

AN ALGORITHM FOR THE SOLUTION OF THE
GENERAL SET-COVERING PROBLEM BY EUCLIDEAN MEANS

A THESIS

Presented to

The Faculty of the Division of Graduate Studies

by

Frank H. Cullen, Jr.

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in Operations Research

Georgia Institute of Technology

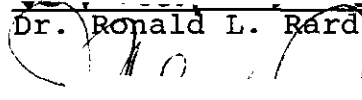
December, 1975

AN ALGORITHM FOR THE SOLUTION OF THE
GENERAL SET-COVERING PROBLEM BY EUCLIDEAN MEANS

Approved:



Dr. Ronald L. Bardin, Chairman



Dr. John J. Jarvis

Dr. V. Edinger

Date approved by Chairman: 12-15-75

ACKNOWLEDGMENTS

The author wishes to express his sincere appreciation to his thesis advisor, Dr. R. L. Rardin, whose patience, sympathy, and counsel are no doubt responsible for the completion of this work. The author also wishes to extend his thanks to the members of his reading committee, Dr. J. J. Jarvis and Dr. V. E. Unger, whose excellent criticisms and continued support throughout this and other matters has proved invaluable and is greatly appreciated.

I also want to thank Miss Jiovanna Cobb for her fine job on the typing and preparation of this thesis.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
LIST OF ILLUSTRATIONS	iv
SUMMARY	v
Chapter	
I. INTRODUCTION AND LITERATURE REVIEW	1
Formulation	
Reduction Properties	
Solution Approaches	
The Number-Theoretical Approach	
II. ADAPTATION OF RICHMOND'S ALGORITHM TO THE GENERAL SET-COVERING PROBLEM	16
Richmond's Algorithm	
Notion of a Fundamental Matrix	
Fundamental Matrix in Set-Covering	
Fourier-Motzkin Elimination Method	
Forward Variable Elimination	
Forward Variable Elimination in Set-Covering	
Backtracking for Integer Solutions	
Backtracking for the Set-Covering Problem	
The Adapted Algorithm as Implicit Enumeration	
III. CONCLUSIONS AND EXTENSIONS	46
BIBLIOGRAPHY.	49

LIST OF ILLUSTRATIONS

Figure	Page
2.1 A Side-by-side Comparison of the Algorithms	42

SUMMARY

The specialization of Richmond's algorithm to the general set-covering problem is accomplished in three parts in this thesis. The first part of Richmond's algorithm, which concerns the transformation of the original problem to canonical form with Rubin's Sequential Algorithm, is shown to degenerate to the original problem. The notion of a fundamental matrix, although quite interesting due to the structure which it exhibits, does not supply any previously-unknown information about problem structure for set-covering constraints.

The second part of Richmond's algorithm, which deals with the use of the classical Fourier-Motzkin elimination procedure to obtain bounds on each of the problem variables, is shown to degenerate primarily to the intuitive Forced Variables reduction property. The addition of some common sense rules to the direct results the Fourier-Motzkin forward elimination produce the final upper and lower bounding expressions used in the third step of Richmond's algorithm specialized to the general set-covering problem.

The third part of Richmond's algorithm, that of backtracking for integer solutions with the Cook and Cooper procedure, is shown to be closely related to the general method of implicit enumeration. Using the bounding expressions obtained from the second part, the specialized algorithm and

general implicit enumeration are compared side-by-side. The mechanics of the two procedures are shown to be virtually identical, differing only in the basic philosophies underlying them. The general implicit enumeration procedure bounds the optimal solution value with the values of successively better values of feasible solutions discovered during the expansion of partial solutions. In the context of the set-covering problem, these successively better values bound the value of the optimal solution from the top. The philosophy of the Richmond algorithm, and hence of the specialization of the algorithm to the general set-covering problem, is to approach the optimal solution value from the bottom, with the hope that the tighter bounds on the variables will cause faster fathoming to arrive more quickly at an optimal solution.

The combined impact of the above results is that the apparently quite unusual, number-theoretical approach of Richmond's algorithm degenerates in the set-covering case to little more than well-known implicit enumeration techniques. Thus, while not producing a new algorithm, the effort of this thesis to specialize the Richmond algorithm to the set-covering case produces a unification of two quite separate branches of integer programming research.

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

The general set-covering problem is a well-established problem in mathematical programming that has considerable application in such diverse areas as information retrieval, emergency facilities location, switching theory, political redistricting and apportionment, and scheduling problems.

Formulation

Intuitively, the general set-covering problem can be formulated as follows:

Given a set W with m elements e_i , a collection of n subsets S_j of W , and associated positive costs c_j for each of the subsets S_j ;
Find the minimal cost collection C^* of subsets S_j of W which "covers" the set W , so that every element e_i of W is in some subset S_j of C^* .

Mathematically, the above formulation can be written

as

$$\begin{aligned} & \min cx \\ & \text{subject to } Ax \geq \underline{1} \\ & 0 \leq x_j \leq 1 \quad j = 1, 2, \dots, n \\ & x_j \text{ integer} \quad j = 1, 2, \dots, n \end{aligned} \tag{1-1}$$

where $\underline{1}$ is a column m -vector of 1's, x is a column n -vector, each component x_j representing a decision whether subset S_j is a member of the object collection of subsets. At optimality, the indication of each x_j is

$$\text{if } x_j = \begin{cases} 0, & \text{then } S_j \notin C^* \\ 1, & \text{then } S_j \in C^* . \end{cases}$$

The n columns of the matrix A are associated with the subsets S_j , and the m rows of A are associated with the elements e_i of W . The elements a_{ij} of A are defined

$$a_{ij} = \begin{cases} 0 & \text{if } e_i \notin S_j \\ 1 & \text{if } e_i \in S_j . \end{cases}$$

The objective of the problem, as stated in the intuitive formulation, is to obtain the minimal cost collection of the subsets, this being merely the inner product of the vector x with c , a row n -vector consisting of the positive c_j 's; subject to the "covering constraints" embodied by the matrix inequality system, $Ax \geq \underline{1}$, and the 0,1 restrictions on the x_j .

A specialization of the general set-covering problem, the "set-partitioning" problem, is formulated as follows:

$$\begin{aligned} & \min cx \\ & \text{subject to } Ax = \underline{1} \\ & 0 \leq x_j \leq 1 \quad j = 1, 2, \dots, n \quad (1-2) \\ & x_j \text{ integer} \quad j = 1, 2, \dots, n . \end{aligned}$$

Thus this formulation differs from the general set-covering problem by stipulating that each element e_i in W can be an element of exactly one subset S_j in C^* .

Lemke, Salkin, and Spielberg [15] have demonstrated that every feasible set-partitioning problem can be solved by first transforming it into an equivalent set-covering formulation with a simple adjustment of the cost vector. The transformation consists of defining

$$t_j = \sum_{i=1}^m a_{ij}$$

and choosing any value L such that

$$L > \sum_{j=1}^n c_j$$

and defining new cost coefficients

$$c'_j = c_j + Lt_j$$

The optimal solution to formulation (1-2) (if any does indeed exist) is easily shown to be the same as the optimal solution to the set-covering problem

$$\min c'x$$

$$\text{subject to } Ax \geq \underline{1}$$

$$x_j = 0 \text{ or } 1 \text{ for all } j = 1, 2, \dots, n .$$

In the sense that the set-partitioning problem can be solved

as a set-covering problem, the set-covering problem is the more encompassing of the two formulations.

Reduction Properties

Because of the special structure of the general set-covering problem, there are certain properties which can be exploited so that a set-covering formulation may be more easily solved. Garfinkel and Nemhauser [11] discuss several properties which help characterize optimal solutions. Of prime importance among these are

1. Feasibility - each covering constraint must include at least one covering element. This is to say that every e_i must be contained in some S_j .
2. Forced variables - if there exists some e_i which is contained in exactly one S_j , that subset must be in any feasible cover, and hence can be eliminated from the problem (as a column) along with any covering constraint it may satisfy.
3. Row dominance - if there are two rows r_p and r_q of the constraint matrix such that $r_p \geq r_q$, which is to say that satisfaction of r_q automatically satisfies r_p , then r_p may be eliminated from the problem.
4. Column dominance - if there is some subcollection of subsets S such that this subcollection covers every constraint covered by some subset S_j and further that the total cost of that subcollection

is less than or equal to the cost of the subset S_j , then S_j (as a column) may be eliminated from the problem.

Another property of the general set-covering problem is that the upper bound of 1 on x_j in (1-1) can be relaxed without adverse effect on an optimal solution. Assume there existed an optimal solution $x^* = \{x_1^*, x_2^*, \dots, x_{j-1}^*, x_j^*, x_{j+1}^*, \dots, x_n^*\}$ such that $x_j^* > 1$. A solution $x' = \{x_1^*, x_2^*, \dots, x_{j-1}^*, 1, x_{j+1}^*, \dots, x_n^*\}$ yields a better objective function value as $cx^* - cx' = c_j(x_j^* - 1) > 0$ and so $cx^* > cx'$. Further, if x^* is feasible, then x' is also feasible, as the maximum value required of x_j to satisfy any covering constraint in which it appears is 1. Hence (1-1) can be re-written as

$$\begin{aligned} & \text{minimize } cx \\ & \text{subject to } Ax \geq \underline{1} \\ & x_j \geq 0 \text{ and integer for all } j = 1, 2, \dots, n . \end{aligned} \tag{1-3}$$

Solution Approaches

The published literature in the area of the general set-covering problem treats the problem in a wide variety of contexts and from quite a few different vantage points. Garfinkel and Nemhauser [11] present a fairly current survey of solution procedures for the general set-covering problem. These various methods generally break down into two classes: cutting plane algorithm and enumerative methods.

Cutting Planes and Involutory Bases

Principally, a cutting plane algorithm in integer programming is one in which an additional constraint (or cut) is applied to some present set of constraints (usually in an optimal simplex tableau) which eliminates part of the feasible region on the constraint set, but which does not exclude an integer value which may be optimal from the reduced feasible region. A cut is said to be "strong" if it eliminates the current optimal linear programming solution from the reduced feasible region, and "weak" otherwise.

An interesting cutting plane approach to the set-covering problem has been developed by Bellmore and Ratliff [2]. The essence of this approach is the notion that for the set-covering problem (1-1), the optimal solution (if the problem has a feasible solution) will be a non-redundant (prime) cover, a non-redundant cover being simply a feasible solution to the covering constraints with the property that no subset S_k in the cover C can be wholly contained in the union of the other subsets S_j in C . Given a non-redundant cover, a basis matrix B can then be associated with this prime cover. By simply interchanging rows of this basic matrix (in effect interchanging inequalities in the original constraint set $Ax \geq \underline{1}$), a new basis matrix B can be obtained with the special form

$$\hat{B} = \begin{bmatrix} I_1 & 0 \\ P & -I_2 \end{bmatrix}$$

Since $\hat{\hat{B}}\hat{\hat{B}} = I$, B is its own inverse and is said to be involutory.

The involutory basis cutting-plane technique arises from this result. The following equational cut is imposed:

$$\sum_{j \in Q} x_j \geq 1 \quad \text{where } Q = \{j: (c_B B^{-1} A_j - c_j) > 0\} \quad (1-4)$$

Essentially this cut means that, in the spirit of the revised simplex algorithm, the non-basic columns (corresponding to the subsets S_j not used in the prime cover) are "priced out" in terms of their dual variables. Those columns which price out favorably are considered to be in the set Q . The condition implied by the cut (1-4) is that at least one of the elements of Q appear in the next prime cover when the process is completed. When Q is empty, the last prime cover is optimal, a situation analogous to dual feasibility in the revised simplex algorithm.

The strength of the involutory bases approach is in the determination of which non-basis columns should be members of Q . When the rows of the constraint set are interchanged so that the basis B associated with the current prime cover is involutory, the set Q may alternately be described as

$$Q = \{j: (C_B^{\hat{\hat{B}}} A_j - c_j) > 0\} \quad (1-5)$$

where the $\hat{\hat{}}$ symbol indicates that both the basis matrix B

and the constraint matrix A are to be considered in their rearranged form. Since (1-5) does not require the computation of B^{-1} , the most significant computational effort is involved in the rearrangement of the rows of the current basis matrix and constraint set so that the basis matrix is in involutory form.

Enumerative Methods

The method of implicit enumeration originally introduced by Balas [1] and Geoffrion [12] is rather extensively used in solutional procedures for both the general set-covering problem and specialized applications of the general formulation with exploitable special structure. In general terms, the method of implicit enumeration proceeds in the following manner:

- Step 1. An initial incumbent solution is set either to any feasible solutional value or alternately to some pessimistic bound for an optimal solution.
- Step 2. A "free" variable, or a variable not fixed in the current partial solution is chosen by some predefined entry and is fixed at a value (either 0 or 1). If there are no "free" variables, the procedure branches to Step 4.
- Step 3. Bounds on the value of any solution obtained by "completing" the partial solution reached by fixing this variable are computed by some bounding procedure. If the bounds for the

particular "node" or partial solution indicate that further expansion cannot result in a better partial solution, the node is "fathomed" (Step 4); otherwise the partial solution is expanded by a return to Step 2. If there is a completion of the partial solution which is feasible, and has a lower solution value than the incumbent solution, it becomes the new incumbent solution, and the procedure resumes.

Step 4. If the current node in the solution tree was reached by the "forward" movement of Step 2, the last variable fixed at some value is now refixed at the opposite value and the procedure resumes at Step 2. If the current node was reached by the "backward" movement of this step (Step 4), the last variable fixed in the partial solution is returned to the "free" variables, and the new "current" node becomes the node directly preceding the old "current" node, and the procedure returns to Step 4. If, during the execution of Step 4, all variables are returned to the "free" variables, the procedure stops with the last incumbent solution being optimal.

Lemke, Salkin, and Spielberg [15] present an algorithm

which approaches the general set-covering problem in a rather straightforward application of implicit enumeration. The specialization of the basic implicit enumeration scheme to the special structure of the general set-covering problem occurs primarily in two areas: the variable entry rule of Step 2, and the lower and upper bounding procedures of Step 3. The existence of a partial solution for a set-covering formulation results in a very nice reduction of the size of the sub-problem, as and constraints "covered" by some variable fixed at the value 1 in the partial solution can be eliminated from consideration in the sub-problem, as well as any variable which does not "cover" some constraint already covered by a variable fixed at 1 in the partial solution. With these considerations in hand, the variable entry rule and the bounding procedures apply only to the reduced sub-problem and are as follows:

- a. The variable entry rule is a simple minimum cost per number of uncovered rows heuristic, so the entering variable x_e is defined

$$x_e : \sum_{i \in G} \frac{c_e}{a_{ie}} = \min_{j \in F} \left\{ \frac{c_j}{\sum_{i \in G} a_{ij}} \right\}$$

where F is the set of "free" variables in the sub-problem, and G is the set of constraints unsatisfied in the current partial solution.

- b. The lower bound for the sub-problem is simply the optimal linear programming solution for the sub-problem plus the objective function value of the partial solution.
- c. The upper bounding procedure for the sub-problem heuristically obtains a prime cover from the optimal linear programming solution by rounding up the fractional value in the optimal LP basis and eliminating the redundancy from the resulting cover.

The strength of the Lemke, Salkin, and Spielberg algorithm lies in the fact that the sub-problems are of reduced size and therefore difficulty, a strong asset in enumeration methods. Also, the bounding procedures do not absolutely require a great deal of computational overhead time, as the reduction to the sub-problem can be an implicit operation to tableaus constructed for the parent problem.

A Specialization of the General Set-Covering Problem

A specialized application of the set-covering formulation is an emergency service facilities location model discussed by Toregas, Swain, ReVelle, and Bergman [20]. In this model there are two main assumptions:

1. All user points (associated with covering constraints) may also be emergency facilities locations (associated with the covering variables) and vice versa, and

2. The objective is to minimize the total number of facilities required to "cover" all user points.

The first assumption, coupled with the fact that the coefficients of the constraint matrix are based on metric distances between points means that the constraint matrix is symmetric, or $a_{ij} = a_{ji}$. The second assumption means that all $c_j = 1$, this being referred to as "unit costs".

The solution method of [20] is divided into two main steps:

1. Solve the problem as a linear programming problem and obtain a real solution. If the real solution is also integer, fine. Otherwise,
2. Impose the following constraint:

$$\sum_{j=1}^n x_j \geq [m_0] + 1$$

where $[m_0]$ is the greatest integer less than or equal to m_0 , the optimal linear programming solution. This cut essentially is along the objective function with the intent of forcing one or more of the fractional values for the basic non-surplus variables to round up to 1, and to evoke an all-integer solution to the problem.

Toregas, et al claim that this procedure (at the writing of their paper) has not failed to yield an integer

solution. And although counterexamples to this claim to exist, the simplicity and success of this approach is appealing. What is further interesting about the nature of the imposed cut is that, although the motivation for the cut is very similar to that used in the method of involutory bases, the imposed cut here is more closely tied to the objective function rather than appearing as an additional covering constraint.

The Number-Theoretical Approach

As the formulation of integer linear programs involves systems of diophantine equations and inequalities, considerable interest in the literature has been directed toward the application of many ideas from the well-developed field of linear diophantine analysis to the study of integer linear programs. Bradley ([3] and [4]) is exemplary of this spirit as he examines the equivalence classes of diophantine equational and inequality systems. Kendall and Zionts [14] and Glover and Woolsey [13] have used well-known number-theoretic concepts to discuss the aggregation of diophantine constraints. Bradley and Wahi [5] present a combined number-theoretic and enumerative algorithmic approach to the solution of the general integer programming problem. This algorithm involves the transformation of the diophantine inequality system to a canonical form (closely related to Hermite normal form) and applying the Fourier-Motzkin elimination method. The solution of the transformed formulation of the problem is

then solved by an efficient implicit enumeration scheme.

Richmond and Ravindran [17] use basically the same philosophy in their development of an algorithm for the solution of the general integer program in equality form. Like the Bradley and Wahi approach, the Richmond and Ravindran algorithm employs a transformation to a canonical form and a subsequent use of a form of the Fourier-Motzkin elimination procedure. However, it is here that the two approaches differ. Where Bradley and Wahi use the Fourier-Motzkin elimination procedure as a preparation for an efficient implicit enumeration algorithm, Richmond and Ravindran use a modified backtracking scheme developed by Cook and Cooper [6] at the completion of the forward elimination of the Fourier-Motzkin method to arrive at an integer solution. The computational results reported for this algorithm are not particularly outstanding, but there are several facets of the manufacture of the algorithm which make it rather interesting as a point of departure for investigations into specialized problems where problem structure might aid in algorithmic simplifications. The crux of the performance of the algorithm rests in the modified backtracking scheme, as this is the part of the algorithm which is executed recursively. Because of the way in which Richmond and Ravindran transform the parent problem, the backtracking procedure resolves to the imposition of an objective function bound followed by a search for a feasible integer solution - an approach which, in view of the success

of the Toregas approach to a specialization of the set-covering problem and their similar treatment of the objective function, would seem to indicate promise for an adaptation of the Richmond algorithm to the general set-covering problem. The remainder of this thesis is directed toward an investigation of such an adaptation.

CHAPTER II

ADAPTATION OF RICHMOND'S ALGORITHM TO
THE GENERAL SET-COVERING PROBLEM

In this chapter Richmond's algorithm will be specialized using known special structure inherent in the general set-covering formulation. The specialized algorithm, once constructed, will be compared with the well-known method of implicit enumeration described in Chapter I. The results of this comparison will be used to draw conclusions about the adapted algorithm.

Richmond's Algorithm

The general pure integer programming may be described as

$$\begin{aligned}
 &\text{minimize} && z = cx \\
 &\text{subject to} && Ax = b && (2-1) \\
 &&& x \text{ is a non-negative integer vector} \\
 &&& (x_j \geq 0 \text{ and integer})
 \end{aligned}$$

where A is an m by n matrix, b is an m by 1 column vector, c is a 1 by n row vector, and A , b and c are integer. This formulation can also be written

$$\begin{aligned}
 &\text{minimize} && z \\
 &\text{subject to} && cx + z = 0
 \end{aligned}$$

$$Ax = b \quad (2-2)$$

$$x_j \geq 0 \text{ and integer for}$$

$$\text{all } j = 1, 2, \dots, n$$

If z_0 is defined to be some integer lower bound on z (assuming z is indeed bounded), then the problem may be re-written

minimize

$$\text{subject to } \begin{bmatrix} -c \\ A \end{bmatrix} x + \begin{bmatrix} z_0 + k \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \quad (2-3)$$

$$x_j \geq 0 \text{ and integer for all } j=1, 2, \dots, n$$

$$k \geq 0 \text{ and integer}$$

or, in a more generic form,

minimize k

$$\text{subject to } Hx = d$$

$$x_j \geq 0 \text{ and integer for all } j=1, 2, \dots, n$$

$$k \geq 0 \text{ and integer}$$

$$\text{where } H = \begin{bmatrix} -c \\ A \end{bmatrix} \quad \text{and } d = \begin{bmatrix} -z_0 + k \\ b \end{bmatrix} .$$

In general terms, the Richmond algorithm for such problems operates by first setting $k = 0$ and finding a good initial value for z_0 (usually the value of the linear programming solution to (2-1)). The procedure then checks for the existence of an integer solution to (2-4). If one exists, the optimal solution is realized; otherwise, k is incremented

by 1 and the process is repeated until such time as the system (2-4) is feasible in integers for some k .

The thrust of the algorithm is thus seated in the procedure which checks for the existence of an integer solution for a fixed k value. The development of the mechanics of this procedure is divided up into three parts:

1. The transformation to fundamental form
(Rubin's Sequential Algorithm)
2. Forward variable elimination
(Fourier-Motzkin)
3. Backtracking scheme
(revised Fourier-Motzkin - Cook and Cooper)

Notion of a Fundamental Matrix

Given the system

$$Ax = b$$

there is an associated homogeneous system

$$Ax = 0 . \tag{2-5}$$

If A has m rows and n columns and has rank m , ($m \leq n$), then it is a well-established fact that (2-5) has $n-m$ independent solutions. These solutions form what is known as a "fundamental set" and an n by $n-m$ matrix whose columns consist of the separate members of a fundamental set is called a "fundamental matrix" for the matrix A . An elementary property of the fundamental matrix F of A is that the product

$$M \triangleq AF = 0 .$$

Hence x^* is a solution to the system $Ax = b$ if and only if $x^* + Fy$ is also a solution of $Ax = b$ as

$$A(x^* + Fy) = Ax^* + AFy = b + 0 = b .$$

Thus given a particular solution x^* , and a fundamental matrix F of H , the problem of finding a feasible integer solution to

$$Hx = d \tag{2-6}$$

$$x \text{ a vector } \geq 0 \text{ and integer} \tag{2-7}$$

can be solved by finding an integer solution to the problem

$$x^* + Fy \geq 0 . \tag{2-8}$$

Without loss of generality, the elements of F can be assumed to be integer, as the components of any column of F can be multiplied by the least common multiple of their respective denominators without affecting the independence of the columns with respect to each other. Given that x^* is integer, if y is also chosen as integer, then the resultant value for $x^* + Fy$ is trivially integer. However, it is not true that all feasible integer solutions to (2-8) correspond to integer values of y . Consider the system of equations

$$x_1 + 2x_2 = 1 \tag{2-9}$$

$$x_1 + x_3 = 2$$

where $H = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ and $d = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

Now in this case, $F = \begin{bmatrix} -2p \\ p \\ 2p \end{bmatrix}$ where p is some number,

and a particular solution $x^* = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

In this example, choose $p = 2$; so all the feasible solutions to (2-9) can be described as

$$x = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -4 \\ 2 \\ 4 \end{bmatrix} y$$

where y is (in this case) a scalar. Notice that the vector

$$\begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix} \text{ is an integer solution to (2-9), but the}$$

corresponding value of y which would yield that solution vector is $y = 1/2$, a non-integer quantity, hence care must be taken in the construction of any enumeration technique based on the manipulation of the vector y if no feasible integer points are to be overlooked.

Smith [19] demonstrated that a suitable fundamental set for any m by n matrix H can be selected such that all

integral solutions to the homogeneous problem can be expressed as integral linear combinations of the members of that fundamental set. Hence if the fundamental matrix F of (2-8) is constructed so that the columns are members of a Smith fundamental set, it can be shown that the integer values for $x = x^* + Fy$ will occur on integer values of y . In the above numerical example (2-9) the Smith normal form of the fundamental matrix coincides to the value $p=1$.

To transform the problem (2-4) into the equivalent formulation (2-8), Richmond's Algorithm employs a procedure they call Rubin's Sequential Algorithm. Rubin's algorithm essentially uses basic-number-theoretic concepts to inductively compute an integer solution x^* to (2-6) and simultaneously compute the Smith fundamental matrix, F .

Fundamental Matrix in Set-Covering

The notion of a fundamental matrix in the context of the set-covering problem is a slightly extended concept from the straightforward presentation above. Again, the general set-covering may be written as

$$\begin{aligned} & \text{minimize } cx \\ & \text{subject to } Ax \geq \underline{1} \\ & \quad x_j \geq 0 \text{ and integer } j=1,2,\dots,n \end{aligned} \tag{2-10}$$

where A and $\underline{1}$ are as described before. This formulation is in inequality form, and so the equality form of (2-10) is

$$\begin{aligned}
& \text{minimize } cx \\
& \text{subject to } Ax - Is = \underline{1} \\
& x_j \geq 0 \text{ and integer for all } j=1,2,\dots,n \\
& s_i \geq 0 \text{ for all } i = 1,2,\dots,m
\end{aligned}$$

In the form of (2-3) this is

$$\begin{aligned}
& \text{minimize } k \\
& \text{subject to } \begin{bmatrix} -c & 0 \\ A & -I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} -z_0 & -k \\ \underline{1} \end{bmatrix} \\
& x_j \geq 0 \text{ and integer } j=1,2,\dots,n \\
& s_i \geq 0 \text{ and integer } i=1,2,\dots,m \quad (2-11) \\
& k \geq 0 \text{ and integer.}
\end{aligned}$$

So, in the sense of (2-4),

$$H = \begin{bmatrix} -c & 0 \\ A & -I \end{bmatrix} \quad \hat{x} = [x \quad s]^T, \text{ and } d = \begin{bmatrix} -z_0 & -k \\ \underline{1} \end{bmatrix}$$

H is here an $m+1$ by $m+n$ matrix having a row rank of $m+1$. Thus F, the fundamental matrix of H, will be an $m+n$ by $n-1$ matrix. Denoting the element in the i^{th} row and j^{th} column of F as f_{ij} , it is clear that the matrix $M = HF$ can be written

$$M = \begin{bmatrix} \sum_{j=1}^n -c_j f_{j1} & \sum_{j=1}^n -c_j f_{j2} & \sum_{j=1}^n -c_j f_{j,n-1} \\ \sum_{j=1}^n a_{1j} f_{j1}^{-f_{n+1,1}} & \sum_{j=1}^n a_{1j} f_{j2}^{-f_{n+1,2}} & \sum_{j=1}^n a_{1j} f_{j,n-1}^{-f_{n+1,m}} \\ \sum_{j=1}^n a_{mj} f_{j1}^{-f_{n+m,1}} & \sum_{j=1}^n a_{mj} f_{j2}^{-f_{n+m,2}} & \sum_{j=1}^n a_{mj} f_{j,n-1}^{-f_{n+m,m}} \end{bmatrix} \cdot$$

This notation can be simplified greatly if the p^{th} column of F is denoted by $F(p)$ and the i^{th} row of A is denoted by $A(i)$. The resultant form of M would be

$$M = \begin{bmatrix} -cF(1) & -cF(2) & \cdots & -cF(n-1) \\ A(1)F(1)^{-f_{n+1,1}} & A(1)F(2)^{-f_{n+1,2}} & \cdots & A(1)F(n-1)^{-f_{n+1,m}} \\ \vdots & \vdots & & \vdots \\ A(m)F(1)^{-f_{n+m,1}} & A(m)F(2)^{-f_{n+m,2}} & \cdots & A(m)F(n-1)^{-f_{n+m,m}} \end{bmatrix} \quad (2-12)$$

Recalling that $M=HF=0$, two primary results can be derived from (2-12):

1. $-cF(p) = 0$ for all $p=1,2,\dots,n-1$
2. $A(i)F(p) = f_{n+i,p}$ for all $i=1,2,\dots,m$ and $p=1,2,\dots,n-1$ (2-13)

The first result is consistent with the idea that the columns of the fundamental matrix F are associated with the independent solutions of the homogeneous system $\hat{H}x = 0$ of (2-11). The second result identifies the surplus variable

$f_{n+i,p}$ with i^{th} constraint and the p^{th} independent solution. This in effect means that $f_{n+i,p}$ is the value of the surplus variable in the i^{th} equation given the p^{th} independent solution. This is consistent with the idea that the $n-1$ columns of F span the solution space of (2-11).

The significance of the vector y in this application is that of an index of participation of each of the $n-1$ independent solutions with the particular solution to form the current solution. Thus y may be said to be a "generating" vector for all solutions (in the sense of (2-11)) of the system $\hat{H}x = d$.

The properties of the fundamental matrix F of H presented in (2-13) are derived assuming k is a fixed value. A different fundamental matrix applies when k is considered as a variable. Multiplying the second constraint by -1 , (2-11) can be re-written as

$$\begin{array}{ll} \text{minimize } k \\ \text{subject to } \end{array} \begin{bmatrix} 1 & 0 & -c \\ 0 & I & -A \end{bmatrix} \begin{bmatrix} k \\ s \\ x \end{bmatrix} = \begin{bmatrix} -z_0 \\ -\underline{1} \end{bmatrix} \quad (2-14)$$

$$x_j \geq 0 \text{ and integer for } j=1,2,\dots,n$$

$$s_i \geq 0 \text{ and integer for } i=1,2,\dots,m$$

$$k \geq 0$$

Here, in the sense of (2-4),

$$H = \begin{bmatrix} 1 & 0 & -c \\ 0 & I & -A \end{bmatrix}, \quad \hat{x} = \begin{bmatrix} k \\ s \\ x \end{bmatrix} \quad \text{and } d = \begin{bmatrix} -z_0 \\ -1 \end{bmatrix} .$$

The fundamental matrix F of this matrix H is $n+1$ by m since H is $n+m+1$ by $n+1$. Further, since $HF = 0$, F can be easily verified to be of the form.

$$F = \begin{bmatrix} c \\ A \\ I \end{bmatrix} . \quad (2-15)$$

Rubin's Sequential Algorithm, given H as in (2-14), will arrive at F of H as in (2-15) and compute x^* as

$$\hat{x}^* = \begin{bmatrix} -z_0 \\ -1 \\ 0 \end{bmatrix} \quad (2-16)$$

This particular solution to (2-14) is interesting in that it is equivalent to letting s and k be basic in the linear programming relaxation of (2-14). Further, k and s in x^* are both negative and hence infeasible.

With the computation of the fundamental matrix F of H as in (2-14) and (2-15) and a particular solution \hat{x}^* (2-16), Richmond's algorithm seeks a solution to the inequality system (2-8) which is in this case

$$\hat{x}^* + Fy \geq 0 . \quad (2-17)$$

Substituting (2-16) and (2-15), the relationship

$$\begin{bmatrix} -z_0 \\ \underline{1} \\ 0 \end{bmatrix} + \begin{bmatrix} c \\ A \\ I \end{bmatrix} y \geq 0 \quad (2-18)$$

is obtained. Subtracting \hat{x}^* from both sides yields the inequality.

$$\begin{bmatrix} c \\ A \\ I \end{bmatrix} y \geq \begin{bmatrix} z_0 \\ \underline{1} \\ 0 \end{bmatrix} \quad (2-19)$$

With $y = x$, the inequality system (2-19) in effect yields back the original problem in the form (2-11). It would thus appear that in the set covering case the entire first phase of Richmond's algorithm (that of transformation of the system to fundamental form via Rubin's Sequential Algorithm) does not result in any reductions and can be entirely eliminated by the direct result (2-19).

Fourier-Motzkin Elimination Method

To solve the system

$$\hat{x}^* + Fy \geq 0, \text{ or } Fy \geq -\hat{x}^* \quad (2-20)$$

for integer solutions, Richmond's algorithm uses the classical Fourier-Motzkin Elimination procedure. This classical procedure has received a substantial amount of attention in the mathematical programming literature, both as a free-

standing method (Dantzig [7], Garfinkel and Nemhauser [11], and Bradley [4]), and in conjunction with other canonical transformations (Bradley and Wahi [5]). The classical procedure itself is solely involved with a forward variable-by-variable elimination of (2-20) in favor of expressional upper and lower bounds on each variable in a manner as described below. As the computed bounds for each variable are not restricted to be integer in the classical approach, modern application to integer programming has popularized a second phase to the procedure - namely a back-substitution method to locate an all-integer solution. This back-substitution method was originally introduced by Cook and Cooper [6].

Forward Variable Elimination

Forward variable elimination is the process by which the system

$$Hy \geq d \quad (2-21)$$

is solved in real-valued variables. Taking the dimension of H to be $m+1$ by n , (2-21) can be written

$$\sum_{j=1}^n h_{ij}y_j \geq d_i \quad \text{for all } i=1,2,\dots,m+1 \quad (2-22)$$

A single iteration of the forward variable elimination yields a system of inequalities with 1 variable eliminated (hence a system of $n-1$ variables) and sets of both upper and lower bounds for the eliminated variable in terms of the other $n-1$

variables. Basically, the forward elimination procedure can be described as a three-step procedure.

Step 1. Choose the lowest subscript value of the yet-uneliminated variables (say e). Partition or divide the current set of inequalities into three groups according to the sign of the coefficient in the current constraint matrix:

$$\begin{aligned} g_1(e) &= \{r : h_{re} > 0\} \\ g_2(e) &= \{t : h_{te} < 0\} \\ g_3(e) &= \{u : h_{ue} = 0\} \end{aligned} \quad (2-23)$$

Step 2. Every $r \in g_1(e)$ represents a lower bound for y_e in terms of the yet-uneliminated variables, as division of both sides of the inequality by h_{re} yields

$$\frac{d_r}{h_{re}} - \sum_{j=e+1}^n \frac{h_{rj}y_j}{h_{re}} \leq y_e \quad \text{for all } r \in g_1(e) \quad (2-24)$$

Every $t \in g_2(e)$ represents an upper bound for y_e in terms of the uneliminated variables, as division of both sides of the inequality by h_{te} yields

$$y_e \leq \frac{d_t}{h_{te}} - \sum_{j=e+1}^n \frac{h_{tj}y_j}{h_{te}} \quad \text{for all } t \in g_2(e) \quad (2-25)$$

Every $u \in g_3(e)$ represents no restriction on the variable y_e , and therefore generates no bound.

Step 3. A new set of inequalities is constructed by letting each and every upper bound of y_e be greater than or equal to every lower bound of y_e . Added to this set are the inequalities represented in $g_3(e)$ which did not play an active role in bounding y_e . Hence the new "current" set of inequalities is

$$\sum_{j=e+1}^n \begin{pmatrix} h_{rj} & - & h_{tj} \\ h_{re} & & h_{te} \end{pmatrix} y_j \geq \begin{pmatrix} d_r & - & d_t \\ h_{re} & & h_{te} \end{pmatrix}$$

for all $r \in g_1(e)$, $t \in g_2(e)$ (2-26)

$$\sum_{j=e+1}^n h_{uj} y_j \geq d_u \quad \text{for all } u \in g_3(e) \quad (2-27)$$

For ease of notation, the new "current" set of inequalities is also written $Hy \geq d$. If there are any uneliminated variables left, the procedure returns to Step 1.

At the end of the forward variable elimination procedure, y_n has upper and lower bounds which are real numbers. The least upper bound and the greatest lower bound are the true values which y_n can assume in a feasible solution. If there are no values for y_n with these bounds, the entire sys-

tem (2-22) has no real solution.

The bounds for the variable y_{n-1} as computed in Step 2 are a function only of y_n . Therefore, given some value for y_n , the set of upper and lower bounds can be computed for y_{n-1} . The least upper bound and the greatest lower bound form the "true" bounds between which all feasible values of y_{n-1} must fall given the fixed value of y_n . In a similar fashion, the bounds for any variable y_j can be computed given the partial solution $\{y_n, y_{n-1}, \dots, y_{j+1}\}$. This procedure is known as "back-substitution", and is the basis for the work of Cook and Cooper.

The essence of Forward Variable Elimination is that the new "current" system represented by (2-26) and (2-27) represents the necessary and sufficient conditions for the existence of a solution to the old "current" system $Hy \geq d$. This notion is intuitive and can be best illustrated with an example.

Consider the system

$$3x_1 + 4x_2 \geq 11$$

$$-x_1 + x_2 \geq 1$$

If x_1 is eliminated, the resultant bounds are

$$x_1 \geq \frac{11}{3} - \frac{4}{3}x_2$$

$$x_1 \geq -1 + x_2$$

and the reduced system in only the variable x_2 is

$$-1 + x_2 \geq \frac{11}{3} - \frac{4}{3} x_2$$

or $x_2 \geq 2$. Any value of $x_2 \geq 2$ will generate a non-empty interval for x_1 , and any value of $x_2 < 2$ will generate an empty interval for x_1 . For example, $x_2 = -1$ generates bounds on x_1 as

$$x_1 \geq 5 \quad \text{and} \quad x_1 \leq -2$$

to which there is no real solution.

The primary drawback to the use of the Fourier-Motzkin Elimination procedure is that the number of inequalities generated may grow exponentially at each stage of the elimination at the rate of $\left(\frac{q}{2}\right)^2$, where q is the number of inequalities at the previous state. If n_1, n_2 , and n_3 are defined to be the number of inequalities represented in $g_1(e)$, $g_2(e)$, and $g_3(e)$, respectively; the number of inequalities generated at each stage will be equal to $n_1 n_2 + n_3$. This potentially rapid growth in the sheer problem size is what is probably primarily responsible for the relative absence of successful algorithms employing a Fourier-Motzkin-based approach.

Forward Variable Elimination in Set-Covering

The special structure of the set-covering problem allows a substantial simplification in the forward variable elimination process. The system to be solved is (from (2-19)),

$$\begin{bmatrix} c \\ A \\ I \end{bmatrix} y \geq \begin{bmatrix} v \\ \underline{1} \\ 0 \end{bmatrix} \quad (2-28)$$

where $v \geq z_0$ is some candidate objective function value. This is equivalent to the system

$$\begin{bmatrix} c \\ A \end{bmatrix} y \geq \begin{bmatrix} v \\ \underline{1} \end{bmatrix} \quad (2-29)$$

$$y \geq 0$$

Eliminating first on y_1 ($e=1$ in Step 1), the inequality

$$cy \geq v \quad (2-30)$$

is in set $g_1(1)$, as $c_1 > 0$. Likewise, in the sense of (1-3)

$$i \in g_1(1) \quad \text{for all } e_i \in S_1$$

$$i \in g_3(1) \quad \text{for all } e_i \notin S_1$$

which is to say that all inequalities "covered" by S_1 are also in $g_1(1)$. All other rows of A are in $g_3(1)$. Thus after elimination, only the inequalities for those rows in A which did not involve y_1 will be preserved. The bounds generated for y_1 in Step 2 are

$$\frac{v}{c_1} - \sum_{j=2}^n \frac{c_j y_j}{c_1} \leq y_1 \quad (2-31)$$

and

$$1 - \sum_{j=2}^n a_{ij}y_j \leq y_1 \quad \text{for all } i \text{ such that } a_{i1} = 1 \quad (2-32)$$

Assuming all y_j 's are non-negative, interpretations of (2-31) can be made. The bound represented by (2-31) simply says that y_1 must account for the amount the objective function value is short, given the partial solution $\{y_n, y_{n-1}, \dots, y_2\}$. The bound represented by (2-32) indicates that if the partial solution $\{y_n, y_{n-1}, \dots, y_2\}$ does not "cover" some row i which y_1 can cover, y_1 must account for the remainder of 1 if a feasible solution is to be obtained.

Continuing the elimination with variable y_2 , the remaining system is

$$\sum_{j=2}^n a_{ij}y_j \geq 1 \quad \text{for all } i \text{ such that } a_{i1} = 0 \quad (2-33)$$

Here again, all constraints which are "coverable" by $S_2(a_{i2} = 1)$ are in group $g_1(2)$, while all other constraints $i(a_{i2} = 0)$ are in group $g_3(2)$. So the bounds for y_2 are (from (2-24))

$$1 - \sum_{j=p+1}^n a_{ij}y_j \leq y_p \quad \text{for all } i \text{ such that} \\ a_{ip} = 1 \text{ and } a_{i1} = a_{i2} = \dots = a_{ip-1} = 0 \quad (2-35)$$

This is a general statement of the "last opportunity" restriction imposed by the Fourier-Motzkin procedure which closely parallels the Forced Variable reduction property discussed in Chapter I.

Since y is in effect the x vector of (1-3), the integrality condition on x is directly imposable on y . With all y_j required to be integer, it can be said that (2-35) forces y_p to be at least 1, rather than the sum of a number of fractions subtracted from 1.

Insofar as the objective in (2-14) is to minimize the value of the variable k , there is a noticeable lack of any mention of k in the immediately preceding discussion. From (2-14), k is seen to satisfy the expression

$$k = cz - z_0 \quad (2-36)$$

Substituting this into (2-30) and equating x with y , the result obtained is

$$k \geq v - z_0 \quad (2-37)$$

As the objective is to make k the smallest non-negative integer possible, it is also the objective to force v as close to z_0 (from the positive side, as $v \geq z_0$) as possible. In this sense, there may appear to be an inconsistency with (2-29) as any $y = w\underline{1}$ with w sufficiently large will satisfy the relationship but be utterly meaningless as a problem solution. The way in which the objective of the minimization is

achieved is inherent in the backtracking procedure and not in a simple satisfaction of the relationship (2-29).

Backtracking for Integer Solutions

A modified back-substitution method for finding integer solutions to the system (2-22)

$$\sum_{j=1}^n h_{ij}y_j \geq d_i \quad \text{for all } i=1,2,\dots,m+1 \quad (2-38)$$

has been refined by Cook and Cooper [6]. If an effective lower bound L_e is defined for y_e given a partial solution $\{y_n, y_{n-1}, \dots, y_{e+1}\}$ as

$$\begin{aligned} L_e &= \text{the smallest integer greater than or equal to} \\ &\quad \lambda_e \text{ where } \lambda_e \text{ is defined from (2-24) as} \\ \lambda_e &= \max_{r \in g_1(e)} \left\{ \frac{d_e}{h_{re}} - \sum_{j=e+1}^n \frac{h_{rj}y_j}{h_{re}} \right\} \end{aligned} \quad (2-39)$$

and similarly define the effective upper bound U_e for y_e

$$\begin{aligned} U_e &= \text{the greatest integer less than or equal to} \\ &\quad u_e \text{ where } u_e \text{ is defined from (2-25) as} \\ u_e &= \min_{t \in g_2(e)} \left\{ \frac{d_e}{h_{te}} - \sum_{j=e+1}^n \frac{h_{tj}y_j}{h_{te}} \right\} \end{aligned}$$

the modified Cook and Cooper procedure can be described as a four-step process as follows:

Step 1. The index j of the "current" variable is set

to $n + 1$ and the current partial solution is set as empty so that the procedure may begin correctly with consideration of y_n at Step 2. ($j = n + 1$)

Step 2. The next variable to be considered in the expansion of the current partial solution is determined to be y_{j-1} , ($j = j - 1$), and the procedure resumes at Step 3. If there is no next variable ($j = 1$), then the current partial solution is complete and is a solution to (2-38).

Step 3. If the bounds on the current variable indicate that there does not exist a feasible integer completion ($L_j > U_j$), then the procedure branches to Step 4 to backtrack. Otherwise, the current partial solution is augmented by fixing $y_j = L_j$, the lower bound. The procedure then proceeds to Step 2 to continue the expansion of the new current solution.

Step 4. Considering the last variable fixed in the partial solution as the current variable ($j = j + 1$), if y_j is currently at its upper bound, then no completion of the partial solution $\{y_j, y_{j+1}, \dots, y_n\}$ can be feasible. Thus, y_j is "freed" and y_{j+1} is examined for

the same characteristics by a return to Step 4. If y_j is below its upper bound ($y_j < U_j$), then y_j is re-fixed at the next highest integer value ($y_j = y_j + 1$) and the procedure resumes with Step 2. If at the time the procedure reached Step 4 there was no current partial solution, then there is no feasible integer solution to the system (2-38).

The application of this backtracking scheme in Richmond's algorithm is augmented by k as the $n+1^{\text{st}}$ variable in the elimination process. This is to say that the value of k is the first variable in a given partial solution.

Backtracking for the Set-Covering Problem

In the context of the set-covering problem, the addition of k in the backtracking scheme has a direct effect upon the lower bound for y_1 represented by (2-31). The intent of this lower bound is to insure that the value of the objective function is at least some value v . With the explicit inclusion of k , v can be represented as $z_0 + k$. Since the purpose of the backtracking scheme is to determine if an integer feasible solution exists for a fixed value of k , (2-31) can be written

$$y_1 = \frac{k + z_0}{c_1} - \sum_{j=2}^n \frac{c_j y_j}{c_1} \quad (2-41)$$

and the term L_1 (in the sense of (2-39)) can be described

$$L_1 = \begin{cases} 0 & \text{if (2-41) } \Rightarrow y_1 = 0 \text{ and } \sum_{j=2}^n a_{ij}y_j \geq 1 \\ & \text{for all } i \in g_1(1) \\ 1 & \text{if (2-41) } \Rightarrow y_1 = 1 \text{ and } \sum_{j=2}^n a_{ij}y_j = 0 \\ & \text{for some } i \in g_1(1) \\ 2 & \text{otherwise} \end{cases} \quad (2-42)$$

The possible assignment of 2 to L_1 is to indicate an infeasibility in integers of a particular partial solution $\{k, y_n, y_{n-1}, \dots, y_2\}$ and in coordination with the definition of U_1 below, is designed to cause a regression in the expansion of the particular partial solution as in Step 4 of the modified Cook and Cooper scheme as described above.

For the y_q with $q = 2, 3, \dots, n$; the bounding expression (2-35) applies, so L_q for $q = 2, 3, \dots, n$ can be described as

$$L_q = \begin{cases} 1 & \text{if } \sum_{j=q+1}^n a_{ij}y_j = 0 \text{ for some } i \in g_1(q) \\ 0 & \text{otherwise} \end{cases}$$

The generation of the variable-bounding expressions for (2-29), due to the non-negative nature of both c and A , do not include any upper bounding expressions. However, the 0-1 requirement of the solution vector x in (1-1) is in itself an imposition of an upper bound on all x_j (and all y_j) where $j = 1, 2, \dots, n$. Further, as $c_j > 0$ for all $j = 1, 2, \dots, n$;

any partial solution $\{y_n, y_{n-1}, \dots, y_q\}$ with the property

$$\sum_{j=q}^n c_j y_j > k + z_0 \quad (2-44)$$

cannot have a completion $\{y_{q-1}, \dots, y_1\}$ which satisfies

$$\sum_{j=1}^n c_j y_j = k + z_0 \quad (2-45)$$

so each y_q is also upper bounded by

$$y_q \leq \frac{k + z_0 - \sum_{j=q+1}^n c_j y_j}{c_q} \quad (2-46)$$

Thus, in the sense of (2-40), U_q for $q = 1, 2, \dots, n$ can be described

$$U_q = \begin{cases} 0 & \text{if } \sum_{j=q}^n c_j y_j > k + z_0 \\ 1 & \text{otherwise} \end{cases} \quad (2-47)$$

In summary, L_q in the set-covering environment represents the Forced Variable reduction property discussed in Chapter I. In addition, L_1 also encompasses (2-41) as an objective function value-related restriction on y_1 . U_q in the set-covering environment represents an objective function value ceiling on every partial solution which says that a

partial solution cannot be expanded productively if the sum of the cost coefficients of the y_j fixed at 1 in the partial solution already exceeds the target objective function value, $k + z_0$. The use of L_q and U_q in the modified Cook and Cooper procedure constitutes the specialization of the third part of Richmond's algorithm to the general set-covering problem.

Example. So that the actual mechanics of the adapted algorithm can be easily understood, consider the following numerical example:

$$\begin{aligned}
 &\text{minimize } 3x_1 + 4x_2 + 5x_3 \\
 &\text{subject to} \quad \quad \quad x_3 \geq 1 \\
 &\quad \quad \quad \quad \quad \quad x_1 + x_2 \geq 1 \quad \quad \quad (2-48) \\
 &\quad \quad \quad \quad \quad \quad x_2 + x_3 \geq 1 \\
 &x_i = 0 \text{ or } 1 \text{ for } i = 1, 2, 3.
 \end{aligned}$$

For this example, consider $v = z_0 + k = 7$ and let the procedure operate from Step 1:

- Step 1. $j = n+1 = 4$.
- Step 2. Current partial solution = $\{\emptyset\}$.
- Step 3. $L_3 = 1, U_3 = 1, L_3 \leq U_3$, go to Step 2.
- Step 2. Current partial solution = $\{x_3=1\}$ $j = 3-1 = 2$.
- Step 3. $L_2 = 0, U_2 = 0, L_2 \leq U_2$, go to Step 2.
- Step 2. Current partial solution = $\{x_3=1, x_2=0\}$
 $j = 2-1 = 1$
- Step 3. $L_1 = 1, U_1 = 0, L_1 > U_1$, go to Step 4. (2-49)

- Step 4. $j = 1+1 = 2, x_2 = U_2$
 Current partial solution = $\{x_3=1\}$ go to Step 4.
- Step 4. $j = 2+1 = 3, x_3 = U_3$
 Current partial solution = $\{x_3=1\}$ go to Step 4.
- Step 4. $j = 2+1 = 3, x_3 = U_3$
 Current partial solution = $\{\emptyset\}$ go to Step 4.
- Step 4. There is no feasible solution with $v = 7$.

If the target objective function value v is incremented by 1 from 7 to 8, and the procedure is repeated starting from Step 1, the results are the same until the procedure reaches statement (2-49). From there, the procedure is as follows:

- Step 3. $L_1 = 1, U_1 = 1, L_1 \leq U_1$, go to Step 2.
- Step 2. Current partial solution is $\{x_3=1, x_2=0, x_1=1\}$
 $j = 1-1 = 0$
 The current partial solution is complete.

The Adapted Algorithm as Implicit Enumeration

The third part of Richmond's algorithm dealing with backtracking for integer solutions closely resembles the general procedure for implicit enumeration as described in Chapter I in such areas as the development of partial solutions and the use of bounding procedures. In this section it shall be shown that the two procedures are virtually identical in the set-covering case. For the sake of clarity and brevity, the central ideas of the method of implicit enumeration and the modified Cook and Cooper procedure adapted to the general set-covering problem are summarized in Figure 2-1.

IMPLICIT ENUMERATIONMODIFIED COOK AND COOPER

Step 1: Initialization

An incumbent solution is selected. The current partial solution is empty.

The current j is set to $n+1$. The current partial solution is empty.

Step 2: Variable Entry Rule

A variable entry rule selects some y_j to be considered for entry into the partial solution and is fixed at either 0 or 1. If no variable can enter, the process branches to Step 4.

The next current variable is determined as y_{j-1} so that the new current j is equal to $j-1$. If the new current value of j is 0, the old current partial solution is a complete solution.

Step 3: Forward Partial Solution Expansion

If the current partial solution is complete and has a better objective function value than the old incumbent, it replaces the old incumbent and the process resumes at Step 4. If the computed bounds indicate that further expansion of the partial solution cannot lead to a better incumbent, the process resumes at Step 4; otherwise it returns to Step 2.

If the bounds on y_j are such that $L_j > U_j$, the process branches to Step 4. Otherwise, y_j is fixed at L_j in the new partial solution and the process returns to Step 2.

Step 4: Backtracking on the Partial Solution

If y_j has the value originally assigned to it by Step 2, y_j is fixed at the opposite value and the process resumes at Step 2. Otherwise, y_j is "freed" and the procedure returns to Step 4. If all variables are "free", the last incumbent solution is optimal.

The current variable y_j becomes the last variable fixed in the partial solution. If this variable is at least its upper bound, then y_j is "freed" from the current partial solution and the process examines y_{j+1} by returning to Step 4. Otherwise, the fixed value of y_j is incremented by 1 in the current partial solution, the lower bound L_j is set to y_j , and the process returns to Step 2.

Figure 2-1: A Side-by-side Comparison of the Algorithms

It is clear from Figure 2-1 that the two algorithms generally break down into the same procedural steps. The concept of developing a partial solution into a complete solution is the essence of both algorithms and largely dictates their similar procedural flow. However, even though the means by which these algorithms achieve their ends are alike, their respective goals are different. The method of implicit enumeration is constantly looking for a completion which is better than some known incumbent solution, while the modified Cook and Cooper procedure stops upon reaching the first completion. This difference is seen most easily in noting the places where these two procedures terminate. The method of implicit enumeration terminates in Step 4 when it is found that all branches of the solution tree have been searched. The Cook and Cooper procedure stops at Step 2, when the first complete integer solution has been constructed.

The precise details of how the partial solutions are constructed are somewhat different in the two procedures and demand closer examination. The variable entry rule for the modified Cook and Cooper could be considered a special case of the more generic statement in the implicit enumeration procedure. The entering variable is merely the one with the next lowest subscript. More important is the fact that the bounds used in Step 3 of the implicit enumeration scheme are directly related to the value of the objective function, while the bounds on Step 3 of the modified Cook and Cooper

procedure are related only to a particular variable y_j . For the set-covering case, this difference can be reconciled by remembering that U_j as stated in (2-47) is really an objective function-related bound. Further, L_j as stated in (2-43) is primarily the result of a reduction property, and for y_1 is also an objective function-related bound.

Perhaps the most interesting difference between the two algorithms is that fact that the method of implicit enumeration is designed for 0-1 variables (see however Trotter and Shetty [21] for extensions to the general integer case), while the modified Cook and Cooper procedure is restricted only to integer problems. Restricting the values of the variables to 0-1, as in the set-covering case, has interesting effects upon the performance of the Cook and Cooper procedure. First, the assignment of the opposite value to a variable in Step 4 of the implicit enumeration scheme is duplicated by Step 4 of the Cook and Cooper procedure when the value of the variable being considered is changed from 0 to 1. When this change is subsequently from 1 to 2, or 2 to 3, this causes y_j to be greater than U_j and a branch to Step 4, which is exactly what takes place when the variable being considered in Step 4 of the implicit enumeration procedure is not found to be at its originally-set value.

The above observations concerning the similarities and differences between the method of implicit enumeration and the modified Cook and Cooper procedure serve to point out the

fact that the two approaches are very close and virtually indistinguishable in many respects.

CHAPTER III

CONCLUSIONS AND EXTENSIONS

The specialization of Richmond's algorithm to the general set-covering problem was accomplished in three parts in this thesis. The first part of Richmond's algorithm, which concerns the transformation of the original problem to canonical form with Rubin's Sequential Algorithm, was shown to degenerate to the original problem. The notion of a fundamental matrix, although quite interesting due to the structure which it exhibits, does not supply any previously-unknown information about problem structure for set-covering constraints.

The second part of Richmond's algorithm, which deals with the use of the classical Fourier-Motzkin elimination procedure to obtain bounds on each of the problem variables, was shown to degenerate primarily to the intuitive Forced Variable reduction property. The addition of some common sense rules to the direct results the Fourier-Motzkin forward elimination produced the final upper and lower bounding expressions used in the third step of Richmond's algorithm specialized to the general set-covering problem.

The third part of Richmond's algorithm, that of backtracking for integer solutions with the Cook and Cooper procedure, was shown to be closely related to the general method of implicit enumeration. Using the bounding expressions ob-

tained from the second part, the specialized algorithm and general implicit enumeration were compared side-by-side. The mechanics of the two procedures were shown to virtually identical, differing only in the basic philosophies underlying them. The general implicit enumeration procedure bounds the optimal solution value with the values of successively better values of feasible solutions discovered during the expansion of partial solutions. In the context of the set-covering problem, these successively better values bound the value of the optimal solution from the top. The philosophy of the Richmond algorithm, and hence of the specialization of the algorithm to the general set-covering problem, is to approach the optimal solution value from the bottom, with the hope that the tighter bounds on the variables will cause faster fathoming to arrive more quickly at an optimal solution.

The combined impact of the above results is that the apparently quite unusual, number-theoretical approach of Richmond's algorithm degenerates in the set-covering case to little more than well-known implicit enumeration techniques. Thus, while not producing a new algorithm, the effort of this thesis to specialize the Richmond algorithm to the set-covering case has produced a unification of two quite separate branches of integer programming research.

It is conjectured that other researchers could substantially expand on this unification. The analysis leading to the conclusion that Rubin's Sequential Algorithm provides

no new information in the set-covering case would seem to equally applicable to any integer program with inequality constraints. Similarly, the parallel between the mechanics of implicit enumeration and the Cook and Cooper procedure would appear to hold for any 0-1 integer programming problem.

BIBLIOGRAPHY

1. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," Operations Research 13 (1965), pp. 517-546.
2. Bellmore, M. and H. D. Ratliff, "Set-Covering and Involutionary Bases," Management Science 18, pp. 194-206.
3. Bradley, G. H., "Transformation of Integer Programs to Knapsack Problems," Mathematics, Vol. 1 (1971).
4. Bradley, G. H., "Equivalent Integer Programs and Canonical Problems," Management Science 17 (1971), pp. 354-366.
5. Bradley, G. H., and P. N. Wahi, "An Algorithm for Integer Linear Programming: A Combined Algebraic and Enumerative Approach," Report No. 29, Dept. of Administrative Sciences, Yale University, December 1969.
6. Cook, R. A. and L. Cooper, "An Algorithm for Integer Linear Programming," Report No. AM 65-2, Washington University, November 1965.
7. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1963, pp. 84-85.
8. Dickson, L. E., History of the Theory of Numbers, Volume II: Diophantine Analysis, Stechert and Co., New York, 1934, pp. 83-84.
9. Garfinkel, R. S., and G. L. Nemhauser, "The Set-Partitioning Problem: Set-Covering with Equality Constraints," Operations Research 17, 1969, pp. 848-856.
10. Garfinkel, R. S., and G. L. Nemhauser, "Optimal Political Districting by Implicit Enumeration Techniques," Management Science Vol. 16, No. 8 (April 1970), pp. B495-B508.
11. Garfinkel, R. S., and G. L. Nemhauser, Integer Programming, John Wiley & Sons, New York, 1972, Chapters 7 & 8.
12. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas' Method," SIAM Review Vol. 9, No. 2 (April 1967), pp. 178-190.
13. Glover, F., and R. E. D. Woolsey, "Aggregating Diophantine Equations," Zeitschrift fur Operations Research, Vol. 16 (1972).

14. Kendall, K. E., and S. Zionts, "Solving Integer Programming Problems by Aggregating Constraints," Presented at the Joint National Meeting of the AIIE, ORSA, and the Institute of Management Science, Atlantic City, N. J., November 1972.
15. Lemke, C. E., H. M. Salkin, and K. Spielburg, "Set-Covering by Single-Branch Enumeration with Linear-Programming Subproblems," Operations Research 19 (1971), pp. 998-1022.
16. Mordell, L. J., Diophantine Equations, Academic Press, London, 1969.
17. Richmond, T. R., and Ravindran, A., "A Generalized Euclidean Procedure for Integer Linear Programs," Graduate School of Business, Howard University, Washington, D. C., 1972.
18. Saaty, T. L., Optimization in Integers and Related Extremal Problems, McGraw-Hill, New York, 1970.
19. Smith, H. J. S., Report on the Theory of Numbers, Chelsea Press, Bronx, N. Y., 1965 and 1894.
20. Toregas, C., R. Swain, C. ReVelle, and L. Bergman, "The Location of Emergency Service Facilities," Operations Research Vol. 19 (1971), pp. 1363-1373.
21. Trotter, L. E. Jr., and C. M. Shetty, "An Algorithm for the Bounded Variable Integer Programming Problem," Dept. of Industrial and Systems Engineering, Georgia Institute of Technology, 1971.
22. Uspenski, J. V., Elementary Number Theory, McGraw-Hill, New York, 1939.