorithm is verifying optimality of an incumbent solution) by making the partial solution obtained at the next iteration more susceptible to fathoming tests based on infeasibility.

(3) Another approach would be to fix that variable (at either its upper or lower bound) which minimizes d_j for $j \notin S$ since this would minimize the number of additional branches added into the explicit solution tree on the subsequent iteration. The computational comparison

of this rule and (1) above given in Chapter V seems to indicate that this is a worthwhile heuristic.

(4) One might alternatively consider a minimum cost augmentation discipline, based on the heuristic that an algorithm using this rule would tend to explore more quickly those solutions with lower objective values and hence obtain a good feasible solution more quickly. Such a rule would augment at its lower bound that variable which minimized c_j for all $j \in S$.

Fathoming Several Partial Solutions Simultaneously

The necessary means for a simple implementation of the enumerative algorithm are now at hand; tests for fathoming partial solutions and rules for augmentation can be chosen from those presented above with considerable flexibility. Before considering more powerful methods of fathoming based on infeasibility of systems of constraints, we first consider some straightforward means of extending the above tests in order to fathom several partial solutions simultaneously, taking advantage of an appropriate arrangement of the admissible values of the variable whose index is rightmost in S . We note that such tests are, of course, applicable only to an algorithm which deals with bounded integer variables directly rather than in piecemeal fashion as in the binary conversion of a problem with bounded integer variables. Success in fathoming is always followed by backtracking, and consequently one might suspect that fathoming several solutions simultaneously should tend to accelerate the enumeration process.

Optimality Tests

Fathoming by optimality occurs when the best possible solution is feasible (as described earlier). Suppose that a partial solution X_S is fathomed by optimality, and let j be the rightmost index of S . In this case, if the fixed variable x_j is being sequenced through increasing values (arrangement $0, 1, \dots, d_j$), we may immediately eliminate j from S and backtrack accordingly, since $c_j \geq 0$ implies that an increase in x_j will lead to partial solutions with a non-optimal objective value. Alternatively, suppose this same situation is encountered and x_j is currently being decreased (arrangement $d_j, d_j-1, \dots, 1, 0$). Then considerations of feasibility imply that x_j may be reduced to

$$x_j' = \text{Max} \begin{cases} 0, & \text{for } i \text{ such that } a_{ij} \leq 0 \\ x_j - \left[\frac{b_i^S}{a_{ij}} \right] - 1, & \text{for } i \text{ such that } a_{ij} > 0 \end{cases}$$

before violating feasibility while improving the incumbent solution.

Infeasibility Tests

Another means of fathoming developed for single partial solutions was by constraint infeasibility. The extension in this case to fathoming multiple partial solutions is quite natural. Suppose that the i th constraint is infeasible relative to the partial solution X_S ; i.e., $\beta = b_i^S + \sum_{j \in S} \text{Max}(0, a_{ij} d_j) < 0$. Let j_0 be the current rightmost index in S . If x_{j_0} is being increased (arrangement $0, 1, \dots, d_j$) and $a_{ij_0} \leq 0$ then all admissible values of x_{j_0} have been fathomed, since to increase x_{j_0}

under such conditions cannot diminish the infeasibility of the i th constraint (i.e., cannot increase β). A similar result is obtained if x_{j_0} is being decreased (arrangement $d_j, d_{j-1}, \dots, 1, 0$) and $a_{ij_0} \geq 0$. On the other hand, suppose that x_{j_0} is increasing and $a_{ij_0} > 0$. In this case we can only increase x_{j_0} by $-\left[\frac{\beta}{a_{ij_0}}\right]$, at which point the i th equation becomes feasible. Again, a similar result holds if x_{j_0} is being decreased and $a_{ij_0} < 0$. It is useful to note the applicability of these results to surrogate constraints, which are designed to be specifically susceptible to such tests (see Chapter IV).

Tests for Relative Bound Redefinition

Another form of fathoming several partial solutions simultaneously is by recomputing the bounds on the free variables relative to a given partial solution. At any stage of the enumeration process the question of interest in attempting to redefine bounds on the free variables is: What are the upper and lower bounds on x_j , $j \notin S$ for all feasible completions to the current partial solution which improve on the incumbent value of the objective? It is clear that sufficient tests for answering this question will provide means for simultaneously fathoming several partial solutions since they will limit the range of a variable to be augmented.

Certainly no variable should be fixed at a value which implies a non-optimality (multiple fathoming via optimality); i.e., no variable should be fixed at one of its admissible values such that $z^S + c_j x_j \geq \bar{z}$. Similarly, no variable should be fixed at a value which forces one of the constraints of (5) to be infeasible (multiple fathoming via

infeasibility); i.e., no x_j should be fixed at a value such that $b_i^s + a_{ij}x_j + \sum_{k \neq j, k \in S} \text{Max}(0, a_{ik}d_k) < 0$. These considerations lead to the possibility of defining a new (and possibly strengthened) upper bound on each free variable; such bounds will be valid in any improving feasible solution to the current partial solution. This upper bound redefinition may be accomplished by computing a temporary upper bound d_j' for $x_j, j \notin S$ by considering the following constraints:

$$x_j \geq 0 \text{ and integer} \quad (6)$$

$$x_j \leq d_j$$

$$x_j < \frac{\bar{z} - z^s}{c_j}$$

$$x_j \leq \frac{b_i^s + \sum_{k \neq j, k \in S} \text{Max}(0, a_{ik}d_k)}{-a_{ij}}, \text{ for all } i \text{ such that } a_{ij} < 0.$$

In addition, it is easily seen that a new lower bound redefinition may be accomplished by computing a temporary lower bound ℓ_j' for $x_j, j \notin S$ by considering the following constraints:

$$x_j \geq 0 \text{ and integer} \quad (7)$$

$$x_j \leq d_j$$

$$x_j \geq \frac{b_i^s + \sum_{k \neq j, k \in S} \text{Max}(0, a_{ik}d_k)}{-a_{ij}}, \text{ for all } i \text{ such that } a_{ij} > 0.$$

For notational convenience it is assumed that d'_j and l'_j represent the upper and lower bounds, respectively, for x_j at any given stage of the enumeration. The use of such bounds plays an important part in deciding which free variable should be augmented when attempts to fathom a partial solution fail.

The incorporation of such temporary bounds into the enumerative algorithm is easily effected by computing such bounds via (6) and (7) and resetting the temporary upper and lower bounds for each free variable $x_j, j \in S$ to d'_j and l'_j , respectively, after fathoming. The inclusion of such temporary bounds on free variables is straightforward in each of the tests described earlier; for example, the necessary and sufficient condition for the infeasibility of the i th constraint of (5)

$$\text{becomes } b_i^S + \sum_{j \in S} \text{Max}(a_{ij} l'_j, a_{ij} d'_j) < 0.$$

Inclusion of these extensions to permit fathoming of multiple partial solutions should certainly enhance the fathoming capabilities (and hence the implicit nature) of the algorithm. To provide additional means for fathoming single partial solutions by determining the feasibility of (5), we next consider the possibility of producing infeasible constraints via non-negative linear combinations of the constraints of (5). This approach will also be shown to lead conveniently to further techniques of fathoming multiple partial solutions by relative bound redefinition for free variables.

CHAPTER IV

SURROGATE CONSTRAINTS

Introduction

As previously stated (see Chapter III), a necessary and sufficient condition for the infeasibility of a given constraint

$b_i^s + \sum_{j \in S} a_{ij} x_j \geq 0$ is that $b_i^s + \text{Max}_{x_j} \sum_{j \in S} a_{ij} x_j < 0$. Such considerations have yielded tests which are useful in fathoming partial solutions.

However, previous developments concerning constraint infeasibility are meaningful only as applied to constraint inequalities individually. It is quite natural to attempt to extend these tests in order that several inequalities may be treated simultaneously, since a system of inequalities may be infeasible although each inequality is feasible. Consider

the following example:
$$\begin{cases} -4 - x_1 + 2x_2 \geq 0 \\ -4 + 2x_1 - x_2 \geq 0 \end{cases} \quad \text{where } 0 \leq x_1, x_2 \leq 3, \text{ and}$$

the x_j are integral. Clearly, the points (0,2), (0,3), (1,3), and (2,3) satisfy the first inequality and the points (2,0), (3,0), (3,1), and (3,2) satisfy the second inequality, while there is no point (x_1, x_2) such that $0 \leq x_1, x_2 \leq 3$ which satisfies both inequalities simultaneously. Since any point which satisfies a system of inequalities must necessarily satisfy any non-negative linear combination of those inequalities, it is reasonable to consider the ability of such constraints to reflect the infeasibility of an entire system. For instance, in the example given above, the inequality $-8 + x_1 + x_2 \geq 0$,

generated as a linear combination of the two inequalities with unit weight on each, is clearly infeasible for $0 \leq x_1, x_2 \leq 3$ and hence reflects the infeasibility of the original system. Specifically then, we wish to consider the conditions under which an infeasible constraint can be generated by non-negative linear combinations of a system of constraints; as such constraints are generated directly from the original system of constraints, they will be called surrogate constraints [2,8,10,11,12,14].

Strongest Surrogate Constraints

At any given stage of the enumeration process, the constraint set which must be satisfied by any solution improving on the current best solution is (assuming current partial solution X_s):

$$b_i^s + \sum_{j \notin S} a_{ij} x_j \geq 0, \quad i=1, \dots, m \quad (8)$$

$$z^s + \sum_{j \notin S} c_j x_j < \bar{z} \quad (9)$$

$$0 \leq x_j \leq d_j \quad \text{and} \quad x_j \text{ integral, } j \notin S \quad (10)$$

where b_i^s , z^s , and \bar{z} are as defined in Chapter III. After Geoffrion [10], we will define the strength of a given surrogate constraint with respect to its proximity to infeasibility.

Definition: The surrogate constraint $\mu^1(b+Ax) + (\bar{z}-cx) > 0$ is said to be *stronger relative to X_s* than the surrogate constraint $\mu^2(b+Ax) + (\bar{z}-cx) > 0$ if

$$\text{Max}_{x_j (j \notin S)} \{ \mu^1 (b + Ax) + (\bar{z} - cx) \} < \text{Max}_{x_j (j \notin S)} \{ \mu^2 (b + Ax) + (\bar{z} - cx) \}.$$

Since any solution satisfying (8)-(10) must also satisfy any non-negative linear combination of (8) and (9), demonstration of $\mu_i \geq 0$, $i=1, \dots, m$ for which

$$\left\{ \sum_{i=1}^m \mu_i (b_i^S + \sum_{j \notin S} a_{ij} x_j) + \bar{z} - z^S - \sum_{j \notin S} c_j x_j \right\} \leq 0 \text{ for every } x_j \in \Gamma_j, j \notin S$$

where $\Gamma_j = \{x_j | x_j \text{ integer}, 0 \leq x_j \leq d_j\}$ is sufficient to show the infeasibility of (8)-(10). The above purpose is served if we show that

$$\text{Max}_{x_j \in \Gamma_j} \left\{ \sum_{i=1}^m \mu_i (b_i^S + \sum_{j \notin S} a_{ij} x_j) + \bar{z} - z^S - \sum_{j \notin S} c_j x_j \right\} \leq 0.$$

In other words, such an inequality proves the non-existence of a completion to the current partial solution which simultaneously verifies the original problem constraints and reduces (strictly) the current best objective value.

Applying the derivation given in [10], the problem of determining a strongest surrogate constraint relative to the partial solution X_S may be written in the following form:

$$\text{Min}_{\mu} \text{Max}_{x_j \in \Gamma_j} \left\{ \sum_{i=1}^m \mu_i (b_i^S + \sum_{j \notin S} a_{ij} x_j) + (\bar{z} - z^S - \sum_{j \notin S} c_j x_j) \right\} \quad (11)$$

subject to $\mu_i \geq 0$, $i=1, \dots, m$, where $\Gamma_j = \{x_j | x_j \text{ integer}, 0 \leq x_j \leq d_j\}$, as defined earlier.

Rearranging terms we get:

$$\bar{z} - z^S + \min_{\mu} \left\{ \sum_{i=1}^m \mu_i b_i^S + \max_{x_j \in \Gamma_j} \left\{ \sum_{j \notin S} \left(\sum_{i=1}^m \mu_i a_{ij} - c_j \right) x_j \right\} \right\} \quad (12)$$

where $\mu_i \geq 0$, $i=1, \dots, m$.

For fixed μ_i , $i=1, \dots, m$ we may treat the inner maximization in (12) as an integer program:

$$\max_{x_j \in \Gamma_j} \sum_{j \notin S} \left(\sum_{i=1}^m \mu_i a_{ij} - c_j \right) x_j \quad (13)$$

The optimal value of (13) is independent of the integrality restrictions on the x_j (the bounds d_j are integral) so that (13) becomes a linear program:

$$\max_{x_j} \sum_{j \notin S} \left(\sum_{i=1}^m \mu_i a_{ij} - c_j \right) x_j \quad (14)$$

subject to $0 \leq x_j \leq d_j$, $j \notin S$.

We may rewrite (14) using the duality theorem of Linear Programming to obtain:

$$\begin{aligned} & \min_{\omega_j} \sum_{j \notin S} \omega_j d_j & (15) \\ & \text{subject to } \omega_j \geq \sum_{i=1}^m \mu_i a_{ij} - c_j, \quad j \notin S \\ & \omega_j \geq 0, \quad j \notin S. \end{aligned}$$

Finally, substituting (15) into (12) we obtain the following linear program which is equivalent to the original problem (11) of determining a strongest surrogate constraint relative to the partial solution X_S :

$$\text{Min}_{\mu, \omega} v = \bar{z} - z^S + \sum_{i=1}^m \mu_i b_i^S + \sum_{j \notin S} \omega_j d_j \quad (\text{LP}_{X_S})$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i=1}^m \mu_i a_{ij} - \omega_j \leq c_j, \quad j \notin S \\ & \mu_i \geq 0, \quad i=1, \dots, m \\ & \omega_j \geq 0, \quad j \notin S. \end{aligned}$$

Means for fathoming a given partial solution are immediately available from the solution to (LP_{X_S}) . Suppose $v^* = \text{Min}_{\mu, \omega} v \leq 0$. This can be true if and only if there exist $\mu_i \geq 0, i=1, \dots, m$ such that

$$\text{Max}_{x_j \in \Gamma_j} \{ \bar{z} - z^S + \sum_{i=1}^m \mu_i b_i^S + \sum_{j \notin S} (\sum_{i=1}^m \mu_i a_{ij} - c_j) x_j \} \leq 0 \quad \text{for } \Gamma_j$$

defined as above. Hence for every $x_j \in \Gamma_j, j \notin S$, the expression within brackets in the above inequality is non-positive and the constraint

$$\bar{z} - z^S + \sum_{i=1}^m \mu_i b_i^S + \sum_{j \notin S} (\sum_{i=1}^m \mu_i a_{ij} - c_j) x_j > 0$$

which is only a rearrangement of the surrogate constraint

$$\sum_{i=1}^m \mu_i (b_i^S + \sum_{j \notin S} a_{ij} x_j) + (\bar{z} - z^S - \sum_{j \notin S} c_j x_j) > 0$$

is infeasible for every $x_j \in \Gamma_j$, $j \notin S$.

The role played by (LP_{X_S}) in the solution process is the following: At any iteration one may attempt to fathom the current partial solution by performing simplex iterations on (LP_{X_S}) until $v \leq 0$ in which case the current partial solution X_S has been fathomed, or until an optimal solution $v^* > 0$ to (LP_{X_S}) is obtained implying the non-existence of an infeasible surrogate constraint. Since such surrogate constraints are highly susceptible to the fathoming tests for constraint infeasibility developed in Chapter III, we note that augmenting such constraints to the original problem may prove effective with respect to these tests. Thus a surrogate constraint need not be discarded after its generation in solving (LP_{X_S}) --the k (where k is a specified number) most recently generated surrogate constraints may be maintained throughout the enumeration process. This heuristic appears to have proven useful in the computational results of [10].

Sequential Solutions to the Imbedded Linear Program

Implementation of fathoming via (LP_{X_S}) may be easily effected by using the revised simplex method for solving (LP_{X_S}) . The revised simplex method also provides a convenient framework in which one can take advantage of the fact that (LP_{X_S}) need not be solved from its initial tableau each time it is used in the solution process. Specifically, suppose (LP_{X_S}) is utilized on some iteration and at some future iteration with partial solution X'_S , we wish to attempt to fathom using $(LP_{X'_S})$. From the information available from (LP_{X_S}) we can construct an advanced tableau for $(LP_{X'_S})$ in one of two ways:

(1) If the same variables are fixed in X_S and $X_{S'}$, (i.e., $S = S'$) then only the values of these variables can be different (i.e., $X_S \neq X_{S'}$). Hence $(LP_{X_{S'}})$ is identical to (LP_{X_S}) except possibly for the elements $z^{S'}$, $b_i^{S'}$, $i=1, \dots, m$ and d_j , $j \notin S'$. In other words, generation of an advanced tableau for $(LP_{X_{S'}})$ may be effected merely by an appropriate change of the objective function for (LP_{X_S}) .

(2) If $S' \neq S$ then it is necessary to alter the (LP_{X_S}) tableau by column and row deletions and/or additions depending on how S' differs from S . Suppose first that $j \in S'$ and $j \notin S$ (i.e., x_j is currently fixed but was free at the time (LP_{X_S}) was used). Then both the constraint $\sum_{i=1}^m \mu_i a_{ij} - \omega_j \leq c_j$ and its corresponding slack variable s_j as well as the variable ω_j must be deleted from the (LP_{X_S}) tableau. Means for such deletions are most easily explained by considering whether s_j and ω_j are basic or non-basic variables in the final tableau for (LP_{X_S}) considered:

Case (i): Suppose s_j and ω_j are both basic in the final tableau of (LP_{X_S}) . This cannot be since the columns corresponding to s_j and ω_j in (LP_{X_S}) are linearly dependent.

Case (ii): Suppose either s_j or ω_j , but not both, to be basic in the final tableau of (LP_{X_S}) . Then the columns corresponding to s_j and ω_j and the constraint in which s_j or ω_j is basic are simply deleted after which techniques described in (1) above are used to obtain an advanced tableau for $(LP_{X_{S'}})$.

Case (iii): Suppose neither s_j nor ω_j is basic in the final tableau of (LP_{X_S}) . In this case, from the structure of (LP_{X_S}) it is evident that those columns in the final tableau of (LP_{X_S}) which

correspond to s_j and ω_j must be non-zero. Furthermore, the structure of (LP_{X_S}) yields the convenient property that the ω_j -column must be the negative of the s_j -column in each tableau of (LP_{X_S}) . Hence one is assured that a pivot can be made in the final tableau of (LP_{X_S}) which forces either s_j or ω_j to become basic while maintaining primal feasibility. After such a pivot we are once again in Case (ii) considered above.

On the other hand, suppose that $j \notin S'$ but $j \in S$ (i.e., x_j is currently free but was fixed during the iteration on which (LP_{X_S}) was being used). In this case we must alter the final tableau of (LP_{X_S}) by adding the constraint $\sum_{i=1}^m \mu_i a_{ij} - \omega_j \leq c_j$ and the s_j - and ω_j -columns in order to obtain an advanced tableau for $(LP_{X_{S'}})$. This is easily accomplished using standard methods; in the event that the solution derived in this manner for $(LP_{X_{S'}})$ is not primal feasible, the fact that the ω_j -activity is the negative of the s_j -activity provides a convenient means for obtaining primal feasibility-- s_j will be basic, so merely pivot to let ω_j replace s_j in the set of basic variables. This feature eliminates the need for a dual simplex algorithm in obtaining $(LP_{X_{S'}})$ from (LP_{X_S}) . After such an addition as that described above we once again use the means given in (1) to obtain the appropriate advanced tableau for $(LP_{X_{S'}})$.

These considerations can be used to considerable advantage in making (LP_{X_S}) a computationally efficient means of attempting to fathom the partial solution X_S . In addition, the ease of transforming a basic feasible solution for (LP_{X_S}) into a basic feasible solution for $(LP_{X_{S'}})$

facilitates the implementation of such schemes as using the imbedded linear program as a fathoming tool only on every k th (where k is a specified constant) iteration of the basic enumeration process. This idea leads to apparent computational success in some instances as described by Geoffrion in [10].

Fathoming by Optimality Considerations

Still further fathoming information is available from (LP_{X_S}) .

Writing (LP_{X_S}) as a maximization problem we have:

$$\text{Max } (-v) = z^S - \bar{z} - \sum_{i=1}^m \mu_i b_i^S - \sum_{j \notin S} \omega_j d_j \quad (16)$$

$$\begin{aligned} \text{subject to } \quad & \sum_{i=1}^m \mu_i a_{ij} - \omega_j \leq c_j, \quad j \notin S \\ & \mu_i \geq 0, \quad i=1, \dots, m \\ & \omega_j \geq 0, \quad j \notin S. \end{aligned}$$

By duality considerations we obtain the following equivalent of (16):

$$\text{Min } \sum_{j \notin S} c_j x_j + z^S - \bar{z} \quad (17)$$

$$\begin{aligned} \text{subject to } \quad & \sum_{j \notin S} a_{ij} x_j \geq -b_i^S, \quad i=1, \dots, m \\ & -x_j \geq -d_j, \quad j \notin S \\ & x_j \geq 0, \quad j \notin S. \end{aligned}$$

Rewriting (17) we see that (LP_{X_s}) is merely the dual (with the constant $z^s - \bar{z}$ added to the objective function) of the linear program obtained at each stage of the solution process by omitting the integrality restrictions on x_j , $j \notin S$. Namely,

$$\text{Min } \sum_{j \notin S} c_j x_j \quad (P_{X_s})$$

$$\text{subject to } b_i^s + \sum_{j \notin S} a_{ij} x_j \geq 0, \quad i=1, \dots, m$$

$$0 \leq x_j \leq d_j, \quad j \notin S.$$

Thus the dual variables of (LP_{X_s}) may be used to provide another sufficient condition for fathoming X_s : If an optimal solution to (LP_{X_s}) has integral dual variables, then these variables provide an optimal solution to (P_{X_s}) with the integrality restrictions on the x_j , $j \notin S$ included, and hence provide a best feasible completion to X_s , thus fathoming the partial solution X_s . Incorporation of these considerations into the enumeration procedure is easily accomplished (see Figures 1 and 2 in Chapter V).

Relative Bound Redefinition

In addition to the useful features already described, an optimal tableau for (LP_{X_s}) may provide information which allows one to strengthen the upper and lower bounds on the free variables. As discussed earlier in Chapter III, this capability leads to fathoming several partial

solutions simultaneously. Such means for reducing the range of admissible values for free variables is of considerable importance, since this reduces (enormously in some instances--refer to Chapter V) the number of partial solutions which must be explicitly examined in the event that augmentation is necessary. Suppose then that an optimal solution for (LP_{X_S}) is at hand, and let x_{j_0} be free (i.e., $j_0 \notin S$). A lower bound on admissible values of x_{j_0} in any feasible completion to X_S is provided by the smallest integer greater than or equal to the solution to the following linear program:

$$\begin{aligned} & \text{Min } x_{j_0} && (18) \\ & \text{subject to } b_i^S + \sum_{j \notin S} a_{ij} x_j \geq 0, \quad i=1, \dots, m \\ & && 0 \leq x_j \leq d_j, \quad j \notin S. \end{aligned}$$

Similarly, an upper bound on admissible values of x_{j_0} in any feasible completion to X_S is provided by the largest integer less than or equal to the solution to:

$$\begin{aligned} & \text{Min } (-x_{j_0}) && (19) \\ & \text{subject to } b_i^S + \sum_{j \notin S} a_{ij} x_j \geq 0, \quad i=1, \dots, m \\ & && 0 \leq x_j \leq d_j, \quad j \notin S. \end{aligned}$$

Solutions to the linear programs (18) and (19) may be obtained by a simple alteration of the objective function of (P_{X_s}) , and hence by an alteration of the right-hand side vector in (LP_{X_s}) . Using the terminology of the revised simplex method, let B^{-1} be the current basis inverse matrix for (LP_{X_s}) , and let π_{j_0} be the dual variable associated with the column of B^{-1} which corresponds to the index j_0 . The structure of (LP_{X_s}) indicates that replacement of the original right-hand side vector for (LP_{X_s}) by the objective function of problem (18) (of problem (19)) is accomplished by replacing the current right-hand side vector for (LP_{X_s}) by a positive (negative) copy of the column corresponding to j_0 in B^{-1} . Consequently this replacement provides an immediately optimal solution for (18) (for (19)) if the elements of the column corresponding to j_0 in B^{-1} are all non-negative (non-positive). Optimality of (LP_{X_s}) implies that $0 \leq \pi_{j_0} \leq d_{j_0}$ so that (LP_{X_s}) provides convenient means for redefining* (and possibly improving) the bounds for the free variable x_{j_0} : If the column corresponding to x_{j_0} in B^{-1} is non-negative (non-positive) then $[\pi_{j_0}]$ provides a new lower (upper) bound on the admissible values of x_{j_0} in any feasible completion to X_s .

Near Optimal Solutions

The dual relation shared by (LP_{X_s}) and (P_{X_s}) also provides convenient means for introducing fathoming techniques based on the maximum possible objective improvement resulting from any completion to X_s .

* Note that fathoming can occur here. Specifically, for any free variable if bound redefinition results in a relative upper bound smaller than the relative lower bound then certainly X_s has been fathomed.

Denote by (I_{X_S}) the integer programming problem relative to X_S at each stage of the enumeration procedure; i.e., (I_{X_S}) is merely (P_{X_S}) with integrality restrictions on the x_j , $j \in S$. Then any optimal solution v^* to (LP_{X_S}) provides a lower bound, $z_L = \bar{z} - z^S - v^*$ on the objective value attainable by any feasible solution to (I_{X_S}) . This lower bound is useful in constructing an algorithm which insures that the best integral solution found to a given problem will be within $p\%$, where p is specified and $0 \leq p \leq 100$, of the true optimal solution to that problem. Such an algorithm is easily obtained within the framework of our current algorithm if, at any stage of the enumeration one allows a partial solution to be fathomed by demonstration that every feasible completion of that partial solution admits to a percentage objective improvement of less than $p\%$. This condition can be tested using z_L quite easily: If (LP_{X_S}) is optimal and $\frac{(\bar{z} - z_L - z^S)}{\bar{z}} (100) < p$ then any feasible completion to X_S will be within $p\%$ of \bar{z} .

The inclusion of such a feature as this, allowing a prescribed percentage error in final results, is very worthwhile in an industrial context where near optimal solutions are sometimes satisfactory when considered against the additional cost of computation required to discover and verify the true optimal solution to a problem. Hence this capability has been included in the present algorithm as indicated in Figures 1 and 2 in Chapter V.

In summary, the above considerations attest to the usefulness of (LP_{X_S}) as a tool for fathoming both single and multiple partial solutions, as well as to its utility in constructing an algorithm which will

produce near optimal solutions to problems of the form (1)-(4). We next consider an equivalent formulation which provides further insight into the effectiveness of (LP_{X_s}) as a tool for fathoming.

An Alternative View of the Imbedded Linear Program

Consider again the problem of fathoming a partial solution X_s ; this may be accomplished by demonstrating that there exists no assignment of admissible values to the free variables verifying (8)-(10) or by producing an optimal solution (i.e., one which minimizes $\sum_{j \notin S} c_j x_j$) which satisfies (8)-(10). Since fathoming by determination of an optimal solution is itself an integer programming problem, it is perhaps easier to direct effort towards determining sufficient conditions for the non-existence of solutions to (8)-(10). This is precisely the approach followed in attempting to show via (LP_{X_s}) that there is no completion to X_s which simultaneously improves the incumbent objective value and satisfies feasibility requirements. In this regard, confronted with the integer programming problem (I_{X_s}) , it seems quite natural that one might attempt to fathom X_s by exhibiting the non-existence of improving (i.e., relative to an incumbent solution) solutions to the associated linear program (P_{X_s}) , since it is clear that if (P_{X_s}) admits to no improving solution, then neither can (I_{X_s}) . The duality theorem of Linear Programming provides a convenient tool for accomplishing this goal while producing a continually improved lower bound on feasible solutions to (P_{X_s}) (and hence to (I_{X_s})), namely, (LP_{X_s}) , the dual of (P_{X_s}) which has already been discussed in this chapter with reference to computing the strongest surrogate constraints

available relative to the partial solution X_S . The equivalence of this approach and that of determining a strongest surrogate constraint via (LP_{X_S}) is provided by the following theorem; the algorithmic equivalence of the two approaches is obvious.

Theorem 3: There exists a feasible solution to (P_{X_S}) which improves on $\bar{z} \iff$ there does not exist a surrogate constraint which fathoms X_S .

Proof. Let v^* denote the infimal value of (LP_{X_S}) , which is necessarily feasible, and let z^* be the infimal value of (P_{X_S}) .

(\implies) Assume (P_{X_S}) feasible and $z^* + z^S < \bar{z}$. Then reference to the linear programs (LP_{X_S}) , (16), (17), and (P_{X_S}) shows immediately that $-v^* = z^* + z^S - \bar{z}$. Hence $-v^* < 0$, so that $v^* > 0$. Hence (LP_{X_S}) fails to produce a surrogate constraint which fathoms X_S .

(\impliedby) If there does not exist a surrogate constraint which fathoms X_S , then $v^* > 0$. Since (LP_{X_S}) is a minimization problem and the solution $\mu_i = 0, i=1, \dots, m$ and $\omega_j = 0, j \notin S$ is feasible for (LP_{X_S}) , it is clear that $\bar{z} - z^S$ provides a finite upper bound on v^* . Thus we obtain the feasibility of (P_{X_S}) from the duality theorem of Linear Programming since $0 < v^* \leq \bar{z} - z^S$. To see that there exists a feasible solution to (P_{X_S}) which improves on \bar{z} note that $v^* > 0 \implies -v^* = z^S - \bar{z} + \text{Min} \sum_{j \notin S} c_j x_j < 0 \implies z^S + z^* < \bar{z}$, as required.

The equivalence of the existence of solutions to (P_{X_S}) which improve upon the current best feasible solution and the non-existence of a surrogate constraint which fathoms X_S also demonstrates readily that although $v^* \leq 0$ provides a sufficient condition for fathoming X_S , this condition cannot also be necessary. This is clear since

feasibility of (P_{X_S}) does not always imply feasibility of (I_{X_S}) . This line of reasoning leads one quite naturally to consider, after failing to fathom X_S via (LP_{X_S}) , attempts to exploit bound redefinition capabilities of (LP_{X_S}) on the free variables in order to enhance the possibilities for fathoming after augmentation of some free variable. The most striking feature, however, of the above equivalence is that although addressing oneself to the feasibility of (P_{X_S}) seems far more natural and certainly less complicated than considering the existence of strongest surrogate constraints, this has apparently gone unnoticed. In addition to its simplicity, the approach presented here suggests the interesting theoretical possibility of its application in other areas of mathematical programming. We now turn to yet another interpretation, that provided by geometric considerations, of the development of surrogate constraints for fathoming X_S via (LP_{X_S}) .

Geometric Interpretation of Surrogate Constraints

Further insight into the process of fathoming a partial solution by the construction of an infeasible surrogate constraint is provided by a geometric interpretation of this process. Suppose, without loss of generality, that the variables x_1, \dots, x_k are free. Ignoring integrality, the feasible region of (P_{X_S}) at this stage can be represented by $B \cap P$ where

$$B = \{y = (y_1, \dots, y_k) \mid 0 \leq y_j \leq d_j, j=1, \dots, k\}, \text{ and}$$

$$P = \{y = (y_1, \dots, y_k) \mid b_i^s + \sum_{j=1}^k a_{ij} y_j \geq 0, \quad i=1, \dots, m\}.$$

We also consider the hyperplane H_μ and the closed half-space H_μ^+ defined as follows:

$$H_\mu = \{y = (y_1, \dots, y_k) \mid \sum_{i=1}^m \mu_i (b_i^s + \sum_{j=1}^k a_{ij} y_j) = 0, \quad \mu_i \geq 0, \quad i=1, \dots, m\}$$

$$H_\mu^+ = \{y = (y_1, \dots, y_k) \mid \sum_{i=1}^m \mu_i (b_i^s + \sum_{j=1}^k a_{ij} y_j) \geq 0, \quad \mu_i \geq 0, \quad i=1, \dots, m\}.$$

The following theorem establishes the fact that, for linear constraints, a single constraint always exists which can be used to reflect infeasibility of an entire system of constraints.

Theorem 4: $B \cap P = \phi \iff$ there exist $\mu_i \geq 0, \quad i=1, \dots, m$ such that

$$B \cap H_\mu^+ = \phi.$$

Proof. (\Leftarrow) For given $\mu_i, \quad i=1, \dots, m, H_\mu^+$ defines a closed half-space in E^k . Clearly, $y \in P$ implies $y \in H_\mu^+$, so that $P \subset H_\mu^+$. Hence

$$B \cap H_\mu^+ = \phi \text{ implies } B \cap P = \phi.$$

(\Rightarrow) Here we consider two cases. First assume that $P \neq \phi$, then since B is compact and P is closed, if $B \cap P = \phi$ there exists a hyperplane $K = \{y = (y_1, \dots, y_k) \mid \sum_{j=1}^k \alpha_j y_j + \beta = 0, \quad \alpha \neq 0\}$ strictly separating B and P ([23], page 50) such that $B \cap K^+ = \phi$ and $P \cap K^- = \phi$ where

$$K^+ = \{y = (y_1, \dots, y_k) \mid \sum_{j=1}^k \alpha_j y_j + \beta \geq 0\}$$

and

$$K^- = \{y = (y_1, \dots, y_k) \mid \sum_{j=1}^k \alpha_j y_j + \beta \leq 0\}.$$

If we let $\text{int}(K^-)$ denote the open half-space defined by

$$\text{int}(K^-) = \{y = (y_1, \dots, y_k) \mid \sum_{j=1}^k \alpha_j y_j + \beta < 0\}$$

then we also have that $P \cap \text{int}(K^-) = \emptyset$ since $K^- \supset \text{int}(K^-)$. The fact that $P \cap \text{int}(K^-) = \emptyset$ can be expressed by stating that the system

$$\sum_{j=1}^k a_{ij} y_j \geq -b_i^s, \quad i=1, \dots, m$$

$$\sum_{j=1}^k \alpha_j y_j < -\beta$$

has no solution $y \in E^k$. Using the non-homogeneous version of Farkas' theorem ([23], page 32), this implies that either

$$\sum_{i=1}^m \mu_i a_{ij} = \alpha_j, \quad j=1, \dots, k \tag{20}$$

$$\sum_{i=1}^m \mu_i b_i^s \leq \beta$$

or

$$\sum_{i=1}^m \mu_i a_{ij} = 0, \quad j=1, \dots, k \tag{21}$$

$$\sum_{i=1}^m \mu_i b_i^s < 0$$

has a solution such that $\mu_i \geq 0$, $i=1, \dots, m$. If (21) holds then for any $y \in E^k$ we have that $\mu(Ay + b^S) < 0$. But $P \neq \emptyset$ implies that for any $\mu_i \geq 0$, $i=1, \dots, m$, there exists some $y \in E^k$ such that $\mu(Ay + b^S) \geq 0$. Hence (21) cannot hold for $P \neq \emptyset$. Thus (20) must hold. Now $y \in K$ implies that $\alpha y + \beta = 0$, so that by (20) there exist $\mu_i \geq 0$, $i=1, \dots, m$ for which $\mu Ay = -\beta$. But (20) also implies that $\beta \geq \mu b^S$ so that we have $y \in K$ implies $\mu(Ay + b^S) \leq 0$. Hence $y \in K$ implies that $y \in H_\mu^-$ or, in other words, $K \subset H_\mu^-$. Thus $K^+ \supset H_\mu^+$, so that $B \cap H_\mu^+ = \emptyset$, which is the desired result.

On the other hand suppose $P = \emptyset$. This implies that there exists no $y \in E^k$ such that $-Ay \leq b^S$. From Gale's theorem for linear inequalities ([23], page 33) this implies the existence of $\mu_i \geq 0$, $i=1, \dots, m$ which satisfy

$$\sum_{i=1}^m \mu_i a_{ij} = 0, \quad j=1, \dots, k$$

$$\sum_{i=1}^m \mu_i b_i^S = -1.$$

Hence for any $y \in E^k$ we must have $\mu(Ay + b^S) = -1$. Thus, for this particular μ we have that $H_\mu^+ = \emptyset$. Hence $H_\mu^+ \cap B = \emptyset$, as required.

We now extend the results of Theorem 4 to include surrogate constraints of the form $\mu(b^S + Ay) + (\bar{z} - z^S - cy) > 0$. We will denote the feasible region of solution for (P_{X_S}) as R ; i.e., let $R = P \cap B$. We also consider the objective hyperplane

$$Z = \{y = (y_1, \dots, y_k) \mid \sum_{j=1}^k c_j y_j = \bar{z} - z^S\} \text{ and}$$

$$\text{int}(Z^-) = \{y = (y_1, \dots, y_k) \mid \sum_{j=1}^k c_j y_j < \bar{z} - z^S\}.$$

Then the following theorem is a restatement of the existence of a fathoming surrogate constraint if and only if there is no improving solution to (P_{X_S}) .

Theorem 5: $R \cap \text{int}(Z^-) = \phi \iff$ there exist $\mu_i \geq 0, i=1, \dots, m$ such that for all $y \in B, \mu(b^S + Ay) + (\bar{z} - z^S - cy) \leq 0$.

Proof. (\Leftarrow) Assume existence of $\mu_i \geq 0, i=1, \dots, m$ such that for no $y \in B, \mu(b^S + Ay) + (\bar{z} - z^S - cy) > 0$. Further assume that $R \cap \text{int}(Z^-) \neq \phi$. If this is the case, there exists some $y \in B$ which satisfies $b^S + Ay \geq 0$ and $\bar{z} - z^S - cy > 0$. But if this is true, then for any $\mu_i \geq 0, i=1, \dots, m$ we have that for some $y \in B, \mu(b^S + Ay) + (\bar{z} - z^S - cy) > 0$, which is a contradiction. Hence $R \cap \text{int}(Z^-) = \phi$.

(\Rightarrow) We consider two cases; first let $R \neq \phi$. Then $R \cap \text{int}(Z^-) = \phi$ implies that the system

$$-cy > z^S - \bar{z}$$

$$\begin{bmatrix} -A \\ I \\ -I \end{bmatrix} y \leq \begin{bmatrix} b^S \\ d \\ 0 \end{bmatrix}$$

where $d = (d_1, \dots, d_k)$, I is the $k \times k$ identity matrix, and 0 is a $k \times 1$ vector of zeroes, has no solution $y \in E^k$. Thus the non-homogeneous version of Farkas' theorem implies that

either
$$[\mu \ \omega \ v] \begin{bmatrix} -A \\ I \\ -I \end{bmatrix} = -c; \quad [\mu \ \omega \ v] \begin{bmatrix} b^S \\ d \\ 0 \end{bmatrix} \leq z^S - \bar{z} \quad (22)$$

or
$$[\mu \ \omega \ v] \begin{bmatrix} -A \\ I \\ -I \end{bmatrix} = 0; \quad [\mu \ \omega \ v] \begin{bmatrix} b^S \\ d \\ 0 \end{bmatrix} < 0 \quad (23)$$

has a solution $\mu_i \geq 0, i=1, \dots, m; \omega_j \geq 0, j=1, \dots, k; v_j \geq 0, j=1, \dots, k$. If (23) holds then for any $y \in E^k$ we must have $\mu Ay - \omega y + v y = 0$ and $\mu b^S + \omega d < 0$. Hence for any $y \in E^k$ we have $\mu(Ay + b^S) + \omega(d - y) + v y < 0$ if (23) is to hold. On the other hand $R \neq \phi$ implies that for any $y \in R$ we have $\mu(Ay + b^S) \geq 0; \omega(d - y) \geq 0; v y \geq 0$. Hence for $y \in R$ we obtain that $\mu(Ay + b^S) + \omega(d - y) + v y \geq 0$. This contradiction implies that (23) cannot hold; hence (22) must hold. Pick $y \in B$. Then from (22) we get $\mu Ay - \omega y + v y = cy$ and $\mu b^S + \omega d \leq z^S - \bar{z}$. Hence we have $\mu(Ay + b^S) + \omega(d - y) + v y + \bar{z} - z^S - cy \leq 0$. Further, since we have chosen $y \in B$ and since (22) assures us that $\omega \geq 0$ and $v \geq 0$ we may write that $\mu(Ay + b^S) + \bar{z} - z^S - cy \leq 0$ which is the desired result. This establishes the theorem for $R \neq \phi$.

Next let $R = \phi$. Since $R = P \cap B = \phi$, we know by Theorem 4 that there exist $\mu_i \geq 0, i=1, \dots, m$ such that $B \cap H_\mu^+ = \phi$; i.e., there exist $\mu_i \geq 0, i=1, \dots, m$ such that for all $y \in B, \mu(b^S + Ay) < 0$. Since B is compact and $\mu(b^S + Ay)$ is a continuous function of y for fixed μ , we know

there exists $\delta > 0$ such that $\mu(b^S + Ay) \leq -\delta < 0$. Now pick $k > 0$ such that $k\delta \geq \bar{z}$. Then if we define $\mu' = k\mu$ it is clear that for all $y \in B$,

$$\mu'(b^S + Ay) + (\bar{z} - z^S - cy) \leq \mu'(b^S + Ay) + \bar{z} \leq -k\delta + \bar{z} \leq 0$$

as required, since $z^S \geq 0$, $c \geq 0$, and $y \in B$. This completes the proof.

Note that $B \cap P = \emptyset$ implies that there is no feasible solution to (P_{X_S}) and that $B \cap H_\mu^+ = \emptyset$ ($P \subset H_\mu^+$), implies the existence of a separating hyperplane H_μ between B and P (this hyperplane becomes degenerate if $P = \emptyset$). Thus determination of a surrogate constraint of the form $\mu(b^S + Ay) \geq 0$ which is satisfied by no $y \in B$ is equivalent to specifying a hyperplane strictly separating B and P since both establish infeasibility of the problem under consideration. Similarly we note that determination of a surrogate constraint of the form $\mu(b^S + Ay) + (\bar{z} - z^S - cy) > 0$ which is satisfied by no $y \in B$ is equivalent to specifying a hyperplane which separates $P \cap \text{int}(Z^-)$ and B .

If we now denote by $[\bar{R}]$ the convex hull of the integer points in R , then since $R \supset [\bar{R}]$, $R \cap \text{int}(Z^-) = \emptyset$ implies $[\bar{R}] \cap \text{int}(Z^-) = \emptyset$ while it is clear that the converse does not necessarily hold. Hence, it is precisely those instances in which $[\bar{R}] \cap \text{int}(Z^-) = \emptyset$ but $R \cap \text{int}(Z^-) \neq \emptyset$ that the imbedded linear program fails to fathom although there exists no feasible completion of the current partial solution which improves the incumbent objective value.

This concludes our investigation into the various means of implementing the basic algorithm presented in Chapter II. The various

tests considered in Chapter III and their extensions presented in this chapter appear to imply that their implementation should result in quite a powerful means of treating problems of the form (1)-(4). This implication is indeed true, as is indicated by the computational results exhibited in the following chapter.

CHAPTER V

COMPUTATIONAL RESULTS

Basic Considerations

The implicit enumeration algorithm described in Chapters II, III, and IV has been programmed in order to provide a general evaluation of the algorithm as a whole and a specific investigation into certain important alternative steps for implementation discussed in Chapters III and IV. The computer program (refer to Appendix B for documentation) is coded in FORTRAN for the CDC 6600 and is currently capable of handling integer programming problems on the order of 70 constraints and 70 variables; a minor program change allows an increase in problem size subject only to the core memory restriction of the object computer. A general flowchart of the particular version of the implicit enumeration algorithm used in the computer program is given in Figure 1; a more detailed flow description of the specific variant of the imbedded linear program used is given by Figure 2.

A few general comments are in order before discussing the results of the computational study. The solution times presented in Tables 2-5 are the CDC 6600 CPU seconds required for solution of the individual examples. The time required for initialization (program loading and input of initial problem) and termination (output of final results) is

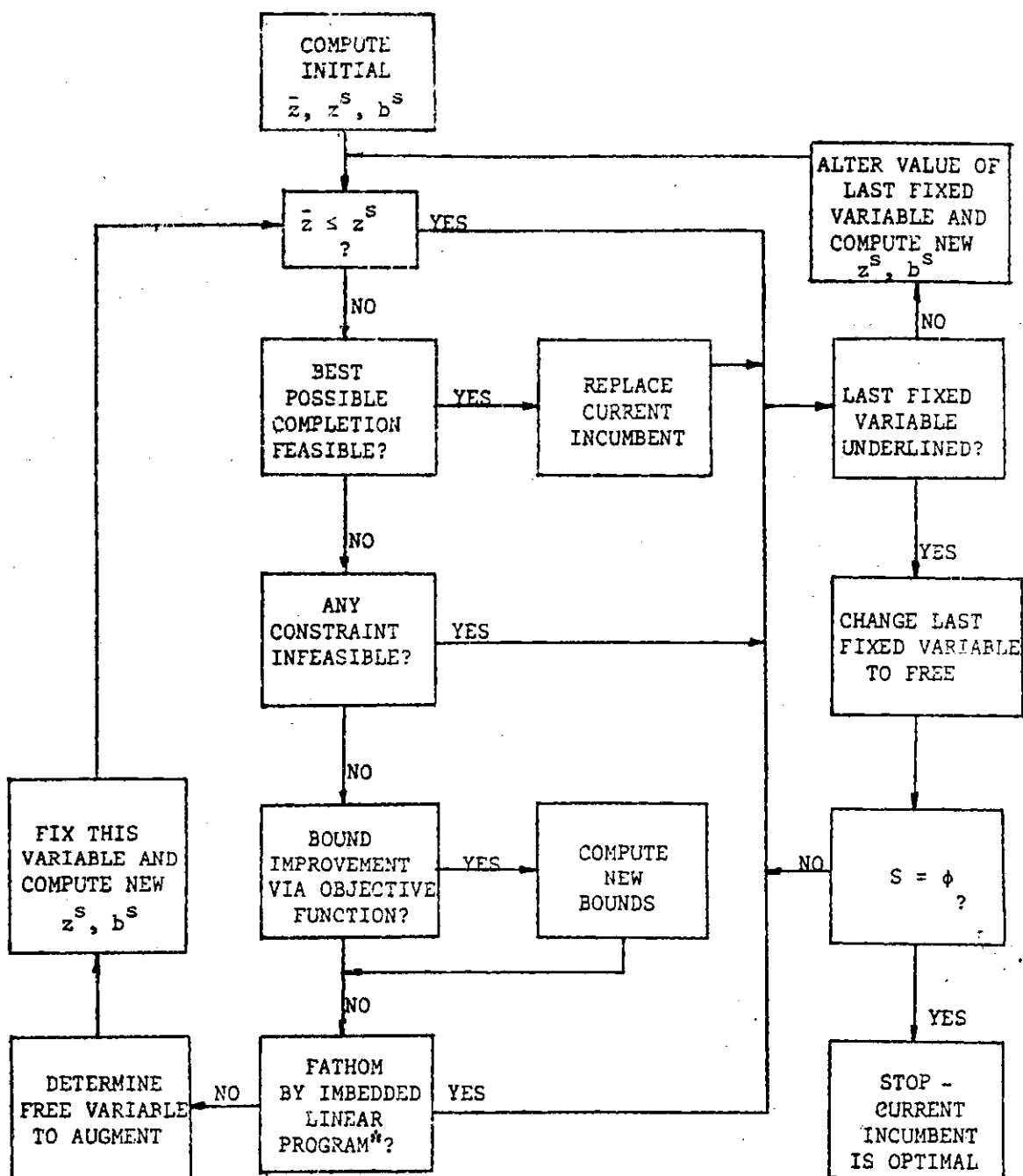


Figure 1. Flow Diagram of Implicit Enumeration Algorithm

* See Figure 2 for a more detailed description of fathoming via the imbedded linear program.

not included* in order to provide a more accurate comparison with the results of [10]. In order to insure uniformity with respect to the effects of factors other than those under investigation, the following solution considerations were identical throughout the study. Computation was begun in each instance with $S_0 = \{1, \dots, n\}$ and $X_{S_0} = (0, \dots, 0)$. This initialization scheme proved generally more efficient on a few test cases, and hence was adopted for the entire study. The imbedded linear program was utilized at each iteration on which the simple fathoming tests indicated in Figure 1 failed, in order to provide results consistent with those presented in [10]; initial test cases confirmed the conclusion of Geoffrion in [10] that using the imbedded linear program proved far more effective than ignoring the imbedded linear program as a tool for fathoming. As indicated in Chapter IV, surrogate constraints computed by the imbedded linear program may be appended to the original problem in the hope that the simple fathoming tests will prove useful on such constraints. No such constraints were added in the problems solved since it was felt that the additional computation required to actually compute a surrogate constraint once (LP_{X_S}) had been used would be approximately equal to that required to re-enter the imbedded linear program at a successive iteration. In addition, the near optimal solution feature of the imbedded linear program was not exploited, so that each problem was solved to an optimal solution.

In order to provide results consistent with naïve program

*This time is very small (.04 sec.) and is essentially the same for each example considered.

utilization, the initial bounds used on program variables were taken as those immediately available via optimality and feasibility considerations on the variables of the starting tableau. For example, from the inequality $11 - 3x_1 - 5x_2 \geq 0$ one readily obtains upper bounds of 3 and 2 on x_1 and x_2 , respectively. In addition, the initial upper bound on the objective value was taken as $\bar{z} = \sum_{j=1}^n c_j d_j$, although stronger bounds on optimality were available from many of the problems. The ability to ascertain strong initial bounds is, however, a factor of undeniable importance (due to the consequent size reduction of the enumeration problem), as is dramatically illustrated by comparing the results of problems 12, 14, and 16 to those of 13, 15, and 17 in Tables 2-5.

Tables 2-5 also give the total number of iterations (an iteration is defined as a single pass through the flow diagram of Figure 1) required for complete solution of each problem. In addition, the iteration number on which the optimal solution was discovered is given in order to provide an indication of the effort required in verifying optimality. Finally, an indication of the algorithm's ability to examine a large proportion of the admissible solutions implicitly is given as a percentage computed as follows:

$$\frac{(\text{number of admissible solutions} - \text{number of iterations}) \times 100}{\text{number of admissible solutions}}$$

Results

The specific items under consideration in the computational study are:

- (1) whether bound redefinition via the imbedded linear program as discussed in Chapter IV is computationally expedient;
- (2) whether any general conclusions can be reached regarding a comparison between a minimum-branch augmentation rule and Balas' augmentation rule (refer to Chapter III); and
- (3) whether the direct means of treating bounded variables presented here is significantly more efficient than treatment via conversion of problem variables to sums of binary variables.

The minimum-branch augmentation discipline used specified the augmentation of variables at their lower bound to provide further contrast with the Balasian rule of augmentation at the upper bound.

The general characteristics of the problems solved are indicated by Table 1; the test problems themselves are reproduced in Appendix C. Each problem was solved using each of the four possible combinations of the factors described above for completeness. The results (Tables 2-6) indeed seem to indicate several general conclusions for the problems solved.

Utilization of the bound redefinition capability of the imbedded linear program appears to offer a substantial gain in the overall performance of the algorithm, regardless of the augmentation rule with which it is used. This improvement occurs not only in solution times, but also in the total number of iterations required for solution and in the number of iterations required to reach an optimal solution. The

Table 1. Problem Characteristics

Problem		Number of Variables	Number of Constraints	Number of Admissible Solutions
HALDI [19]				
1	1	5	4	1.792×10^3
2	2	5	4	2.592×10^3
3	3	5	4	4.400×10^3
4	4	5	4	2.016×10^3
5	7	5	4	1.409×10^6
6	8	5	4	3.511×10^6
7	9	6	6	5.832×10^3
8	10	12	10	2.741×10^8
IBM TEST [19]				
9	1	7	7	2.799×10^5
10	2	7	7	7.813×10^4
11	3	4	3	3.119×10^5
12	4a	15	15	4.748×10^{12}
13	4b	15	15	1.074×10^9
14	5a	15	15	2.059×10^{14}
15	5b	15	15	1.074×10^9
16	6a	31	31	1.919×10^{32}
17	6b	31	31	2.147×10^9
18	7	50	12	2.931×10^{31}
19	8	37	12	1.553×10^{35}
20	9	15	35	3.277×10^4
21	DIET PROBLEM	14	12	7.465×10^7

improvement rendered by the bound redefinition capability is quite dramatic in some instances (e.g., see 5, 6, 18, and 19 in Tables 3 and 5 and 8, 17, 19, and 21 in Tables 2 and 4). The noticeable area in which virtually no improvement resulted from redefining bounds on free variables in the imbedded linear program was in problems 12-17; even in these cases, only a slight increase in solution time was suffered due to the tests for bound redefinition. The apparent conclusion is that the capability of the imbedded linear program to aid in bound redefinition is a computationally expedient means of improving the enumeration process.

In comparing the data obtained by varying the augmentation rule, results are not nearly so well defined as above. The minimum-branch rule is apparently superior on problems 1-8, 16, and 18 while Balas' rule proved more effective on problems 9-15, 17, and 19-20; results on problem 21 were mixed. Although these results do not indicate any general preference of which augmentation rule should be used, it is interesting to note that if the bound redefinition feature of the imbedded linear program is used, the algorithm generally appears to perform more efficiently with the minimum-branch rule; the only significant exceptions to this statement are problems 15 and 19 in Tables 4 and 5. This tends to suggest that for naïve program usage, the bound redefinition feature of the imbedded linear program and the minimum-branch augmentation rule should be used.

The general performance of the algorithm appears quite satisfactory, both in terms of solution times as well as in terms of the

ability of the algorithm to implicitly enumerate a large percentage of the admissible solutions to each problem. In this regard, however, one cannot underestimate the consequence of providing good initial bounds on the problem variables in order to reduce the number of admissible solutions; reference to problems 12, 14, and 16* versus problems 13, 15, and 17** in any of the Tables 2-5 attests to this fact. It is also of significance to note that every problem was solved (refer to Table 4), and within reasonable limits on the required amount of computation.

An excellent summary of recent computational experience with other algorithms is provided in [10]. In order to provide a more meaningful comparison between the solution times presented in [10] and those given here, the computer code given in [9] has been run*** on the CDC 6600 for problem 20, which is a binary integer programming problem. The CDC 6600 CPU time**** required for solving this problem on the code given in [9] was 7.443 sec. A direct comparison between codes is hazardous due to programming differences and different implementations of the basic enumerative procedure. It is of interest to note, however, that solution times in Tables 2-5 vary from 4.045 sec. to 5.228 sec., indicating that the algorithm presented here is slightly more effective

* Relatively weak initial bounds are used in these cases.

** The considerably stronger bounds used by Geoffrion in [10] are used in these examples.

*** With $S_0 = \{1, \dots, 15\}$, $X_{S_0} = (0, \dots, 0)$; no surrogate constraints are carried in the solution process; the imbedded linear program is used at each opportunity.

**** Subtracting .06 sec. for program initialization and termination.

on this particular problem. Hence one would suspect that on integer programs with larger than binary bounds on the problem variables where relative bound redefinition plays an important role (problems 18 and 19, for example), the algorithm presented here should compare equally well, if not better, than existing solution methods. It is also worthwhile to reiterate the ability of an algorithm such as that developed in this thesis to deal with problems in which bounds on program variables are so large as to make binary conversion impossible under computer storage restrictions.

To provide a more decisive comparison between the algorithm presented here which treats bounded variables directly and existing algorithms which treat only binary problems, all of the problems solved* were converted to binary form and then solved on the CDC 6600. The results of this investigation, which conclusively indicate the efficiency gained by handling such problems directly, are presented in Table 6. Table 6 gives problem solution time for each binary problem on the code (ENUMER8) for the algorithm presented in this thesis as well as the solution time required on the code (RIP30C) presented by Geoffrion in [9]. In addition, Table 6 provides the problem size (in both binary and direct form) of each example considered.

The results from Table 6 indicate quite emphatically the advantages rendered by solving such problems directly; on each problem except

*The IBM TEST [19] problems 4b, 5b, and 6b are not considered here; their counterparts with larger bounds (problems 4a, 5a, and 6a) were converted, however, both to give an indication of the ability of the algorithm presented here to deal with large problems and to demonstrate the dramatic effect of solving such problems directly.

number 16, solution times increased by a factor of 4-12 when binary conversion was used. Reference to problems 8, 12, 13, 14, 15, and 18 indicates that the superiority of the direct technique over binary conversion techniques appears to increase with the size of the problem. It is also interesting to note that for most of the larger problems (e.g., problems 12, 13, 14, 15, and 17) in binary form ENUMER8 appears to perform significantly better than RIP30C. On the other hand, for problems 5, 6, 8, and 16 in binary form, RIP30C is more efficient than ENUMER8. These discrepancies are most likely due to different augmentation rules* and slightly different implementations of the basic enumerative process. The dramatic improvements obtained by solving problems directly over solving problems in binary form on either RIP30C or ENUMER8, however, is attributable only to the conclusion that, for most problems, the direct method of solution is far superior. The single exception to this statement among those problems solved is problem 16; a reasonable explanation of this exception is indicated in Chapter VI and an enhancement to the direct treatment of bounded variables is also indicated there which should mitigate solution times on such problems.

In summary, the implementation of the enumerative algorithm presented in Chapters II-IV has shown this method to be among the best available for solving general integer programming problems of the form (1)-(4); computational success is certainly exhibited on those problems

*The particular augmentation discipline used can effect results quite substantially; e.g., refer to problems 15-20 in Tables 2 and 3.

considered. In fact, utilization of the algorithm presented here makes core storage limitations of present day computers a more realistic restriction than computation time limitations in determining which integer programs can be solved economically. Furthermore, this implementation has indicated that the capability of using the imbedded linear program to redefine relative bounds on free program variables is a worthwhile incorporation which can lead to significant improvement in some instances at only slightly increased computational effort on each iteration. We next consider conclusions indicated by this thesis as well as areas it suggests for future research.

Table 2. Results Using Minimum-Branch Augmentation
and No Bound Redefinition

Problem		Solution Time* (sec.)	Number of Iterations	Iteration on Which Optimum Solution Was Found	Percentage of Admissible Solutions Implicitly Enumerated
HALDI [19]					
1	1	.025	54	21	96.986
2	2	.017	36	24	98.611
3	3	.021	53	29	98.795
4	4	.016	26	23	98.710
5	7	.097	352	252	99.975
6	8	.126	472	327	99.987
7	9	.029	32	32	99.451
8	10	.372	206	188	100.000
IBM TEST [19]					
9	1	.060	36	22	99.987
10	2	.130	88	14	99.887
11	3	.022	85	18	99.973
12	4a	1.545	409	304	100.000
13	4b	1.119	234	177	100.000
14	5a	52.238	14549	553	100.000
15	5b	40.792	8035	136	99.999
16	6a	63.115	6924	5450	100.000
17	6b	20.984	482	22	100.000
18	7	3.737	1478	1257	100.000
19	8	>500.000	-	-	-
20	9	5.228	414	20	98.737
21	DIET PROBLEM	.938	477	448	99.999

*Excluding .04 sec. for program initialization and termination.

Table 3. Results Using Balas' Augmentation
Rule and No Bound Redefinition

Problem		Solution Time* (sec.)	Number of Iterations	Iteration on Which Optimum Solution Was Found	Percentage of Admissible Solutions Implicitly Enumerated
HALDI [19]					
1	1	.043	106	21	94.085
2	2	.031	66	24	97.454
3	3	.045	106	29	97.591
4	4	.022	41	23	97.966
5	7	2.567	10363	7529	99.265
6	8	3.627	14625	695	99.583
7	9	.108	153	87	97.377
8	10	2.043	1430	344	99.999
IBM TEST [19]					
9	1	.059	36	22	99.987
10	2	.094	61	14	99.922
11	3	.020	92	18	99.970
12	4a	1.415	409	334	100.000
13	4b	.922	190	158	100.000
14	5a	51.112	12920	622	100.000
15	5b	27.276	4794	158	100.000
16	6a	126.582	22133	20852	100.000
17	6b	10.603	254	22	100.000
18	7	25.297	7053	6783	100.000
19	8	185.573	46577	46566	100.000
20	9	4.045	290	18	99.115
21	DIET PROBLEM	.820	310	257	100.000

*Excluding .04 sec. for program initialization and termination.

Table 4. Results Using the Minimum-Branch Augmentation Rule with Bound Redefinition

Problem		Solution Time* (sec.)	Number of Iterations	Iteration on Which Optimum Solution Was Found	Percentage of Admissible Solutions Implicitly Enumerated
HALDI [19]					
1	1	.025	46	21	97.433
2	2	.026	36	24	98.611
3	3	.025	53	29	98.795
4	4	.015	26	23	98.710
5	7	.108	352	252	99.975
6	8	.135	435	327	99.988
7	9	.033	32	32	99.451
8	10	.285	127	107	100.000
IBM TEST [19]					
9	1	.058	36	22	99.987
10	2	.143	88	14	99.887
11	3	.024	84	18	99.973
12	4a	1.504	399	308	100.000
13	4b	1.098	220	171	100.000
14	5a	53.372	14606	557	100.000
15	5b	42.773	7832	152	99.999
16	6a	62.920	6832	5358	100.000
17	6b	12.647	244	22	100.000
18	7	3.328	725	516	100.000
19	8	79.331	12638	12619	100.000
20	9	4.749	375	16	98.856
21	DIET PROBLEM	.603	143	121	100.000

*Excluding .04 sec. for program initialization and termination.

Table 5. Results Using Balas' Augmentation
Rule with Bound Redefinition

Problem		Solution Time* (sec.)	Number of Iterations	Iteration on Which Optimum Solution Was Found	Percentage of Admissible Solutions Implicitly Enumerated
HALDI [19]					
1	1	.041	73	21	95.926
2	2	.033	66	24	97.454
3	3	.052	100	29	97.727
4	4	.025	41	23	97.966
5	7	1.102	2368	1511	99.832
6	8	1.771	3972	380	99.887
7	9	.101	95	64	98.371
8	10	1.936	1051	148	100.000
IBM TEST [19]					
9	1	.060	36	22	99.987
10	2	.100	61	14	99.922
11	3	.023	92	18	99.970
12	4a	1.496	404	329	100.000
13	4b	.957	183	151	100.000
14	5a	52.909	12921	612	100.000
15	5b	28.604	4794	158	100.000
16	6a	125.224	22076	20795	100.000
17	6b	10.895	254	22	100.000
18	7	16.415	3907	3647	100.000
19	8	47.029	2744	2726	100.000
20	9	4.132	251	18	99.234
21	DIET PROBLEM	.826	243	198	100.000

* Excluding .04 sec. for program initialization and termination.

Table 6. A Comparison Between Binary and Direct Implicit Enumeration

Problem	Problem Size		CDC 6600 CPU Secs.* ENUMER8 Direct**	CDC 6600 CPU Secs.* ENUMER8 Binary**	CDC 6600 CPU Secs.* RIP30C**	IBM 7044 Execution Time [10] in Secs.	
	Problem Size Direct (# Constraints x # Variables)	Problem Size Binary (# Constraints x # Variables)					
HALDI [19]							
1	1	4 5	4 11	.025	.124	.178	-
2	2	4 5	4 13	.026	.131	.095	-
3	3	4 5	4 14	.025	.200	.204	-
4	4	4 5	4 12	.015	.080	.107	-
5	7	4 5	4 22	.108	2.031	.524	1.8
6	8	4 5	4 23	.135	1.571	.557	3.0
7	9	6 6	6 15	.033	.234	.169	-
8	10	10 12	10 30	.285	4.221	1.112	3.6
IBM TEST [19]							
9	1	7 7	7 21	.058	.268	.505	.6
10	2	7 7	7 21	.143	.893	.809	1.2
11	3	3 4	3 21	.024	.242	.339	.6
12	4a	15 15	15 45	1.504	9.225	16.887	-
13	5a	15 15	15 60	53.372	519.661***	577.920	-
14	6a	31 31	31 124	62.920	747.728	>1024.000	-
15	7	12 50	12 121	3.328	19.104	20.263	-
16	8	12 37	12 133	79.331	22.678	8.963	-
17	9	35 15	35 15	4.749	4.749	7.443	26.4
18	DIET PROBLEM	12 14	12 30	.603	3.246	3.212	-

*Excluding program initialization and termination time; .06 sec. for ENUMER8 binary and RIP30C; .04 sec. for ENUMER8 direct (same results as Table 4).

**Initial partial solutions used contained every variable set at its lower bound. The imbedded linear program was used at each opportunity; no surrogate constraints were carried in the solution process. In ENUMER8 the minimum branch augmentation rule was used and the imbedded linear program was utilized for bound redefinition.

***This problem was solved using Balas' augmentation rule in 228.038 sec.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

The conclusions indicated by this study are of two varieties:

(1) those which are general in nature, relating to the use of explicit enumeration as a computationally expedient tool for solving the bounded variable pure integer programming problem and (2) those which are related to the specific features of the algorithm examined herein. Even though any general conclusions available are, in fact, extrapolations of the experience gained by working with a specific algorithm there are nonetheless some valid statements which can be made in this regard. First, the algorithm presented possesses the attributes of computational reliability and efficiency required for solving problems in the small to medium scale range. Reasonable computational expediency should be expected with problems up to the order of several hundred variables and/or constraints; the computational efficiency in many instances will be directly related to the original bounds determined for the problem variables. A reasonable working hypothesis for naïve usage of this algorithm appears to be that given in Chapter V of utilizing the minimum-branch augmentation rule in conjunction with the bound redefinition feature of the imbedded linear program.

For the specific conclusions relating to the usefulness of bound redefinition in the imbedded linear program and to the efficiency of the Balasian and minimum-branch augmentation rules, the reader is referred

to the discussion in Chapter V. A reasonable item for further investigation in this area might be the development and testing of an augmentation rule which combined the characteristics of a minimum-branch and a Balasian augmentation rule. For instance, one might consider a minimum-branch rule which used Balas' rule to break ties among candidates for augmentation under the minimum-branch rule. An investigation into such a branching discipline and into more of the augmentation rules presented in Chapter III would no doubt prove worthwhile.

Another area for further investigation concerns the functional dependence of solution time on the number of problem variables.* The results of Tables 2-5 indicate that this dependence is somewhat erratic, but perhaps some more specific statement could be made after further investigation into solution times required for larger problems. Another question of interest here concerns the functional dependence of solution time on the original bounds specified for each of the variables. Reference to problems 12-17 in Table 4, for instance, indicates that this dependence may be less than linear, which is an important indication, especially for larger problems.

The number of explicit evaluations made in the solution process is very closely connected with the range of feasible values for free variables at the time an augmentation is to be performed; hence it seems reasonable that considerable improvement in the efficiency of the enumeration process should result from improved means of relative bound

*Geoffrion's results in [10] indicate a linear or low-order monomial dependence.

redefinition. In other words, one possible avenue for future research concerns development and verification of computationally expedient means for answering, or attempting to answer the following question: What are the least upper bound and the greatest lower bound for each free variable in all completions of the current partial solution which are feasible and which improve on the current best solution? One means of approaching such a question is by recourse to problems (18) and (19) which provide a sufficient answer to the question in terms of (P_{X_S}) , the linear program associated with the current partial solution. The appropriate consideration here, of course, is how far one should go in attempting to redefine bounds on free variables. Should bound redefinition be attempted for all free variables, or just for a subset of the free variables? One approach would be to determine the solutions of problems (18) and (19) for only that variable which has been chosen for augmentation. Perhaps a heuristic rule such as attempting a redefinition of bounds only for those last k (where k is parametric) variables most recently fixed but currently free will prove useful. Another possibility here is to attempt to strengthen the bound redefinition rule (refer to Chapter IV) currently being exploited. A possible extension is to attempt bound redefinition for not only all of those variables whose column is non-negative or non-positive in the current basis inverse matrix (B^{-1}) of the imbedded linear program but also those variables whose respective columns in B^{-1} admit to a simple calculation (i.e., without a pivot operation) based on the current basis of the imbedded linear program in order to redefine upper and/or lower bounds.

In other words, one might also use the structural properties of the imbedded linear program as an additional aid in attempting to redefine bounds on free variables.*

In addition there are many questions concerning the effect of the various decision steps in the implicit enumeration algorithm which can be conveniently posed within the framework of a further computational study. For example: Once a variable is selected for augmentation, should it be fixed at its current lower or upper bound? How many surrogate constraints should be maintained throughout the solution process? How should one decide on the appropriate iteration frequency for attempting to fathom via the imbedded linear program? The information gained from such a study would undoubtedly lead to improved usage of the current algorithm.

The algorithm presented here spends only a small fraction of its time in the backtracking phase of the enumerative procedure, due to the fact that only the two simplest arrangements $(0, 1, \dots, d_j$ and $d_j, d_j - 1, \dots, 1, 0)$ available are allowed for fixed variables. This requirement that a variable be fixed at either its current upper or lower bound restricts one's ability to use rounded variables (or naturally integral variables) obtained from the solution to (P_{X_s}) as candidates for augmentation at intermediate values.** This improvement

* Refer to Chapter IV. Recall that each ω_j activity is the negative of the corresponding s_j activity, which may alleviate the necessity for dual simplex pivots in solving (18) and (19).

** This heuristic is available for a binary algorithm. See ref. [10], for example.

can easily be included with more complex backtracking schemes; its use should certainly lead to improvements in those problems where many variables assume intermediate (i.e., between their lower and upper bounds) values in optimal solutions. For problems of this variety one might expect a binary algorithm to prove more effective than the direct algorithm presented here since binary conversion facilitates more rapid examination of these intermediate values; it is interesting to note that the optimal solution to problem 16 in Table 6 is of exactly this nature.

Undoubtedly the most important theoretical considerations of this study are the new interpretations of surrogate constraints considered in Chapter IV. These interpretations suggest many areas for further investigation, some of which are the following:

(1) Perhaps the geometric interpretation of a surrogate constraint as a separating hyperplane can be exploited in order to produce such surrogate constraints by means other than linear programming.

(2) Either interpretation of surrogate constraints given in Chapter IV may lead to improved means of bound redefinition, which should in turn improve the algorithm presented here. Such considerations need not be limited to problems within the current framework of (1)-(4).

(3) The generation of a separating surface rather than a separating hyperplane may prove useful in the context of nonlinear programming algorithms.

(4) Perhaps the technique of manipulating the feasibility of linear programs could prove useful in other branch and bound applications.

(5) The technique of separating the regions corresponding to general problem constraints and problem bounding constraints might prove useful in the development of an algorithm for solving linear programming problems in which some variables are bounded.

Hence, the avenues opened by these considerations certainly seem fertile in terms of future research possibilities which should not only improve the enumerative procedure presented in this thesis, but also extend the techniques used in the algorithm presented here to other areas.

As a final possibility for extending the results of this thesis, one might consider an investigation into the mixed integer programming version of the algorithm presented here. This extension is straightforward for problems of the form:

$$\begin{aligned} & \text{Min } c^1 x^1 + c^2 x^2 \\ & \text{subject to } b + A^1 x^1 + A^2 x^2 \geq 0 \\ & \qquad \qquad \qquad x^1 \leq d^1, x^1 \geq 0 \text{ and integer} \\ & \qquad \qquad \qquad x^2 \leq d^2, x^2 \geq 0 \end{aligned}$$

where x^1 , c^1 , d^1 are n_1 -vectors; x^2 , c^2 , d^2 are n_2 -vectors; b is an m -vector; A^1 and A^2 are $m \times n_1$ and $m \times n_2$ matrices, respectively. The question of fathoming a partial solution X_s^1 using an imbedded linear program in this case would be related to the feasibility of the linear program obtained by omitting the integrality restriction on x^1 ; the

duality theory of Linear Programming would once again provide convenient means for answering this question. In particular, one would use $(LP_{X'_S})$ in order to generate surrogate constraints of the form $\mu(b + A^1 x^1 + A^2 x^2) + (\bar{z} - c^1 x^1 - c^2 x^2) > 0$ in an attempt to fathom X'_S .

$$\text{Min } \bar{z} - z^S + \sum_{i=1}^m \mu_i b_i^S + \sum_{j \notin S} d_j^1 \omega_j^1 + \sum_{j=1}^{n_2} d_j^2 \omega_j^2 \quad (LP_{X'_S})$$

$$\text{subject to } \sum_{i=1}^m \mu_i a_{ij}^1 - \omega_j^1 \leq c_j^1, \quad j \notin S$$

$$\sum_{i=1}^m \mu_i a_{ij}^2 - \omega_j^2 \leq c_j^2, \quad j=1, \dots, n_2$$

$$\mu_i \geq 0, \quad i=1, \dots, m; \quad \omega_j^1 \geq 0, \quad j \notin S; \quad \omega_j^2 \geq 0, \quad j=1, \dots, n_2.$$

In summary, although the algorithm presented in this thesis provides an extremely effective means for solving the bounded variable pure integer programming problem, many avenues for future investigation into the results of its refinement and extension are open. Such investigations will undoubtedly extend the results of this study in relation to the general question of the effectiveness of implicit enumeration as a means for solving general discrete variable programming problems.

APPENDIX A

EXAMPLE

The following example is solved using one of the possible variations of the algorithm presented (see Chapters II-V). The following decision rules are used:

- (1) Assume that initially all variables are free.
- (2) Attempt to fathom using the imbedded linear program as frequently as possible in the solution process.
- (3) When augmentation is necessary, augment (at its upper bound) a variable which minimizes the total number of branches required for explicit enumeration.
- (4) Attempt redefinition of upper bounds for free variables at each iteration using optimality considerations only.
- (5) Initialize $\bar{z} = \sum_{j=1}^n c_j d_j$.

Initial Problem

$$\begin{aligned} & \text{Min } x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\ \text{s.t. } & -5 + x_1 \qquad \qquad \qquad + x_4 + x_5 \qquad \qquad + x_7 \geq 0 \\ & -5 \qquad \qquad \qquad + x_3 \qquad \qquad + x_5 \qquad \qquad + x_7 \geq 0 \\ & -5 + x_1 \qquad \qquad \qquad + x_5 \qquad \qquad + x_7 \geq 0 \\ & -4 + x_1 + x_2 \qquad \qquad + x_4 + x_5 \qquad \qquad \geq 0 \\ & -4 + x_1 + x_2 \qquad \qquad \qquad + x_5 \qquad \qquad + x_7 \geq 0 \\ & -4 + x_1 \qquad \qquad + x_3 \qquad \qquad \qquad + x_7 \geq 0 \\ & -3 + x_1 + x_2 + x_3 \qquad \qquad \qquad + x_7 \geq 0 \\ & \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad x_i \geq 0, \quad i=1, \dots, 7 \end{aligned}$$

$$x_1 \leq 3, x_2 \leq 3, x_3 \leq 3, x_4 \leq 4, x_5 \leq 4, x_6 \leq 4, x_7 \leq 3$$

Iteration 1

$$S = \phi, z^S = 0, b^S = (-5, -5, -5, -4, -4, -4, -3), \bar{z} = 24$$

Step 1: Fathom by $\bar{z} \leq z^S$? No.

Step 2: Best possible completion feasible? No.

Step 3: Infeasible constraint? No.

Step 4: Bound redefinition possible? No.

Step 5: Solution of imbedded linear program.

Initial Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_1	s_2	s_3	s_4	s_5	s_6	s_7	
s_1	1	1	0	1	1	1	1	1	-1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
s_2	1	0	0	0	1	1	0	1	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	0
s_3	1	0	1	0	0	0	1	1	0	0	-1	0	0	0	0	0	0	1	0	0	0	0	0
s_4	1	1	0	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0	0
s_5	1	1	1	1	1	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0
s_7	1	1	1	1	0	1	1	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	1
(-v)	-24	-5	-5	-5	-4	-4	-4	-3	3	3	3	4	4	4	3	0	0	0	0	0	0	0	0

⋮

Optimal Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_1	s_2	s_3	s_4	s_5	s_6	s_7	
μ_4	.5	.5	0	.5	1	.5	0	0	-.5	0	0	0	-.5	0	.5	.5	0	0	0	.5	0	-.5	0
s_2	0	-.5	0	-.5	0	.5	0	1	.5	-1	0	0	.5	0	-.5	-.5	1	0	0	-.5	0	.5	0
s_3	0	-1	0	-1	0	-1	0	0	0	0	-1	0	0	0	1	0	0	1	0	0	0	-1	0
s_4	.5	.5	0	-.5	0	-.5	0	0	.5	0	0	-1	.5	0	-.5	-.5	0	0	1	-.5	0	.5	0
μ_2	.5	.5	1	.5	0	.5	0	0	.5	0	0	0	-.5	0	-.5	-.5	0	0	0	.5	0	.5	0
s_6	1	0	0	0	0	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0
μ_6	.5	.5	0	.5	0	.5	0	1	-.5	0	0	0	.5	0	-.5	.5	0	0	0	-.5	0	.5	0
(-v)	-17.5	1.5	0	1.5	0	2.5	0	2	1.5	3	3	4	1.5	4	.5	1.5	0	0	0	2.5	0	2.5	0

Fail to fathom by surrogate constraint.

Step 6: Dual variables of imbedded linear program integral? No.

Step 7: Fix variable 1 at value 3.

Iteration 2

$$S = \{1\}, X_S = (3), z^S = 3, b^S = (-2, -5, -2, -1, -1, -1, 0), \bar{z} = 24$$

Step 1: Fathom by $\bar{z} \leq z^S$? No.

Step 2: Best possible completion feasible? No.

Step 3: Infeasible constraint? No.

Step 4: Bound redefinition possible? No.

Step 5: Solution of imbedded linear program.

Initial Tableau (After Deleting s_1)

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
s_2	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
s_3	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	1	1	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0
μ_2	1	1	1	1	1	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	0	0	0	0	-1	0	1	1	0	0	0	1	0	-1	0	0	0	-1	0	1
(-v)	-16	3	0	3	0	4	3	4	3	3	4	3	4	-1	0	0	0	1	0	4

Optimal Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
s_2	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
ω_7	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	1	1	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0
μ_2	1	1	1	1	1	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	0	-1	0	-1	-1	-1	1	1	0	-1	0	1	0	0	0	1	0	-1	0	0
(-v)	-16	2	0	2	0	3	3	4	3	2	4	3	4	0	0	1	0	1	0	3

Fail to fathom by surrogate constraint.

Step 6: Dual variables of imbedded linear program integral? Yes;

hence fathom.

Iteration 3

$$S = \{1\}, X_S = (2), z^S = 2, b^S = (-3, -5, -3, -2, -2, -2, -1),$$

$$\bar{z} = 8, X^* = (3, 0, 1, 0, 1, 0, 3)$$

Step 1: Fathom by $\bar{z} \leq z^S$? No.

Step 2: Best possible completion feasible? No.

Step 3: Infeasible constraint? No.

Step 4: Bound redefinition possible? No.

Step 5: Solution of imbedded linear program.

Initial Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
s_2	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
ω_7	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	1	1	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0
μ_2	1	1	1	1	1	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	0	-1	0	-1	-1	-1	1	1	0	-1	0	1	0	0	0	1	0	-1	0	0
(-v)	-1	3	0	3	1	4	0	1	3	4	4	1	0	0	0	-1	0	3	0	3

Optimal Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
s_2	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
s_3	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	1	1	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0
μ_2	1	1	1	1	1	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	0	0	0	0	-1	0	1	1	0	0	0	1	0	-1	0	0	0	-1	0	1
(-v)	-1	2	0	2	1	3	0	1	3	3	4	1	4	1	0	0	0	3	0	2

Fail to fathom by surrogate constraint.

Step 6: Dual variables of imbedded linear program integral?

Yes; hence fathom.

Iteration 4

$$S = \{1\}, X_S = (1), z^S = 1, b^S = (-4, -5, -4, -3, -3, -3, -2),$$

$$\bar{z} = 7, X^* = (2, 0, 0, 0, 3, 0, 2)$$

Step 1: Fathom by $\bar{z} \leq z^S$? No.

Step 2: Best possible completion feasible? No.

Step 3: Infeasible constraint? No.

Step 4: Bound redefinition possible? No.

Step 5: Solution of imbedded linear program.

Initial Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
s_2	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
s_3	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	1	1	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0
μ_2	1	1	1	1	1	1	0	0	0	0	0	-1	0	0	0	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	0	0	0	0	-1	0	1	1	0	0	0	1	0	-1	0	0	0	-1	0	1
(-v)	-1	1	0	1	-1	2	0	1	3	3	4	2	4	0	0	0	0	2	0	3

Optimal Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
μ_4	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
s_3	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	0	1	0	0	0	-1	0	-1	1	0	-1	0	0	0	-1	0	1	0	0	0
μ_2	0	1	1	1	0	0	0	-1	1	0	0	-1	0	0	-1	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	1	0	0	0	0	1	1	2	-1	0	0	1	0	-1	1	0	0	-1	0	1
(-v)	0	1	0	1	0	3	0	2	2	3	4	2	4	0	1	0	0	2	0	3

Fathom by surrogate constraint.

Iteration 5

$$S = \{\underline{1}\}, X_S = (0), z^S = 0, b^S = (-5, -5, -5, -4, -4, -4, -3),$$

$$\bar{z} = 7, X^* = (2, 0, 0, 0, 3, 0, 2)$$

Step 1: Fathom by $\bar{z} \leq z^S$? No.

Step 2: Best possible completion feasible? No.

Step 3: Infeasible constraint? No.

Step 4: Bound redefinition possible? No.

Step 5: Solution of imbedded linear program.

Initial Tableau

	r.h.s.	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	s_2	s_3	s_4	s_5	s_6	s_7
μ_4	1	0	0	0	1	1	0	1	-1	0	0	0	0	0	1	0	0	0	0	0
s_3	0	-1	0	-1	0	-1	0	0	0	-1	0	0	0	1	0	1	0	0	0	-1
s_4	0	1	0	0	0	-1	0	-1	1	0	-1	0	0	0	-1	0	1	0	0	0
μ_2	0	1	1	1	0	0	0	-1	1	0	0	-1	0	0	-1	0	0	1	0	0
s_6	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0
μ_6	1	0	0	0	0	1	1	2	-1	0	0	1	0	-1	1	0	0	-1	0	1
(-v)	1	0	0	0	0	4	0	4	0	3	4	3	4	-1	3	0	0	1	0	4

Fathom by surrogate constraint.

Final solution: $X^* = (2, 0, 0, 0, 3, 0, 2), \bar{z} = 7.$

APPENDIX B

USER'S INSTRUCTIONS FOR ENUMER8

Input

The original integer programming problem (in the form of Equations (1)-(4) of Chapter II) and parametric information to control the enumeration are the only items required for input to ENUMER8. Information is input in the order indicated.

CONTROL CARD

The following items are right-justified in each of nine five-column fields; no decimal is required.

- M - number of problem constraints. (≤ 70 , including those surrogate constraints maintained (see NSC)).
- N - number of problem variables, excluding slacks (≤ 70).
- NS - number of fixed variables in the initial partial solution.
- NFQ - input k for utilization of the imbedded linear program at every kth opportunity.
- IO - regulates program output. =1 for output of a detailed solution history at each iteration; =0 for output only of incumbent (current best feasible) solutions as discovered plus output of a final solution history. (See *Output* Section of this Appendix.)
- IPCT - per cent error allowed in the final objective value ($0 \leq IPCT \leq 100$).
- NSC - input k for retention and use in simple fathoming tests of the k most recently generated surrogate constraints. ($M + NSC \leq 70$).
- IBD - option which controls use of the imbedded linear program in bound redefinition. =1 for bound redefinition; =0 otherwise.

IAUG - determines augmentation discipline. =0 for Balas' rule; =1 for a minimum-branch rule.

RIGHT-HAND SIDE CARD(S)

The right-hand side elements of the original (i.e., *not* the current right-hand side relative to an advanced solution if one is input) problem are input in constraint order; entries are placed in eight ten-column fields on as many cards as are required; a decimal point is required with each entry.

OBJECTIVE FUNCTION CARD(S)

The coefficients of the objective function are entered in eight ten-column fields on as many cards as are required; a decimal point is required with each entry.

UPPER BOUNDS CARD(S)

The elements of the original upper bounds vector are entered in eight ten-column fields on as many cards as necessary; a decimal point is required with each entry.

A-MATRIX CARD(S)

The coefficients of the A-matrix are entered in eight ten-column fields on as many cards as are required; these elements are input in consecutive row order (i.e., $a_{11}, a_{12}, \dots, a_{1n}, a_{21}, a_{22}, \dots, a_{2n} \dots a_{m1}, a_{m2}, \dots, a_{mn}$); a decimal point is required with each entry.

The remaining items, which identify an initial partial solution, are input, in the order indicated, if and only if NS \neq 0 on the CONTROL CARD.

VARIABLE TYPE CARD(S)

One entry for each variable, x_j , is input to indicate its status in the initial partial solution; items are input on as many cards as necessary, right-justified in 16 5-column fields with no decimal points included. One entry appears for each variable, regardless of whether that variable is free or fixed in the initial partial solution. The j th entry is:

0 if x_j is free.

1 if x_j is fixed and increasing in the initial partial solution.

-1 if x_j is fixed and decreasing in the initial partial solution.

INITIAL PARTIAL SOLUTION CARD(S)

One entry for each variable, x_j , in the initial partial solution is input; items are input on as many cards as necessary, right-justified in 16 5-column fields with no decimal points included. Exactly NS entries should appear (refer to CONTROL CARD). Input the index j if x_j is fixed in the initial partial solution.

UNDERLINED INDICES CARD(S)

One entry for each variable, x_j , is input to indicate whether its index is underlined in the initial partial solution; items are right-justified in 16 5-column fields with no decimal points included on as many cards as necessary. One entry appears for each variable, regardless of whether that variable is free or fixed initially. The j th entry is:

1 if x_j is fixed and j is underlined in the initial partial solution.

0, otherwise.

INITIAL SOLUTION VECTOR CARD(S)

One entry for each variable, x_j , is input to specify the initial solution vector; items are input with decimal point in eight ten-column fields on as many cards as necessary. One entry should appear for each x_j ; the j th entry should be:

0, if x_j is free initially.

The value at which x_j is fixed if x_j is fixed initially.

Output

Program output is regulated by the IO option in the input. If IO = 1 a detailed solution history will be printed out at each iteration of the algorithm; items output will include:

- (1) The current partial solution.
- (2) Indication of a successful means for fathoming, if any.
- (3) Indication of which variable is augmented, if any.
- (4) Column and row deletions and/or additions to the imbedded linear program.

In any event, incumbent solutions are output by ENUMER8 as they are discovered. In addition, final solution information is also output:

- (1) The number of iterations (i.e., number of explicit enumerations) required for the complete enumeration.
- (2) The iteration number on which the optimal solution was discovered.
- (3) The optimal solution.

- (4) Total number of admissible solutions.
- (5) The percentage of admissible solutions implicitly enumerated.
- (6) The number of times fathoming occurred by each of the various tests for fathoming.
- (7) The number of augmentations required.

Program Modification

ENUMER8 may be easily modified to handle larger problems, subject only to the core restrictions of the computer being used. Comment cards (refer to the program listing) at the beginning of the routines ENUMER8 and BOUND indicate those modifications necessary.

COMPUTER PROGRAM LISTING

ENUMER8

```

PROGRAM ENIMPR(INPUT,OUTPUT,TAPF60=INPUT,TAPF61=OUTPUT)
C
C*****TO ALTER PROGRAM CAPACITY, CHANGE ALL DIMENSIONED VARIABLES ACCORDINGLY
C*****EXCEPT NCH(50), ALSO CHANGE DO-LOOP INDEX IN 100-LOOP.
C
      DIMENSION A(70,70),RS(70),XSTAR(70),OJ(70),OJP(70),XLR(70),
*           C(70),IS1(70),IS2(70),X(70),IUL(70),
*           RTP(70),RINV(70,70),CJP1(70),
*           CJP2(70),CJP3(70),PCOL(70),TXX(70),TIJ(70),NCH(50)
      COMMON/STOR/NE,IND,ICTR8,TIJ,XLR,DJP,CJP3,RINV
C*****DEFINITION OF PROGRAM VARIABLES*****
C ITER --- ITERATION COUNT. ***
C ITOPT --- ITERATION COUNT OF DISCOVERY OF CURRENT OPTIMUM. ***
C M --- NUMBER OF ROWS IN ORIGINAL PROBLEM. ***
C N --- NUMBER OF COLUMNS IN ORIGINAL PROBLEM. ***
C NS --- NUMBER OF ELEMENTS IN THE SET S. (NO. FIXED VARIABLES) ***
C NFO --- ITERATION EPED. FOR SOLVING IMBEDDED I.L.P. ***
C JND --- SUBSCRIPT OF THE MOST RECENTLY FIXED VARIABLE. ***
C MCP --- POINTS TO A-MATRIX ROW SUBSCRIPT OF MOST RECENT SURROGATE ***
C           CONSTRAINT ADDED. ***
C ICTR1 --- NO. FATHOMS BY ZRAB LESS THAN OR EQUAL TO ZS. ***
C ICTR2 --- NO. OF FATHOMS BY FEASIBLE BEST POSSIBLE COMPLETION. ***
C ICTR3 --- NO. FATHOMS BY SINGLE CONSTRAINT INFEASIBILITY. ***
C ICTR4 --- NO. FATHOMS BY PERCENT ERROR FACTOR. ***
C ICTR5 --- NO. FATHOMS BY INTEGER DUALS TO I.L.P. ***
C ICTR6 --- NO. FATHOMS BY GENERATION OF INFEASIBLE SURROGATE CONSTRAINT. ***
C ICTR7 --- NO. AUGMENTATIONS NECESSARY. ***
C ICTR8 --- NO. FATHOMS BY ROUNDS INCONSISTENCIES. ***
C ZS --- CURRENT PARTIAL SOLUTION OBJECTIVE VALUE. ***
C ZRAB --- ORI. FON. VALUE OF BEST FEASIBLE SOLUTION FOUND CURRENTLY. ***
C IO --- =1 AN ITERATION HISTORY WILL BE PRINTED. ***
C           OTHERWISE --- ONLY FINAL SOLUTION INFORMATION IS PRINTED. ***
C IPCT --- PERCENT ERROR ALLOWABLE IN OPTIMAL SOLUTION. ***
C PCT --- SAME IN FLOATING POINT. ***
C NSC --- NO. OF SURROGATE CONSTRAINTS TO BE CARRIED IN ALGORITHM. ***
C TRD --- =1 I.L.P. WILL ALSO BE USED FOR ROUND REDEFINITION. ***
C           OTHERWISE --- I.L.P. USED TO DEVELOP S.C. ONLY. ***
C TAIG --- =1 AUGMENTATION RULE IS MINIMUM BRANCH. ***
C           =0 AUGMENT BY KALAS AUGMENTATION RULE. ***
C IND --- =1 FATHOMED IN ROUND REDEFINITION ROUTINE. ***
C           =0 OTHERWISE. ***
C X(J) --- I-TH VALUE IN CURRENT PARTIAL SOLUTION VECTOR. ***
C RS(I) --- I-TH ELEMENT OF R.H.S. OF CURRENT PARTIAL SOLUTION. ***
C C(J) --- J-TH ELEMENT OF ORIGINAL PROBLEM COST ROW. ***
C DJ(J) --- J-TH ELEMENT OF ORIGINAL UPPER BOUNDS VECTOR. ***
C DJP(J) --- J-TH ELEMENT OF CURRENT UPPER BOUNDS VECTOR. ***
C XLR(J) --- J-TH ELEMENT OF CURRENT LOWER BOUNDS VECTOR. ***
C A(I,J) --- I-TH ROW, J-TH COLUMN ELEMENT OF ORIGINAL PROBLEM MATRIX. ***
C IS1(J) --- =0 J-TH VARIABLE CURRENTLY FREE. ***
C           =+1 J-TH VARIABLE CURRENTLY FIXED AND INCREASING. ***
C           =-1 J-TH VARIABLE CURRENTLY FIXED AND DECREASING. ***
C IS2(I) --- SUBSCRIPT OF I-TH ELEMENT TO ENTER THE SET S. ***
C IUL(I) --- =0 CURRENT I-TH ELEMENT OF THE SET S IS NOT UNDERLINED. ***
C           =1 CURRENT I-TH ELEMENT OF THE SET S IS UNDERLINED. ***
C MI --- CURRENT NO. ROWS (INCL. SURROGATE CONSTRAINTS) IN PROBLEM. ***
C NE --- CURRENT NO. OF FREE VARIABLES (I.E., NO. ROWS IN I.L.P.). ***
C ZGAP --- CURRENT ZRAB MINUS ZS. ***
C XSTAR(J) --- J-TH ELEMENT OF CURRENT BEST FEASIBLE SOLUTION. ***
C IFO --- COUNTS NO. ITERATIONS ON WHICH ENTRY TO I.L.P. WAS POSSIBLE. ***
C NEP --- ROW SIZE OF I.L.P. ON LAST EXIT FROM I.L.P. ***
C NC --- COUNTS THE NO. OF ADDITIONS AND DELETIONS SINCE LAST ***

```

```

C          EXIT FROM I.L.P. ***
C NCH(I)   --- SUBSCRIPT OF I-TH VARIABLE FIXED OR FREED SINCE LAST ***
C          I.L.P. EXIT. ***
C IMR      --- =1 ON INITIAL ENTRY TO I.L.P. ***
C          =2 WHEN ENTRY TO I.L.P. ONLY INVOLVES OBJ. FCN. CHANGE. ***
C          =3 WHEN ENTRY TO I.L.P. REQUIRES DELETION AND/OR ***
C          ADDITION OF NEW PROGRAM VARIABLES. ***
C ZNEG     --- VALUE OF OBJ. FCN. OF I.L.P. FROM MOST RECENT ITERATION. ***
C ITPCT    --- NO. I.P. ITERATIONS ON CURRENT ENTRY TO I.L.P. ***
C RIP(I)   --- I-TH ELEMENT OF MOST RECENT R.H.S. OF I.L.P. ***
C RINV(I,J)--- I-TH ROW, J-TH COL. ELEMENT OF BASIS INVERSE MATRIX OF ***
C          MOST RECENT I.L.P. ITERATION. ***
C CJP1(J)  --- J-TH ELEMENT OF COST COEFFICIENTS FOR MULTIPLIER ACTIVITIES ***
C          IN MOST RECENT TABLEAU OF I.L.P. ***
C CJP2(J)  --- SAME FOR BOUNDS ACTIVITIES OF I.L.P. ***
C CJP3(J)  --- SAME FOR SLACK ACTIVITIES (I.E., DUAL VARIABLES) OF I.L.P. ***
C PCOL(I)  --- I-TH ELEMENT OF CURRENT PIVOT COL. IN I.L.P. ***
C IXX(K)   --- K-TH BASIC VARIABLE IN I.L.P. ***
C          MULTIPLIER ACTIVITIES IN RANGE 1-300. ***
C          BOUND ACTIVITIES IN RANGE 301-600. ***
C          SLACK ACTIVITIES IN RANGE 601-900. ***
C IJ(K)    --- SUBSCRIPT OF FREE VARIABLE TO WHICH K-TH SLACK ACTIVITY ***
C          OF I.L.P. CORRESPONDS (ALSO GIVES BOUND ACTIVITY ***
C          CORRESPONDENCE). ***
C JP       --- SUBSCRIPT OF PIVOT COL. IN CURRENT I.L.P. ITERATION. ***
C IP       --- SUBSCRIPT OF PIVOT ROW IN CURRENT I.L.P. ITERATION. ***
C PIVOT    --- RINV(IP,JP). ***
C ZLRD     --- LOWER BOUND ON FEASIBLE COMPLETIONS TO CURRENT ***
C          PARTIAL SOLUTION. ***
C*****
C
C*****PART I --- STORAGE INITIALIZATION.
C
C          WRITE(61,4001)
4001  FORMAT(1H1)
C          IMR=1
C          ITPR=0
C          JND=0
C          NC=0
C          IF0=0
C          NCP=0
C          ICTR1=0
C          ICTR2=0
C          ICTR3=0
C          ICTR4=0
C          ICTR5=0
C          ICTR6=0
C          ICTR7=0
C          ICTR8=0
C          ITOPT=0
C          ZS=0.
C          ZRAP=0.
C          DO 100 I=1,70
C          XLR(I)=0.
100   X(I)=0.
C
C*****PART II --- DATA INPUT.
C
C          READ(60,9000)M,N,NS,NEQ,IO,IPCT,NSC,TRD,IAUG
C          READ(60,9001)(RS(I),I=1,M)
C          READ(60,9002)(C(I),I=1,N)
C          READ(60,9003)(D(I),I=1,N)
C          READ(60,9004)(A(I,J),J=1,N),I=1,M)
C          IF(NS,FO,0)GO TO 255
C          READ(60,9005)(IS1(I),I=1,N)
C          READ(60,9006)(IS2(I),I=1,NS)

```

```

      READ(60,9000) (TJ) (T),T=1,N)
      READ(60,9001) (X(I),I=1,N)
      JND=IS2(NS)
C
C*****PART III --- SOLUTION INITIALIZATION.
C
      DO 250 I=1,NS
      K=IS2(J)
      DO 240 I=1,M
240    RS(I)=RS(I)+A(I,K)*X(K)
250    ZS=ZS+C(K)*X(K)
255    DO 260 I=1,N
      ZRAP=ZRAP+D(I)*C(I)
260    DJP(I)=D(I)
      PCT=IPCT
      M]=M
      NSC=M+NSC
      NCP=M
300    ITPR=ITPR+1
      NF=N-NS
      ZGAP=ZRAP-ZS
C
C*****PART IV --- SIMPLE FATHOMING TEST.
C
      IF (I0,NF,1) GO TO 305
      WRITE(61,6050) ITPR,ZRAP,ZS,(IS)(I),I=1,N)
      WRITE(61,6052) (X(I),I=1,N)
305    IF (ZRAP,GT,ZS) GO TO 400
      ICTP=ICTP+1
      IF (I0,F0,1) WRITE(61,6020)
      IF (IS)(JND),F0,1) GO TO 350
      IF (C(JND),F0,0.) GO TO 350
      DEL=AINT(-ZGAP/C(JND))
      IF (DEL,LE,0.) GO TO 1000
      X(JND)=X(JND)-DEL
      DO 310 I=1,M]
310    RS(I)=RS(I)-DEL*A(I,JND)
      ZS=ZS-DEL*C(JND)
      IF (X(JND),GT,0.) GO TO 1000
350    IJ(NS)=1
      GO TO 1000
C
C*****PART V --- FATHOMING TEST FOR FEASIBILITY OF BEST POSSIBLE COMPLETION.
C
400    DO 410 I=1,M
      IF (RS(I),LT,0.) GO TO 500
410    CONTINUE
      ICTP2=ICTP2+1
      ITOPT=ITPR
      IF (I0,F0,1) WRITE(61,6021)
      DO 420 I=1,N
420    XSTAR(I)=X(I)
      ZRAP=ZS
      WRITE(61,4000) ITPR,ZRAP,(XSTAR(I),I=1,N)
4000   FORMAT(6H ITPR=,16,5X,5H ZRAP=,F10,2,6HXSTAR=/(1X,20F6,1))
      IF (I0,F0,1) WRITE(61,5011) ZRAP,(XSTAR(I),I=1,N)
      IF (IS)(JND),F0,1) GO TO 350
      IF (X(JND),LE,1.) GO TO 1000
      DEL=X(JND)
      DO 430 IJ=1,M
      IF (A(IJ,JND),LE,0.) GO TO 430
      DT=AINT(BS(IJ)/A(IJ,JND))
      IF (DT,GE,DEL) GO TO 430
      DEL=DT
      IF (DEL,F0,0.) GO TO 1000
430    CONTINUE

```

```

X(JND)=X(JND)-DFI
DO 440 I=1,M]
440 RS(I)=RS(I)-DFI*A(I,JND)
    7S=7S-DFI*C(JND)
    XSTAR(JND)=X(JND)
    7RAP=7S
    WRITE(6,4000)ITER,7RAP,(XSTAR(I),I=1,N)
    IF(10.F0.)WRITE(6,5011)7RAP,(XSTAR(I),I=1,N)
    IF(X(JND).LE.0.)TUJ(NS)=1
    GO TO 1000
C
C*****PART VI --- REDEFINITION OF UPPER BOUNDS.
C
500 DO 550 I=1,N
    IF(1S1(I).NE.0)GO TO 550
    IF(C(I).EQ.0.)GO TO 550
    DJTEMP=AJNT(7GAP/C(I)-1).F-7)
    IF(DJTEMP.I.T.DJP(I))DJP(I)=DJTEMP
550 CONTINUE
C
C*****PART VII --- TEST FOR FATHOMING BY CONSTRAINT INFEASIBILITY.
C
DO 680 I=1,M]
IF(RS(I).GE.0.)GO TO 680
VAL=RS(I)
DO 610 J=1,N
IF(1S1(J).NE.0)GO TO 610
IF(A(I,J).LE.0.)GO TO 610
VAL=VAL+A(I,J)*DJP(J)
610 CONTINUE
IF(VAL.GE.0.)GO TO 680
ICTR3=ICTR3+1
IF(10.F0.)WRITE(6,6051)I
GO TO 1000
680 CONTINUE
C
C*****PART VIII --- SOLUTION OF IMBEDDED LINEAR PROGRAM.
C
IF(NF.GT.0)GO TO 699
WRITE(6,5017)
STOP
699 IFO=IFO+1
IF((1FO/NEQ)*NEQ.NE.JEQ)GO TO 900
GO TO (700,750,800).JMR
700 7NEG=-7GAP
C
C*****JMR = 1 --- INITIALIZE AND SOLVE THE IMBEDDED LINEAR PROGRAM.
C
IJLPC1=0
K=0
DO 703 I=1,N
IF(1S1(I).NE.0)GO TO 703
K=K+1
IJ(J(K))=I
RIP(K)=C(I)
CJP2(K)=DJP(I)
CJP3(K)=0.
703 CONTINUE
NF]=NF-1
DO 705 I=1,NF]
K=I+1
DO 704 J=K,NF
RINV(I,J)=0.
704 RINV(J,I)=0.
IXX(I)=600+IJ(J(I))
705 RINV(I,I)=1.

```

```

      IXX(NF)=600+ITJ(NF)
      RTMV(NF,NF)=1.
      DO 706 I=1,M
706   C(JP1(I))=RS(I)
707   JP=0
      ILPCT=ILPCT+1
      IF(I0,FO,1)WRITE(6,5004)ILPCT,7NEG
      IC=1
      CC=-1,F-5
      DO 708 I=1,M
      IF(C(JP1(I),GF,CC)GO TO 708
      JP=1
      CC=C(JP1(I)
708  CONTINUE
      DO 710 I=1,NF
      IF(C(JP2(I),GF,CC)GO TO 709
      IC=2
      JP=1
      CC=C(JP2(I)
709  CONTINUE
      IF(C(JP3(I),GF,CC)GO TO 710
      IC=3
      JP=1
      CC=C(JP3(I)
710  CONTINUE
      IF(JP,NF,0)GO TO 720
      IF(I0,FO,1)WRITE(6,5000)
      IF(NSC,FO,M)GO TO 714
      NCP=NCP+1
      IF(NCP,GT,NSC)NCP=M+1
      IF(M1,|T,NSC)M1=NCP
      RS(NCP)=ZGAP-1,F-7
      DO 711 I=1,NF
      JJ=IXX(I)
      IF(JJ,GT,300)GO TO 711
      RS(NCP)=RS(NCP)+RTP(I)*RS(JJ)
711  CONTINUE
      DO 713 I=1,N
      A(NCP,I)=-C(I)
      DO 712 J=1,NF
      JJ=IXX(J)
      IF(JJ,GT,300)GO TO 712
      A(NCP,I)=A(NCP,I)+RTP(J)*A(JJ,I)
712  CONTINUE
713  CONTINUE
714  ZLRD=7NEG+7RAD
      NFP=NF
      NC=0
C
C*****TEST FOR INTEGER DUAL VARIABLES FOR THE IMBEDDED I. P.
C
      DO 716 I=1,NF
      CC=ABS(C(JP3(I))-AIMT(C(JP3(I))))
      IF(CC,LT,.0001)GO TO 716
      IF(CC,GT,.9999)GO TO 716
      IMR=2
      IF(-100.*7NEG,|F,7LRD*PCT)GO TO 715
      IF(ITD,GT,0)CALL ROUND
      IF(ITD,FO,0)GO TO 900
      IF(I0,FO,1)WRITE(6,5014)
      GO TO 1000
715  ICTP4=ICTP4+1
      IF(I0,FO,1)WRITE(6,5012)
      GO TO 1000
716  CONTINUE
      ICTP5=ICTP5+1
      ITOPT=ITFP

```

```

IF (IO.EQ.1) WRITE (6,5005) (C(JP2(I),I=1,N)
7RAD=0.
DO 717 I=1,NE
  J=I,J(I)
717 XSTAR(I)=6INT(C(JP2(I)+.5)
DO 718 I=1,N
  IF (TS(I),EQ.0) GO TO 718
  XSTAR(I)=Y(I)
718 7RAD=7RAD+C(I)*XSTAR(I)
  WRITE (6,4000) I,7RAD,(XSTAR(I),I=1,N)
  IF (IO.EQ.1) WRITE (6,5011) 7RAD,(XSTAR(I),I=1,N)
  IMR=2
  GO TO 1000
720 7COI=00
  IF (IO.EQ.2) GO TO 724
  IF (IO.EQ.3) GO TO 724
DO 721 I=1,NE
721 7COI(I)=0.
DO 722 I=1,NE
  K=I,I(I)
DO 723 I=1,IE
722 7COI(I)=7COI(I)+FINV(I,I)*A(I,I)*K)
723 CONTINUE
  JPP=JP
  GO TO 728
724 DO 725 I=1,IE
725 7COI(I)=RINV(I,IP)
  JPP=300+I(I,JP)
  GO TO 728
726 DO 727 I=1,NE
727 7COI(I)=RINV(I,IP)
  JPP=600+I(I,JP)
728 IP=0
  RATIO=1,EPF
DO 729 I=1,NE
  IF (7COI(I),LE.1,EP-5) GO TO 729
  RNUMBER(I)/7COI(I)
  IF (R-EP,GE,RATIO) GO TO 729
  RATIO=R-EP
  IP=I
729 CONTINUE
  IF (IP,GT,1) GO TO 724
C
C*****MINI 7 2 = (THE IN)TY) --- BACK TO 1.
C
IF (IO.EQ.1) WRITE (6,5001)
730 IMR=2
  NCF=
  NCF=NE
  ICF=ICF+1
  IF (NCF,GT,100) GO TO 1000
  NCF=ICF+1
  IF (NCF,GT,NCF) GO TO 1000
  IF (N1,GT,NCF) NCF=N1
  RS(NCF)=7300-1,EP-7
DO 731 I=1,NE
  JJ=I*Y(I)
  IF (I,GT,300) GO TO 731
  RS(NCF)=RS(NCF)+RIP(I)*RS(I)
731 CONTINUE
DO 732 I=1,NE
  A(NCF,I)=A(I)
DO 732 I=1,NE
  JJ=I*Y(I)
  IF (I,GT,300) GO TO 732
  A(NCF,I)=A(NCF,I)+RIP(I)*A(I,I)

```

```

732 CONTINUE
733 CONTINUE
GO TO 1000
734 PIVOT=PCOL(ITR)
ZNEG=ZNEG-ZCOL*PIVOT/PIVOT
IXY(ITR)=IPR
DO 735 I=1,NF
IF(I,FO,ITR)GO TO 733
RATIO=PCOL(I)/PIVOT
DO 737 I=1,NF
737 RINV(I,I)=RINV(I,I)-RATIO*INVTM(ITR,I)
RIP(I)=RIP(I)-RATIO*INVTM(ITR)
738 CJP2(I)=CJP2(I)-ZCOL*RINV(ITR,I)/PIVOT
DO 739 I=1,NF
739 RINV(ITR,I)=RINV(ITR,I)/PIVOT
RIP(ITR)=RIP(ITR)/PIVOT
740 DO 741 I=1,I
741 CJP1(I)=RS(I)
DO 742 I=1,NF
K=I+I(I)
DO 742 I=1,NF
742 CJP1(I)=CJP1(I)+CJP2(I)*R(I,K)
743 CONTINUE
DO 744 I=1,NF
J=I+I(I)
744 CJP2(I)=CJP2(I)-CJP1(I)
IF(ZNEG,IXY,=1,5-IXY)GO TO 707
IF(ITO,FO,1)WRITE(61,5002)
GO TO 730
750 CONTINUE
C
C*****IMR2 --- DELETION TO I,J,P. -- ONLY ONE, FOL, HAS BEEN ALTERED.
C
IF(ITO,FO,1)WRITE(61,5003)
IIPCT=0
DO 751 I=1,NF
751 CJP2(I)=0
DO 757 I=1,NF
IF(IXY(I),GT,200)GO TO 750
J=IXY(I)
CC=RS(I)
GO TO 754
752 IF(IXY(I),GT,600)GO TO 757
I=IXY(I)-200
CC=DIP(I)
754 DO 755 J=1,NF
755 CJP2(I)=CJP2(I)-CC*RINV(I,J)
757 CONTINUE
ZNEG=-ZCOL
DO 760 I=1,NF
K=I+I(I)
760 ZNEG=ZNEG+CJP2(I)*C(K)
GO TO 740
C
C*****IMR3 --- DELETION TO I,J,P. -- AUGMENTATION OR DELETION REQUIRED.
C
800 IF(NC,GT,50,OR,(IFC/200)*200,FO,IFC)GO TO 700
IF(ITO,FO,1)WRITE(61,5004)
DO 805 K=1,NC
IF(NCH(K),FO,0)GO TO 805
IF(K,FO,NC)GO TO 802
DO 801 J=K,NC
IF(NCH(K)+NCH(J),NE,0)GO TO 801
NCH(K)=0
NCH(J)=0
GO TO 805

```

```

R01  CONTINUE
R02  IF (NCH(K),LT,0) GO TO R24
C
C*****PERFORM NECESSARY DELETIONS.
C
      J=NCH(K)
      DO R03 I=1,NFR
      I=I,I(I)
      IF (J,EQ,0) GO TO R04
R03  CONTINUE
      WRITE (61,5003) I
      WRITE (61,5010)
      GO TO 700
R04  IOR=600+I
      IORC=300+I
      ID=I
      DO R05 J=1,NFR
      IF (IXX(I),NE, IOR) GO TO R07
      ID=J
      GO TO R17
R07  IF (IXX(I),NE, IOR) GO TO R09
      ID=J
      ZIFC=ZIFC+RIP(ID)*CJPR(I)
      GO TO R17
R09  CONTINUE
      IFLAG=I
R09  ZCOL=CJPR(ID)
      IS=0
      RATIO=1,FRY
      DO R10 J=1,NFR
      IF (RINV(I, ID),LE,1,FRY) GO TO R11
      RHE=RID(I)/RINV(I, ID)
      IF (RHE,GE, RATIO) GO TO R11
      RATIO=RHE
      ID=J
R10  CONTINUE
      IF (ID,NE,0) GO TO R13
      IFLAG=IFLAG+1
      IF (IFLAG,EQ,2) GO TO R12
      DO R11 I=1,NFR
      RINV(I, ID)=RINV(I, ID)
      CJPR(ID)=CJPR(I)-CJPR(ID)
      GO TO R09
R12  WRITE (61,5010)
      GO TO 700
R13  RIVOT=RINV(ID, ID)
      ZIFC=ZIFC-ZCOL*RIVOT(ID)/RIVOT
      DO R14 I=1,NFR
      IF (I,EQ, ID) GO TO R15
      RATIO=RINV(I, ID)/RIVOT
      DO R14 J=1,NFR
      RINV(I, J)=RINV(I, J)-RATIO*(RIVOT-1)
      RIP(I)=RIP(I)-RATIO*RIP(ID)
R15  CJPR(I)=CJPR(I)-ZCOL*RINV(ID, I)/RIVOT
      DO R14 I=1,NFR
      RINV(ID, I)=RINV(ID, I)/RIVOT
      RIP(ID)=RIP(ID)/RIVOT
R17  IF (ID,EQ, NFR) GO TO R20
      IF (ID)EQ(I) NFR)
      CJPR(ID)=CJPR(NFR)
      DO R18 J=1,NFR
R18  RINV(I, ID)=RINV(I, NFR)
R20  IF (ID,EQ, NFR) GO TO R22
      IXX(ID)=IXX(NFR)
      RIP(ID)=RIP(NFR)
      DO R21 J=1,NFR

```

```

R21 RTNV(IP,J)=RTNV(MEP,J)
R22 NEP=MEP-1
    IF(TO,FO,1)WRITE(A,5007) JJ
    IF(MEP,NE,1)GO TO R25
    WRITE(A,5018)
    GO TO 730
R25 CONTINUE
    DO R50 I=1,NC
    IF(MCH(I),GE,0)GO TO R50
C
C*****PERFORM NECESSARY ADDITIONS.
C
    JJ=-MCH(I)
    K=MEP+1
    RTNV(K,K)=1.
    DO R26 J=1,NEP
    RTNV(K,J)=0.
R26 RTNV(J,K)=0.
    DO R27 I=1,MEP
    IF(IYX(I),GT,300)GO TO R27
    I=IYX(I)
    CC=AI(I)
    DO R28 J=1,MEP
R28 RTNV(K,J)=RTNV(K,J)-CC*RTNV(I,J)
R29 CONTINUE
    CJP2(K)=0.
    RTD(K)=C(I)
    DO R34 J=1,NEP
    I=I(I,J)
R34 RTD(K)=RTD(K)+C(I)*RTNV(K,I)
    TI(K)=JJ
    IYX(K)=600+JJ
    IF(RTD(K),GE,0.)GO TO R40
    IYX(K)=300+JJ
    RTD(K)=-RTD(K)
    ZNEG=7956-RTD(K)*CJP(I)
    DO R36 J=1,K
    RTNV(K,J)=RTNV(K,J)
R36 CJP2(I)=CJP2(I)+RTNV(K,I)*CJP(I)
R40 NEP=K
    IF(TO,FO,1)WRITE(A,5008) JJ
R50 CONTINUE
    TIGOT=0
    GO TO 730
C
C*****PART IX --- AUGMENTATION.
C
R900 IF(TANG,FO,0)GO TO R932
    ITEST=0
    DO R930 I=1,N
    IF(TSI(I),NE,0)GO TO R930
    IF(ITEST,FO,0)JND=I
    ITEST=1
    IF(DIP(I)-X(I),GE,0)IF(JND)-X(JND))GO TO R930
    JND=I
R930 CONTINUE
    X(JND)=X(JND)
    TSI(JND)=1
    GO TO R932
R932 VAL=-1.E20
    DO R935 I=1,N
    IF(TSI(I),NE,0)GO TO R935
    CC=0.
    DO R933 J=1,N
    C)TEMP=RS(J)+CJP(I)*S(J,I)
    IF(C)TEMP,(T,I)CC=CC+D)TEMP

```

```

033  CONTINUE
     IF (CC,IF,VAL) GO TO 035
     VAL=CC
     JND=J
035  CONTINUE
     X(JND)=DJP(JND)
     IS1(JND)=1
038  IF (IC,FO,1) WRITE (6,6025) JND
     ICTR7=ICTR7+1
     NS=NC+1
     IS2(NS)=JND
     ZS=ZS+C(JND)*X(JND)
     DO 040 I=1,M1
040  RS(I)=RS(I)+X(JND)*A(I,JND)
     TH(NS)=0
     IF (DJP(JND),FO,XI(JND)) TH(NS)=1
     IF (IMR,ME,1) IMR=3
     NC=NC+1
     IF (NC,LT,50) NCH(NC)=JND
     GO TO 300
C
C*****PART X --- BACKTRACK.
C
1000 IF (NS,FO,0) GO TO 1006
1005 IF (TH(NS),ME,1) GO TO 1010
     IF (IMR,ME,1) IMR=2
     NC=NC+1
     IF (NC,LT,50) NCH(NC)=JND
     ZS=ZS-C(JND)*X(JND)
     DO 1007 I=1,M1
1007 RS(I)=RS(I)-X(JND)*A(I,JND)
     X(JND)=0.
1009 NS=NS-1
     IS1(JND)=0
     JND=IS2(NS)
     IF (NS,FO,0) GO TO 1000
     GO TO 1006
1010 DO 1015 I=1,N
     IF (IS1(I),ME,0) GO TO 1015
     XI R(I)=0.
     DJP(I)=0.111
1015 CONTINUE
     IF (IS1(JND),FO,1) GO TO 1050
     X(JND)=Y(JND)+1.
     DO 1020 I=1,M1
1020 RS(I)=RS(I)+A(I,JND)
     ZS=ZS+C(JND)
     IF (X(JND),GT,XI R(JND)) GO TO 300
     TH(NS)=1
     GO TO 300
1050 Y(JND)=Y(JND)+1.
     DO 1060 I=1,M1
1060 RS(I)=RS(I)+A(I,JND)
     ZS=ZS+C(JND)
     IF (X(JND),LT,DJP(JND)) GO TO 300
     TH(NS)=1
     GO TO 300
C
C*****PART XT --- TERMINATION OF FOUR-BEAT] IN PROCESS.
C
2000 IF (ICTR2+ICTR3,ME,0) GO TO 2005
     WRITE (6,5017)
     GO TO 2007
2005 WRITE (6,8000) (XSTAR(I),I=1,N)
     WRITE (6,8001) ZPAR,ITEP
2007 CC=1.

```

```

      DO 2010 I=1,N
2010  CC=CC*(DJ(I)+1.)
      VAL=ITFR
      PCT=(CC-VAL)*100./CC
      WRITE(61,5050)CC,VAL,PCT,ICTR1,ICTR2,ICTR3,ICTR4,ICTR5,ICTR6,
*          ICTR7,ICTR7,ITORT
      STOP
5000  FORMAT(41H OPTIMAL SOLUTION OF IMPROVED L. P. FOUND)
5001  FORMAT(16H MIN Z = -(1E.1))
5002  FORMAT(12H FATHOMED BY SURROGATE CONSTRAINT.)
5003  FORMAT(45H ENTERED OBJ. FCN. UPDATE PORTION OF T. I. P.)
5004  FORMAT(20H ITP ITERATION NO. =.14,5X,7H ZNEG =.E20,8)
5005  FORMAT(24H INTEGER EQUALS TO T.I.P./(1X,22F6.2))
5006  FORMAT(28H ENTERED L.P. MODIFICATION)
5007  FORMAT(17H DELETED VARIABLE,15)
5008  FORMAT(16H ADDED VARIABLE,15)
5009  FORMAT(24H UNABLE TO FIND VARIABLE,14,32H FOR DELETION -- RE-INIT
*TA(17F.1)
5010  FORMAT(58H UNABLE TO RE-ENTER T.I.P. SUCCESSFULLY --- RE-INITIALIZ
*F.)
5011  FORMAT(41H CURRENT BEST FEASIBLE SOLUTION --- ZRAB=.F10,3/
*      5X,2HY=.12F10,3/(7X,12F10,3))
5012  FORMAT(48H FATHOMED BY PERCENT POSSIBLE IMPROVEMENT FACTOR)
5013  FORMAT(29H NO FEASIBLE SOLUTION EXISTS.)
5014  FORMAT(35H FATHOMED BY BOUND INCONSISTENCIES.)
5017  FORMAT(70H*****T.I.P. END PROGRAM --- ATTEMPT TO ENTER T.I.P
* WITH NE=0000)
5019  FORMAT(54H*****CURRENT PROGRAM FLOOD --- ATTEMPT TO GET NE=0000)
5050  FORMAT(///36H ADDITIONAL SOLUTION INFORMATION ---/
* 7X,21HTOTAL NUMBER OF NODES,22X,F15,8/
* 7X,37HNUMBER OF NODES EXPLICITLY ENUMERATED,4X,F15,8/
* 7X,41HPERCENTAGE OF NODES IMPLICITLY ENUMERATED,2X,F15,8//
* 7X,27HNUMBER OF TIMES FATHOMED BY/
* 10X,17H7RAB LESS THAN ZS,33X,110/
* 10X,30HEFFASIBILITY OF BEST POSSIBLE COMPLETION,11X,110/
* 10X,26HAM INFEASIBLE CONSTRAINT,26X,110/
* 10X,29HPERCENTAGE OF IMPROVEMENT FACTOR,21X,110/
* 10X,36HATTACHED VALUE OF THE IMPROVED L.P.,16X,110/
* 10X,34HIMPROVED L.P. SURROGATE CONSTRAINT,16X,110/
* 10X,36HUPPER AND LOWER BOUND INCONSISTENCIES,14X,110/
* 7X,32HNUMBER OF AUGMENTATIONS REQUIRED,21X,110/
* 7X,43HITERATION NUMBER ON WHICH OPTIMUM WAS FOUND,10X,110)
6020  FORMAT(25H FATHOMED BY ZRAB,1E.75)
6021  FORMAT(44H FATHOMED BY FEAS. BEST POSSIBLE COMPLETION.)
6025  FORMAT(38H AUGMENTATION STEP --- VARIABLE FIXED,15)
6050  FORMAT(110H ITERATION,15,3X,5H7RAB=.F15,8,3X,3H7CS=.F15,8/5X,4HIS1=.
*3X,30I3/(12X,30I3))
6051  FORMAT(38H FATHOMED BY FEASIBLE CONSTRAINT NO.,15)
6052  FORMAT(5X,2HY=.3X,12F10,3/(10X,12F10,3))
8070  FORMAT(21H1OPTIMAL SOLUTION ---/(25X,E20,8))
8101  FORMAT(28H0OPTIMAL VALUE OF OBJECTIVE=.E20,8,5X,17HITERATION COUNT
* =.110)
9000  FORMAT(16I5)
9001  FORMAT(8E10,0)
      END
      SUBROUTINE BOUND
C
C*****BOUND-- ATTEMPTS TO REDEFINE UPPER AND LOWER BOUNDS ON FREE
C*****PROGRAM VARIABLES VIA INFORMATION IMMEDIATELY AVAILABLE FROM
C*****THE IMPROVED LINEAR PROGRAM.
C*****TO ALTER PROGRAM CAPACITY, CHANGE ALL DIMENSIONED VARIABLES ACCORDINGLY.
C
      COMMON/STOR/NE,IND,ICTR8,IT,I,YI,R,DJO,CJR3,RTM
      DIMENSION IT(70), JO(70),CJR2(70),RTM(70,70),XLR(70)
      IND=0
      DO 200 I=1,NE

```

```

      K=IT(I)
C
C*****ATTEMPT NEW LOWER BOUND DEFINITION.
C
      IF(CJPR(I).LE.X)R(K)+1,E-5)GO TO 100
      DO 50 J=1,NF
      IF(RINV(J,I).LT.-1,E-5)GO TO 100
50    CONTINUE
      X)R(K)=AINT(CJPR(I)+.79999)
      IF(X)R(K).LE.CJPR(K)+1,E-5)GO TO 100
      IND=1
      ICTPR=ICTPR+1
      RETURN
C
C*****ATTEMPT NEW UPPER BOUND DEFINITION.
C
100   IF(CJPR(I).GE.CJPR(K)-1,E-5)GO TO 200
      DO 150 J=1,NF
      IF(RINV(J,I).GT.1,E-5)GO TO 200
150   CONTINUE
      CJPR(K)=AINT(CJPR(I)+1,E-5)
      IF(X)R(K).LE.CJPR(K)+1,E-5)GO TO 200
      IND=1
      ICTPR=ICTPR+1
      RETURN
200   CONTINUE
      RETURN
      END

```

APPENDIX C

SAMPLE PROBLEMS

The following problems were solved using the computer code ENUMER8; computational results are given in Chapter V. All problems reproduced here are listed in a detached coefficient form of the following integer program in bounded variables: Minimize cx subject to $b + Ax \geq 0$ and $d \geq x \geq 0$ where c , x , and d are n -vectors, b is an m -vector, and A is an $m \times n$ matrix.

PROBLEM 1 --- HALDI, FIXED CHARGE 1.

R.H.S. DETACHED COEFFICIENT ARRAY

-16	-2	-3	1	2	2
-18	-3	-2	2	1	2
-6	6	0	1	0	0
-7	0	7	0	1	0

OBJ. FCN. 0 0 1 1 1

BOUNDS 1 1 6 7 7

OPTIMAL SOLUTIONS

1	1	5	3	5
0	0	6	7	0
1	1	5	1	7

OPTIMAL OBJECTIVE VALUE = 13.00

PROBLEM 2 --- HALDI, FIXED CHARGE 2.

R.H.S. DETACHED COEFFICIENT ARRAY

-20	-2	-3	1	2	2
-22	-3	-2	2	1	2
-8	9	0	1	0	0
-7	0	7	0	1	0

OBJ. FCN. 0 0 1 1 1

BOUNDS 1 1 8 7 8

OPTIMAL SOLUTIONS

1	1	5	3	7
1	1	4	3	8
0	0	8	7	0

OPTIMAL OBJECTIVE VALUE = 15.00

PROBLEM 3 --- HALDI, FIXED CHARGE 3.

R.H.S. DETACHED COEFFICIENT ARRAY

-26	-2	-3	1	2	2
-26	-3	-2	2	1	2
-9	9	0	1	0	0
-9	0	9	0	1	0

OBJ. FCN. 0 0 1 1 1

BOUNDS 1 1 9 9 10

OPTIMAL SOLUTIONS

1	1	5	3	10
0	0	9	9	0

OPTIMAL OBJECTIVE VALUE = 13.00

PROBLEM 4 --- HALDI, FIXED CHARGE 4.

R.H.S. DETACHED COEFFICIENT ARRAY

-17	-2	-3	1	2	2
-19	-3	-2	2	1	2
-6	6	0	1	0	0
-8	0	9	0	1	0

OBJ. FCN. 0 0 1 1 1

BOUNDS 1 1 6 8 7

OPTIMAL SOLUTIONS

1	1	4	2	7
0	1	6	5	2

OPTIMAL OBJECTIVE VALUE = 13.00

PROBLEM 5 --- HALDI, FIXED CHARGE 7.

R.H.S. DETACHED COEFFICIENT ARRAY

-180	-20	-30	1	2	2
-195	-30	-20	2	1	2
-60	60	0	1	0	0
-75	0	75	0	1	0

OBJ. FCN.	0	0	1	1	1
-----------	---	---	---	---	---

BOUNDS	1	1	60	75	75
--------	---	---	----	----	----

OPTIMAL SOLUTIONS

1	1	38	23	73
1	1	38	21	75
1	1	36	23	75

OPTIMAL OBJECTIVE VALUE = 134.00

PROBLEM 6 --- HALDI, FIXED CHARGE 8.

R.H.S. DETACHED COEFFICIENT ARRAY

-270	-20	-30	1	2	2
-270	-30	-20	2	1	2
-90	90	0	1	0	0
-90	0	90	0	1	0

OBJ. FCN.	0	0	1	1	1
-----------	---	---	---	---	---

BOUNDS	1	1	90	90	105
--------	---	---	----	----	-----

OPTIMAL SOLUTIONS

1	1	38	36	105
1	1	36	38	105

OPTIMAL OBJECTIVE VALUE = 179.00

PROBLEM 7 --- HALDI, FIXED CHARGE 9

R.H.S. DETACHED COEFFICIENT ARRAY

-6	-2	-2	0	1	1	0
-6	-2	0	-2	1	0	1
-6	0	-2	-2	0	1	1
-8	8	0	0	1	0	0
-8	0	8	0	0	1	0
-8	0	0	8	0	0	1

OBJ. FCN. 0 0 0 1 1 1

BOUNDS 1 1 1 8 8 8

OPTIMAL SOLUTIONS

1 1 1 5 5 5

OPTIMAL OBJECTIVE VALUE = 15.00

PROBLEM 8 --- HALDI, FIXED CHARGE 10.

R.H.S. DETACHED COEFFICIENT ARRAY

-288	-9	-7	-16	-8	-24	-5	3	7	8	4	6	5
-232	-12	-6	-6	-2	-20	-8	4	6	3	1	5	8
-213	-15	-5	-12	-4	-4	-5	5	5	6	2	1	5
-234	-18	-4	-4	-18	-28	-1	6	4	2	9	7	1
-12	12	0	0	0	0	0	1	0	0	0	0	0
-15	0	15	0	0	0	0	0	1	0	0	0	0
-12	0	0	12	0	0	0	0	0	1	0	0	0
-10	0	0	0	10	0	0	0	0	0	1	0	0
-11	0	0	0	0	11	0	0	0	0	0	1	0
-11	0	0	0	0	0	11	0	0	0	0	0	1

OBJ. FCN. 0 0 0 0 0 0 1 1 1 1 1 1

BOUNDS 1 1 1 1 1 1 12 15 12 10 11 11

OPTIMAL SOLUTIONS

0 0 0 1 0 1 12 15 12 2 11 2

OPTIMAL OBJECTIVE VALUE = 54.00

PROBLEM 9 --- IBM TEST 1.

R.H.S. DETACHED COEFFICIENT ARRAY

-5	1	0	0	1	1	0	1
-5	0	1	0	1	0	1	1
-5	0	0	1	0	1	1	1
-4	1	1	0	0	1	1	0
-4	1	0	1	1	0	1	0
-4	0	1	1	1	1	0	0
-3	1	1	1	0	0	0	1

OBJ. FCN.	1	1	1	1	1	1	1
-----------	---	---	---	---	---	---	---

BOUNDS	5	5	5	5	5	5	5
--------	---	---	---	---	---	---	---

OPTIMAL SOLUTIONS

1	0	0	2	2	1	2
0	0	1	1	2	2	2

OPTIMAL OBJECTIVE VALUE = 8.00

PROBLEM 10 --- IBM TEST 2.

R.H.S. DETACHED COEFFICIENT ARRAY

-4	1	0	0	1	1	0	1
-4	0	1	0	1	0	1	1
-4	0	0	1	0	1	1	1
-3	1	1	0	0	1	1	0
-3	1	0	1	1	0	1	0
-3	0	1	1	1	1	0	0
-2	1	1	1	0	0	0	1

OBJ. FCN.	1	1	1	1	1	1	1
-----------	---	---	---	---	---	---	---

BOUNDS	4	4	4	4	4	4	4
--------	---	---	---	---	---	---	---

OPTIMAL SOLUTIONS

1	0	0	2	2	1	1
1	1	0	1	2	2	0
0	0	0	1	2	2	2

OPTIMAL OBJECTIVE VALUE = 7.00

PROBLEM 11 --- IBM TEST 3.

R.H.S. DETACHED COEFFICIENT ARRAY

-96	4	5	3	6
-200	20	21	17	12
-101	11	12	12	7

OBJ. FCN.	13	15	14	11
-----------	----	----	----	----

BOUNDS	24	20	32	17
--------	----	----	----	----

OPTIMAL SOLUTIONS

0	0	0	17
---	---	---	----

OPTIMAL OBJECTIVE VALUE = 187.00

PROBLEM 21 --- DIET PROBLEM.

R.H.S. DETACHED COEFFICIENT ARRAY

-300	40	40	40	48	60	60	56	60	0	0	0	0	0	60
350	-40	-40	-40	-44	-60	-60	-56	-60	0	0	0	0	0	-60
-300	0	0	0	30	0	0	45	36	45	45	45	45	45	0
360	0	0	0	-90	0	0	-45	-36	-45	-45	-45	-45	-45	0
-110	0	0	0	32	8	8	20	16	0	28	24	0	28	8
145	0	0	0	-32	-8	-8	-20	-16	0	-28	-24	0	-28	-8
0	0	0	0	2	-1	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	-1	-1	-1	0	0
2	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	-1	0	0	0	0	1
0	0	0	0	0	0	-1	-1	-1	1	0	0	0	0	-1
-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

OBJ. FCN.	4.3	5.0	4.5	9.0	1.3	1.2	1.6	1.5	2.0	5.0	3.5	4.3	5.0	1.5
-----------	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

BOUNDS	2	1	2	4	3	2	2	2	4	3	3	3	3	5
--------	---	---	---	---	---	---	---	---	---	---	---	---	---	---

OPTIMAL SOLUTIONS

0	0	0	1	2	2	1	0	3	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

OPTIMAL OBJECTIVE VALUE = 26.60

PROBLEMS 14-15 --- ID: TEST 5.

S.P.S. DETACHED COEFFICIENT ARRAY

-6	1	0	0	0	1	1	1	0	0	0	0	1	0	1	0	1
-8	0	1	0	0	0	1	1	0	0	0	0	0	1	1	1	1
-8	0	0	1	0	0	0	1	0	0	0	0	0	1	1	1	1
-8	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1
-7	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
-7	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0
-7	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0
-7	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0
-6	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
-6	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1
-6	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1
-5	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	0
OBJ. FCN.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
BOUNDS*	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
OPTIMAL SOLUTIONS																
	1	0	1	0	0	2	2	3	3	0	2	2	2	0	2	
	1	0	1	0	2	1	1	1	1	0	1	1	1	1	1	2
	1	1	1	1	1	1	1	1	1	1	2	2	2	1	2	1
	0	0	0	0	2	2	1	1	1	1	2	2	0	1	2	1

OPTIMAL OBJECTIVE VALUE = 15.00

* Bounds given are for problem 14. For problem 15, all bounds are 3.

PROBLEM 18 --- IMA TEST 7.

OBJ. FCN.	TRANSPOSE OF DETACHED COEFFICIENT ARRAY												BOUNDS	OPTIMAL SOLUTION
20	46	0	0	0	0	0	0	0	0	0	0	0	16	0
25	68	0	0	0	0	0	0	0	0	0	0	0	11	0
30	91	0	0	0	0	0	0	0	0	0	0	0	8	0
38	137	0	0	0	0	0	0	0	0	0	0	0	6	0
46	182	0	0	0	0	0	0	0	0	0	0	0	4	0
61	273	0	0	0	0	0	0	0	0	0	0	0	3	0
74	364	0	0	0	0	0	0	0	0	0	0	0	2	0
101	545	0	0	0	0	0	0	0	0	0	0	0	2	0
137	818	0	0	0	0	0	0	0	0	0	0	0	1	0
15	0	46	0	0	0	0	0	0	0	0	0	0	11	0
17	0	68	0	0	0	0	0	0	0	0	0	0	8	0
21	0	91	0	0	0	0	0	0	0	0	0	0	6	0
29	0	137	0	0	0	0	0	0	0	0	0	0	4	0
31	0	182	0	0	0	0	0	0	0	0	0	0	3	0
41	0	273	0	0	0	0	0	0	0	0	0	0	2	0
50	0	364	0	0	0	0	0	0	0	0	0	0	2	0
67	0	545	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	46	0	0	0	0	0	0	0	0	0	6	0
11	0	0	68	0	0	0	0	0	0	0	0	0	4	0
14	0	0	91	0	0	0	0	0	0	0	0	0	3	0
19	0	0	137	0	0	0	0	0	0	0	0	0	2	0
20	0	0	182	0	0	0	0	0	0	0	0	0	2	0
27	0	0	273	0	0	0	0	0	0	0	0	0	1	0
15	0	0	0	46	0	0	0	0	0	0	0	0	7	0
18	0	0	0	68	0	0	0	0	0	0	0	0	4	0
22	0	0	0	91	0	0	0	0	0	0	0	0	3	0
26	0	0	0	137	0	0	0	0	0	0	0	0	2	0
37	0	0	0	182	0	0	0	0	0	0	0	0	1	0
41	0	0	0	273	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	46	0	0	0	0	0	0	0	8	0
11	0	0	0	0	68	0	0	0	0	0	0	0	4	0
14	0	0	0	0	91	0	0	0	0	0	0	0	3	0
16	0	0	0	0	137	0	0	0	0	0	0	0	2	0
20	0	0	0	0	182	0	0	0	0	0	0	0	2	0
25	0	0	0	0	273	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	46	0	0	0	0	0	0	2	0
10	0	0	0	0	0	68	0	0	0	0	0	0	2	0
12	0	0	0	0	0	91	0	0	0	0	0	0	1	0
5	20	0	0	0	0	0	-21	0	0	0	0	0	30	0
14	70	0	0	0	0	0	-85	0	0	0	0	0	7	3
6	20	20	0	0	0	0	0	-21	0	0	0	0	10	1
14	70	70	0	0	0	0	0	-85	0	0	0	0	2	2
5	20	20	20	0	0	0	0	0	-21	0	0	0	2	2
14	70	70	70	0	0	0	0	0	0	-85	0	0	0	0
6	20	20	20	20	0	0	0	0	0	-21	0	0	7	3
14	70	70	70	70	0	0	0	0	0	0	-85	0	1	1
5	20	20	20	20	20	0	0	0	0	0	-21	0	6	2
14	70	70	70	70	70	0	0	0	0	0	0	-85	1	1
6	20	20	20	20	20	20	0	0	0	0	0	-21	4	0
14	70	70	70	70	70	70	0	0	0	0	0	-85	1	1

---S. -717 -498 -200 -150 -162 -56 630 211 51 158 137 68

OPTIMAL OBJECTIVE VALUE = 152.00

PROBLEM 19 --- IBM TEST P.

OBJ. FCN.	TRANSPOSE OF DETACHED COEFFICIENT ARRAY												BOUNDS	OPTIMAL SOLUTION
1	1	1	1	1	1	1	0	0	0	0	0	0	1	0
0	-12	0	0	0	0	0	1	0	0	0	0	0	8	0
0	0	-12	0	0	0	0	1	0	0	0	0	0	8	8
0	0	0	-12	0	0	0	1	0	0	0	0	0	8	0
0	0	0	0	-12	0	0	1	0	0	0	0	0	8	0
0	0	0	0	0	-12	0	1	0	0	0	0	0	8	0
0	-14	0	0	0	0	0	0	1	0	0	0	0	7	0
0	0	-14	0	0	0	0	0	1	0	0	0	0	7	0
0	0	0	-14	0	0	0	0	1	0	0	0	0	7	7
0	0	0	0	-14	0	0	0	1	0	0	0	0	7	3
0	0	0	0	0	-14	0	0	1	0	0	0	0	7	0
0	0	0	0	0	0	-14	0	1	0	0	0	0	7	0
0	-17	0	0	0	0	0	0	0	1	0	0	0	6	0
0	0	-17	0	0	0	0	0	0	1	0	0	0	6	0
0	0	0	-17	0	0	0	0	0	1	0	0	0	6	0
0	0	0	0	-17	0	0	0	0	1	0	0	0	6	0
0	0	0	0	0	-17	0	0	0	1	0	0	0	6	5
0	-5	0	0	0	0	0	0	0	0	1	0	0	17	3
0	0	-5	0	0	0	0	0	0	0	1	0	0	17	0
0	0	0	-5	0	0	0	0	0	0	1	0	0	17	1
0	0	0	0	-5	0	0	0	0	0	1	0	0	17	0
0	0	0	0	0	-5	0	0	0	0	1	0	0	17	0
0	-20	0	0	0	0	0	0	0	0	0	1	0	5	3
0	0	-20	0	0	0	0	0	0	0	0	1	0	5	0
0	0	0	-20	0	0	0	0	0	0	0	1	0	5	0
0	0	0	0	-20	0	0	0	0	0	0	1	0	5	0
0	0	0	0	0	-20	0	0	0	0	0	1	0	5	3
0	-9	0	0	0	0	0	0	0	0	0	0	1	11	3
0	0	-9	0	0	0	0	0	0	0	0	0	1	11	1
0	0	0	-9	0	0	0	0	0	0	0	0	1	11	0
0	0	0	0	-9	0	0	0	0	0	0	0	1	11	7
0	0	0	0	0	-9	0	0	0	0	0	0	1	11	5
0	0	0	0	0	0	-9	0	0	0	0	0	1	11	2
R.H.S.	105	105	105	105	105	105	-8	-10	-5	-4	-6	-18		

OPTIMAL OBJECTIVE VALUE = 0.00

PROBLEM 20 --- IBM TEST 9.

R.H.S. DETACHED COEFFICIENT ARRAY

-1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
-1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
-1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
-1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
-1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0
-1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0
-1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
-1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0
-1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
-1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
-1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0
-1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
-1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
-1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0
-1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0
-1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0
-1	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
-1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0
-1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
-1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
-1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0
-1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0
-1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1
-1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
-1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0
-1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
-1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
-1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
-1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
-1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0
-1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0
-1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
-1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
-1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
-1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
-1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
OBJ. FCN.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
BOUNDS	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
OPTIMAL SOLUTIONS	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0
	1	1	1	0	1	0	1	1	1	0	0	1	1	0	0

OPTIMAL OBJECTIVE VALUE = 9.00

BIBLIOGRAPHY

- [1] Balas, Egon, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Operations Research*, Vol. 13, pp. 517-546 (1965).
- [2] Balas, Egon, "Discrete Programming by the Filter Method," *Operations Research*, Vol. 15 (1967), pp. 915-957.
- [3] Balinski, M. L., "Integer Programming: Methods, Uses, Computation," *Management Science*, Vol. 12 (1965), pp. 253-313.
- [4] Balinski, M. L., "On Recent Developments in Integer Programming," to appear in *Proceedings of the International Symposium on Mathematical Programming*, Princeton, N.J., August, 1967.
- [5] Dantzig, G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.
- [6] Eastman, W. L., "A Solution to the Traveling-Salesman Problem," presented at the *American Summer Meeting of the Econometric Society*, Cambridge, Massachusetts, Aug., 1958.
- [7] Geoffrion, Arthur M., "Integer Programming by Implicit Enumeration and Balas' Method," *SIAM Review*, Vol. 9 (1967), pp. 178-190.
- [8] Geoffrion, Arthur M., "Implicit Enumeration Using an Imbedded Linear Program," RAND Memorandum RM-5406-PR, September (1967), the RAND Corp., Santa Monica, California.
- [9] Geoffrion, Arthur M., and A. B. Nelson, "User's Instructions for 0-1 Integer Linear Programming Code RIP30C," RAND Memorandum RM-5627-PR, May (1968), The RAND Corp., Santa Monica, California.
- [10] Geoffrion, Arthur M., "An Improved Implicit Enumeration Approach for Integer Programming," RAND Memorandum RM-5644-PR, June (1968), The RAND Corp., Santa Monica, California.
- [11] Geoffrion, Arthur M., "An Improved Implicit Enumeration Approach for Integer Programming," *Operations Research*, Vol. 17 (1969), pp. 437-454.
- [12] Glover, Fred, "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," *Operations Research*, Vol. 13 (1965), pp. 879-919.

- [13] Glover, Fred, "A New Foundation for a Simplified Primal Integer Programming Algorithm," *Operations Research*, Vol. 16 (1968), pp. 727-740.
- [14] Glover, Fred, "Surrogate Constraints," *Operations Research*, Vol. 16 (1968), pp. 741-749.
- [15] Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs," in R. L. Graves and P. Wolfe, *Recent Advances in Mathematical Programming*, pp. 269-302, McGraw-Hill, New York, 1963.
- [16] Gomory, R. E., "An Algorithm for the Mixed Integer Problem," RAND Memorandum RM-2597, July (1970), The RAND Corporation, Santa Monica, California.
- [17] Gomory, R. E., "All-Integer Integer Programming Algorithm," IBM Research Center, Research Report RC-189, January (1960).
- [18] Gomory, R. E., "Faces of an Integer Polyhedron," *Proceedings of the National Academy of Sciences*, Vol. 57 (1967), pp. 16-18.
- [19] Haldi, J., "25 Integer Programming Test Problems," Stanford University, Working Paper No. 43, December (1964).
- [20] Land, A. H. and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, Vol. 28 (1960), pp. 497-520.
- [21] Lawler, E. L. and D. E. Wood, "Branch-and-Bound Methods: A Survey," *Operations Research*, Vol. 14 (1966), pp. 699-719.
- [22] Little, J. D. C. *et al.*, "An Algorithm for the Traveling Salesman Problem," *Operations Research*, Vol. 11 (1963), pp. 972-989.
- [23] Mangasarian, O. L., *Nonlinear Programming*, McGraw-Hill, New York, 1969.
- [24] Mitten, L. G., "Branch-and-Bound Methods: General Formulation and Properties," *Operations Research*, Vol. 18 (1970), pp. 24-34.
- [25] Woolsey, R. E. and C. A. Trauth, Jr., "IPSC, A Machine Independent Integer Linear Program," SANDIA Corporation Report SC-RR-66-433, July (1966).
- [26] Woolsey, R. E. and C. A. Trauth, Jr., "Integer Linear Programming: A Study in Computational Efficiency," *Management Science*, Vol. 15 (1969), pp. 481-493.

- [27] Young, Richard D., "A Simplified Primal (All-Integer) Integer Programming Algorithm," *Operations Research*, Vol. 16 (1968), pp. 750-782.
- [28] Zions, Stanley, "Toward a Unifying Theory for Integer Linear Programming," *Operations Research*, Vol. 17 (1969), pp. 359-367.