To the Graduate Council:

I am submitting herewith a thesis written by Christine Chau-yan Chen entitled "Some Selected Methods for Solution to the Integer Linear Programming Problem." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Yueh-er Kuo

Yueh-er Kuo, Major Professor

We have read this thesis and recommend its acceptance:

Steven Mark Serbin

Charles E Clark

Accepted for the Council:

L Evans Peck

Vice Chancellor
Graduate Studies and Research

Thesis
78
.C375
Cop. 2

SOME SELECTED METHODS FOR SOLUTION TO THE INTEGER LINEAR PROGRAMMING PROBLEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Christine Chau-yan Chen

March 1978

1356501

ACKNOWLEDGEMENT

The author would like to express her thanks to Dr. Yueh-er Kuo who directed this thesis.

ABSTRACT

The purpose of this thesis is to consider some algorithms for the solution to the integer linear programming problem.

Chapter I introduces the linear programming problem, the simplex method and the dual simplex method, along with some theoretical basis for these methods.

Chapter II addressed the primal integer programming algorithm to solve primal feasible, but not optimal, integer linear programming problem. The finiteness of iterations in this algorithm is also proved.

Chapter III discusses the primal - dual integer programming algorithm for solving the primally and dually infeasible integer linear programming problem.

Chapter IV introduces Balas' additive algorithm for solving the 0 - 1 linear programming problem.

Since the integer linear programming problem may have many variables and constraints, sometimes it is impossible to solve them by hand. Flowcharts for the primal integer programming algorithm and Balas' additive algorithm are presented in Chapter II and Chapter IV. Computer programs can be easily developed based on these flowcharts to execute the algorithms for solving the linear integer programming problem.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
	The Linear Programming Problem	1
	The Simplex Method	4
	The Dual Simplex Method	9
II.	THE PRIMAL INTEGER PROGRAMMING ALGORITHM	13
	The Rudimentary Primal Algorithm	14
	Geometrical Interpretation of the Primal Integer Programming Algorithm	19
	The Simplified Primal Algorithm	23
	The Proof of Finiteness of the Simplified Primal Algorithm	37
III.	A PRIMAL - DUAL INTEGER PROGRAMMING ALGORITHM	46
	Introduction	46
	The Primal - Dual Integer Programming Algorithm	47
IV.	THE ZERO - ONE LINEAR IMPLICIT ENUMERATION	55
	The Concept of Implicit Enumeration	55
	The Enumeration Scheme	58
	Balas' Additive Algorithm for Solving 0 - 1 Linear Programming Problem	61
	Additional Tests for the Implicit Enumeration Method	77
	Surrogate Constraints	80
BIBLIOGRAPHY		86
VITA		88

LIST OF FIGURES

FIGURE		PAGE
2.1	Graphical Interpretation Revealing the Solution of the Integer Linear Programming Problem (2.8)	22
2.2	Flowchart for the Simplified Primal Integer Programming Algorithm	32
4.1	Graphical Interpretation Revealing the Implicit Enumeration	56
4.2	Flowchart for Glover's Enumeration Scheme .	59
4.3	Flowchart for Balas' Additive Algorithm . .	67

CHAPTER I

INTRODUCTION

The Linear Integer Programming Problem is that of finding solution to a set of linear equations or inequalities that maximize (or minimize) a given linear function subject to the added condition that all or some of the variables may only assume integer values. When all of the variables must be integers, then it is called a Pure Integer Linear Programming Problem. When only certain specific variables must be integers, then it is called a Mixed Integer Linear Programming Problem.

In 1958 Gomory first devised a method, known as the method of Integer Forms [6], for solving the pure integer linear programming problem. In 1960, he devised another method, the All Integer Method [7]. Since then, four main distinct approaches capable of solving this kind of problem effectively have been demonstrated. These can be described briefly as: the cutting plane method, the approximation method, the enumeration method and the combinatorial method [2]. Here we will limit our discussion to the primal method and the enumeration method.

I. THE LINEAR PROGRAMMING PROBLEM

The linear programming problem may be stated as follows:

$$\text{Maximize} \quad z = c_1 x_1 + c_2 x_2 + \dots + c_n x_n, \tag{1.1}$$

$$\text{subject to} \quad a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1,$$

$$\vdots \tag{1.2}$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m,$$

$$x_1, x_2, \dots, x_n \geq 0, \tag{1.3}$$

where the a_{ij}, b_i, c_j are given constants and $m < n$. A minimization problem can be converted into a maximization problem by multiplying the objective function by a negative sign. All constraint inequalities can be in the same form by multiplying the both sides of inequalities by negative signs.

After adding slack variables, we can state the previous problem as follows :

$$\text{Maximize} \quad z = c_1 x_1 + \dots + c_n x_n + \dots + c_{n+m} x_{n+m},$$

$$\text{subject to} \quad a_{11}x_1 + \dots + a_{1n}x_n + x_{n+1} = b_1,$$

$$a_{21}x_1 + \dots + a_{2n}x_n + x_{n+2} = b_2, \tag{1.4}$$

$$\vdots$$

$$a_{m1}x_{11} + \dots + a_{mn}x_n + x_{n+m} = b_m,$$

$$\text{where} \quad x_1, x_2, \dots, x_{n+m} \geq 0.$$

The previous program can also be stated as follows:

$$\text{Maximize} \quad z = c\,x$$

$$\text{subject to} \quad \tilde{A}\,x = b \tag{1.5}$$

$$\text{and} \quad x \geq 0.$$

where $c = (c_1, c_2, \dots c_{n+m})$ for $c_{n+1} = \dots = c_{n+m} = 0$, $x = (x_1, \dots x_n, x_{n+1}, \dots x_{n+m})^T$, $b = (b_1, \dots b_m)^T$, the superscript T means the transpose of a vector, $\tilde{A} = (a_{ij})$ and 0 is an n-dimensional null column vector.

It can also be stated as follows :

Maximize $z = c\,x$,

subject to $x \geqslant 0$,

and $x_1P_1 + \cdots + x_nP_n + x_{n+1}P_{n+1} + \cdots + x_{n+m}P_{n+m} = P_0$,

where P_j for $j = 1,2,\cdots,n+m$ is the j-th column of the matrix $\tilde{A}$ and $P_0 = b$.

A vector x, which satisfies conditions (1.2), (1.3) or which satisfies $x_1P_1 + \cdots + x_{n+m}P_{n+m} = 0$, $x \geqslant 0$ is called a feasible solution to the linear programming problem.

If x is a feasible solution and the vectors P_i corresponding to $x_i > 0$ are linear independent, then x is called a basic feasible solution.

A vector x, which satisfies conditions (1.1), (1.2) and (1.3) is called a feasible and optimal solution.

We say that a vector $x = (x_1, \cdots, x_{n+m})^T$ is lexicographically positive ($x \succ 0$) if the first nonzero element of x is positive, x is lexicographically greater than y if $x-y \succ 0$.

We shall state a number of theorems concerning the linear programming problem (which is stated in (1.4)). Detail proof can be found in Gass' book [5] or in any introductory linear programming text book.

Theorem 1.1: The set of all feasible solutions to the linear programming problem is a convex set.

Theorem 1.2: The objective function assumes its maximum at an extreme point of the convex set K. If it assumes its maximum at more than one extreme point, then it takes on the same value for every convex combination of these points.

Theorem 1.3: If a set of linearly independent vectors

$P_1, P_2, \cdots, P_k$, $k \leq m$, can be found such that $x_{10}P_1 + x_{20}P_2 + \cdots + x_{k0}P_k = P_0$ for $x_{i0} \geq 0$, $i = 1,2,\cdots,k$, then the vector $x_0 = (x_{10}, x_{20}, \cdots, x_{k0}, 0,\cdots, 0)$ is an extreme point of the convext set of feasible solutions.

Theorem 1.4: Associated with every extreme point of K (the set of feasible solutions) is a set of m linearly independent vectors from the given set $P_1, P_2, \cdots, P_{n+m}$.

As a result, we reach the following conclusion: Every basic feasible solution corresponds to an extreme point of K.

II. THE SIMPLEX METHOD

The simplex method is a procedure of finding an extreme point of K and determining if it yields a maximum solution. If it does not, we then find another extreme point whose corresponding value of the objective function is no less than the preceding one. (The theoretical and computational development of the simplex method can be found in Gass's book [5].). The following is a brief summary of the procedure.

We shall assume that the linear programming problem is feasible, that every basic feasible solution is nondegenerate (it has exactly n zeros) and that we are given a basic feasible solution. Let the given solution be $x_0 = b = (x_{10}, x_{20},\cdots,x_{m0}) = (b_1,b_2,\cdots,b_m)$ and the associated set of linearly independent vectors be $P_{n+1},P_{n+2},\cdots,P_{n+m}$. We then have

$$x_{1,0}P_{n+1}+x_{2,0}P_{n+2}+\cdots+x_{m,0}P_{n+m}=P_0 \qquad (1.6)$$

$$x_{1,0}c_{n+1}+x_{2,0}c_{n+2}+\cdots+x_{m,0}c_{n+m}=z_0 \qquad (1.7)$$

where all $x_{i,0}>0$ for $i=1,\ldots,m$, z_0 is the corresponding value of the objective function for the given solution. Since $P_{n+1},\ldots,P_{n+m}$ are linearly independent, we can express every vector from the set $P_1,\ldots,P_n,P_{n+1},\ldots,P_{n+m}$ in terms of $P_{n+1},P_{n+2},\ldots,P_{n+m}$. Let P_j be given by

$$x_{1,j}P_{n+1}+x_{2,j}P_{n+2}+\cdots+x_{m,j}P_{n+m}=P_j,\ j=1,\ldots,n. \qquad (1.8)$$

and define

$$x_{1,j}c_{n+1}+x_{2,j}c_{n+2}+\cdots+x_{m,j}c_{n+m}=z_j,\ j=1,\ldots,n. \qquad (1.9)$$

Initially, the set of linearly independent vectors forms a $m\times m$ identity matrix, hence we have $x_{i,j}=a_{i,j}$ for $i=1,..,m$ and $j=1,2,\ldots,n$. Also, $x_{i,0}=b_i$ for $i=1,2,..,m$. Also $c_{n+1}=\ldots=c_{n+m}=0$, we then have $z_j-c_j=0-c_j=-c_j$ for $j=1,..,n$.

The initial simplex tableau will take the following general form:

			c_1	. . . .	c_n	c_{n+1}	. . .	c_{n+m}
i	Basis	c	P_0	P_1	P_n	P_{n+1}		P_{n+m}
1	P_{n+1}	c_{n+1}	b_1	$a_{1,1}$	$a_{1,n}$	1		0
2	P_{n+2}	c_{n+2}	b_2	$a_{2,1}$	$a_{2,n}$	0		0
.	.	.	.	.	.	.		.
.	.	.	.	.	.	.		.
.	.	.	.	.	.	.		.
.	.	.	.	.	.	.		.
m	P_{n+m}	c_{n+m}	b_m	$a_{m,1}$	$a_{m,n}$	0		1
m+1			z_0	(z_1-c_1)	. . . (z_n-c_n)	0	. . .	0

(1.10)

Once an initial computational tableau has been constructed, the simplex procedure calls for the successive iteration of :

1. Check for optimality. The current solution is optimal if $z_j - c_j \geq 0$ for $j=1,..,n$, otherwise, go to step 2.
2. We shall construct a new basis by inserting a new basic vector P_r in place of P_k,
   where P_k, P_r are selected corresponding to
   (i) $z_r - c_r = \underset{j=1,..n}{\text{Min.}} (z_j - c_j) < 0$,
   (ii) $a_{k0}/a_{kr} = \underset{i}{\text{Min.}} (a_{i0}/a_{ir})$ for $a_{ir} > 0$.
   If $a_{ir} \leq 0$ for all i, there exists no finite optimal solution, otherwise, go to step 3.
3. Transformation of the tableau including P_0 column and the (m+1)th row by the complete elimination procedure using x_{kr} as a pivot element.
4. Go to step 1.

Each such iteration produces a new basic feasible solution and eventually we shall obtain a maximum solution or determine an unbounded solution.

After rearranaging Tableau 1.1 , we get the tableau of Tucker formulation as follows :

		$-x_1$	$-x_2$	$\cdots$	$-x_n$
z	z_0	z_1-c_1	z_2-c_2	$\cdots$	z_n-c_n
x_{n+1}	b_1	a_{11}	a_{12}	$\cdots$	a_{1n}
x_{n+2}	b_2	a_{21}	a_{22}	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_{n+m}	b_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}
x_1	0	-1	0	$\cdots$	0
x_2	0	0	-1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_n	0	0	0	$\cdots$	-1

(1.11)

We define x_1, x_2, $\cdots$, x_n as nonbasic variables, x_{1+n}, $\cdots$, x_{n+m} as basic variables. The procedure of the simplex method in Tucker formulation tableau can be stated as follows:

1. If $z_j - c_j \geqslant 0$ for all j, then the maximum solution has been found. Otherwise, go to step 2.
2. We choose x_r as a new basic variable in place of x_k, where x_r, x_k are selected corresponding to
   (i) $z_r - c_r = \underset{j=1,\cdots n}{\text{Min.}} (z_j - c_j) < 0$
   (ii) $b_k/a_{kr} = \text{Min.} (b_i/a_{ir})$ for $a_{ir} > 0$.
3. Transformation of the tableau by the complete elimination procedure using a_{kr} as a pivot element.
4. Go to step 1.

EXAMPLE 1.1:

$$\text{Maximize} \quad z = 2x_1 + 4x_2 + x_3 + x_4,$$

$$\text{subject to} \quad x_1 + 3x_2 + x_4 \leq 4,$$

$$2x_1 + x_2 \leq 3,$$

$$x_2 + 4x_3 + x_4 \leq 3,$$

where $x_i \geq 0$ for $i = 1,2,3,4$.

SOLUTION:

After adding slack variables (t_1, t_2, t_3), the problem is expressed in (1.12).

		$-x_1$	$-x_2$	$-x_3$	$-x_4$
z	0	-2	-4	-1	-1
t_1	4	1	3	0	1
t_2	3	2	1	0	0
t_3	3	0	1	4	1
x_1	0	-1	0	0	0
x_2	0	0	-1	0	0
x_3	0	0	0	-1	0
x_4	0	0	0	0	-1

(1.12)

		$-x_1$	$-t_1$	$-x_3$	$-x_4$
z	16/3	-2/3	4/3	-1	1/3
t_1	0	0	-1	0	0
t_2	5/3	5/3	-1/3	0	-1/3
t_3	5/3	-1/3	-1/3	4	2/3
x_1	0	-1	0	0	0
x_2	4/3	1/3	1/3	0	1/3
x_3	0	0	0	-1	0
x_4	0	0	0	0	-1

(1.13)

In tableau (1.12) x_2 is selected to enter as a basic variable and t_1 is selected to leave. Then the complete elimination procedure is performed using 3 (circled in tableau 1.12) as the pivot element to derive the second tableau (1.13).

		$-x_1$	$-t_1$	$-t_3$	$-x_4$
z	23/4	-3/4	5/4	1/4	1/2
t_1	0	0	-1	0	0
t_2	5/3	5/3	-1/3	0	-1/3
t_3	0	0	0	-1	0
x_1	0	-1	0	0	0
x_2	4/3	1/3	1/3	0	1/3
x_3	5/12	-1/12	-1/12	1/4	1/6
x_4	0	0	0	0	-1

(1.14)

		$-t_2$	$-t_1$	$-t_3$	$-x_4$
z	13/2	9/20	11/10	1/4	7/20
t_1	0	0	-1	0	0
t_2	0	-1	0	0	0
t_3	0	0	0	-1	0
x_1	1	3/5	-1/5	0	-1/5
x_2	1	-1/5	2/5	0	2/5
x_3	1/2	1/20	-1/10	1/4	3/20
x_4	0	0	0	0	-1

(1.15)

In (1.15), since $z_j - c_j \geqslant 0$ for all j, so $x_1 = 1$, $x_2 = 1$, $x_3 = 1/2$ and $x_4 = 0$ is the solution and the optimal value is $z = 13/2$.

III. THE DUAL SIMPLEX METHOD

The dual simplex method is used to solve the linear programming problem, when no basic feasible solution can be found for the primal problem and problem is primal optimal.

If the primal problem is stated as (1.16), then its associated dual problem is stated as (1.17).

$$\text{Maximize} \quad f(x) = c\,x, \tag{1.16}$$

subject to $\quad \tilde{A}\,x = b,$

where $x_i \geqslant 0$ for $i=1,2,\cdots,n+m$, $\tilde{A}$ is a matrix of dimension $m \times (n+m)$, $b=(b_1,\cdots,b_m)$ and $c=(c_1,\cdots,c_n,c_{n+1},\cdots,c_{n+m})$.

$$\text{Minimize} \quad g(w) = w\,b = w_1b_1 + \cdots + w_mb_m, \tag{1.17}$$

subject to $\quad w\tilde{A} \leqslant c,$

where $w=(w_1,\cdots,w_m)$, $w\tilde{A}$ is a matrix of dimension $1 \times (m+n)$.

Associated with these problems is the following important theorem (The Duality Theorem).

Theorem 1.5: If either the primal or the dual problem has a finite feasible and optimal solution, then the other problem has a finite feasible and optimal solution, and the optimal values of the primal and the dual problems are equal. That is, Maximize $f(x)$ = Minimize $g(w)$.

Now, using this principle of duality, the procedure of dual simplex method in Tucker formulation tableau has been established for solving the dual problem.

1. Check for primal feasibility, i.e., whether $b_i \geq 0$ for all i. If $b_i \geq 0$ for all i, then the feasible and optimal solution has been found. Otherwise, go to step 2.
2. We choose x_k as a new basic variable in place of x_r, where x_r, x_k are selected corresponding to

   (i) $b_r = \text{Min.}\{b_i \mid b_i < 0\}$.

   (ii) $(z_k - c_k)/-a_{rk} = \underset{j}{\text{Min.}} \{(z_j - c_j)/-a_{rj}\}$ for $a_{rj} < 0$.
4. Transformation of the tableau by the complete elimination procedure using a_{rk} as a pivot element.
5. Go to step 1.

EXAMPLE 1.2:

Maximize $z = 2(-x_1) + 3(-x_2) + 2(-x_4)$

subject to $-3(-x_1) + (-2)(-x_2) + 1.(-x_3) + (-1)(-x_4) \geq 6,$

$2(-x_1) + 1.(-x_2) + (-2)(-x_3) + (-1)(-x_4) \geq 5,$

where $x_i \geq 0$ for $i = 1,2,3,4.$

SOLUTION:

		$-x_1$	$-x_2$	$-x_3$	$-x_4$
z	0	2	3	0	2
t_1	-6	(-3)	-2	1	-1
t_2	-5	2	1	-2	-1
x_1	0	-1	0	0	0
x_2	0	0	-1	0	0
x_3	0	0	0	-1	0
x_4	0	0	0	0	-1

(1.18)

		$-t_1$	$-x_2$	$-x_3$	$-x_4$
z	-4	2/3	5/3	2/3	4/3
t_1	0	-1	0	0	0
t_2	-9	2/3	-1/3	(-4/3)	-5/3
x_1	2	-1/3	2/3	-1/3	1/3
x_2	0	0	-1	0	0
x_3	0	0	0	-1	0
x_4	0	0	0	0	-1

(1.19)

		$-t_1$	$-x_2$	$-t_2$	$-x_4$
z	-17/2	1	3/2	1/2	1/2
t_1	0	-1	0	0	0
t_2	0	0	0	-1	0
x_1	17/4	-1/2	3/4	-1/4	3/4
x_2	0	0	-1	0	0
x_3	0	0	0	-1	0
x_4	0	0	0	0	-1

(1.20)

In tableau(1.18) $z_j - c_j \geq 0$ for all j. The problem is primal optimal and we want to obtain primal feasibility. The dual simplex method is used to solve the problem. In (1.18) x_1 is selected to enter as a basic variable to replace t_1. In (1.19), x_3 is selected to enter as a basic variable to replace t_2. In tableau(1.19)bi ≥ 0 for all i, hence $x_1 = 17/4$, $x_2 = 0$, $x_3 = 27/4$, $x_4 = 0$ is the solution and the optimal value is $z = -17/2$.

The simplex method can be used on any problem with a feasible but not optimal solution in the initial tableau. The dual simplex method is just the opposite of the simplex method with the initial tableau for the dual simplex method having an optimal, but infeasible, solution and then maintaining optimality while gradually making the solution feasible.

CHAPTER II

THE PRIMAL INTEGER PROGRAMMING ALGORITHM

The term "primal" applied to an integer programming algorithm denotes a method that proceeds to an optimal solution through a sequence of better solutions that are all primal feasible. For the integer linear programming problem, we want to maintain the integer value properties of the complete tableau to avoid the redundancy for computer coding. So we modify the simplex method and the dual simplex method.

Gomory"s All Integer Algorithm is a variant of the dual simplex method and Young's Primal Integer Programming Algorithm [9] is a variant of the simplex method. The all integer algorithm generates a sequence of dual feasible,all integer solutions and the primal integer programming algorithm generates a sequence of primal feasible, all integer solutions. In the primal integer programming approach, we start with a primal feasible integer solutinn, then proceed iteratively, if possible, until an optimum solution is attained or else the algorithm can be terminated at any point with the best solution obtained.

Both the all integer algorithm and the primal integer programming algorithm are cutting-plane methods. In the iteration, if the new solution is an integer value, the process stops. If the new solution is not an integer value, a new constraint (cutting-plane) is then introduced and the

algorithms are repeated until an integer solution is found or until an indication is produced that no integer solution exists. Both algorithms preserve the integer value characteristic of the complete tableau for all iterations.

In this chapter we shall discuss two primal all-integer programming algorithms: the rudimentary primal algorithm and the simplified primal algorithm. The simplified primal algorithm differs essentially from the rudimentary primal algorithm at only two choice points: selection of the pivot column and selection of the source row for constraint cut. The primal purpose of the additional restriction that characterizes the simplified primal algorithm is to provide a foundation for a proof of finiteness.

I. THE RUDIMENTARY PRIMAL ALGORITHM

Assume that we are given the following bounded and solvable integer programming problem:

$$\begin{aligned} \text{Maximize} \quad & z = a_{00} - a_{01}x_1 - \cdots - a_{0n}x_n, \\ \text{subject to} \quad & a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1, \\ & a_{21}x_1 + \cdots + a_{2n}x_n \leq b_2, \\ & \qquad \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m, \end{aligned} \tag{2.1}$$

where

$$x_i \geq 0 \text{ and integer for } i = 1,2,\cdots,n.$$

After adding slack variables $t^T = (t_1, t_2, \cdots, t_m)^T$, the previous problem can be stated as follows:

$$\text{Maximize} \quad z = a_{00} - a_{01} x_1 - \cdots - a_{0n} x_n ,$$

$$\text{Subject to} \quad a_{11} x_1 + \cdots + a_{1n} x_n + t_1 + 0 \cdots = b_1,$$

$$a_{21} x_1 + \cdots + a_{2n} x_n + 0 + t_2 + 0 = b_2,$$

$$\vdots$$

$$a_{m1} x_1 + \cdots + a_{mn} x_n + 0 + 0 \cdots + t_m = b_m,$$

where $x_i \geq 0$, $t_j \geq 0$ and integer for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. (2.2)

The tableau form of Tucker formulation is as follows:

	1	$-x_1$	$-x_2$	$\cdots$	$-x_n$
z	a_{00}	a_{01}	a_{02}	$\cdots$	a_{0n}
t_1	b_1	a_{11}	a_{12}	$\cdots$	a_{1n}
t_2	b_2	a_{21}	a_{22}	$\cdots$	a_{2n}
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
t_m	b_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}
x_1	0	-1	0	$\cdots$	0
x_2	0	0	-1	$\cdots$	0
.	.	.	.	.	.
.	.	.	.	.	.
x_n	0	0	0		-1

(2.3)

For the convenience of discussion we rearrange the system of equations (2.1) as follows :

$$\begin{aligned} &\text{Maximize} \quad z = x_0 , \\ &\text{Subject to} \quad x_0 = a_{00} - \sum_{j=1}^{n} a_{0j} x_j , \\ &\qquad\qquad t_i = b_i - \sum_{j=1}^{n} a_{ij} x_j , \end{aligned} \tag{2.4}$$

where $x_j \geq 0$ and integer for $j=1,2,\cdots,n$,

$t_i \geq 0$ and integer for $i=1,2,\cdots,m$.

The matrix of tableau form is as follows :

$$I_{m+1} Z_B + A Z_N = A_0 , \tag{2.5}$$

where $Z_B^T = (x_0, t_1, \cdots, t_m)$,

$Z_N^T = (x_1, x_2, \cdots, x_n)$,

$A_0^T = (a_{00}, b_1, \cdots, b_m) = (a_{00}, a_{10}, \cdots, a_{m0})$.

I_{m+1} is an identity matrix of dimension m+1, and A is an (m+1 by n) matrix.

A typical subsequent tableau will be symbolized by

$$I_{m+1} Z_B + A Z_N = A_0 \tag{2.6}$$

In the matrix equations (2.5) and (2.6) Z_B is called a set of basic variables and Z_N is called a set of nonbasic variables. Since the primal integer programming algorithm is a cutting plane method, it routinely adds variables and constraints to the system. The set of nonbasic variables is a set which contains either original variables x_i or

subsequently adjoined slack variables S_I and it changes from tableau to tableau.

For the convenience of calculation, we drop constraint cut from the tableau before adding another constraint cut. We use $\mathcal{J}$ to symbolize the index set of nonbasic variables in (2.6); and $\mathcal{V}$ to represent the index set of rows in (2.6). We assume that $A_0 \geqslant 0$ and that A and A_0 consist entirely of integer constants. In the following description of the rudimentary primal algorithm, we use the symbol [y] to denote the integer part of y. We use index J to denote the index of the pivot column and we let A_j denote the column of A in (2.6) that corresponds to a nonbasic variable u_j.

The Rudimentary Primal Algorithm

1. Check for optimality. The current solution is optimal if $a_{0j} \geqslant 0$ for all $j \in \mathcal{J}$, otherwise go to step 2.
2. Select some column A_J, for which $a_{0J} < 0$ as the pivot column.
3. Check for an unbounded solution. If $a_{iJ} \leqslant 0$ for all i there exists no finite optimal solution, otherwise go to step 4.
4. Calculate the pivot ratio $\theta_J = \underset{a_{iJ}>0}{\text{Min}} (a_{i0}/a_{iJ})$. Select as the source row for a constraint cut some row v for which $0 \leqslant [a_{v0}/a_{vJ}] \leqslant \theta_J$.
5. Adjoin to the tableau the following constraint cut which is generated from the source row v. If we set the index of the cut row as I, then the constraint equation

is as follows :

$$S_I + \sum_{j \neq J} [a_{vj}/a_{vJ}]\, u_j = [a_{v0}/a_{vJ}].$$

We set $b_I = [a_{v0}/a_{vJ}]$ and $a_{Ij} = [a_{vj}/a_{vJ}]$.

6. Transformation of the tableau: Use A_J as the pivot column and the S_I row as the pivot row.

$$\left.\begin{aligned} \hat{a}_{i0} &= a_{i0} - a_{iJ}\, b_I \\ \hat{a}_{ij} &= a_{ij} - a_{iJ}\, a_{Ij} \qquad j \neq J \\ \hat{a}_{iJ} &= - a_{iJ} \end{aligned}\right\} \quad i = 0,1,\cdots,m+n \qquad (2.7)$$

Here we use symbol $\wedge$ to distinguish the data and the index set of the subsequent tableau.

7. Go to step 1.

In the rudimentary primal algorithm if $[a_{v0}/a_{vJ}] \geq 1$, then the cycle is called a transition cycle. If $[a_{v0}/a_{vJ}] < 1$, then the cycle is called a stationary cycle. In a transition cycle, a_{00} is increased by the amount $[a_{v0}/a_{vJ}] \cdot (-a_{0J}) \geq 1$. a_{00} is unchanged under stationary cycles. To obtain a finite algorithm, it is sufficient to augment the rudimentary primal algorithm in three main respects :

(i) Augment step 2 with a test to determine whether a transition cycle can be executed. If a transition cycle is possible, then the above procedure can be carried out without essential modification. If a transition cycle is not possible, then the above procedure must be modified.

(ii) Augment step 2 by introducing an additional restriction (consistent with $a_{0J} < 0$) on the selection of the nonbasic variable of the corresponding pivot column.

(iii) Augment step 4 by additional restrictions on the selection of the source row. These restrictions will be discussed at a later point.

II. GEOMETRICAL INTERPRETATION OF THE PRIMAL INTEGER PROGRAMMING ALGORITHM

To gain further insight into the pivot operation, let us consider the geometrical interpretation of the problem. We shall examine a two-dimensional example.

EXAMPLE 2.1:

$$\begin{aligned} \text{Maximize} \quad & z = 2x_1 + x_2, \\ \text{subject to} \quad & x_1 - x_2 \leq 3, \\ & -5x_1 + 3x_2 \leq 15, \\ & 2x_1 + 3x_2 \leq 30, \end{aligned} \tag{2.8}$$

where $x_1, x_2 \geq 0$ and integers.

SOLUTION:

After adding slack variables (t_1, t_2, t_3), we get :

$$\begin{aligned} t_1 &= 3 + 1 \cdot (-x_1) - 1 \cdot (-x_2), \\ t_2 &= 30 + 2 \cdot (-x_1) + 3 \cdot (-x_2), \\ \text{and} \quad t_3 &= 15 - 5 \cdot (-x_1) + 3 \cdot (-x_2). \end{aligned}$$

The problem has the following tableau form :

		$-x_1$ ↓	$-x_2$	
z	0	-2	-1	
x_1	0	-1	0	
x_2	0	0	-1	
t_1	3	(1)	-1	←
t_2	30	2	3	
t_3	15	-5	3	
s_1	3	1	-1	

(2.9)

		$-s_1$	$-x_2$ ↓	
z	6	2	-3	
x_1	3	1	-1	
x_2	0	0	-1	
t_1	0	-1	0	
t_2	24	-2	(5)	←
t_3	30	5	-2	
s_2	4	-1	1	

(2.10)

In tableau (2.9) x_1 is selected to enter as a basic variable and t_1 is selected to leave. We proceed the complete elimination procedure using $(-x_1)$ column as the pivot column and s_1 row as the pivot row to obtain tableau(2.10).In tableau (2.9), the element 1 is circled to denote $(-x_1)$ column as the pivot column and t_1 row as the source row. The primal integer programming algorithm then proceeds to tableau(2.11) and tableau (2.12).

		$-s_1$ ↓	$-s_2$	
z	18	-1	3	
x_1	7	0	1	
x_2	4	-1	1	
t_1	0	-1	0	
t_2	4	(3)	-5	←
t_3	38	3	2	
s_3	1	1	-2	

(2.11)

		$-s_3$	$-s_2$
z	19	1	1
x_1	7	0	1
x_2	5	1	-1
t_1	1	-1	-2
t_2	1	-3	1
t_3	35	-3	8

(2.12)

In tableau(2.12) since $z_j - c_j \geq 0$ for all $j \in \mathcal{J}$, hence $x_1 = 7$, $x_2 = 5$ is the solution and optimal value is 19.

The set of feasible solutions of problem (2.8) not including the condition that x_1 and x_2 are integers is a polygon bounded by constraints $x_1 \geq 0$ and $x_2 \geq 0$. If x_1 and x_2 are restricted to be integers, then the set of feasible solutions contains all integer points that are inside or on the boundary of the polygon.

When the cut intersects the edge of polygon at a new integer point, a transition cycle results; otherwise, it intersects the edge at the current solution point and we have a stationary cycle. When a transition cycle occurs, the feasible solution point moves to a new point where the cut intersects the edge of polygon. In a sequence of stationary cycles the algorithm is generating cuts until it finds a cut which intersects the edge of polygon at a new feasible integer point.

Figure 2.1 illustrates the progress of the algorithm in solving the above mentioned problem (2.8).

(1) At point $P_0(0,0)$, the cut 1, $s_1 = 3 - x_1 + x_2 \geq 0$, is generated, and the cut intersects the polygon at a new integer point (3,0). So the feasible solution point moves from (0,0) to (3,0).

(2) At point $P_1(3,0)$, the cut 2, $s_2 = 4 + s_1 - x_2$ $= 4 + (3 - x_1 + x_2) - x_2 = 7 - x_1 \geq 0$, is generated. The cut 1 intersects the polygon at a new integer point (7,4). So the feasible solution point moves from (3,0) to (7,4).

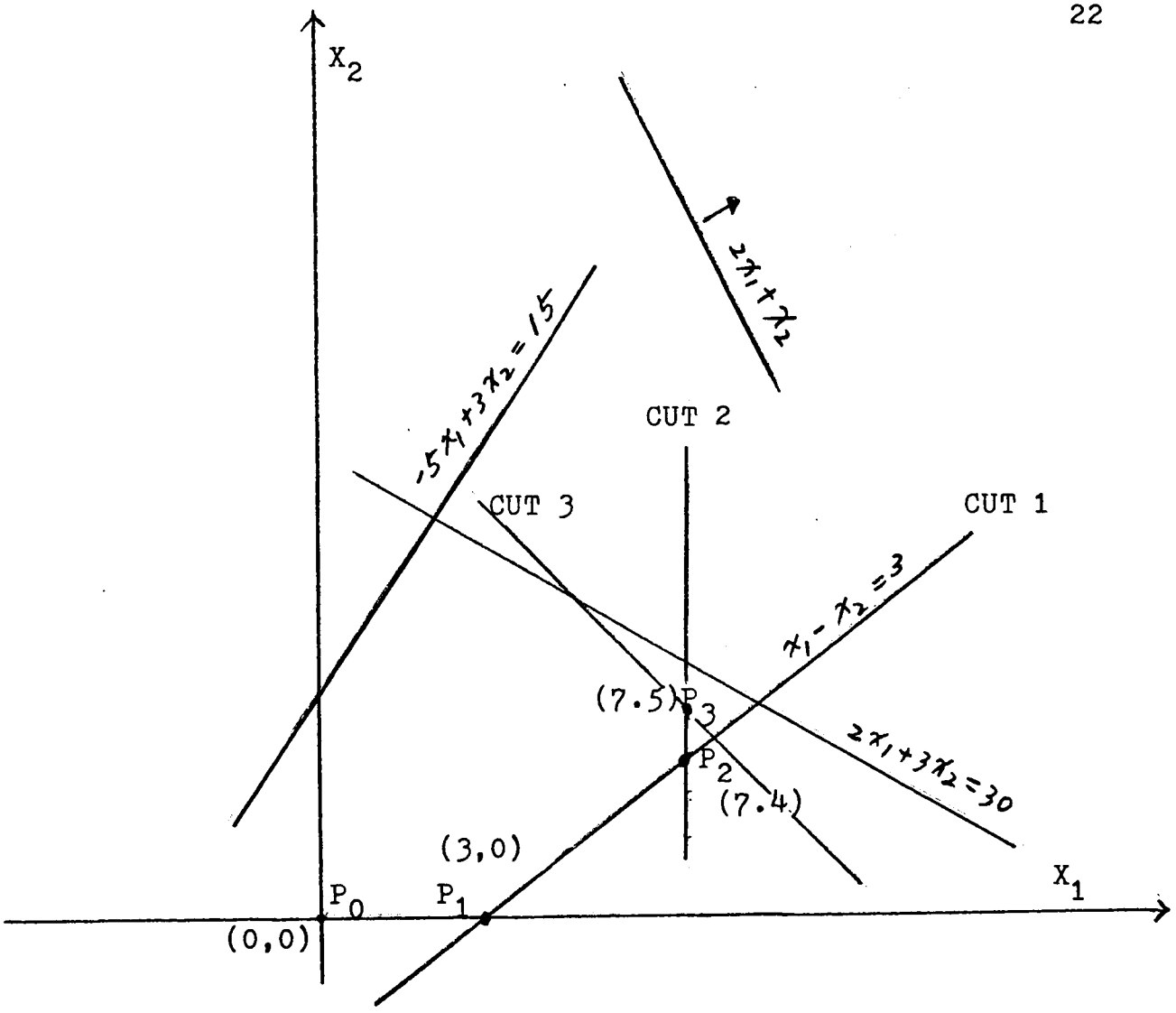


Figure 2.1 Graphical interpretation revealing the solution of the Integer Linear Programming problem (2.8)

(3) At point P_2, (7,4), the cut 3, $s_3 = 1 - s_1 + 2 s_2 = 12 - x_1 - x_2 \geq 0$ is generated. The cut 3 intersects the polygon at a new integer point (7,5). So the feasible solution point moves from (7,4) to (7,5).

(4) The tableau (2.12) reveals that point P_3, (7,5), is optimal.

III. THE SIMPLIFIED PRIMAL ALGORITHM

We have already indicated that the simplified primal algorithm is distinguished from the rudimentary primal algorithm by more restricted rule for pivot column and source row selection. The essential new requirement is satisfied by the following procedure. We adjoin a new constraint to the original Tableau (2.9). The constraint is:

$$\sum_{j=1}^{n} x_j \leq a_{L0} \tag{2.13}$$

This is merely a limit on the sum of the nonbasic variables. The positive constant a_{L0} is chosen sufficiently large to preserve all feasible integer solutions to (2.1). After the first transition cycle the constraint (2.13)is dropped from the system and replaced by the following constraint :

$$\sum_{j \in \mathcal{J}} u_j \leq a_{L0} \tag{2,14}$$

Following each succeeding transition cycle we drop the L row that was adjoined after the previous transition cycle and adjoin a new L row (2.14) with the index set and the constant a_{L0} determined from the current tableau. The reason for the above change is the basis for proving finiteness of the simplified primal algorithm.

The typical tableau will have the form :

$$x_0 + \sum_{j \in \mathcal{J}} a_{0j} u_j = a_{00}$$

$$t_i + \sum_{j \in \mathcal{J}} a_{ij} u_j = b_i \quad (i=1,2,\cdots,m) \qquad (2.15)$$

$$x_L + \sum_{j \in \mathcal{J}} a_{Lj} u_j = a_{L0}$$

where x_L is the slack variable for (2.14) and $a_{Lj} = 1$ for all $j \in \mathcal{J}$ in the original tableau and in each tableau that results from a transition cycle.

Selection of the pivot column in a stationary cycle of the simplified primal algorithm is influenced by the row L. Before describing the pivot column selection rule we need the following definition. For $j \in \mathcal{J}$ we define :

$$R_j^T = (a_{0j}/a_{Lj},\ a_{1j}/a_{Lj},\ \cdots,\ a_{m+n,j}/a_{Lj},\ a_{Lj}/a_{Lj})$$

The pivot column is selected in stationary cycles of the simplified primal algorithm by requiring that $a_{Lj} > 0$ and $R_J \overset{\mathcal{L}}{\prec} R_k$ for all $k \in \mathcal{J}$ with $a_{Lk} > 0$.

The simplified primal algorithm does not prescribe a deterministic rule for source row selection, The simplified primal algorithm only requires that the source row selection be accomplished in compliance with an appropriate source row selection rule. The appropriate source row selection rule is generated on the basis of Theorem 2.1 below.

Theorem 2.1: If on some iterations row v is selected as the source row and column J as the pivot column then the cut

$$S_I = \lceil b_v/a_{vJ} \rceil - \lceil a_{v1}/a_{vJ} \rceil u_1 - \cdots - u_J - (\cdots - \lceil a_{vn}/a_{vJ} \rceil u_n \qquad (2.16)$$

is the pivot row and the coefficients of row v after pivot operation satisfy :

$$\text{(i)}\quad \hat{a}_{vJ} = -\,a_{vJ} \qquad (2.17)$$

$$\text{(ii)}\quad a_{vJ} > \hat{a}_{vj} \geq 0 \quad \forall\, j \neq J.$$

Proof: (i) is trivial.

(ii) By pivot operation, $\hat{a}_{vj} = a_{vj} - [a_{vj}/a_{vJ}]\cdot a_{vJ}$, Since $(a_{vj}/a_{vJ}) \geq [a_{vj}/a_{vJ}] \geq (a_{vj}/a_{vJ}) - 1$, (ii) above follows immediately by substitution of these bounds for $[a_{vj}/a_{vJ}]$.

To connect the Theorem with our criterion for source row selection rules we begin by noting that in a stationary cycle $[a_{v0}/a_{vJ}] = 0$, so that $\hat{a}_{v0} = a_{v0} = \underline{a}_{v0}$, where $\underline{a}_{v0}$ is the value in the constant column of tableau that resulted from the most recent transition cycle. Now if $a_{vJ} - \underline{a}_{v0} = t > 0$, then as a consequence of (2.17) we have, for any selection of $\hat{J}$, $\hat{a}_{v\hat{J}} - \underline{a}_{v0} = \hat{t} < t$. Therefore, if row v is repeatedly selected as the source row so long as $a_{vJ} > a_{v0} = \underline{a}_{v0}$, where J is the index of the pivot column, then a finite sequence of such selection must lead to a tableau in which $a_{vJ} \leq \underline{a}_{v0}$.

Appropriate Source Row Selection Rule

(a) Select as the source row $v \in V(u_J) = \{i \mid 0 \leq a_{i0}/a_{iJ} \leq \theta_J\}$ that has been a member of $V(u_J)$ through the longest sequence of immediate preceeding tableaus, break ties by an arbitrary selection. Go to (b).

(b) Repeatedly select the row v chosen in (a) so long as $v \in V(u_J)$. In the first tableau for which $v \notin V(u_J)$, go to (a).

The Simplified Primal Algorithm

0. Adjoin the L row $\left(\sum_{j=1}^{n} x_j \leq a_{L0}\right)$. Go to step 1.

½. Revise the L row $\left(\sum_{j \in \mathcal{J}} u_j \leq a_{L0}\right)$. Go to step 1.

1. Check for optimality. The current solution is optimal if $a_{0j} \geq 0$ for all $j \in \mathcal{J}$, otherwise go to step 2a.

2a. For each $j \in \mathcal{J}$ that has $a_{Lj} > 0$ and $a_{0j} < 0$, generate the vector R_j and select A_J such that $a_{LJ} > 0$ and $R_J \overset{L}{\leq} R_k$ for all k with $a_{Lk} > 0$. Go to step 2b.

2b. Check for a transition cycle by calculating pivot ratio $\theta_J = \min(a_{i0}/a_{iJ})$, for $a_{iJ} > 0$ and $i \in \mathcal{I}$. If $a_{iJ} \leq 0$ for $i \in \mathcal{I}$, then there is no bounded solution. If $\theta_J \geq 1$, then go to step 3a, otherwise, go to 3b.

3a. Generate the set $V(u_J) = \left\{ i \mid 0 \leq a_{i0}/a_{iJ} \leq \theta_J \right\}$. Select any row $v \in V(u_J)$ as the source row for a constraint cut. Go to step 4.

3b. Generate the set $V(u_J) = \left\{ i \mid a_{iJ} > a_{i0} \right\}$. Use an appropriate source row selection rule to select some $v \in V(u_J)$ as the source row for a constraint cut. Go to step 4.

4. Adjoin to the tableau the following constraint cut generated from the source row v :

$$S_I + \sum_{j \in \mathcal{J}} \left[a_{vj}/a_{vJ} \right] u_j = \left[a_{v0}/a_{vJ} \right].$$

and we set $b_I = [a_{v0}/a_{vJ}]$ and $a_{Ij} = [a_{vj}/a_{vJ}]$.

5. Transformation of tableau : Use A_J as the pivot column and S_I row as the pivot row.

$$\left.\begin{aligned} \hat{a}_{i0} &= a_{i0} - a_{iJ}\, b_I \\ \hat{a}_{ij} &= a_{ij} - a_{iJ}\, a_{Ij} \quad j \neq J \\ \hat{a}_{iJ} &= - a_{iJ} \end{aligned}\right\} \quad i = 0,1,\cdots,m+n,L \qquad (2.18)$$

6. If the cycle just completed was a transition cycle, go to step ½; if the cycle was a stationary cycle go to step 1.

EXAMPLE 2.2:

$$\begin{aligned} \text{Maximize} \quad z = {} & 4x_1 + 6x_2 + 3x_3, \\ \text{subject to} \quad & x_1 + 2x_2 \leq 5, \\ & 9x_1 + 2x_2 - 4x_3 \leq 8, \\ & -3x_1 - 2x_2 + 2x_3 \leq 1, \\ & -5x_1 + 4x_2 + 6x_3 \leq 16, \end{aligned} \qquad (2.19)$$

where $x_i \geq 0$ and are integers for i=1,2,3.

SOLUTION:

Adding slack variables (t_1, t_2, t_3, t_4), we get :

$$\begin{aligned} t_1 &= 5 + 1 \cdot (-x_1) + 2 \cdot (-x_2), \\ t_2 &= 8 + 9 \cdot (-x_1) + 2 \cdot (-x_2) - 4 \cdot (-x_3), \\ t_3 &= 1 + (-3) \cdot (-x_1) - 2 \cdot (-x_2) + 2 \cdot (-x_3), \\ t_4 &= 16 + (-5) \cdot (-x_1) + 4 \cdot (-x_2) + 6 \cdot (-x_3). \end{aligned}$$

		$-x_1$	$-x_2$ ↓	$-x_3$	
z	0	-4	-6	-3	
t_1	5	1	(2)	0	←
t_2	8	9	2	-4	
t_3	1	-3	-2	2	
t_4	16	-5	4	6	
x_1	0	-1	0	0	
x_2	0	0	-1	0	
x_3	0	0	0	-1	
x_L	50	1	1	1	
s_1	2	0	1	0	

$V(x_2) = \{ t_1 \text{ row} \}$

(2.20)

		$-x_1$ ↓	$-s_1$	$-x_3$	
z	12	-4	6	-3	
t_1	1	1	-2	0	
t_2	4	(9)	-2	-4	←
t_3	5	-3	2	2	
t_4	8	-5	-4	6	
x_1	0	-1	0	0	
x_2	2	0	-1	0	
x_3	0	0	0	-1	
x_L	50	1	1	1	
s_2	0	1	-1	-1	

$V(x_1) = \{ t_2 \text{ row} \}$

(2.21)

		$-s_2$	$-s_1$	$-x_3$ ↓	
z	12	4	2	-7	
t_1	1	-1	-1	1	
t_2	4	-9	7	(5)	←
t_3	5	3	-1	-1	
t_4	8	5	-9	1	
x_1	0	1	-1	-1	
x_2	2	0	1	0	
x_3	0	0	0	-1	
x_L	50	-1	2	2	
s_3	0	-2	1	1	

$V(x_3) = \{ t_2 \text{ row} \}$

(2.22)

		$-s_2$ ↓	$-s_1$	$-s_3$	
z	12	-10	9	7	
t_1	1	(1)	-2	-1	←
t_2	4	1	2	-5	
t_3	5	1	0	1	
t_4	8	7	-10	-1	
x_1	0	-1	0	1	
x_2	2	0	1	0	
x_3	0	-2	1	1	
x_L	50	3	0	-2	
s_4	1	1	-2	-1	

$V(s_2) = \{ t_1, t_4 \text{ rows} \}$

(2.23)

		$-s_4$ ↓	$-s_1$	$-s_3$	
z	22	10	-11	-3	
t_1	0	-1	0	0	
t_2	3	-1	4	-4	
t_3	4	-1	2	2	
t_4	1	-7	(4)	1	←
x_1	1	1	-2	0	
x_2	2	0	1	0	
x_3	2	2	-3	-1	
x_L	50	1	1	1	
s_5	0	-2	1	1	

$V(s_1) = \{ t_2, t_4 \text{ rows} \}$

(2.24)

		$-s_4$ ↓	$-s_5$	$-s_3$	
z	22	-12	11	8	
t_1	0	-1	0	0	
t_2	3	(7)	-4	-8	←
t_3	4	3	-2	0	
t_4	1	1	-4	2	
x_1	1	-3	2	2	
x_2	2	2	-1	-1	
x_3	2	-4	3	2	
x_L	50	3	-1	0	
s_6	0	1	-1	-2	

$V(s_4) = \{ t_2 \text{ row} \}$

(2.25)

		$-s_6$	$-s_5$	$-s_3$ ↓	
z	22	12	-1	-16	
t_1	0	1	-1	-2	
t_2	3	-7	3	(6)	←
t_3	4	-3	1	6	
t_4	1	-1	-3	4	
x_1	1	3	-1	-4	
x_2	2	-2	1	3	
x_3	2	4	-1	-6	
x_L	50	-3	2	6	
s_7	0	-2	0	1	

$V(s_3) = \{ x_2, t_2, t_3, t_4 \text{ rows} \}$

(2.26)

		$-s_6$ ↓	$-s_5$	$-s_7$	
z	22	-20	-1	16	
t_1	0	-3	-1	2	
t_2	3	(5)	3	-6	←
t_3	4	9	1	-6	
t_4	1	7	-3	-4	
x_1	1	-5	-1	4	
x_2	2	4	1	-3	
x_3	2	-8	-1	6	
x_L	50	9	2	-6	
s_8	0	1	0	-2	

$V(s_6) = \{ x_2, t_2, t_3, t_4 \text{ rows} \}$

(2.27)

		$-s_8$	$-s_5$	$-s_7$ ↓	
z	22	20	-1	-24	
t_1	0	3	-1	-4	
t_2	3	-5	3	(4)	←
t_3	4	-9	1	12	
t_4	1	-7	-3	10	
x_1	1	5	-1	-6	
x_2	2	-4	1	5	
x_3	2	8	-1	-10	
x_L	50	-9	2	12	
s_9	0	-2	0	1	

$V(s_7)=\{x_2, t_2, t_3, t_4$ rows$\}$

(2.28)

		$-s_8$ ↓	$-s_5$	$-s_9$	
z	22	-28	-1	24	
t_1	0	-5	-1	4	
t_2	3	3	3	-4	
t_3	4	15	1	-12	
t_4	1	13	-3	-10	
x_1	1	-7	-1	6	
x_2	2	(6)	1	-5	←
x_3	2	-12	-1	10	
x_L	50	15	2	-12	
s_{10}	0	1	0	-1	

$V(s_8)=\{x_2, t_3, t_4$ rows$\}$

(2.29)

		$-s_{10}$	$-s_5$	$-s_9$ ↓	
z	22	28	-1	-4	
t_1	0	5	-1	-1	
t_2	3	-3	3	-1	
t_3	4	-15	1	3	
t_4	1	-13	-3	(3)	←
x_1	1	7	-1	-1	
x_2	2	-6	1	1	
x_3	2	12	-1	-2	
x_L	50	-15	2	3	
s_{11}	0	-5	-1	1	

$V(s_9) = \{t_4$ row$\}$

(2.30)

		$-s_{10}$	$-s_5$ ↓	$-s_{11}$	
z	22	8	-5	4	
t_1	0	0	-2	1	
t_2	3	-8	2	1	
t_3	4	0	4	-3	
t_4	1	2	0	-3	
x_1	1	2	-2	1	
x_2	2	-1	(2)	-1	←
x_3	2	2	-3	2	
x_L	50	0	5	-3	
s_{12}	1	-1	1	-1	

$V(s_{12})=\{x_2, t_2, t_3$ rows$\}$

(2.31)

		$-s_{10}$	$-s_{12}$	$-s_{11}$
z	27	3	5	-1
t_1	2	-2	2	-1
t_2	1	-6	-2	3
t_3	0	4	-4	1
t_4	1	2	0	-3
x_1	3	0	2	-1
x_2	0	1	-2	1
x_3	5	-1	3	-1
x_L	50	1	1	1
s_{13}	0	1	-2	1

$V(s_{11}) = \{x_2, t_2, t_3 \text{ rows}\}$

(2.32)

		$-s_{10}$	$-s_{12}$	$-s_{13}$
z	27	4	3	1
t_1	2	-1	0	1
t_2	1	-9	4	-3
t_3	0	3	-2	-1
t_4	1	5	-6	3
x_1	3	1	0	1
x_2	0	0	8	-1
x_3	5	0	1	1
x_L	50	0	3	-1

(2.33)

In Tableau (2.20), $-x_2$ column is selected as the pivot column and t_1 row is selected as the source row, then the constraint cut, s_1 row, is generated. After complete elimination by using s_1 row as the pivot row, we get tableau (2.21).

Tableau (2.24) through tableau (2.31) is a stationary cycle and the appropriate source row selection rule is used to select t_2 row as the source row in Tableau (2.26). In Tableau(2.33), since all $z_j - c_j \geq 0$, hence $x_1 = 3$, $x_2 = 0$, $x_3 = 5$ and $z = 27$ is the optimal solution. A flowchart for the simplified primal algorithm is given in Figure 2.2.

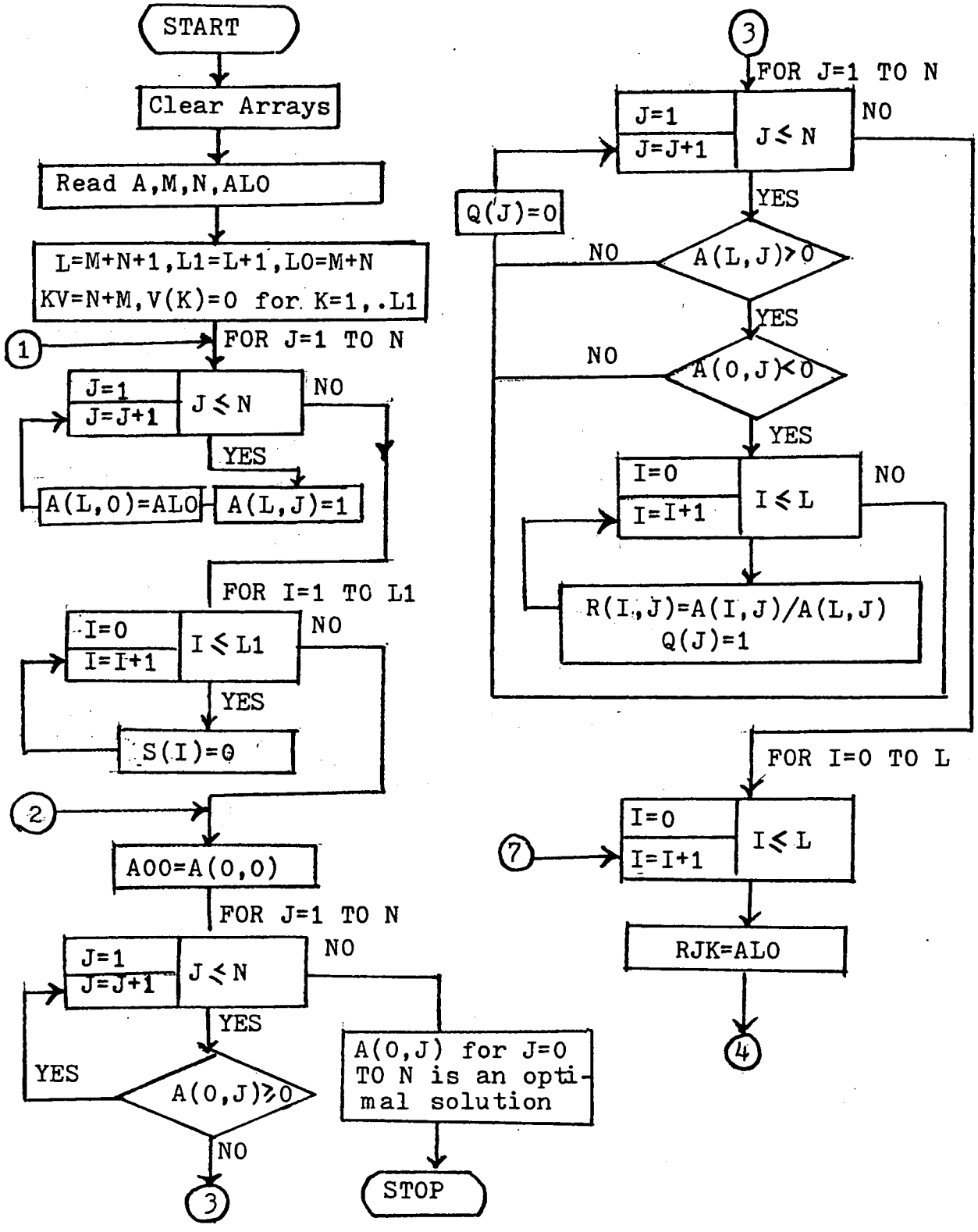


Figure 2.2 Flowchart for the simplified primal integer programming algorithm

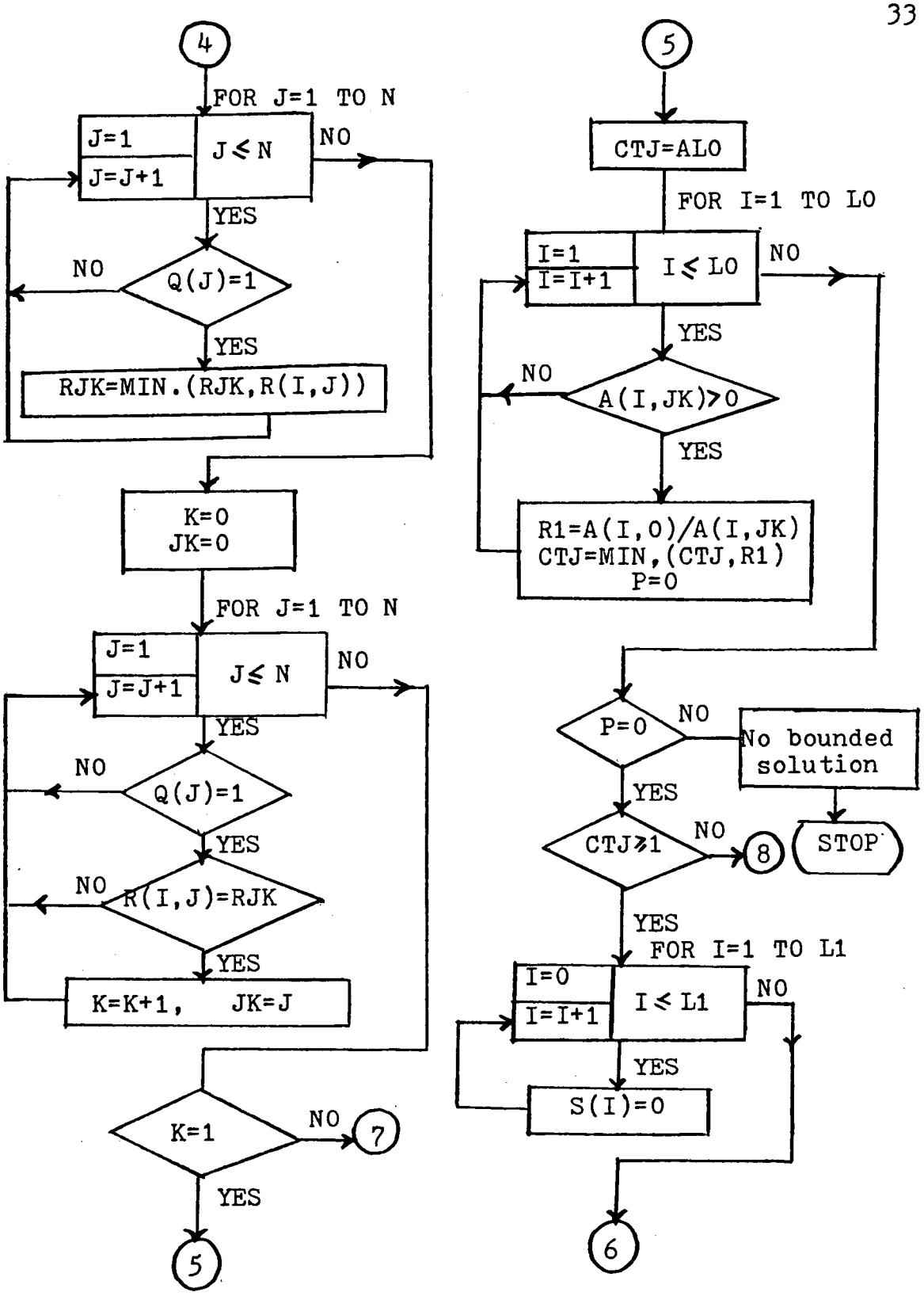


Figure 2.2 (continued)

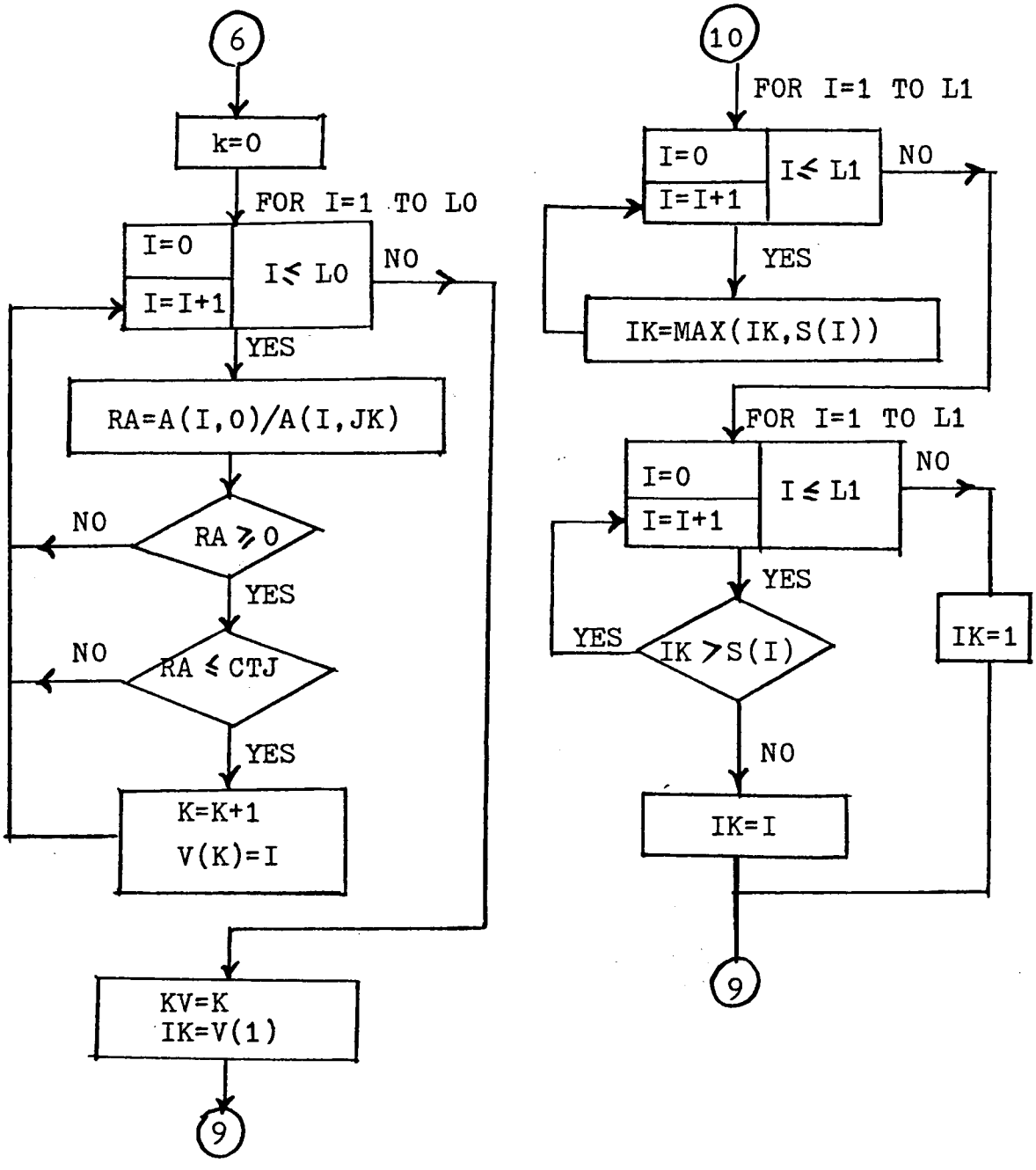


Figure 2.2 (continued)

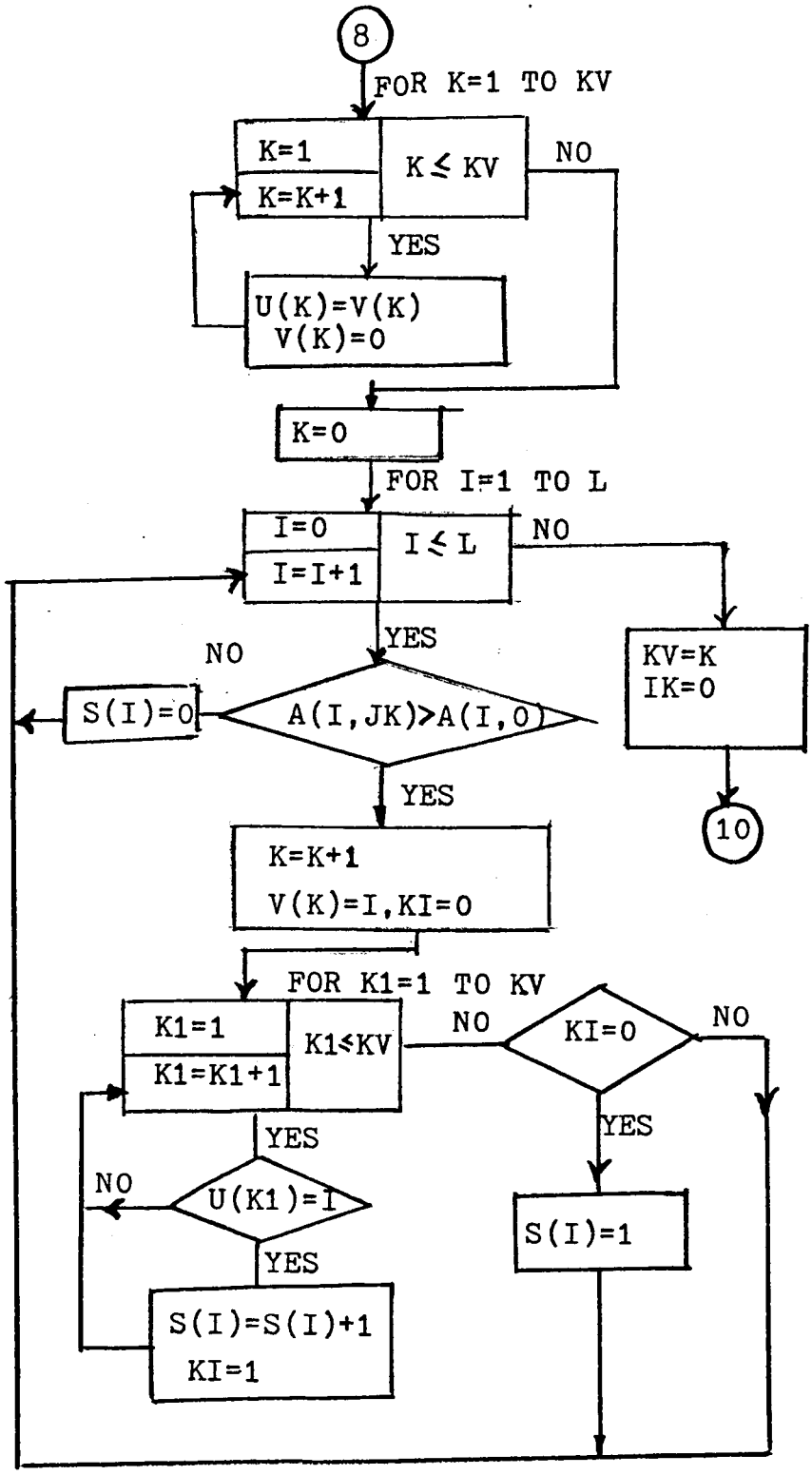


Figure 2.2 (continued)

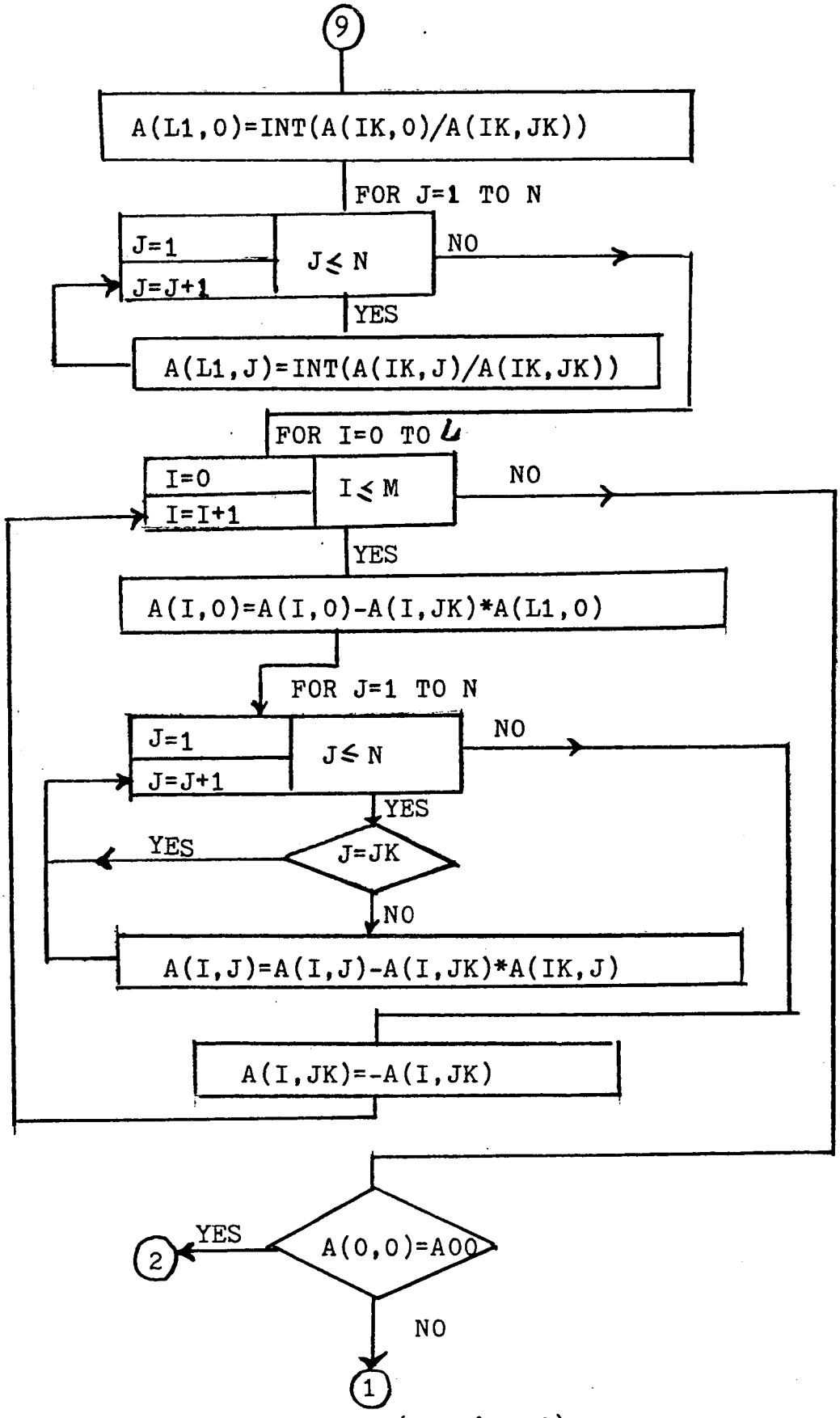


Figure 2.2 (continued)

IV. THE PROOF OF FINITENESS OF THE SIMPLIFIED PRIMAL ALGORITHM

The strategy of our finiteness proof is simple. We show that every sequence of stationary cycles must eventually - after a finite number of iterations - leads to an optimal tableau. This implies that if the basic solution that results from a given transition cycle is not an integer optimum then another transition cycle must occur in the finite interval before an optimal tableau has been obtained.

First we show that during any sequence of stationary cycles, the vector R_J (J is the index of the pivot column) must undergo a strictly lexicographic increase from tableau to tableau, then we show that the first component of R_J must increase at finite intervals and that a finite number of such increase is sufficient to yield a tableau in which the first component of R_J is nonnegative.

Our discussion here will focus on the relations connecting two typical successive tableaus in a sequence of stationary cycles. We rewrite typical tableau as follows:

$$I\, X_B + A\, U_N = A_0, \tag{2.34}$$

or,

$$\begin{aligned} x_0 + \sum_{j\in\mathcal{J}} a_{0j}u_j &= a_{00}, \\ t_i + \sum_{j\in\mathcal{J}} a_{ij}u_j &= a_{i0}, \quad (i=1,2,\cdots,m) \\ x_L + \sum_{j\in\mathcal{J}} a_{Lj}u_j &= a_{L0}. \end{aligned} \tag{2.35}$$

We shall let (2.34) above represent the first tableau and use the symbol $\wedge$ to distinguish the set, data and concepts from the second or successor tableau. We shall refer to the second tableau as $(\hat{2.34})$ or $\hat{A}$. It should be noted that $u_{\hat{J}}$ is the incoming variable selected from the second tableau, $\hat{A}_{\hat{J}}$ is the pivot column in the second tableau while $A_{\hat{J}}$ is the column in the first tableau corresponding to the variable $U_{\hat{J}}$.

Define $$\rho_i \equiv a_{iJ}/a_{LJ}, \qquad (i=0,1,2,\cdots,m,L) \tag{2.36}$$

then

$$R_J^T = (\rho_0, \rho_1, \cdots, \rho_m, \rho_L). \tag{2.37}$$

Define

$$d_{ij} = -\rho_i a_{Lj} + a_{ij}, \quad (i=0,1,\cdots,m,L; j\in\mathcal{J}) \tag{2.38}$$

and

$$\Delta_j^T = (d_{0j}, d_{1j}, \cdots, d_{mj}, d_{Lj}), \quad j\in\mathcal{J} \tag{2.39}$$

then

$$a_{Lj} R_J = A_j - \Delta_j, \qquad j\in\mathcal{J}. \tag{2.40}$$

The initial tableau in any sequence of stationary cycles has several properties that are important to inductive arguments that we shall soon develop. The initial tableau must result from a transition cycle or it must be the original tableau. In either case the initial tableau has $a_{Lj} = 1$ for $j\in\mathcal{J}$. This guarantees that R_J exists since we have $a_{Lj} > 0$ for some j. In addition the initial tableau has the following properties :

$$R_J \overset{\mathcal{L}}{\prec} 0, \tag{2.41}$$

$$\Delta_j \overset{\mathcal{L}}{\succ} 0, \qquad \text{for all } j\ (\neq J)\in\mathcal{J} \tag{2.42}$$

and

$$\Delta_J = 0. \tag{2.43}$$

The property (2.43) is definitional and holds for every tableau. In the initial tableau, since $a_{Lj} = 1$ for $j \in \mathcal{J}$, we have $A_j = R_J$ for all $j \in \mathcal{J}$. The rule of selection of A_J implies that $A_j \mathcal{L} A_J = R_J$. If (2.41) does not hold, then initial tableau must be optimal. Accordingly, in a nonoptimal initial tableau, (2.41) must hold. To establish (2.42) we use the relations $A_{Lj} = 1$ and $A_j = R_j$ as a basis for substitution into (2.40), which yields $\Delta_j = R_j - R_J$. The rule of selection of A_J implies that the right side of this expression must be lexicographically positive.

Now we shift our attention from the initial tableau in a sequence of stationary cycles to the first tableau (2.34) in any successive pair. We shall develop a simple relation connecting (2.34) with its successor $(\hat{2.34})$. The change of the basic formula for the simplified primal algorithm is

$$\hat{a}_{ij} = a_{ij} - a_{pj}\, a_{iJ}, \qquad (i=0,1,\cdots,m) \quad (2.44)$$

$$\hat{a}_{Lj} = a_{Lj} - a_{pj}\, a_{LJ}, \qquad (2.45)$$

where the index p denotes the pivot row and a_{pJ} is assumed to be equal to 1. We multiply (2.45) by ρ_i, which gives :

$$\rho_i \hat{a}_{Lj} = \rho_i a_{Lj} - a_{pj}\, \rho_i a_{LJ}.$$

Substituting into this expression on the basis of (2.38) and (2.36), we obtain :

$$\rho_i \hat{a}_{Lj} = a_{ij} - d_{ij} - a_{pj}\, a_{iJ}.$$

This can be simplified by substitution from (2.44) to yield :

$$\rho_i \hat{a}_{Lj} = \hat{a}_{ij} - d_{ij}. \qquad (2.46)$$

The vector form of (2.27) is :

$$\hat{a}_{Lj} R_J = \hat{A}_j - \Delta_j. \qquad (2.47)$$

This preliminary analysis is the basis for these theorems that provide the result $R_J \overset{\mathcal{L}}{\prec} \hat{R}_{\hat{J}}$.

Theorem 2.2:

Given that :

$$R_J \overset{\mathcal{L}}{\prec} 0,$$

$$\Delta_j \overset{\mathcal{L}}{\succ} 0 \quad \text{for all } j\ (\neq J) \quad ,$$

and $\hat{A}$ is not optimal,

then, there exists a $j \in \hat{\mathcal{J}}$ with $\hat{a}_{Lj} > 0$ and $\hat{R}_{\hat{J}} \overset{\mathcal{L}}{\prec} 0$.

Proof:

Suppose first that the hypothesis of Theorem 2.2 is true and all $j \in \hat{\mathcal{J}}$ have $\hat{a}_{Lj} \leq 0$. These conditions ($\hat{a}_{Lj} \leq 0$ for all $j \in \hat{\mathcal{J}}$, $\Delta_j \overset{\mathcal{L}}{\succ} 0$, and $R_J \overset{\mathcal{L}}{\prec} 0$) together with (2.47) implies that $\hat{A}_j \overset{\mathcal{L}}{\succ} 0$ for all $j \in \hat{\mathcal{J}}$. This contradicts the assumption that $\hat{A}$ is not optimal. Therefore there exists some $j \in \hat{\mathcal{J}}$ with $\hat{a}_{Lj} > 0$, and it is possible to define $\hat{R}_{\hat{J}}$. Now suppose that $\hat{R}_{\hat{J}} \overset{\mathcal{L}}{\succ} 0$. Then according to the argument we have just mentioned, all $j \in \hat{\mathcal{J}}$ with $\hat{a}_{Lj} \leq 0$ must have $\hat{A}_j \overset{\mathcal{L}}{\succ} 0$. But if $\hat{R}_{\hat{J}} \overset{\mathcal{L}}{\succ} 0$, all j with $\hat{a}_{Lj} > 0$ have $(1/\hat{a}_{Lj})\hat{A}_j = \hat{R}_j \overset{\mathcal{L}}{\succ} \hat{R}_{\hat{J}} \overset{\mathcal{L}}{\succ} 0$, which implies that $\hat{A}_j \overset{\mathcal{L}}{\succ} 0$ for these remaining columns. This again contradicts the assumption that $\hat{A}$ is not optimal. Hence $\hat{R}_{\hat{J}} \overset{\mathcal{L}}{\prec} 0$,

Theorem 2.3:

Given that :

$$R_J \overset{\mathcal{L}}{\prec} 0,$$

$$\Delta_j \overset{\mathcal{L}}{\succ} 0 \qquad \text{for all } j\ (\neq J) \in \mathcal{J},$$

and $\hat{A}$ is not optimal,

then $\hat{\Delta}_j \overset{\mathcal{L}}{\succ} 0$ for all $j\ (\neq \hat{J}) \in \hat{\mathcal{J}}$.

Proof:

This proof requires some further algebraic deduction as a preliminary. For the convenience of reference we rewrite (2.47) as :

$$\hat{a}_{Lj} R_J = \hat{A}_j - \Delta_j. \tag{2.48}$$

By analogy to (2.40) we write :

$$\hat{a}_{Lj} \hat{R}_{\hat{J}} = \hat{A}_j - \hat{\Delta}_j. \tag{2.49}$$

Subtracting (2.49) from (2.48), we have :

$$\hat{a}_{Lj} (R_J - \hat{R}_{\hat{J}}) = \hat{\Delta}_j - \Delta_j. \tag{2.50}$$

When $j = \hat{J}$, (2.48) becomes :

$$\hat{a}_{L\hat{J}} R_J = \hat{A}_{\hat{J}} - \Delta_{\hat{J}}. \tag{2.51}$$

Since $\hat{a}_{L\hat{J}} > 0$, we can divide (2.51) by this quantity and obtain :

$$R_J = \hat{R}_{\hat{J}} - (1 / \hat{a}_{L\hat{J}}) \Delta_{\hat{J}}. \tag{2.52}$$

We can use (2.52) to eliminate R_J and $\hat{R}_J$ from (2.50) which yields :

$$\hat{\Delta}_j = \Delta_j - (\hat{a}_{Lj}/\hat{a}_{L\hat{J}})\Delta_{\hat{J}}. \qquad (2.53)$$

Finally we can obtain a second expression for $\hat{\Delta}_j$ by using the relation $\hat{A}_j = \hat{a}_{Lj}\hat{R}_j$ (which is legitimate if $\hat{a}_{Lj} \neq 0$) to eliminate A_j from (2.49). The result is :

$$\hat{\Delta}_j = \hat{a}_{Lj}(\hat{R}_j - \hat{R}_{\hat{J}}). \qquad (2.54)$$

The theorem is an immediate consequence of (2.53) and (2.54). If $\hat{a}_{Lj} \leq 0$, $\hat{\Delta}_j \overset{\mathcal{L}}{>} 0$ follows from (2.53) and the hypothesis of the theorem. If $\hat{a}_{Lj} > 0$, then the relation (2.54) is valid and $\hat{\Delta}_j \overset{\mathcal{L}}{>} 0$, since the pivot column selection rule requires $\hat{R}_j \overset{\mathcal{L}}{>} \hat{R}_{\hat{J}}$.

Theorem 2.4:

Given that :

$$R_J \overset{\mathcal{L}}{<} 0,$$
$$\Delta_j \overset{\mathcal{L}}{>} 0 \quad \text{for all } j\ (\neq J) \in \mathcal{J}.$$

and $\hat{A}$ is not optimal, then

$$R_J \overset{\mathcal{L}}{<} \hat{R}_{\hat{J}}.$$

Proof:

The theorem follows immediately from (2.52), since $\Delta_{\hat{J}} \overset{\mathcal{L}}{>} 0$ and $\hat{a}_{L\hat{J}} > 0$ $(\hat{J} \neq J)$.

From the above theorems, we find that $R_J \overset{\mathcal{L}}{<} \hat{R}_{\hat{J}}$ was derived from properties of the pivot column selection rule. Now we want to show that a finite sequence of stationary cycles will lead to an optimal tableau. This property depends on the source row selection procedure and on the implication of $R_J \overset{\mathcal{L}}{<} \hat{R}_{\hat{J}}$.

We observe from Theorem 2.2 that the first tableau in which R_J is either undefined or lexicographically nonnegative must be optimal. This fact together with Theorem 2.5 and Theorem 2.6 below establish the finiteness of the simplified primal algorithm. In the statement and proof of these theorems we use underlined constants to denote data of the initial tableau in a sequence of stationary cycles.

Now we want to develop a simple definition and some elementary conclusions as follows. It will be useful as a preliminary to the proof of Theorem 2.5 and Theorem 2.6.

Let $\underline{a}_{i0}$ be the constant in row i and column 0 in the first tableau of a sequence of stationary cycles. Then for every tableau in the sequence of stationary cycles, $a_{i0} = \underline{a}_{i0}$ for all $i \in \psi$. For any component i of R_J, in any tableau, we define a_{iJ}/a_{LJ} to be within bounds if $a_{iJ} \leq a_{i0} = \underline{a}_{i0}$ and $a_{LJ} \leq a_{L0} = \underline{a}_{L0}$.

Now we note that if $a_{iJ}/a_{LJ} \leq \underline{a}_{i0}/\underline{a}_{L0}$, then $a_{LJ} \leq \underline{a}_{L0}$ implies that a_{iJ}/a_{LJ} is within bounds, and if $a_{iJ}/a_{LJ} \geq \underline{a}_{i0}/\underline{a}_{L0}$, then $a_{iJ} \leq \underline{a}_{i0}$ implies that a_{iJ}/a_{LJ} is within bounds. Finally, we observe that if a_{iJ} and a_{LJ} are always integers, if $a_{LJ} > 0$, if $a_{iJ}/a_{LJ} \geq r$ (a constant), and if a_{iJ}/a_{LJ} is within bounds, then there is only a finite number of possible values for the ratio a_{iJ}/a_{LJ}.

Theorem 2.5:

If an acceptable source row selcetion rule is used, then at finite intervals adjacent pairs of tableaux must occur for which $a_{0J}/a_{LJ} < \hat{a}_{0J}/\hat{a}_{LJ}$.

Proof:

Since $R_J \overset{\mathcal{L}}{\prec} \hat{R}_J$, we know that all components of R_J cannot remain fixed. Without loss of generality, we assume that the first component a_{0J}/a_{LJ} remains constant for an unbounded sequence S of stationary cycles, while a_{1J}/a_{LJ} increases at finite intervals. Then an unbounded number of increases in a_{1J}/a_{LJ} must occur. Let r be the value of a_{1J}/a_{LJ} in the first tableau, then $r \leq a_{1J}/a_{LJ}$ for all succeeding tableaux. Let S_1 consist of the tableaux in S for which $r \leq a_{1J}/a_{LJ} \leq \underline{a}_{10}/\underline{a}_{L0}$, and let S_2 consist of the tableaux in which $a_{1J}/a_{LJ} > \underline{a}_{10}/\underline{a}_{L0}$. Hence $S = S_1 \cup S_2$.

The source row selection rule requires that $a_{LJ} \leq \underline{a}_{L0}$ at finite intervals. Hence in S_1, a_{1J}/a_{LJ} must be within bounds at finite intervals whenever $a_{LJ} \leq \underline{a}_{L0}$, and the subsequence of S_1 during which a_{1J}/a_{LJ} is within bounds must be unbounded if S_1 is unbounded. But this unbounded subsequence of S_1 (during which a_{1J}/a_{LJ} is within bounds) must contain an unbounded number of increases in a_{1J}/a_{LJ}, since a_{1J}/a_{LJ} increases at finite intervals. This is impossible because there is only a finite number of within-bounds values for a_{1J}/a_{LJ}. Hence S_1 must be finite. An analogous argument shows that S_2 must be finite. Thus S must be finite which contradicts the assumption that the first compontnt of R_J remains fixed for an unbounded sequence of stationary cycles.

Theorem 2.6:

If an acceptable source row selection rule is used, then after a finite sequence of cycles, an optimal condition must

occur.

Proof:

Let r be the value for a_{0J}/a_{LJ} in the first tableau in a sequence of stationary cycles. Define a_{0J}/a_{LJ} to be within bounds whenever $a_{LJ} \leq \underline{a}_{L0}$. Then there is only a finite number of within-bounds and negative values for $\underline{a}'_{0J}/a_{LJ}$ if $a_{LJ} > 0$ and $a_{0J}/a_{LJ} \geq r$. The source row selection rule implies that a_{0J}/a_{LJ} will be within bounds at finite intervals. Since a_{0J}/a_{LJ} must also increase at finite intervals, a finite sequence of cycles is sufficient to exhaust the finite number of negative within-bounds values for the ratio a_{0J}/a_{LJ}. This implies, via Theorem 2.2, an optimal tableau. The remaining possiblity is that $a_{Lj} \leq 0$ for all $j \in \mathcal{J}$ in some tableaux. This also, by Theorem 2.2, implies optimality.

CHAPTER III

A PRIMAL - DUAL INTEGER PROGRAMMING ALGORITHM

In this chapter, we shall introduce a method to solve the integer linear programming problem which can be initially both primally and dually infeasible. The method is an application of the primal and dual pivot operations.

I. INTRODUCTION

Gomory's All Integer Algorithm works with dual feasibility until it gets primal feasibility. Young's Primal Integer Linear Programming Method works with primal feasibility until it gets dual feasibility. Ryan's [8] Primal and Dual Integer Programming Algorithm works with primal infeasibility and dual infeasibility and moves toward both primal feasibility and dual feasibility.

Let the integer linear programming problem be given as:

$$\text{Maximize} \qquad z = -c\,x,$$

$$\text{subject to} \qquad A\,x \leq b,$$

where $x \geq 0$ and integers, c is an n x 1 and b is an m x 1 vector.

Suppose on some iteration, we let

$$T_b = \min_{i=1,\cdots,m} b_i, \qquad T_c = \min_{j=1,\cdots,n} c_j,$$

and $T = \min.\{T_b, T_c\} < 0$; then if we define

$$\theta = -T - \varepsilon, \qquad \text{where } \varepsilon > 0 \text{ and } \varepsilon \text{ is arbitrarily}$$

small, we shall have $b_i + \theta > 0$ and $c_j + \theta > 0$ for every $b_i < 0$ and for every $c_j < 0$, with the exception of one b_i or c_j, say b_k. Then using the system with augmented terms we have a problem with primal infeasibility in row k which is dual feasible. A dual pivot will bring this system to primal feasibility. Thus, if we have the problem as :

Maximize $z = - c\,x$,

subject to $A\,x \leq b$, where $x \geq 0$ and integers,

and let

$$\tilde{b} = \left\{ b_i \mid \tilde{b}_i = b_i \text{ for } b_i \geq 0,\ \tilde{b}_i = b_i + \theta \text{ for } b_i < 0 \right\},$$

$$\tilde{c} = \left\{ c_j \mid \tilde{c}_j = c_j \text{ for } c_j \geq 0,\ \tilde{c}_j = c_j + \theta \text{ for } c_j < 0 \right\},$$

then the modified system :

Maximize $z = - \tilde{c}\,x$,

subject to $A\,x \leq \tilde{b}$, where $x \geq 0$ and integers,

has one primal infeasibility, which is removed by the dual pivot.

II. THE PRIMAL - DUAL INTEGER PROGRAMMING ALGORITHM

Since it is impossible to represent θ as an unknown algebraic value in a computer, and if θ were given values they would in general not be integers (thereby destroying the all integer properties of the integer linear programming algorithm). We have constructed a modification of this method to be used in the integer linear programming problems. In actual use, the parameter can be set as the value of the most negative b_i or c_j and temporarily subtracted from all negative b_i's and c_j's. All these resultant terms will be greater than

or equal to zero and may be used to calculate an optimal pivot row and column by the ratio test of the primal or dual simplex method algorithm.

More specifically, we assume that we have the integer linear programming problem as follows :

$$\text{Maximize} \quad z = -(c_1 x_1 + c_2 x_2 + \cdots + c_n x_n),$$

$$\text{subject to} \quad \begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2, \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m, \end{aligned} \tag{3.1}$$

where $x_i \geq 0$ and integer for $i=1,2,\cdots,n$, with all integer coefficients and non-empty subset $\{ i \mid b_i < 0 \}$ and $\{ j \mid c_j < 0 \}$.

Let

$$\begin{aligned} T_b &= \min_{i=1,2,\cdots,m} b_i < 0, \\ T_c &= \min_{j=1,2,\cdots,n} c_j < 0, \end{aligned} \tag{3.2}$$

and

$$T = \min. \{ T_b, T_c \} < 0.$$

Assume that $T = b_r$, thus we have primal infeasibility. Then we subtract T from all the negative c_j such that all $\tilde{c}_j \geq 0$,

$$\tilde{c}_j = \begin{cases} c_j & \text{for } c_j \geq 0, \\ c_j - T & \text{for } c_j < 0. \end{cases} \tag{3.3}$$

Following the dual simplex selection rule we perform the ratio test :

$$\frac{\tilde{c}_s}{-a_{rs}} = \min_{a_{rj} < 0} \left\{ \frac{\tilde{c}_j}{-a_{rj}} \right\}. \tag{3.4}$$

Thus the pivot column is A_s and the source row is b_r row, then we perform a dual pivot.

An analogous procedure holds, if the minimum term T is some c_s. Then the b_i's take the place of the c_j's and

$$\tilde{b}_i = \begin{cases} b_i & \text{for } b_i \geq 0, \\ b_i - T & \text{for } b_i < 0. \end{cases} \tag{3.5}$$

The ratio test is then :

$$\frac{\tilde{b}_r}{a_{rs}} = \min_{a_{is} > 0} \left\{ \frac{\tilde{b}_i}{a_{is}} \right\}, \tag{3.6}$$

and we perform a primal pivot by using A_s as the pivot column and b_r row as the source row. Whether primal or dual pivots are made, the algorithm switches between them as needed, and T as computed is increasing to 0.

In pivoting primally or dually, we must of course make integer pivots. Primally, we use the normal primal integer programming pivoting method described by Young. More specifically, for the purpose of exposition, we shall refer only to the Rudimentary Primal Integer Algorithm.

If the integer linear programming problem is given as follows :

$$\begin{aligned} &\text{Maximize} && z = -c\,x, \\ &\text{subject to} && A\,x + I\,t = b, \end{aligned} \tag{3.7}$$

where c is an n x 1 and b is an n x 1 vectors and I is an identity matrix of dimension m x m, $x^T = (x_1, x_2, \cdots, x_n)$, $t^T = (t_1, t_2, \cdots, t_m)$.

The tableau form of the problem is as follows :

		$-x_1$	$-x_2$	$\cdots$	$-x_n$
z	0	c_1	c_2	$\cdots$	c_n
x_1	0	-1	0	$\cdots$	0
x_2	0	0	-1	$\cdots$	0
.	.			.	
.	.			.	
.	.			.	
x_n	0	0	0	$\cdots$	-1
t_1	b_1	a_{11}	a_{12}	$\cdots$	a_{1n}
t_2	b_2	a_{21}	a_{22}	$\cdots$	a_{2n}
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
t_m	b_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}

(3.8)

The Primal - Dual Integer Linear Programming Algorithm

1. Let $T_b = \min_{i=1,\cdots m} b_i$, $T_c = \min_{j=1,\cdots n} c_j$. If $T_b \geqslant 0$ and $T_c \geqslant 0$, then the current solution is feasible and optimal, otherwise, go to step 2.

2. Let $T = \min. \{T_b, T_c\}$,

$$\tilde{c}_j = \begin{cases} c_j & \text{for } c_j \geqslant 0, \\ c_j - T & \text{for } c_j < 0, \end{cases} \quad \text{and} \quad \tilde{b}_i = \begin{cases} b_i & \text{for } b_i \geqslant 0, \\ b_i - T & \text{for } b_i < 0. \end{cases}$$

If $T = T_c$, then we go to step 3. If $T = T_b$ then we go to step 4. If $T = T_b = T_c$ then we can arbitrarily go to step 3 or 4.

3. Primal pivot : We select the lexicographically smallest column A_s, let $T = c_s$ and let b_r row be the source row resulting from $\tilde{b}_r/a_{rs} = \min_{a_{is} > 0} (\tilde{b}_i/a_{is})$, then the cut adjoined is $S_I = [b_r/a_{rs}] + [a_{r1}/a_{rs}]\cdot(-x_1) + \cdots + 1 \cdot (-x_s) + \cdots + [a_{rn}/a_{rs}]\cdot(-x_n)$. Then a complete elimination procedure using 1 as the pivot element is performed and the cut is then discarded from the tableau. Go to step 1.

4. Dual pivot : We select the smallest b_i's, say b_r, specify b_r row as the source row, breaking ties by arbitrarily choosing one of them, and specify column A_s as the pivot column resulting from $\tilde{c}_s/(-a_{rs}) = \min_{a_{rj} \, 0} (\tilde{c}_j/(-a_{rj}))$, then the cut adjoined is

$S_I = [b_r/a_{rs}] + [a_{r1}/a_{rs}]\cdot(-x_1) + \cdots + 1 \cdot (-x_s) + \cdots + [a_{rn}/a_{rs}]\cdot(-x_n)$. Then a complete elimination procedure using 1 as the pivot element is performed and the cut is then discarded from the tableau. Go to step 1.

EXAMPLE 3.1:

$$\text{Maximize} \quad z = 2x_1 - 3x_2,$$

$$\text{subject to} \quad -x_1 + 2x_2 \leq 4,$$

$$x_1 - x_2 \leq -2,$$

$$\text{and} \quad x_1 - 3x_2 \leq -5,$$

where $x_1, x_2 \geq 0$ and are integers.

SOLUTION:

After adding slack variables (t_1, t_2, t_3), the problem has

the following tableau form :

		$-x_1$	$-x_2$
z	0	-2	3
x_1	0	-1	0
x_2	0	0	-1
t_1	4	-1	2
t_2	-2	1	-1
t_3	-5	1	-3

(3.9)

From tableau(3.9), we obtain :

$T_b = -5,$

$T_c = -2,$

and $T = T_b = -5.$

		$-x_1$	$-x_2$ ↓	
z	0	3^{-5}	3	
x_1	0	-1	0	
x_2	0	0	-1	
t_1	4	-1	2	
t_2	3^{-5}	1	-1	
t_3	0^{-5}	1	(-3)	←
s_1	1	-1	1	

(3.10)

Dual pivot : $\min_{\text{all } i} b_i = -5,$

$\min_{a_{3j} < 0} (\tilde{c}_j/(-a_{3j})) = \min.(3/-3) = -1.$

In tableau(3.10) we circled -3 to denote $-x_2$ column as the pivot column and t_3 row as the source row. So the cut is :

$s_1 = 1 + [\frac{1}{-3}](-x_1) + (1)(-x_2)$

$= 1 + (-1)(-x_1) + 1\cdot(-x_2).$

		$-x_1$	$-s_1$
z	-3	1	-3
x_1	0	-1	0
x_2	1	-1	1
t_1	2	1	-2
t_2	-1	0	1
t_3	-2	-2	3

(3.11)

After complete elimination procedure, we get tableau(3.11), from which we obtain :

$T_b = -2,$

$T_c = -3,$

and $T = T_c = -3.$

	$-x_1$	$-s_1$ ↓	
z	-3	1	0^{-3}
x_1	0	-1	0
x_2	1	-1	1
t_1	-2	1	-2
t_2	2^{-3}	0	1
t_3	1^{-3}	-2	(3) ←
s_2	-1	-1	1

(3.12)

Primal pivot : $\min_{\text{all } j} c_j = -3$,

$\min_{a_{i2}>0}(\tilde{b}_i/a_{i2}) = \min.(\frac{1}{1}, \frac{2}{1}, \frac{1}{3}) = \frac{1}{3}$.

In tableau(3.12)we circled 3 to denote $-s_1$ column as the pivot column and t_3 row as the source row. So the cut is :

$s_2 = [-2/3] + [-2/3](-x_1) + 1(-s_1)$

$= -1 + (-1)(-x_1) + 1\cdot(-s_1)$.

	$-x_1$	$-s_2$
z	-6	-2	3
x_1	0	-1	0
x_2	2	0	-1
t_1	0	-1	2
t_2	0	1	-1
t_3	1	1	-3

(3.13)

After complete eliminatinn procedure we get tableau(3.13), from which we obtain :

$T_b = 0$,

$T_c = -2$,

and $T = T_c = -2$.

	$-x_1$ ↓	$-s_2$	
z	-6	-2	3
x_1	0	-1	0
x_2	2	0	-1
t_1	0	-1	2
t_2	0	(1)	-1 ←
t_3	1	1	-3
s_3	0	1	-1

(3.14)

Primal pivot : $\min_{\text{all } j} c_j = -2$,

$\min_{a_{i1}>0}(\tilde{b}_i/a_{i1}) = \min.(\frac{0}{1}, \frac{1}{1}) = 0$.

In tableau(3.14), we circled 1 to denote $-x_1$ column as the pivot column and t_3 row as the source row. So the cut is :

$s_3 = [\frac{0}{1}] + [\frac{1}{1}](-x_1) + [\frac{-1}{1}](-s_2)$.

		$-s_3$	$-s_2$
z	-6	2	1
x_1	0	1	-1
x_2	2	0	-1
t_1	0	1	1
t_2	0	-1	0
t_3	1	-1	-2

(3.15)

From tableau (3.15), we obtain:

$$T_b = 0 \geq 0,$$

$$T_c = 1 \geq 0.$$

Therefore, $x_1 = 0$, $x_2 = 2$ and $z = -6$ is the optimal solution.

CHAPTER IV

THE ZERO - ONE LINEAR IMPLICIT ENUMERATION

The linear programming problem with zero-one variables may be stated as follows :

$$\text{Minimize} \quad z = c_1x_1 + \ldots + c_nx_n,$$

$$\text{subject to} \quad a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1 , \qquad (4.1)$$

$$\vdots$$

$$a_{m1}x_1 + \ldots + a_{mn}x_n \leq b_m,$$

where $x_i \geq 0$ and $x_i = 0$ or 1, for $i = 1,2,..,n$,

$c_i \geq 0$ for $i = 1,2,..,n$.

Any problem can be brought to this form by necessary operations . Any negative coefficients c_j in the minimization objective function can be made positive by sustituting $y_j = 1 - x_j$ in the entire problem. Suppose the variable x_k is an integer and then assume that $x_k \leq u_k$, where u_k is a known upper bound. Then $x_k = \sum_{P=0}^{T} 2^P y_{kP}$, $y_{kp} = 0$ or 1 , where T is determined such that $2^{T+1} \geq u_k + 1$. By substituting for x_k in terms of y_{kp} , then the problem with variables y_{kp} is a zero-one linear programming problem.

I. THE CONCEPT OF IMPLICIT ENUMERATION

In general, the solution space of an integer linear programming problem can be assumed to possess a finite number of feasible points. Implicit enumeration does not consider complete binary combinations. Rather, it starts with one or more

variable being fixed at a binary value and then gradually builds the solution by augmenting new variables at fixed values. The idea of implicit enumeration calls for considering a portion of all possible solution points while automatically discarding the remaining ones as nonpromising. In order to illustrate the impact of using implicit enumeration, Figure 4.1 gives a graphical representation of the solution to $3x_1 - 8x_2 + 5x_3 \leq -6$, where $x_j = 0$ or 1 for $j = 1,2,3$.

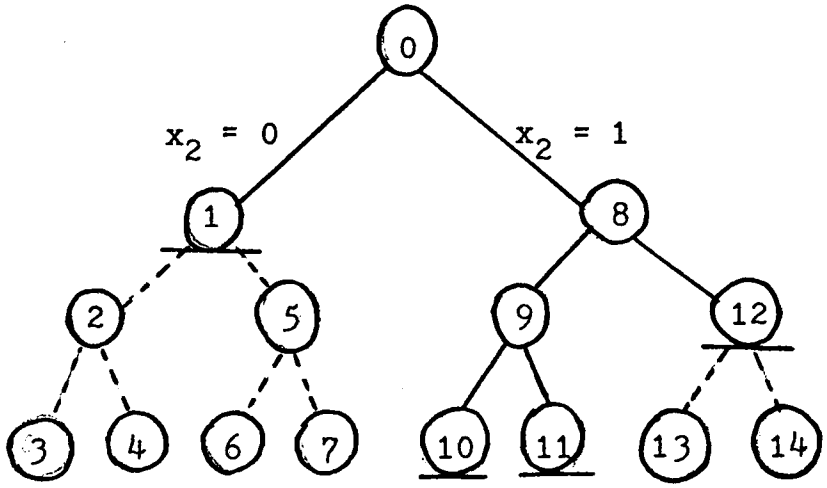


Figure 4.1 Graphical interpretation revealing the implicit enumeration

Implicit enumeration is a tree search procedure. The vertices 0, 1, ···, 14 (sometimes referred to as v_0, v_1, ···, v_{14}) are numbered by the order in which they were considered. Each edge represents a constraint. A line underneath a vertex indicates that no further exploration from that vertex can be profitable, such a vertex is said to be fathomed. As in Figure 4.1 all the solution points that emanate from the branch $x_2=0$ (shown by dotted lines) are discarded. Also $x_2=1$,

$x_1 = 1$ cannot lead to a feasible solution and hence v_{12} is fathomed. The feasible solution point is v_{10} (i.e., $x_2 = 1$, $x_1 = 0$, $x_3 = 0$).

For convenience, the notation +j (-j) is used to indicate that $x_j = 1$ ($x_j = 0$). At the vertex v_k, the path p_k (each path represents the union of constraints) corresponds to an assignment of binary values to a subset of variables such an assignment is called a partial solution. A <u>partial solution</u> is an ordered set, where the order of the components indicate the level in the tree.Let the partial solution at v_k be denoted by J_k.As in figure 4.1,p_{10}corresponding a partial solution (2, -1, -3).

Let the index set of the partial solution J_k be W_k, and then define :

$$S_k^+ = \{ j \mid j \in W_k \text{ and } x_j = 1 \},$$
$$S_k^- = \{ j \mid j \in W_k \text{ and } x_j = 0 \}, \qquad (4.2)$$
$$F_k = \{ j \mid j \notin W_k \}.$$

If $j \in F_k$, then j is called a free variable; a variable that is not assigned a binary value by the partial solution J_k and is free to assume value of zero or one.

A <u>completion</u> of J_k is a complete assignment to all the variables in N and is determined by the assignments in J_k together with an assignment of binary values to the free variables specified by the index set F_k.

II. THE ENUMERATION SCHEME

Two important points for the successful implicit enumeration can be stated as follows :

(1) There is an enumeration scheme which ensures that all the solution points are enumerated (implicitly or explicitly).

(2) A number of fathoming tests must be designed to exclude as many nonpromising solutions as possible.

Glover [5] and Balas [1] give us the enumeration scheme and algorithm to solve zero - one linear programming problem efficiently and nonredundantly.

The flowchart for Glover's enumeration scheme is given in Figure 4.2. Glover's enumeration scheme employs a "backtracking" procedure to generate further solution. Once a feasible partial solution has been found, then backtracking procedure is excuted; we simply replace one of the variables in the current partial solution with its complement. Since objective function is a minimization function with all positive coefficients, if we add more unit value to the current feasible partial solution, then we get worse objective value. Hence no further completion of feasible partial solution is necessary. There are two fathomed cases : (1) J_k is feasible and (2) no completion of J_k is better than J_k. Again, if no completion of J_k is better than J_k, then the backtracking procedure is excuted to avoid redundancy.

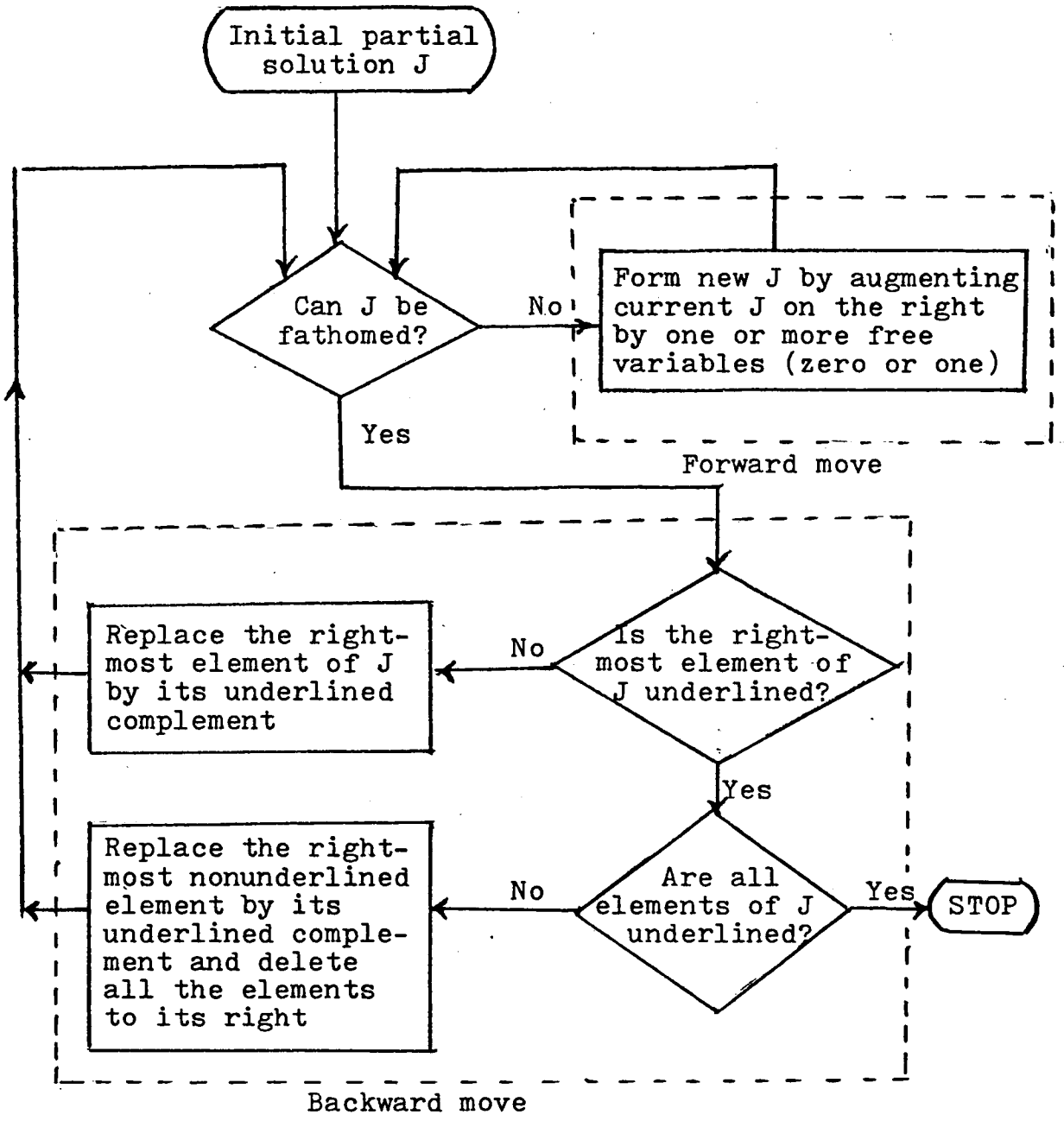


Figure 4.2 Flowchart for Glover's enumeration scheme

To simplify the notation ; for $j \in W_k$, let it appear in J_k as :

$$\begin{cases} j & \text{if } j \in S_k^+ \text{ and } x_j = 0 \text{ has not been considered,} \\ \underline{j} & \text{if } j \in S_k^+ \text{ and } x_j = 0 \text{ has been considered,} \\ -j & \text{if } j \in S_k^- \text{ and } x_j = 1 \text{ has not been considered, and} \\ \underline{-j} & \text{if } j \in S_k^- \text{ and } x_j = 1 \text{ has been considered.} \end{cases}$$

There is another efficient way to scan the list of partial solutions; it is unnecessary to strictly follow the rule of always complementing (and underlining) the right nonunderlined element of the partial solution. To illustrate this idea, consider the partial solution $J = (+2, \underline{-5}, +3, +6, +1, \underline{+4})$. Let $j_1 = 2$ be the element of J that should not be underlined. Next, define J" as the string of elements in J that is not interrupted by an underlined element. Then, any element in J" may be complemented and underlined. This means that the partial solutions generated from J may be one of the following : (+2, $\underline{-5}$, +3, 6, $\underline{-1}$), (+2, $\underline{-5}$, +3, +1, $\underline{-6}$), or (+2, $\underline{-5}$, +6, +1, $\underline{-3}$).

III. BALAS' ADDITIVE ALGORITHM FOR SOLVING 0 - 1 LINEAR PROGRAMMING PROBLEM

In this section we shall introduce some fathoming tests, which are designed to exclude as many nonpromising completions as possible. These tests could be as simple as to allow the explicit enumeration of all 2^n solutions or strong enough to exclude automatically the majority of the solution points.

At the k-th iteration, let J_k be the current partial solution and define :

$$S_i^k = b_i - \sum_{j \in S_k^+} a_{ij}, \qquad i \in M = \{1,2,\cdots,m\}$$

and

$$z_k = \sum_{j \in S_k^+} c_j. \tag{4.3}$$

The objective of tests is to check that either (i) J_k has no feasible completion or (ii) even though J_k may have a feasible completion, it cannot yield a value of z_k that is superior to $\bar{z}$ (the current best objective value). In both cases J_k is fathomed and the backward move must be executed. If on the other hand the tests failed to fathom J_k, then partitioning and branching must be executed in search for a better feasible completion.

Test 1: Is $Q_k = \emptyset$?

Let $Q_k = \{ i \mid S_i^k < 0 \}$. If $Q_k = \emptyset$, then v_k is fathomed. Since J_k is feasible, then the backward move is executed.

Test 2: Is $R_k = \emptyset$?

Let $R_k = \{ j \mid j \in F_k \text{ and } a_{ij} < 0 \text{ for some } i \in Q_k \}$, then $A_k = F_k - R_k = \{ j \mid j \in F_k \text{ and } a_{ij} > 0 \text{ for all } i \text{ such that } S_i^k < 0 \}$. The set A_k represents the free variables, which when elevated to level one cannot possibly improve on the infeasibility of the current solutions. If $R_k = \emptyset$, this means that none of the free variables can be elevated to level one to improve on the infeasibility, which means that J_k is fathomed and the backtracking step is executed.

Test 3: Is $T_k = \emptyset$?

Let $B_k = \{ j \in R_k \mid z_k + c_j \not< \bar{z} \}$. The set B_k identifies those free variables that, although may improve the infeasibility of the problem, cannot possibly lead to an improved value of the objective function. Let $T_k = R_k - B_k$. If $T_k = \emptyset$, then augmentation of J_k, say (J_K, j) for some $j \in B_k$, leads to a nonpromising solution, hence the backtrack move is executed.

Test 4: Is $C_k \neq \emptyset$?

Let $C_k = \left\{ i \in M \mid S_i^k < 0 \text{ and } \sum_{j \in T_k} \min.\{0, a_{ij}\} > S_i^k \right\}$.

If $C_k \neq \emptyset$, then no completion of J_k can satisfy constraint i. So the backward move must be executed.

If J_k pass the tests, then the forward move is executed. to determine the variable to be augmented to J_k. Select the free variable x_p, $p \in T_k$ to be augmented to J_k at level one such that :

$$I_k(p) = \max_{j \in T_k} \{ I_k(j) \}, \qquad (4.4)$$

where $$I_k(j) = \sum_{i \in M} \min. \{ 0, S_i^k - a_{ij} \} \qquad j \in T_k.$$

In the case of a tie, p is selected so that c_p is the smallest c_j. The results of augmentation is $J_{k+1} = (J_k, P)$. There are two cases for $I_k(p)$:

(1) If $I_k(p) = 0$, this means J_{k+1} is feasible. Since the previous tests do not allow an inferior solution, the current solution must yield a better value of $\bar{z}$. Thus $\bar{z} = z_k + c_p$. Since J_{k+1} is feasible, v_{k+1} is fathomed, then we proceed the backward move.

(2) If $I_k(p) < 0$, J_{k+1} must be attempted for fathoming by starting with Test 1 again.

A Rudimentary Linear Enumeration Algorithm

Step 0 : (Initialization) At v_0, $F_0 = 1,2,\cdots,n$. $\bar{z}$ (best known upper bound) = ∞ . $J_0 = \emptyset$.

Step 1 : (Test 1) At v_k, let $z_k = \sum_{j \in S_k^+} c_j$ for $k \neq 0$, $z_0 = 0$.

If $Q_k = \emptyset$, then J_k is feasible. Let $\bar{z} = \min.(\bar{z}, z_k)$ and go to step 5, otherwise go to step 2.

Step 2 : (Test 2) If $R_k = \emptyset$, go to step 5, otherwise go to step 3.

Step 3 : (Test 3) If $T_k = \emptyset$, go to step 5, otherwise go to step 4.

Step 4 : (Test 4) If $C_k \neq \emptyset$, go to step 5, otherwise go to step 7.

Step 5 : (Backtracking) If all elements of J_k are underlined, then go to step 8, otherwise go to step 6.

Step 6 : (Backtracking) Replace the rightmost non-underlined element of J_k by its underlined complement and drop all the elements to the right, then go to step 1.

Step 7 : (Forward move) Let $J_{k+1} = (J_K, P)$. If $I_k(p) \neq 0$, then let $k \leftarrow k + 1$ and go to step 1. If $I_k(p)=0$, then J_{k+1} is a feasible solution, let $k = k + 1$ and go to step 1.

Step 8 : (Termination) If $\bar{z} = \infty$, there is no feasible solution. If $\bar{z} < \infty$, then the feasible solution which yields $\bar{z}$ is optimal.

EXAMPLE 4.1:

$$\text{Minimize} \quad z = 10x_1 + 7x_2 + x_3 + 12x_4 + 2x_5 + 8x_6 + 3x_7 + x_8 + 5x_9 + 3x_{10},$$

subject to

$$3x_1 - 12x_2 - 8x_3 + x_4 - 7x_9 + 2x_{10} \leq -2,$$
$$x_2 - 10x_3 - 5x_5 + x_6 + 7x_7 + x_8 \leq -1,$$
$$5x_1 - 3x_2 - x_3 - 2x_8 + x_{10} \leq -1,$$
$$-5x_1 + 3x_2 + x_3 + 2x_8 - x_{10} \leq 1,$$
$$-4x_3 - 2x_4 - 5x_6 + x_7 - 9x_8 - 2x_9 \leq -3,$$
$$9x_2 - 12x_4 - 7x_5 + 6x_6 + 2x_8 - 15x_9 + 3x_{10} \leq -7,$$
$$-8x_1 + 5x_2 + 2x_3 - 7x_4 - x_5 - 5x_7 - 10x_9 \leq -1,$$

where $x_j = 0$, or 1, $j = 1,2,\cdots,10$.

SOLUTION:

Initially, $F_0 = \{1,2,3,\cdots,10\}$, $J_0 = \emptyset$, $S^0 = (-2, -1, -1, 1, -3, -7, -1)$.

At v_0 : $\bar{z} = \infty$, $z_0 = 0$, $Q_0 = \{1,2,3,5,6,7\}$, $R_0 = T_0 = \{1,2,3,4,5,6,7,8,9\}$, $C_0 = \emptyset$.

$I_0(1) = (-2-3) + (-1-5) = -11$, $I_0(2)=(-2)+(-16)+(-6) =-24$, $I_0(3)=-3-7=-10$, $I_0(4)=-3-1-1-1=-6$,

$I_0(5)=-2-1-3=-6$, $I_0(6)=-2-2-1-13-1=-19$,

$I_0(7)=-2-8-1-4-7=-22$, $I_0(8)=-2-3-1-9-1=-16$,

$I_0(9)=-1-1-1=-3$.

So, $p = 9$ and $J_1 = (9)$.

At v_1 : $z_1 = 5$, $Q_1 = \{2,3,5\}$, $S^1 = (5,-1,-1, 1,-1,8,9)$,

$R_1 = T_1 = \{2,3,4,5,6,8\}$, $C_1 = \emptyset$.

$I_1(2)=-1-1-3+1-1-9+8=-6$, $I_1(3)=0$.

So, $p = 3$ and $J_2 = (9,3)$. J is feasible.

At v_2 : $z_2 = 6$, $S^2 = (13,9,0,0,3,8,7)$, $Q_2 = \emptyset$.

$\bar{z} = 6$. Go to step 5.

At v_3 : $J_3 = (9, \underline{-3})$, $z_3 = 5$, $S^3 = (5,-1,-1,1,-1,8,9)$.

$Q_3 = \{2,3,5\}$, $R_3 = \{2,4,5,6,8\}$, $T_3 = \emptyset$.

Go to step 5 and execute the backward move.

At v_4 : $J_4 = (\underline{-9})$, $z_4 = 0$, $S^4=(-2,-1,-1,1,-3,-7,-1)$.

$Q_4 = \{1,2,3,5,6,7\}$, $R_4 = \{1,2,3,4,5,6,7,8\}$.

Since $c_1 + z_4 = 10 + 0 \geq \bar{z} = 6$, $1 \notin B_4$,

$c_2 + z_4 = 7 + 0 \geq \bar{z}$, $2 \notin B_4$,

$c_4 + z_4 = 12 + 0 \geq \bar{z}$, $4 \notin B_4$,

$c_6 + z_4 = 8 + 0 \geq \bar{z}$, $6 \notin B_4$.

So, $T_4 = R_4 - B_4 = \{3,5,7,8\}$, $C_4 = \emptyset$,

$I_4(3)=-7-2-1=-10$, $I_4(5)=-2-1-3=-6$,

$I_4(7)=-2-1-7-1-3-1-7=-22$, $I_4(8)=-2-1-1+1-2-7-2-1=-15$.

So, $p = 5$.

At v_5 : $J_5 = (\underline{-9}, 5)$, $z_5 = 2$, $S^5=(-2,4,-1,1,-3,0,0)$,

$Q_5 = \{1,3,5\}$, $R_5 = \{2,3,4,6,8\}$, $T_5 = \{3,8\}$, $C_5=\emptyset$.

$I_5(3)=-2$, $I_5(8)=-2-2+1-2=-5$.

So, $p = 3$.

At v_6 : $J_6 = (\underline{-9},5,3)$, $z_6 = 3$, $S^6=(6,14,0,0,1,0,-2)$.

$Q_6 = \{7\}$, $R_6 =\{1,4,7\}$, $T_6 = \emptyset$, then execute the backward move.

At v_7 : $J_7 = (\underline{-9},5,\underline{-3})$, $z_7 = 2$, $S^7=(-2,4,-1,1,-3,0,0)$,

$Q_7 = \{1,3,5\}$, $R_7 =\{2,4,6,8\}$, $T_7 = \{8\}$, $C_7 = \{1\}$,

then go to step 5 and execute the backward move.

At v_8 : $J_8 = (\underline{-9},\underline{-5})$, $z_8 = 0$, $S^8=(-2,-1,-1,1,-3,-7,-1)$,

$Q_8 = \{1,2,3,5,6,7\}$, $R_8 = \{1,2,3,4,6,7,8\}$,

$T_8 = \{3,7,8\}$, $C_8 = \{6\}$,

$i = 6$, $\sum_{j\in T_8} \min.\{0,a_{6j}\} = 0 + 0 + 0 > S_6^8 = -7$.

Go to step 5 and execute the backward move.

At v_9 : $x_3 = x_9 = 1$, all other $x_i = 0$, is feasible and an optimal solution. The optimal value is $z = 6$.

A flowchart summarizing the algorithm is given in Figure 4.3. The algorithm is additive basically because it requires only addition and subtraction operations for its execution.

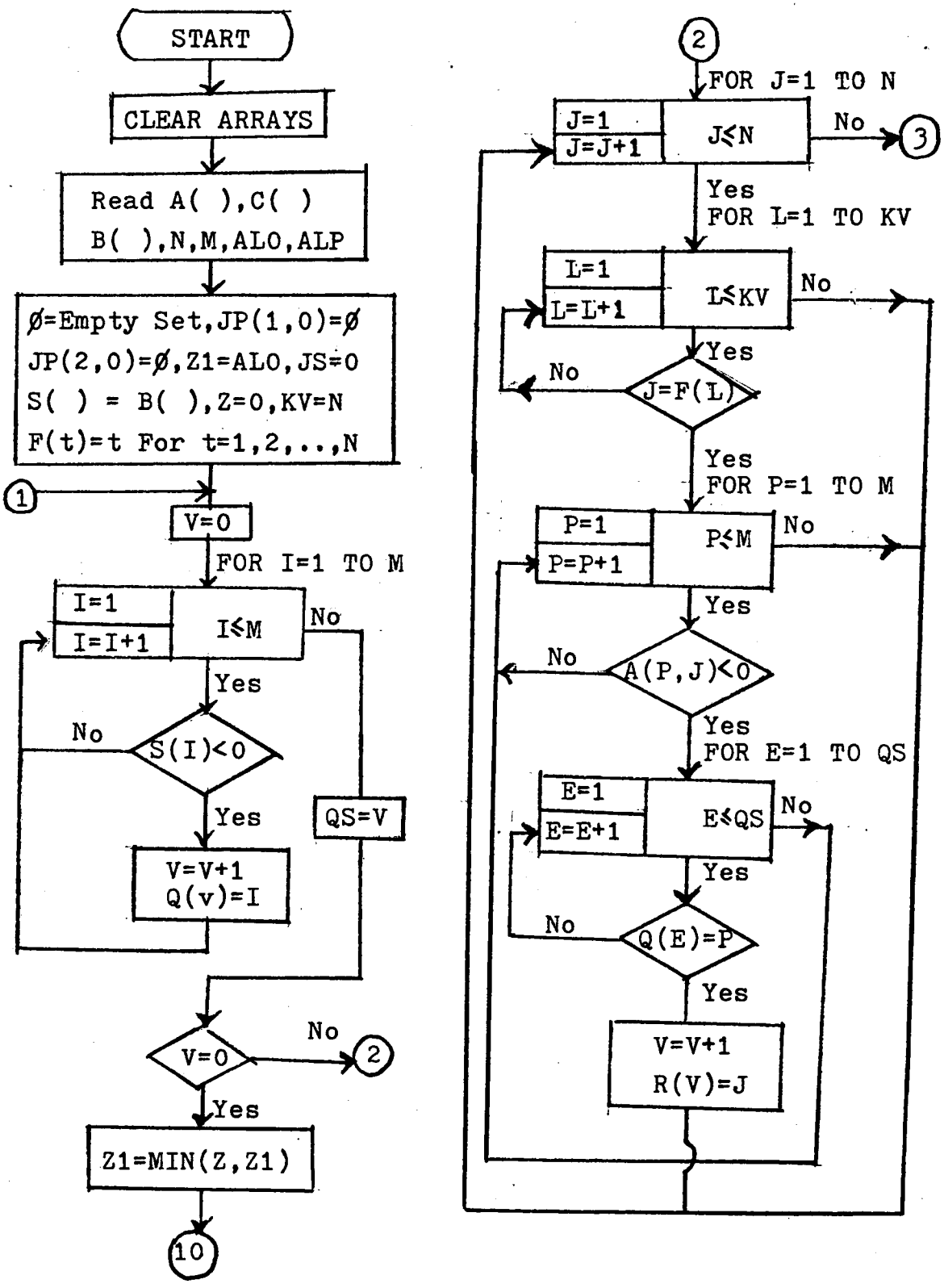


Figure 4.3 Flowchart for Balas' additive algorithm

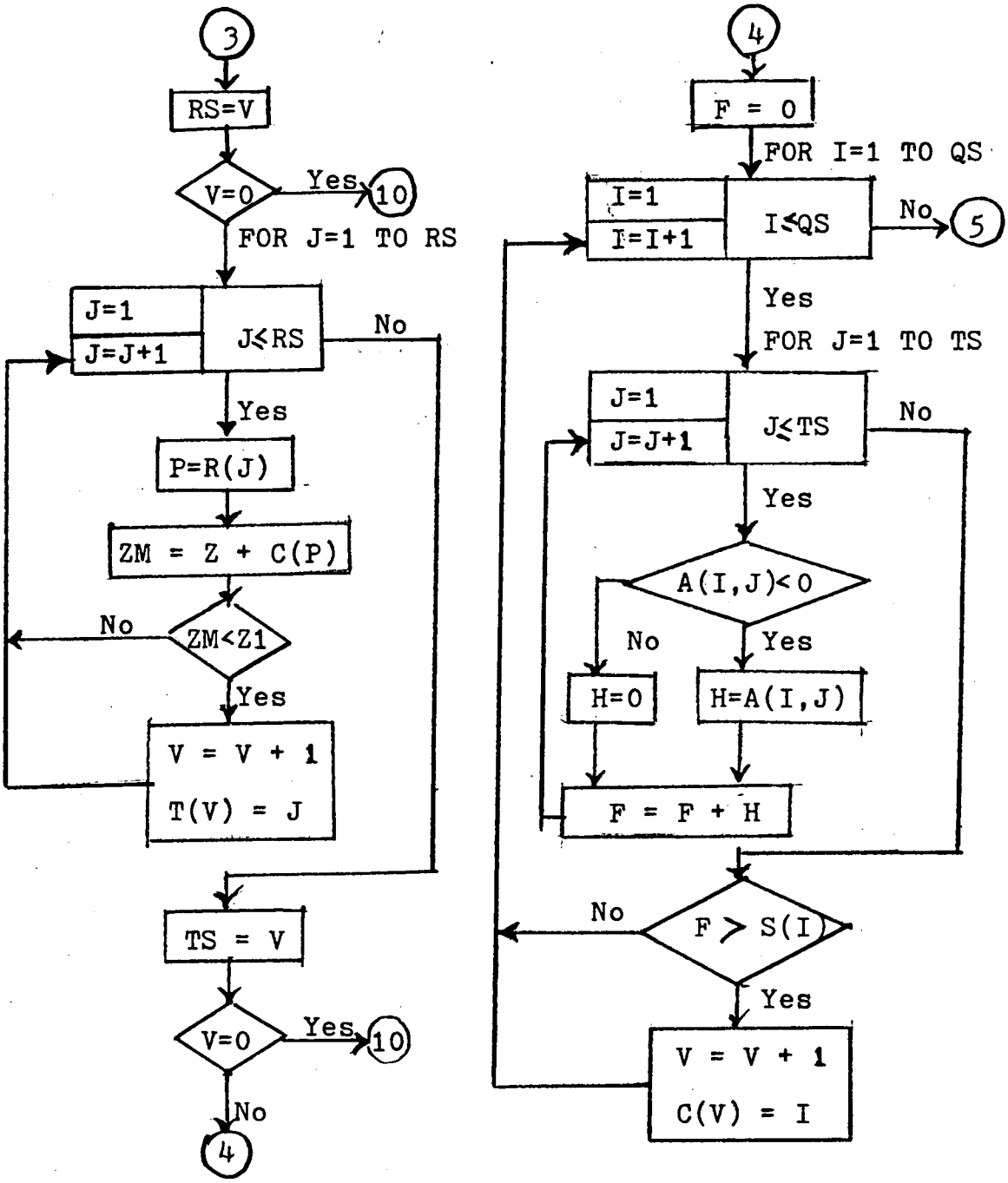


Figure 4.3 (continued)

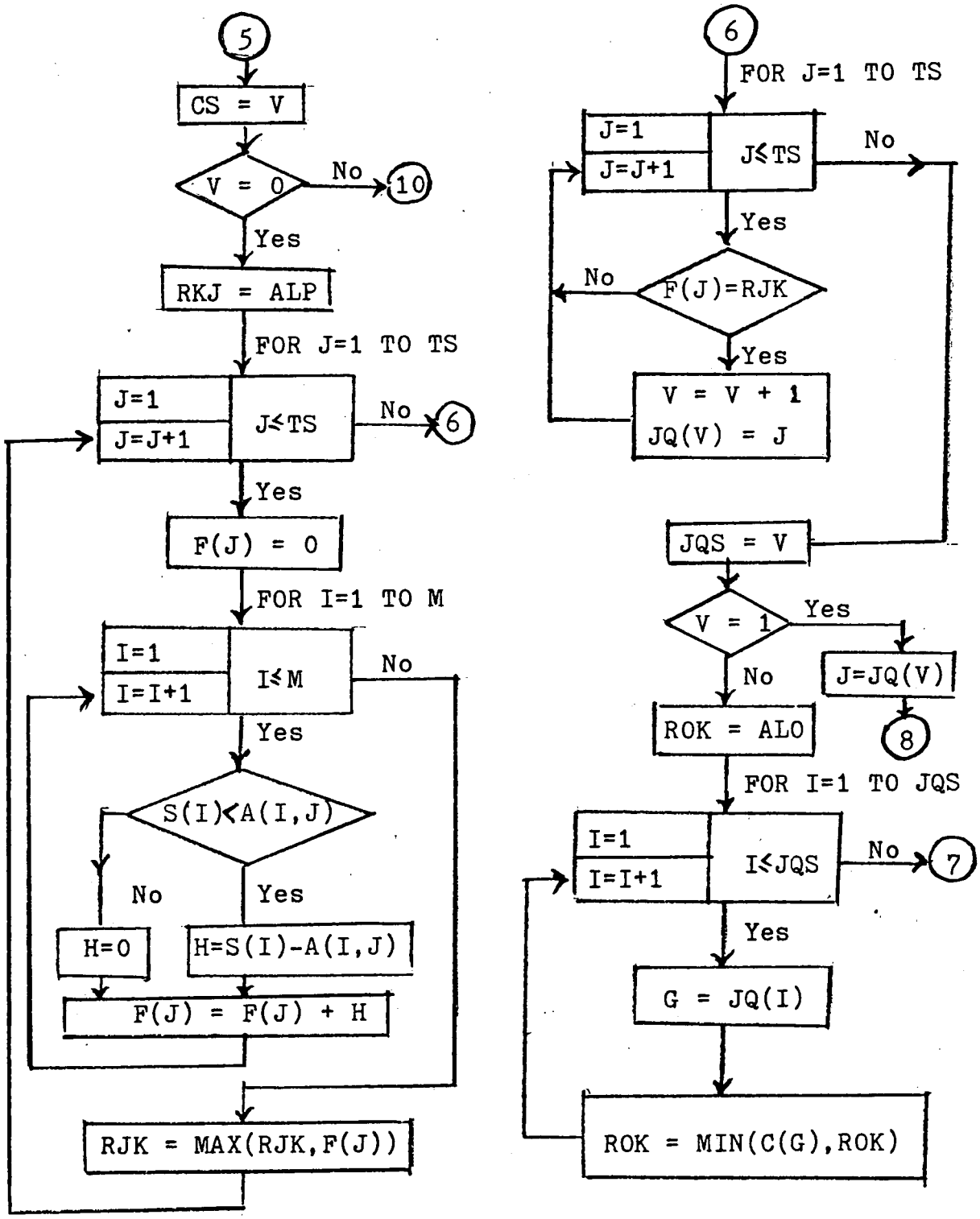


Figure 4.3 (continued)

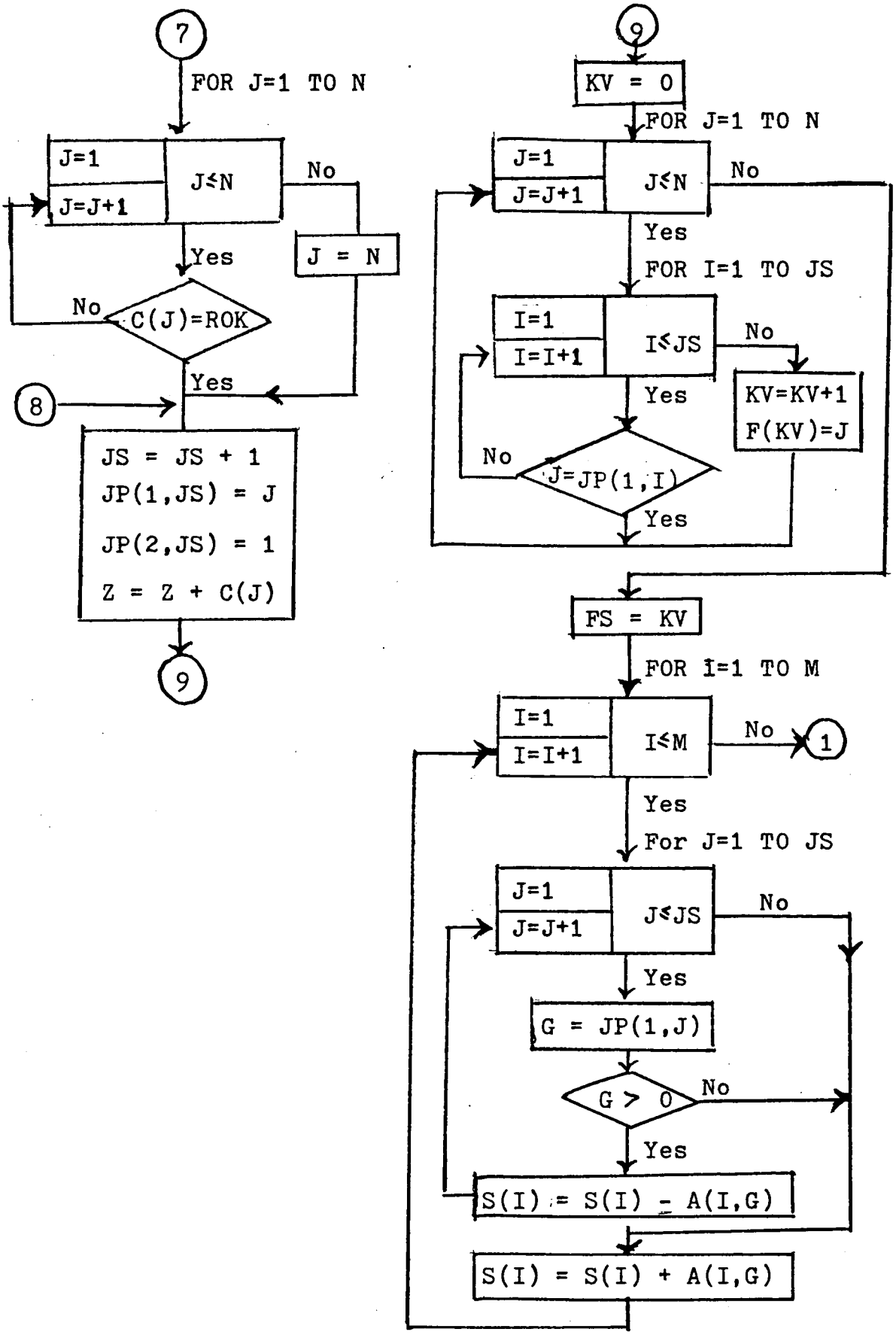


Figure 4.3 (continued)

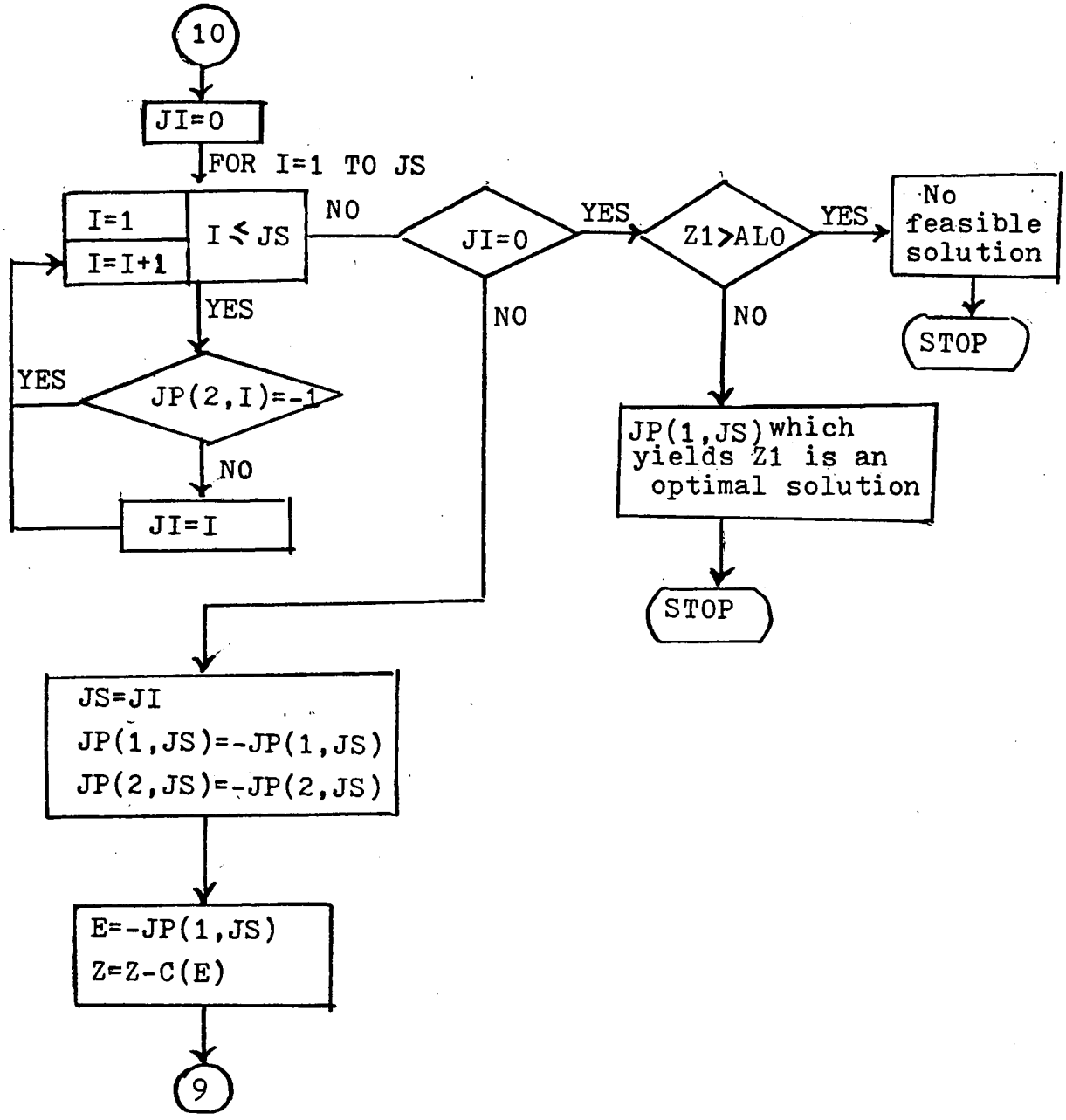


Figure 4.3 (continued)

Now, when we carry out the backward move, we can use another rule. As mentioned above, it is unnecessary to follow the rule strictly by always complementing and underlining the rightmost nonunderlined elements of the partial solutions. For example, if $J_k = (-2,+\underline{3},+5,+4,+7)$ is fathomed, -2 is the element that should not be underlined, then $J_K = \{5,4,7\}$. The basis for selecting a specific element in J_k" is to maintain the condition that are favorable to produce feasible completions quickly. $j \in J_k$" is complemented (ie. set equal to 0) and underlined if the resulting partial solution produces the least amount of infeasibility. Thus, the selected element p must satisfy :

$$w_k(p) = \max_{j \in J_k"} \{w_k(j)\}, \qquad (4.5)$$

where $w_k(j) = \sum_{i \in M} \min.\{0, S_i^k + a_{ij}\}$, $j \in J_k"$,

with $w_k(j)$ the measure of infeasibility when x_j is set back to 0. Again, in case of a tie, p is selected so that c_p is the largest c_j.

EXAMPLE 4.2:

Minimize $z = 2x_1 + 4x_2 + 6x_3 + 8x_4 + 12x_5$,

subject to
$$-8x_1 + 4x_2 + x_3 - x_4 - 5x_5 \leq -5,$$
$$-6x_1 + 3x_2 + 2x_3 - x_5 \leq -2,$$
$$2x_1 - 9x_2 - 3x_3 + 2x_4 - 3x_5 \leq -4,$$

where $x_j = 0$, or 1, $j = 1, 2, 3, 4, 5$.

SOLUTION:

Initially, $F_0 = \{1,2,3,4,5\}$, $J_0 = \emptyset$, $S^0 = (-5,-2,-4)$, $\bar{z} = \infty$.

At v_0 : $Q_0 = \{1,2,3\}$, $R_0 = T_0 = \{1,2,3,4,5\}$, $C_0 = \emptyset$, $z_0=0$,
$I_0(1)=-6$, $I_0(2)=-14$, $I_0(3)=-11$, $I_0(4)=-12$,
$I_0(5)=-2$, so, $p = 5$ and $J_1 = (5)$.

At v_1 : $J_1 = (5)$, $S^1 = (0,-1,-1)$, $Q_1 = \{2,3\}$, $z_1 = 12$,
$R_1 = \{1,2,3\}$, $T_1 = R_1$, $C_1 = \emptyset$.
$I_1(1)=-3$, $I_1(2)=-8$, $I_1(3)=-4$, so, $p = 1$ and $J_2=(5,1)$.

At v_2 : $J_2 = (5,1)$, $S^2 = (8,5,-3)$, $Q_2 = \{3\}$, $z_2 = 14$,
$R_2 = \{2,3\}$, $T_2 = \{2,3\}$, $C_2 = \emptyset$.
$I_2(2) = 0$, $I_2(3)=0$. Since $c_2 < c_3$, thus we branch to $x_2 = 1$.

At v_3 : $J_3 = (5,1,2)$, $S^3 = (4,2,6)$ is feasible.
$\bar{z} = z_3 = 18$, and J_3 is fathomed.
Backward move : $J_3" = \{5,1,2\}$, $w_3(1) = -8$,
$w_3(2) = -3$, $w_3(5) = -1$.
Hence $x_5 = 1$ is complemented.

At v_4 : $J_4 = (1,2,\underline{-5})$, $S^4 = (-1,1,3)$, $Q_4 = \{1\}$, $z_4 = 6$.
$R_4 = T_4 = \{4\}$, $C_4 = \emptyset$.
Foreward move : We branch to $x_4 = 1$.

At v_5 : $J_5 = (1,2,\underline{-5},4)$, $S^5 = (0,1,1)$, $z_5 = 14$.
Since J_5 is feasible, J_5 is fathomed. $\bar{z} = 14$.
Backward move : $J" = \{1,2,4\}$, $w_5(1) = -13$,
$w_5(2) = -8$, $w_5(4) = -1$.
Hence $x_4 = 1$ is complemented.

At v_6 : $J_6 = (1,2,\underline{-5},\underline{-4})$, $S^6 = (-1,1,3)$, $Q_6 = \{1\}$, $z_6 = 6$.

$R_6 = \emptyset$, the backward move is executed.

Backward move : $J_6'' = \{1,2\}$, $w_6(1) = -14$, $w_6(2) = -6$.

Hence $x_2 = 1$ is complemented.

At v_7 : $J_7 = (1,\underline{-2})$, $S^7 = (3,4,-6)$, $Q_7 = \{3\}$, $z_7 = 2$.

$R_7 = \{3,5\}$, $T_7 = \{3\}$, $C_7 = \{3\}$.

Hence the backward move is executed.

Backward move : $J_7'' = \{1\}$.

Hence $x_1 = 1$ is complemented.

At v_8 : $J_8 = (\underline{-1})$, $S^8 = (-5,-2,-4)$, $Q_8 = \{1,2,3\}$, $z_8 = 0$.

$R_8 = \{2,3,4,5\}$, $T_8 = \{2,3,4,5\}$, $C_8 = \{2\}$.

Hence the backward move is executed.

At v_9 : Since all elements of J_8 are underlined, the enumeration is complete and the optimal solution is given by J_5.

Therefore $x_1 = x_2 = x_4 = 1$, $x_3 = x_5 = 0$ and Min. $z = 14$ is the optimal solution.

The solutions of the same problem by using the original Balas' additive algorithm and by using the revised Balas' additive algorithm have the following difference: It is very easy to understand the original Balas' additive algorithm , since we can draw enumeration tree by using the original algorithm , and every vertex of the enumeration tree is one to one correspondence to the partial solution generated from the original algorithm. But it is much quicker by using the revised Balas' additive algorithm.

The solution to the same problem by using the original Balas' additive algorithm is shown as follows:

EXAMPLE 4.3:

Minimize $z = 2x_1 + 4x_2 + 6x_3 + 8x_4 + 12x_5$

subject to $-8x_1 + 4x_2 + x_3 - x_4 - 5x_5 \leq -5$,

$-6x_1 + 3x_2 + 2x_3 - x_5 \leq -2$,

$2x_1 - 9x_2 - 3x_3 + 2x_4 - 3x_5 \leq -4$,

where $x_j = 0$ or 1 for $j = 1,2,3,4,5$.

SOLUTION:

Initially, $F_0=\{1,2,3,4,5\}$, $J_0=\phi$, $S^0=(-5,-2,-4)$, $\bar{z}=\infty$;

At v_0 : $Q_0=\{1,2,3\}$, $R_0=T_0=\{1,2,3,4,5\}$, $C_0=\phi$, $z_0=0$,
$I_0(1)=-6, I_0(2)=-14, I_0(3)=-11, I_0(4)=-12$,
$I_0(5)=-2$, so $p=5$ and $J_1=(5)$.

At v_1 : $J_1=(5)$, $S^1=(0,-1,-1)$, $Q_1=\{2,3\}$, $z_1=12$, $R_1=T_1=\{1,2,3\}$, $C_1=\phi$, $I_1(1)=-3, I_1(2)=-8, I_1(3)=-4$,
so $p=1$ and $J_2=(5,1)$.

At v_2 : $J_2=(5,1)$, $S^2=(8,5,-3)$, $Q_2=\{3\}$, $z_2=14$, $R_2=T_2=\{2,3\}$, $C_2=\phi$, $I_2(2)=0, I_2(3)=0$.
Since $c_2 < c_3$ thus $J_3=(5,1,2)$.

At v_3 : $J_3=(5,1,2)$, $S^3=(4,2,6)$, $\bar{z}=z_3=18$, J_3 is a feasible solution and backward move is executed.

At v_4 : $J_4=(5,1,\underline{2})$, $S^4=(8,5,-3)$, $Q_4=\{3\}$, $z_4=14$, $R_4=T_4$
$T_4=\{3\}$, $C_4=\phi$, $J_5=(5,1,\underline{2},3)$.

At v_5 : $J_5=(5,1,\underline{2},3)$, $S^5=(7,3,0)$, J_5 is feasible, $\bar{z}=z_5=20$, so we execute the backward move.

At v_6 : $J_6=(5,1,\underline{2},\underline{3})$, $S^6=(8,5,-3)$, $Q_6=\{3\}$, $z_6=14$, $R_6=\phi$, then we execute the backward move.

At v_7 : $J_7=(5,-\underline{1})$, $S^7=(0,-1,-1)$, $Q_7=\{2,3\}$, $z_7=12$, $R_7=T_7=\{2,3\}$, $C_7=\phi$, $I_7(2)=-8$, $I_7(3)=-4$, so $J_8=(5,-\underline{1},3)$.

At v_8 : $J_8=(5,-\underline{1},3)$, $S^8=(-1,-3,2)$, $Q_8=\{2,3\}$. $z_8=18$, $R_8=\{4\}$, $T_8=\phi$, so we execute the backward move.

At v_9 : $J_9=(5,-\underline{1},-\underline{3})$, $S^9=(0,-1,-1)$, $Q_9=\{2,3\}$, $z_9=12$, $R_9=T_9=\{2\}$, $C_9\neq\phi$, we execute the backward move.

At v_{10}: $J_{10}=(-\underline{5})$, $S^{10}=(-5,-2,-4)$, $Q_{10}=\{1,2,3\}$, $z_{10}=0$, $R_{10}=T_{10}=\{1,2,3,4\}$, $I_{10}(1)=-6$, $I_{10}(2)=-14$, $I_{10}(3)=-11$.

At v_{11}: $J_{11}=(-\underline{5},1)$, $S^{11}=(3,4,-6)$, $Q_{11}=\{3\}$, $R_{11}=T_{11}=\{2,3\}$, $z_{11}=2$, $C_{11}=\phi$, $I_{11}(2)=-1$, $I_{11}(3)=-1$, $c_2<c_3$, so $p=2$.

At v_{12}: $J_{12}=(-\underline{5},1,2)$, $S^{12}=(-1,1,3)$, $Q_{12}=\{1\}$, $z_{12}=6$, $R_{12}=T_{12}=\{4\}$, $C_{12}=\phi$, $J_{13}=(-\underline{5},1,2,4)$.

At v_{13}: $J_{13}=(-\underline{5},1,2,4)$, $S^{13}=(0,1,1)$, J_{13} is feasible, $\bar{z}=z_{13}=14$, then we execute the backward move.

At v_{14}: $J_{14}=(-\underline{5},1,2,-\underline{4})$, $S^{14}=(-1,1,3)$, $Q_{14}=\{1\}$, $R_{14}=\phi$, then we execute the backward move.

At v_{15}: $J_{15}=(-\underline{5},1,-\underline{2})$, $S^{15}=(3,4,-6)$, $Q_{15}=\{3\}$, $R_{15}=T_{15}=\{3\}$, $z_{15}=2$, $C_{15}\neq\phi$, we execute the backward move.

At v_{16}: $J_{16}=(-\underline{5},-\underline{1})$, $S^{16}=(-5,-2,-4)$, $Q_{16}=\{1,2,3\}$, $R_{16}=T_{16}=\{2,3,4\}$, $C_{16}\neq\phi$, execute the backward move.

At v_{17}: Since all elements of J_{16} are underlined, the enumeration is complete and the optimal solution J_{13}.

By comparing Examples 4.2 and 4.3 it is clear that the revised Balas' additive algorithm works much quicker than the original Balas' additive algorithm.

IV. ADDITIONAL TESTS FOR THE IMPLICIT ENUMERATION METHOD

Many additional tests have been proposed that can reduce the enumeration at the expense of added calculation. Some of these tests are presented as follows :

At the k-th iteration, let J_k be the current partial solution, and define :

$$S_i^k = b_i - \sum_{j \in S_K^+} a_{ij}, \qquad i \in M = \{1,2,\cdots,m\},$$

and

$$z_k = \sum_{j \in S_K^+} c_j. \tag{4.6}$$

Augment (4.6) by the constraint :

$$\sum_{j \in \bar{F}_k} c_j x_j + z_k \leq \bar{z} \tag{4.7}$$

which must be satisfied by any optimal solution. The integer linear programming problem is then :

$$\text{Minimize} \qquad z = \sum_{j \in F_k} a_{0j} x_j + z_k,$$

$$\text{subject to :} \qquad \sum_{j \in F_k} a_{ij} x_j \leq S_i^k, \quad i = 0,1,\cdots,m \tag{4.8}$$

where $S_0^0 = \bar{z} - z_k$, $a_{0j} = c_j$, $x_j = 0$ or 1 for $j \in F_k$. It is assumed that $Q_k \neq \emptyset$ and $\sum_{j \in F_k} \min.\{0, a_{ij}\} \leq S_i^k$ for $i=0,1,\cdots,m$.

Test 5: Suppose that for some $i \in Q_k$ and $p \in F_k$,

$\sum_{j \in F_k} \min.\{0, a_{ij}\} + |a_{ip}| > S_i^k$, then $a_{ip} > 0$ ($a_{ip} < 0$) implies that $x_p = 0$ ($x_p = 1$) in any feasible completion of W_k.

For example, the constraint :

$$-3x_1 + 2x_2 - 5x_3 + 6x_4 \leq -4$$

implies that $x_3 = 1$ and $x_4 = 0$.

Tests 6 - 8 to be presented later are performed by first computing numbers L and U such that the constraint :

$$L \leq \sum_{j \in F_k} x_j \leq U$$

is valid for any optimal completion of W_k.

For each constraint $i = 0,1,\cdots,m$, let the variables be reindexed so that the coefficients a_{ij} are in a non-decreasing order. The reindexed coefficients for row i are ($a_{i,1(i)}$, $a_{i,2(i)}$, $\cdots$, $a_{i,\ell(i)}$) where $\ell = |F_k|$ and if $t < p$ then $a_{i,t(i)} \leq a_{i,p(i)}$. Define $T_i(0) = 0$ and let

$$T_i(p) = \sum_{j=1}^{p} a_{i,j(i)}, \text{ and } T_i(L_i) \leq S_i^k < T_i(L_i - 1),$$

and we define $L = \max_{i \in Q_k} L_i$.

To determine U, for $i = 0,1,\cdots,m$, calculate U_i given

$$T_i(U_i) \leq S_i^k < T_i(U_i + 1) \quad \text{if} \quad T_i(\ell) > S_i^k,$$

$$U_i = \infty \quad \text{if} \quad T_i(\ell) \leq S_i^k,$$

then $U_i \geq U$ and $U = \min. U_i$.

EXAMPLE 4.4:

Suppose $\bar{z} - z_k = 11$, after rearranging the objective function as a constraint, we get the problem as follows:

$$\begin{aligned} 3x_1 + x_2 + 4x_3 + x_4 &\leq 11, \\ -x_1 - x_2 + 2x_3 - 3x_4 &\leq -4, \\ 3x_1 - 4x_2 - 2x_3 + 6x_4 &\leq -1, \end{aligned}$$

It can be easily determined that $L_1 = 2$, $L_2 = 1$, $L = 2$ and $U_0 = \infty$, $U_1 = 3$, $U_2 = 3$, $U = 3$.

Test 6: If $L > U$, v_k is fathomed, since (4.8) cannot be satisfied. Also if $L = F_k$, then each of the free variables can be set to one.

Test 7: Let $T_i(L,U) = \min_{L \leq p \leq U} T_i(p)$, $i = 0,1,\cdots,m$, be the smallest sum that can be obtained using at least L, but not more than U, free variables. Assume that vertex k is not fathomed by Test 6, and a_{ip} is not one of the numbers which determine $T_i(L,U)$. If $T_i(L-1,U-1) > S_i^k - a_{ip}$ then $x_p = 0$.

In Example 4.4, $T_2(1,2) = -6 > S_2^k - a_{24} = -7$. Therefore, $x_4 = 0$ and the new solution is $(J_k, -\underline{4})$.

Test 8: Suppose that vertex k is not fathomed by Test 6, and a_{ip} is one of the numbers that determine $T_i(L,U)$. If $T_i(L+1,U+1) > S_i^k + a_{ip}$ then $x_p = 1$.

In Example 4.4, $T_1(3,4) = -5 > S_1^k + a_{14} = -7$. Therefore, $x_4 = 1$ and the new solution is $(J_k, 4)$. Test 7 and

Test 8 together showed that there is no feasible solution to Example 4.4.

V. SURROGATE CONSTRAINTS

In the binary integer linear programming, consider replacing the set of constraints :

$$\sum_{j \in F_k} a_{ij}\, x_j \leq S_i^k \qquad i = 1,2,\cdots,m \tag{4.9}$$

with the single constraint :

$$\sum_{i=1}^{m} \sum_{j \in F_k} u_i\, a_{ij}\, x_j \leq \sum_{i=1}^{m} u_i\, S_i^k \tag{4.10}$$

where $u = (u_1, u_2, \cdots, u_m) \geq 0$. Now (4.10) is weaker than (4.9) because $S_k(u) \supseteq S_k$, where S_k and $S_k(u)$ are the set of binary solutions to (4.9) and (4.10) respectively. Constraints (4.10) has the advantage, however, of containing information different from that of any individual constraint of (4.9), unless u is a unit vector. Fathoming tests can be performed initially on the surrogate constraint (4.10), since if $S_k(u) = \emptyset$ then $S_k = \emptyset$, hence, we have the following properties :

(i) If z is a feasible solution to the original problem, then z is also feasible for the surrogate constraint.

(ii) If the surrogate constraint has no feasible solution, then neither does the original constraint set.

These properties are useful, since they show that the surrogate constraint can be used to make inferences about the infeasibility of the original problem. To illustrate the usefulness of the surrogate constraints, consider two inequalities :

$$2\,x_1 - x_2 \leq -1, \quad \text{and}$$
$$-\,x_1 + 2\,x_2 \leq -1.$$

Each seperate constraint has a feasible solution. However, when considered simultaneously, letting $u_1 = 1$, $u_2 = 1$, thus yielding the surrogate constraint $x_1 + x_2 \leq -2$. The new constraint immediately ascertains infeasibility.

Since u_i are restricted to be nonnegative values, there are infinete number of surrogate constraints can be developed for the same problem. A good surrogate constraint should have the property of eliminating as many nonoptimal completions as possible. Let

$$g(u) = \min. \sum_{j \in F_k} c_j\, x_j \qquad x \in S_k(u).$$

We say that u generates a stronger constraint than u' if $g(u) > g(u')$, i.e., the minimum (maximum) of a given objective function subject to the stronger surrogate constraint is larger (smaller) than its optimum value subject to the weaker one. Using the above definition, the best surrogate constraint would be generated by u*, where

$$g(u^*) = \max_{u \geq 0}. \; g(u). \qquad (4.11)$$

Solving (4.11) is not easy. A reasonable estimate of u* can be calculated, however, by dropping the integrality restriction in $S_k(u)$. In other words, let

$$g'(u) = \min. \sum_{j \in F_k} c_j x_j,$$

$$\sum_{j \in F_k} \sum_{i=1}^{m} u_i a_{ij} x_j \leq \sum_{i=1}^{m} u_i S_i^k, \qquad (4.12)$$

where $0 \leq x_j \leq 1, \quad j \in F_k,$

and say that the best surrogate constraint is generated by u^0, where

$$g'(u^0) = \max_{u \geq 0}. \; g'(u). \qquad (4.13)$$

The linear programming corresponding to the integer linear programming (4.9) at vertex k is :

$$\text{Minimize} \qquad \gamma_k = \sum_{j \in F_k} c_j x_j,$$

subject to

$$\sum_{j \in F_k} a_{ij} x_j \leq S_i^k, \quad i=1,2,\cdots,m, \qquad (4.14)$$

where $0 \leq x_j \leq 1$ for $j \in F_k$.

The dual problem of (4.14) is :

$$\text{Maximize} \qquad q_k = \sum_{i=1}^{m} y_i S_i^k, \qquad (4.15)$$

subject to

$$\sum_{i=1}^{m} a_{ij} y_i \leq c_j \quad \text{for} \quad j \in F_k,$$

where $y_i \leq 0$ for $i = 1,2,\cdots,m.$

The dual problem of (4.14) can also be as follows:

$$\text{Minimize} \qquad -q_k = \sum_{i=1}^{m} S_i^k v_i,$$

subject to

$$\sum_{i=1}^{m} a_{ij} v_i \geqslant -c_j \quad \text{for} \quad j \in F_k, \qquad (4.15')$$

where $v_i \geqslant 0$ for $i = 1, 2, \cdots, m$.

If (4.14) has no feasible solution, then vertex k is fathomed. If (4.14) has a feasible solution, it must have an optimal solution γ_k^0, since its constraint set is bounded. Assuming that (4.14) is feasible, (4.15) has an optimal solution (q_k^0, v^0) and $q_k^0 = \gamma_k^0$. We are now to show how to calculate the best surrogate constraint by the definition (4.13).

Theorem 4.1: The best surrogate constraint is generated by $u^0 = v^0$.

Proof : Clearly,

$$q_k^0 = \gamma_k^0 = \sum_{j \in F_k} c_j x_j^0 \geqslant g'(u) \quad \text{for any } u \geqslant 0,$$

since any feasible solution to (4.14) is also feasible to (4.12). Letting $u = v^0$ and x' be any feasible solution to (4.12) yields :

$$\sum_{j \in F_k} \sum_{i=1}^{m} v_i^0 a_{ij} x_j' \leqslant \sum_{i=1}^{m} v_i^0 S_i^k = -q_k^0, \qquad (4.16)$$

i.e.,

$$q_k^0 \leqslant \sum_{j \in F_k} \sum_{i=1}^{m} (-v_i^0) a_{ij} x_j'. \qquad (4.17)$$

Now, if v^0 is a feasible solution of (4.15), then

$$\sum_{j \in F_k} \sum_{i=1}^{m} (-v_i^0)\, a_{ij}\, x_j' \leq \sum_{j \in F_k} c_j\, x_j'. \qquad (4.18)$$

From (4.17) and (4.18), we verify that $q_k^0 \leq g'(v^0)$. Finally, $g'(u) \leq q_k^0 \leq g'(v^0)$ for any $u \geq 0$ implies that $g'(u^0) = g'(v^0)$.

Theorem 4.1 says that the best surrogate constraint is generated by the dual variables corresponding to the constraints of (4.9). Solving the linear programming (4.14) yields not only the best surrogate constraint but also bounds which are applicable for all tests. We can calculate the new surrogate constraints periodically or at every vertex, then decide to branch to new vertex or the vertex is fathomed.

EXAMPLE 4.5:

$$\text{Maximize} \quad Z = -5x_1 - 7x_2 - 10x_3 - 3x_4 - x_5$$

$$\text{subject to} \quad -x_1 + 3x_2 - 5x_3 - x_4 + 4x_5 \leq -2,$$

$$2x_1 - 6x_2 + 3x_3 + 2x_4 - 2x_5 \leq 0,$$

$$\text{and} \quad x_2 - 2x_3 + x_4 + x_5 \leq -1,$$

$$\text{where} \quad x_j = 0 \text{ or } 1 \quad \text{for } j = 1,2,\cdots,5.$$

SOLUTION: Solving the corresponding linear programming yields $x_1 = 0$, $x_2 = 1/3$, $x_3 = 2/3$, $x_4 = x_5 = 0$; $\bar{z} = -9$. The associated dual variables are $v_1 = 0$, $v_2 = 8/3$ and $v_3 = 9$. The resulting surrogate constraint is :

$$16/3\, x_1 - 7\, x_2 - 10\, x_3 + 43/3\, x_4 + 11/3\, x_5 \leq -9.$$

Apply Test 7 to this surrogate constraint, called row r, then $L_r = 1$, $U_r = 3$, $T_r(0,2) = -17 < -70/3$ and $x_4 = 0$. Branch to $J_1 = (\underline{-4})$. Now $R_1 = \{1,3\}$, $I_1(1) = 4$, $I_1(3) = 3$. So, we branch to $J_2 = (\underline{-4}, 3)$.

The resulting linear programming problem is :

$$\text{Maximize} \quad Z_2 = -5x_1 - 7x_2 - x_5 - 10,$$

subject to

$$-x_1 + 3x_2 + 4x_5 \leq 3,$$

$$2x_1 - 6x_2 - 2x_5 \leq -3,$$

and

$$x_2 + x_5 \leq 1,$$

where $0 \leq x_1, x_2, x_5 \leq 1$.

The optimal solution is $x_1 = 0$, $x_2 = 1/3$, $x_5 = 1/2$, $z_2 = -77/6$, yielding the bound $z_2 = -13$. The corresponding resulting surrogate constraint is :

$$21/9\, x_1 - 7x_2 - x_5 \leq -17/6.$$

Test 1 applied to this surrogate constraint determines that $x_2 = 1$, so we branch to $J_3 = (-4, 3, 2)$.

At v_3 : $J_3 = (\underline{-4}, 3, 2)$, $S^3 = (0, 3, 0)$, $Q_3 = \phi$.

We execute the backward move.

$w_3(3) = -7$, $w_3(2) = -6$, so, $J_4 = (\underline{-4}, 3, \underline{-2})$.

At v_4 : $J_4 = (\underline{-4}, 3, \underline{-2})$, $S^4 = (3, -3, 1)$, $Q_4 = \{2\}$, $z_4 = -10$, $R_4 = T_4 = \{5\}$, $C_4 \neq \phi$, then we execute the backward move.

At v_5 : $J_5 = (\underline{-4}, \underline{-3})$, $S^5 = (-2, 0, -1)$, $Q_5 = \{1,3\}$, $z_5 = 0$, $R_5 = T_5$ $T_5 = \{1\}$, $C_5 \neq \phi$, then we execute the backwardmove.

At v_6 : Since all elements of J_5 are underlined, $x_2 = x_3 = 1$ $x_1 = x_4 = x_5 = 0$, and max $z = -17$ is the optimal solution.

BIBLIOGRAPHY

BIBLIOGRAPHY

1 . Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero - One Variables," Operation Research, Vol. 13, pp. 517-546, 1965.

2 . Balinski, M. L., 'Integer Programming : Methods, Uses, Computation," Management Science, Vol. 12, pp. 253-313, 1965.

3 . Dantzig, G. B., Linear Programming And Extensions, Princeton University Press, Princeton, N. J., 1963.

4 . Gass, S. I., Linear Programming, McGraw-Hill Book Company, N. Y., 1969.

5 . Glover, F., "A Multiphase - Dual Algorithm for Zero - One Integer Programming Problem," Operation Research, Vol. 13, No. 6, pp. 879-919, 1965.

6 . Gomory, R. E., "Outline of An Algorithm for Integer Solutions to Linear Programs," Bulletin of The American Mathematical Society, vol. 64, pp. 275-278, 1958.

7 . Gomory, R. E., "All - Integer Programming Algorithm," Industrial Scheduling, edited by J. F. Muth and J. L. Thompson, Prentice Hall, Englewood Cliffs, N.J., 1963.

8 . Ryan, P. J., "A Composite Primal - Enumerative Integer Programming Algorithm," Operation Research House Technical Report No. 72-4, Stanford University, California, 1972.

9 . Young, R. D., "A Simplified Primal All Integer Programming Algorithm," Research Report, Rice University, 1966.

VITA

The author was born in Hsin-chu, Taiwan, Rep. of China, on October 5,1946. She was graduated from the First Girls' High School of Taipei in 1965. Then she attended National Tsing-Hua University in Taiwan and received a B.S. degree in Mathematics in 1969. In September, 1973 she entered the graduate school of the University of Tennessee in order to study for a Master's degree with a major in Mathematics. In the summer of 1974 she moved to Hartford,Connecticut.