A FAST ALGORITHM FOR SOLUTION OF TRANSPORTATION PROBLEMS

Ramesh Gupta

TECHNICAL REPORT

in

The Faculty

of

Engineering

Presented in Partial Fulfillment of the Requirements for
the Degree of Master of Engineering at
Sir George Williams University
Montreal, Canada

September, 1972

TABLE OF CONTENTS

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

CHAPTER 9

CHAPTER 10

APPENDIX

A FAST ALGORITHM FOR SOLUTION OF TRANSPORTATION PROBLEMS

ABSTRACT

The transportation problem has been formulated by various investigators and solved to various degrees. The systematic method of solution was first given by Dantzig. In general, the computational procedures are adaptation of the simplex method. However, almost all of the techniques either take too long to be solved by a digital computer or are not readily adaptable for use on digital computers.

The northwest corner rule has been presented for solving the transportation problem. The essentials of the stepping stone method are then reviewed. This technique does not consider costs for determining the initial basic feasible solution. Other techniques which make some use of costs are also described.

A modified technique for solving transportation problem by digital computer is then presented along with the unique features of the method which give its high speed in solving problems. The detailed logic of the method is also explained. This method was implemented and was found to reduce the solution time by 2.6 times as compared to the well known matrix minima method of solution.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to his supervisor, Prof. G. Martin for his encouragement and guidance throughout all stages of this work.

The author extends special thanks to Sir George Williams University for providing computer time which made this entire work possible.

LIST OF SYMBOLS

a	-	Basis table entry number of the search table.
b	-	Basis table entry number of the branch point table.
i	-	Row of basis table.
j	-	Column of basis table.
m	-	Number of sources (plants).
n	-	Number of destinations (customers).
t	-	Entry number of element.
Z	-	Objective function.
Δ	-	Perturbation.
A_a	-	Identifies F_a encountered during search $A_a = 0$ - corner point $= 1$ - branch point $= 2$ - branch point from which a new path was searched.
B_b	-	Indicates whether branch point was found during $B_b = 0$ - row search $= 1$ - column search.
C_{ij}	-	Unit cost from source i to destination j.
$\bar{C}_{ij}$	-	Indirect cost matrix.
ΔC	-	Incremental cost.
D_j	-	Demand of commodity by customer j.
F_a	-	Element number of branch or corner point encountered during search.

G_b - Element number of branch point encountered during search.

I_t - Row of basis element t.

J_t - Column of basis element t.

K_t - Number of an element in the same row of shipment matrix as element t.

L_t - Number of an element in the same column as an element t.

S_i - Supply of commodity at source i.

U_i - Dual variable for row i.

V_j - Dual variable for column j.

X_{ij} - Commodity shipped from source i to customer j.

INTRODUCTION

The purpose of this report is to present and implement a method of solution to a class of problems known as the transportation problems.

The report presents a brief history and definitions of the common terms used. It then describes in detail the stepping stone method which was implemented by the author. This new logic was found to provide a speed improvement over usual methods. Program listing and flow charts are provided in the appendixes.

CHAPTER 1

BACKGROUND

One of the earliest and most fruitful applications of linear programming techniques has been the formulation and solution of the transportation problems as a linear-programming problem.

L.V. Kantorovich showed that a class of problems closely related to the classical transportation case has a remarkable variety of applications concerned typically with the allotment of tasks to machines whose costs and rates of production vary by task and machine type[(5)*]. He gave a useful but incomplete algorithm for solving such problems. In 1942, he wrote a mathematical paper concerned with a continous version of the transportation problem, and in 1948, he authored an applicational study, jointly with Gavurin, on the capacitated transportation problem.

The now standard form of the problem was first formulated, along with a constructive solution, by Frank L. Hitchcock. His paper, "The Distribution of a Product from Several Sources to Numerous Localities"[(18)],

---

* Represents bibliography reference number.

sketched out the partial theory of a technique foreshadowing the simplex method; it did not exploit special properties of a transportation problem except in finding starting solutions. This paper also failed to attract much attention.

Still another investigator, T.C. Koopmans, as a member of the Combined Shipping Board during World War II, became concerned with using solutions of the transportation problem to help reduce overall shipping times, for the shortage of cargo ships constituted a critical bottleneck[10].

In 1947, Koopmans began to spearhead research on the potentialities of linear programs for the study of problems in economics. His historic paper, "Optimum Utilization of the Transportation System"[20], was based on his wartime experience. Because of this and the work done earlier by Hitchcock, the classical case is often referred to as the Hitchcock Koopmans Transportation Problem.

Another, whose work anticipated the recent era of development in linear programming was E. Egervary, a mathematician. His 1931 paper considered the problem of finding a permutation of ones in a matrix composed of zero and one elements[12]. Based on this investigation, Kuhn developed an efficient algorithmic method for solving assignment problems[21]. Kuhn's approach, in its turn, underlies the Ford-Fulkerson Method for solution of the classical transportation problem[11].

The linear-programming formulation and the associated systematic method of solution were first given by Dantzig[5]. The computational procedure is an adaptation of the simplex method applied to the system of equations of the associated linear-programming problem.

This report describes the implementation of a new digital computer technique for solving the classical transportation problem by the stepping stone method. This technique offers considerable advantage in speed over methods currently in use[6,15]. First, the essentials of the stepping stone method are reviewed. Then, the unique features of the method which give its high speed in solving problems with it, are presented. The detailed logic of the method is also given.

CHAPTER 2

FORMULATION OF TRANSPORTATION PROBLEM

The general transportation problem may be formulated as follows:

A Company operates m plants ("origins") producing a commodity, the ith of which can supply S_i units. The company sells its production to n customers ("destinations"), the jth of which demands D_j units of the commodity. The cost of manufacturing and transporting a unit of the commidity from plant i to customers j is C_{ij}. It is desired to find the number of units X_{ij} that should be shipped from each plant to each customer so that the total cost of the operation is a minimum.

To develop the constraints of the problem, set up Table 2.1. The amount shipped from source i to destination j is X_{ij}, the total shipped from source is $S_i \geqslant 0$ and the total received by destination j is $D_j \geqslant 0$.

Imposing temporarily the restriction that the total amount shipped is equal to the total amount received,

that is:

$$\sum_i S_i = \sum_j D_j = A \tag{2.1}$$

The total cost of shipping X_{ij} units is $(C_{ij} \cdot X_{ij})$. Since a

DESTINATIONS

(i) \ (j)	(1)	(2)	...	(3)	...	(n)	
(1)	X_{11}	X_{12}	...	X_{1j}	...	X_{1n}	S_1
(2)	X_{21}	X_{22}	...	X_{2j}	...	X_{2n}	S_2
.	.	.	...	.	...	.	.
.	.	.	...	.	...	.	.
(i)	X_{i1}	X_{i2}	...	X_{ij}	...	X_{in}	S_i
.	.	.	...	.	...	.	.
.	.	.	...	.	...	.	.
(m)	X_{m1}	X_{m2}	...	X_{mj}	...	X_{mn}	S_m
	D_1	D_2	...	D_j	...	D_n	

SOURCES

TABLE 2.1: Transportation Problem Table

negative shipment has no valid interpretation for the problem, each $X_{ij} \geqslant 0$.

Table 2.1 gives the mathematical definition of the problem:

Find values for the variables X_{ij} which minimize the total cost:

$$Z = \sum_{ij} C_{ij} X_{ij} \tag{2.2}$$

Subject to the constraints:

$$\sum_j X_{ij} = S_i \qquad i = 1,2 \ldots\ldots m \tag{2.3}$$

$$\sum_i X_{ij} = D_j \qquad j = 1,2 \ldots\ldots n \tag{2.4}$$

and

$$X_{ij} \geqslant 0 \tag{2.5}$$

Equation 2.3 represents the row sums of Table 2.1 and equation 2.4 the column sums.

In order for equation 2.3 and 2.4 to be consistent, the sum of equation 2.3 must be equal to the sum of equation 2.4.

That is,

$$\sum_i \sum_j X_{ij} = \sum_j \sum_i X_{ij} = \sum_i S_i = \sum_j D_j = A \tag{2.6}$$

System of equations 2.2 to 2.5 is a linear programming problem with $m + n$ equations in mn variables.

For $m = 3$ and $n = 5$, writing the equations corresponding to equations 2.3 and 2.4, gives 8 equations (that is $m + n$) in 15 (that is mn) unknowns as shown in Table 2.2.

THE TRANSPORTATION PROBLEM TABLE:

Consider the Table 2.3. In cell (i,j) enter C_{ij} and X_{ij}. If the X_{ij} entered in the table represents a feasible solution, it must be true that addition of the X_{ij} in row i yields S_i, for $i = 1,\ldots,m$. Similarly, addition of the X_{ij} in column j, yields D_j, for $j = 1,\ldots,n$. Hence, all the constraints are conveniently represented, and it is easy to check whether any set of X_{ij}, is a feasible solution by simply summing the rows and columns.

In the last column enter the origin availabilities and in the last row the destination requirements. It is also convenient to use D_j as a heading for column j to indicate that this column pertains to destination j. Similarly S_i, is placed at the beginning of row i to indicate that this row pertains to origin i.

In the lower right-hand cell of Table 2.3, enter the total amount to be shipped, i.e., $\sum S_i = \sum D_j$.

$$
\begin{array}{lllr}
X_{11} + X_{12} + X_{13} + X_{14} + X_{15} & & & = S_1 \\
 & X_{21} + X_{22} + X_{23} + X_{24} + X_{25} & & = S_2 \\
 & & X_{31} + X_{32} + X_{33} + X_{34} + X_{35} & = S_3 \\
X_{11} & X_{21} & X_{31} & = D_1 \\
\quad X_{12} & \quad X_{22} & \quad X_{32} & = D_2 \\
\qquad X_{13} & \qquad X_{23} & \qquad X_{33} & = D_3 \\
\qquad\quad X_{14} & \qquad\quad X_{24} & \qquad\quad X_{34} & = D_4 \\
\qquad\qquad X_{15} & \qquad\qquad X_{25} & \qquad\qquad X_{35} & = D_5
\end{array}
$$

TABLE 2.2: Equations 2.3 and 2.4 for $m = 3$, $n = 5$

	D_1	D_2		D_j		D_n	S_i
S_1	C_{11} X_{11}	C_{12} X_{12}	...	C_{1j} X_{1j}	...	C_{1n} X_{1n}	S_1
S_2	C_{21} X_{21}	C_{22} X_{22}	...	C_{2j} X_{2j}	...	C_{2n} X_{2n}	S_2
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
S_i	C_{i1} X_{i1}	C_{i2} X_{i2}	...	C_{ij} X_{ij}	...	C_{in} X_{in}	S_i
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
S_m	C_{m1} X_{m1}	C_{m2} X_{m2}	...	C_{mj} X_{mj}	...	C_{mn} X_{mn}	S_m
D_j	D_1	D_2	...	D_j	...	D_n	$S_i = D_j$

TABLE 2.3: Table for Transportation Problem (Including Costs)

When dealing with basic solutions, no more than $m + n - 1$ of the X_{ij} in Table 2.3 will be positive[16]. For basic feasible solutions, no more than $m + n - 1$ of the X_{ij} will ever be >0. Only the values of the basic variables will be entered in the table i.e., not fill in the zeros for the nonbasic variables. However, zero values of the basic variables will be written in.

CHAPTER 3

LOOPS AND TREES

In order to discuss the transportation problem in more detail, it is helpful to define certain terms:

ELEMENT or CELL:

An element is a position in the shipment matrix. The value of an element is the value of X_{ij} for the position of the element in the shipment matrix. Elements of particular interest may be designated by their column, or may be arbitrarily numbered.

DIRECTED PATH JOINING TWO CELLS:

A directed path from the cell (i,j) to the cell (v,w) in Table 2.3 is defined to be an ordered set of cells (i,j), (i,k), (q,k), (q,r),..., (v,w) or (i,j), (s,j), (s,t),..., (v,w) such that any two adjacent cells in the ordered sets lie alternately in the same row, then in the same column, while any three adjacent cells do not lie in the same row or same column. Furthermore, each cell (except the last) must appear only once in the ordered sets. The cell (i,j) is called the initial cell of the path, and (v,w) is called the terminal cell.

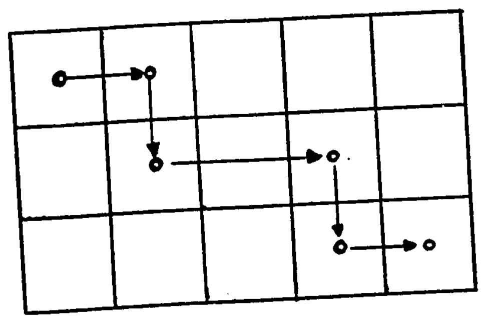


Fig. 3.1 DIRECTED PATH

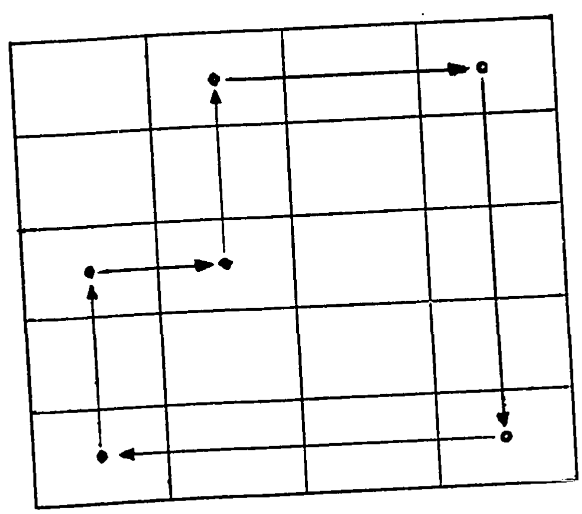


Fig. 3.2 DIRECTED LOOP

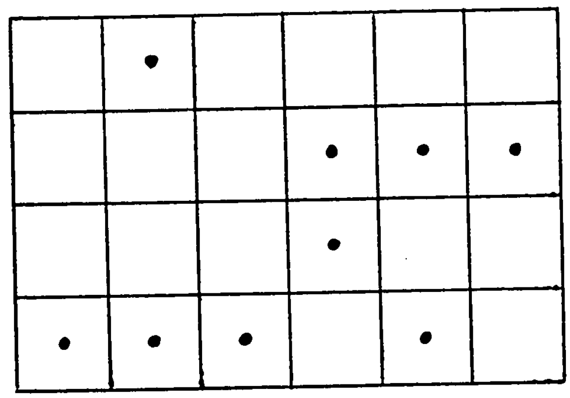


Fig. 3.3 EXAMPLE OF A BASIS

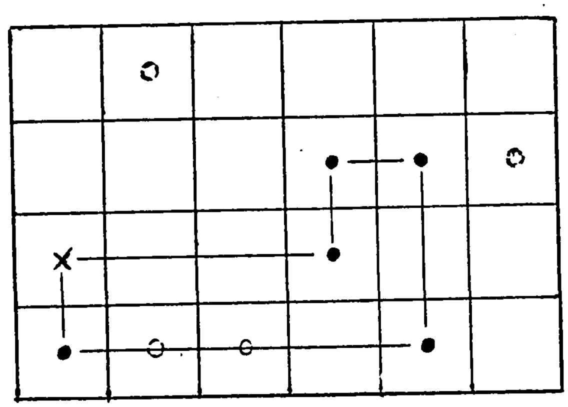


Fig. 3.4 LOOP INCLUDING NEW ELEMENT X

It is convenient to be able to illustrate graphically a path connecting two cells in Table 2.3 (shown in Fig. 3.1). To do this simply join by line segments the ordered set of cells which form the path. The direction is indicated by an arrowhead on the line. These line segments will be called branches.

DIRECTED LOOP:

A directed loop is a directed path such that the first cell in the ordered set is the same as the last cell, (that is, a path starting and terminating with the same cell). A typical directed loop is shown in Fig. 3.2.

TREE:

A tree is a connected set of cells without loops.

BASIS:

A basis is a tree in an m by n shipment matrix containing exactly m + n - 1 elements. A basis has the following properties:

a) There is at least one element of the basis in each row and each column of the shipment matrix.

b) Hence, if a new element is added to the tree, a unique loop is formed including that element.

An example of a basis and the loop formed by adding a new element is shown in Fig. 3.3 and 3.4.

CHAPTER 4

THE DUAL VARIABLES - CHANGE OF BASIS

In order to understand the transportation problem it is convenient to define an auxiliary variable U_i associated with each row and a V_j associated with each column of the cost matrix. These are the dual variables of linear programming theory. Their values are chosen so that

$$U_i + V_j = C_{ij} \tag{4.1}$$

for those combinations of i and j which correspond to elements of the basis.

Let C_{ir}, C_{qr}, C_{qt}, ..., C_{ws}, C_{wj} be the $m + n - 1$ prices corresponding to the variables in any basic feasible solution to a transportation problem with m origins and n destinations. Now suppose that:

$$\begin{aligned} U_i + V_r &= C_{ir}, \\ U_q + V_r &= C_{qr}, \\ U_q + V_t &= C_{qt}, \\ \vdots \quad & \quad \vdots \\ U_w + V_s &= C_{ws}, \\ U_w + V_j &= C_{wj}. \end{aligned} \tag{4.2}$$

Since there are $m + n - 1$ elements in the basis and the number of U's and V's is $m + n$ resulting in one more unknown than number

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1		3		5	5		13
S_2		4	1				5
S_3			7				7
S_4	3		2			6	11
D_j	3	7	10	5	5	6	36

TABLE 4.1 (a) Shipment Matrix

							U_i
	8	5	7	3	3	6	0
	5	6	3	2	5	4	1
	2	4	5	6	4	3	3
	5	3	6	7	8	4	4
V_j	1	5	2	3	3	0	

TABLE 4.1 (b) Cost Matrix

of equations, any one of the variables can be set arbitrarily.

Suppose now that, an element is to be added in cell (3,5) of the shipment matrix in Table 4.1. The circled elements in Table 4.2(a) constitute a basic loop including the new element. If the new element is to have the positive value ΔX, the changes indicated in Table 4.2(b) must be made in the elements of the loop so that the supply and demand requirements remain satisfied. The change in total cost made by bringing in this new element including the effect of the changes in the loop elements is

$$\Delta X\ (C_{35} - C_{15} + C_{12} - C_{22} + C_{23} - C_{33}) = -\Delta X \Delta C \qquad (4.3)$$

Using the definition of the U's and V's, this becomes

$$\begin{aligned} -\Delta X \Delta C &= \Delta X\ [C_{35} - (U_1 + V_5) + (U_1 + V_2) \\ &\qquad - (U_2 + V_2) + (U_2 + V_3) - (U_3 + V_3)] \\ &= \Delta X\ (C_{35} - U_3 - V_5) \end{aligned} \qquad (4.4)$$

Thus, if the quantity $\Delta C = U_i + V_j - C_{ij}$ is negative or zero, no gain can be made by introducing an element in cell (i,j). However, if this quantity is positive, it will pay to make ΔX as large as possible. For the present case $\Delta C = 2$ and a reduction of total cost may be obtained. The value of ΔX is limited by the element in cell (2,2) which becomes zero when $\Delta X = 4$. The new form of the shipment matrix is as in Table 4.3(a). Note that the non-zero elements again form with a tree containing exactly $m + n - 1$ elements which is therefore a new basis. Since the row and column sums of the shipment matrix are the same as before, the values of of the basic elements give a new feasible solution.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1		(3)		5	(5)		13
S_2		(4)	(1)				5
S_3			(7)		X		7
S_4	3		2			6	11
D_j	3	7	10	5	5	6	36

X - New Element.

◯ - Element in Loop Including New Element.

TABLE 4.2 (a): Basic Loop Including New Element.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1		$3+\Delta X$		5	$5-\Delta X$		13
S_2		$4-\Delta X$	$1+\Delta X$				5
S_3			$7-\Delta X$		ΔX		7
S_4	3		2			6	11
D_i	3	7	10	5	5	6	36

TABLE 4.2 (b): Changes in Elements of Loop.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1		7		5	1		13
S_2			(5)				5
S_3			(3)		4		7
S_4	(3)		(2)			(6)	11
D_j	3	7	10	5	5	6	36

TABLE 4.3 (a): Shipment Matrix after one Iteration.

							U_i
	8	5	7	3	3	6	0
	5	6	3	2	5	4	-1
	2	4	5	6	4	3	1
	5	3	6	7	8	4	2
V_j	3	5	4	3	3	2	

Table 4.3 (b): Cost Matrix after one Iteration.

The values of the dual variables U_i and V_j, for the new basis may be computed without reference to the cost data as follows:

Decrease the U for the row of the added element by ΔC. Now select the basic tree consisting of those elements of the basis which may be reached by basic paths from the added element which do not contain other elements in the column of the added element. Consider in turn the basic paths to the elements of the tree. If the last move along the path to an element is across a row, increase V for its column by ΔC; if the last move is up or down a column decrease the U for the row of the element ΔC.

In the example, the circled elements in Table 4.3(a) form the basic tree. The results are given in Table 4.3(b). New U's and V's again satisfy equations 4.1 for the elements of the new basis. The search for another element which, if added to the basis, would reduce the total cost may now be resummed to begin the next iteration.

When $$\Delta C = U_i + V_j - C_{ij} \qquad (4.5)$$

is less than or equal to zero for all combinations of i and j, the solution has been found.

CHAPTER 5

THE STEPPING STONE ALGORITHM

A set of X's which satisfies the restrictions of equations 2.3, 2.4, and 2.5 is called a feasible solution. If the set also minimizes C., it is an optimum feasible solution. One result of the theory of linear programming is that a feasible or optimum feasible solution to a transportation problem need contain no more than $m + n - 1$ non-zero X's [16,22]. The stepping stone method consists of:

(1) Generating a feasible solution having no more than $m + n - 1$ non-zero X's. Such a solution is called a basic feasible solution.

(2) Modifying the basic feasible solution to obtain a new basic feasible solution in a manner that will decrease the total cost.

(3) Repeating step (2) until no further change will result in a decrease of cost.

The theory of the general simplex method proves that this process leads to a basic feasible solution which is an optimum feasible solution[12,16].

Dantzig[5] shows that, for any basic feasible solution, numbers U_i and V_j can be found such that for those X_{ij} in the basic solution, $U_i + V_j = C_{ij}$. Dantzig also shows that, if $U_i + V_j = \bar{C}_{ij}$ for those variables not in the basic solution and if all $\bar{C}_{ij} - C_{ij} \leq 0$, then the basic feasible solution is also a minimum solution. If this condition of optimality is not satisfied, a new basic feasible solution can be readily obtained whose corresponding value of the objective function is less than (nondegeneracy assumed) the preceding value. Dantzig's ingenious computational procedure enables one to obtain basic feasible solutions without setting up the usual simplex table and to test for optimality, i.e., to compute the $Z_{ij} = \bar{C}_{ij}$, in terms of the basis vectors[12] without the explicit representation of the vectors not in the basis.

There are many methods of determining an initial basic feasible solution. It is worth-while to spend some time finding a good initial solution because it can considerably reduce the total number of iterations required to reach an optimal solution.

Most of the methods for determining an initial basic feasible solution assign a positive value to one variable and, at the same time, satisfy either a row or column constraint at each step.

NORTHWEST CORNER RULE

A particularly simple method of determining an initial basic feasible solution is the so-called Northwest Corner Rule, introduced by Charnes and Cooper[2].

Begin with cell (1,1). Set $X_{11} = \min(S_1, D_1)$. At this first step, satisfy either an origin or a destination requirement.

If $S_1 > D_1$, move to cell (1,2) and set $X_{12} = \min(S_1 - D_1, D_2)$. On the other hand, if $D_1 > S_1$, move to cell (2,1) and set $X_{21} = \min(D_1 - S_1, S_2)$. When $S_1 = D_1$, degeneracy occurs, this will be treated later.

At the second step, satisfy either the second origin or the second destination requirement. Continue in this way, satisfying at the k^{th} step either an origin or a destination requirement. Ultimately, this results is a feasible solution.

This method cannot yield more than $m + n - 1$ positive X_{ij} because, at each step, an origin or a destination requirement is satisfied[16]. After $m + n - 1$ steps, $m + n - 1$ of the constraints will be satisfied. In the absence of degeneracy, there are not less than $m + n - 1$ positive X_{ij}. In this case, it is clear that the resulting solution is a basic because the method of constructing the solution rules out any possibility of loops, and at each step, a row or a column constraint is satisfied. In constructing the solution, movement is always down and to the right, and hence a loop cannot be formed.

The basic feasible solution obtained by mean of the Northwest Corner rule may be far from optimal since the costs were completely ignored. Other methods of determining an initial basic feasible solution which do take account of the costs are considered in chapter 6. The additional effort spent in obtaining a good initial basic solution is worth while because it can considerably reduce the number of iterations which will be required to find an optimal solution.

The solutions obtained by the Northwest Corner rule (and similar schemes) are extreme-point solutions, and only such solutions need be considered as candidates for the minimum feasible solution.

Since virtually all applications of the transportation problem require the shipping of only whole units of the item being considered, it is useful to establish the following important property of the transportation problem[16].

Assuming the S_i and D_j are non-negative integers, then every basic feasible solution (i.e., extreme-point solution) has integral values. Any optimal basic solution to transportation problem will have the property that all positive X_{ij} will be integers.

This integrality property is peculiar to the transportation problem. In general, an optimal solution to an arbitrary linear programming problem may not have integral values for the variables. In fact, for the variables to be integers, a linear programming problem usually becomes a nonlinear programming problem[2,7]. Intuitively, this integrality property is expected to follow the physical nature of the problem: if it is profitable to ship a fraction of a unit to any destination, it is generally profitable to ship as large a quantity as possible. Since an integral number of units is required at each destination, an integral number of units will be shipped.

The reduced system of $m + n - 1$ equations in mn variables can, of course, be solved by the general simplex procedure. However, for even small values of m and n, the resulting system of equations becomes unwieldly for manual computation. This dilemma is resolved by the special adaptation of the simplex algorithm to the transportation problem.

EXAMPLE

Consider a problem involving 4 origins and 6 destinations. The origin availabilities, the destination requirements, and the costs are given in Table 5.1.

Note that $\sum S_i = \sum D_j = 181$

An initial basic feasible solution can be obtained by means of Northwest Corner Rule:

Set $X_{11} = \min (S_1, D_1) = \min (50, 30) = 30$

	D_1	D_2	D_3	D_4	D_5	D_6	S_i	U_i
S_1	2 / (30)	1 / (20)	3 / -2	3 / -4	2 / -1	5 / -4	50	2
S_2	3 / 0	2 / (30)	2 / (10)	4 / -4	3 / -1	4 / -2	40	3
S_3	3 / 2	5 / -1	4 / (10)	2 / (40)	4 / (10)	1 / 3	60	5
S_4	4 / -1	2 / 0	2 / 0	1 / -1	2 / (20)	2 / (11)	31	3
D_j	30	50	20	40	30	11	181	
V_j	0	-1	-1	-3	-1	-1		

TABLE 5.1 : Combined Allocation and cost Matrix

Thus satisfying the requirements of destination 1. But there are still units available at origin, hence set:

$$X_{12} = \min (S_1 - D_1, D_2) = \min (20,50) = 20$$

Now the first origin constraint is satisfied. Since an additional 30 units must be shipped to destination 2, set:

$$X_{22} = \min (D_2 - 20, S_2) = \min (30,40) = 30$$

Thus the requirements of destination 2 are satisfied. 10 units are still available at origin 2. Thus:

$$X_{23} = \min (10, D_3) = 10$$

An additional 10 units must be shipped to destination 3,

set: $X_{23} = 10$

This leaves 50 units still to be shipped from origin 3.

Hence set: $X_{34} = 40 = D_4$

and satisfy the requirement of destination 4. Since 10 units remain to be shipped from origin 3,

set: $X_{35} = 10$

The requirement at destination 5 is 30,

set: $X_{45} = 20$

This leaves 11 units to be shipped from origin 5, which is precisely the number of units required at destination 6.

Hence: $X_{46} = 11.$

Circle the X_{ij} values just obtained and note that they are $9 = m + n - 1$ in number. Furthermore, these cells do not form a loop. This is a basic feasible solution. All other X_{ij} are zero, which are not filled in. The initial basic feasible solution is shown in Table 5.1.

Now determine, for those variables in the basic solution, m numbers of U_i and n numbers of V_j such that:

$$
\begin{aligned}
U_1 + V_1 &= C_{11} = 2 \\
U_1 + V_2 &= C_{12} = 1 \\
U_2 + V_2 &= C_{22} = 2 \\
U_2 + V_3 &= C_{23} = 2 \\
U_3 + V_3 &= C_{33} = 4 \\
U_3 + V_4 &= C_{34} = 2 \\
U_3 + V_5 &= C_{35} = 4 \\
U_4 + V_5 &= C_{45} = 2 \\
U_4 + V_6 &= C_{46} = 2
\end{aligned}
\tag{5.1}
$$

Here there are 10 variables in 9 (i.e., m + n - 1) equations. Since equation 5.1 is an underdetermined set of linear equations (i.e., the number of unknowns exceeds the number of equations), the system has an infinite number of solutions. A solution could be determined by letting any one of the variables equal its corresponding C_{ij}. This reduces the number of unknowns by 1 and forces a unique solution of $m + n - 1$ equations in the remaining $m + n - 1$ variables.

Letting $V_1 = 0$, results in

$$\begin{array}{lll} U_1 = 2 & V_1 = 0 & \\ U_2 = 3 & V_2 = 1 & \\ U_3 = 5 & V_3 = -1 & \quad (5.2) \\ U_4 = 3 & V_4 = -3 & \\ & V_5 = -1 & \\ & V_6 = -1 & \end{array}$$

The results are shown in Table 5.1, the corresponding cost coefficients are circled.

Since all the equations of 5.1 are satisfied, $U_i + V_j = C_{ij}$ for those X_{ij} in the basic feasible solution. Now compute $\bar{C}_{ij} = U_i + V_j$ for all combinations (i,j), ($\bar{C}_{ij} = C_{ij}$ for all X_{ij} in the solution).

The next task is to compute $\bar{C}_{ij} - C_{ij}$. These are calculated and are shown in Table 5.1.

Not all $\bar{C}_{ij} - C_{ij} \leqslant 0$, so the initial basic feasible solution is not optimal. The largest $\bar{C}_{ij} - C_{ij}$ is $\bar{C}_{36} - C_{36} = 3$. Hence select X_{36} to be introduced into the solution at an unknown non-negative level ΔX.

As the row and column sums of the variables must equal the corresponding values of S_i and D_j, add or subtract ΔX from some of the other X_{ij} in the first solution.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	30	20					50
S_2		30	10				40
S_3			10	40	$10-\Delta X$	ΔX	60
S_4					$20+\Delta X$	$1i-\Delta X$	31
D_j	30	50	20	40	30	11	181

TABLE 5.2 : Introduction of ΔX

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	30	20					50
S_2		30	10				40
S_3			10	40		10	60
S_4					30	1	31
D_j	30	50	20	40	30	11	181

TABLE 5.3 : Shipment Matrix after one Iteration

Since $X \geqslant 0$ in cell (3,6) subtract ΔX from X_{35}, X_{46} and add ΔX to X_{45} in order to keep the row and column sums correct as shown in Table 5.2. The size of ΔX is restricted by those X_{ij} from which it is subtracted. ΔX cannot be larger than the smallest X_{ij} from which it is subtracted.

Here ΔX must be greater than zero and less than or equal to 10 in order to preserve feasibility. Let $\Delta X = 10$. The new solution is shown in Table 5.3.

The objective function for this solution is equal to:

$$382 - [\max(\bar{C}_{ij} - C_{ij}) > 0] . \Delta X = 382 - (3)(10)$$
$$= 352$$

The corresponding combined allocation and cost matrix table including new (U_i, V_j) is given in Table 5.4.

Here $\max(\bar{C}_{ij} - C_{ij}) > 0] = \bar{C}_{42} - C_{42} = \bar{C}_{43} - C_{43} = 3.$ Hence there is a tie. Cell (4,3) could be arbitrarily selected to enter the basis. Introducing $\Delta X > 0$ in cell (4,3) and adding and subtracting ΔX from X_{ij} results in Tables 5.5 and 5.6.

The objective function becomes

$$352 - [\max(\bar{C}_{ij} - C_{ij}) > 0] . \Delta X = 352 - (3)(1)$$
$$= 349$$

	D_1	D_2	D_3	D_4	D_5	D_6	S_i	U_i
S_1	2 / (30)	1 / (20)	3 / -2	3 / -4	2 / -4	5 / -7	50	2
S_2	3 / 0	2 / (30)	2 / (10)	4 / -4	3 / -4	4 / -5	40	3
S_3	3 / 2	5 / -1	4 / (10)	2 / (40)	4 / -3	1 / (10)	60	5
S_4	4 / 2	2 / 3	2 / 3	1 / 2	2 / (30)	2 / (1)	31	6
D_j	30	50	20	40	30	11	181	
V_j	0	-1	-1	-1	-3	-4	-4	

TABLE 5.4 : Combined Table after First Iteration

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	30	20					50
S_2		30	10				40
S_3			$10-\Delta X$	40		$10+\Delta X$	60
S_4			ΔX		30	$1-\Delta X$	31
D_j	30	50	20	40	30	11	181

TABLE 5.5 : Introduction of ΔX

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	30	20					50
S_2		30	10				40
S_3			9	40		11	60
S_4			1		30		31
D_j	30	50	20	40	30	11	181

TABLE 5.6 : Shipment Matrix after Second Iteration

The combined table with (U_i, V_j) is shown in Table 5.7.

Here $[\max(\bar{C}_{ij} - C_{ij}) > 0] = \bar{C}_{31} - C_{31} = 2$

Introduce $\Delta X > 0$ in the cell (3,1). Add and subtract ΔX as shown in Table 5.8, selecting $\Delta X = 9$ gives new solution as shown in Table 5.9.

The new objective function is:

$$349 - [\max(\bar{C}_{ij} - C_{ij}) > 0] \cdot \Delta X = 349 - (2)(9)$$
$$= 331$$

The new values of (U_i, V_j) are shown in Table 5.10.

Here $[\max(\bar{C}_{ij} - C_{ij}) > 0] = \bar{C}_{44} - C_{44} = 1.$

Introduce $\Delta X > 0$ in the cell (4,4), add and subtract ΔX as shown in Table 5.11. Selecting $\Delta X = 1$ gives the new solution as shown in Table 5.12.

The new objective function is:

$$331 - [\max(\bar{C}_{ij} - C_{ij}) > 0] \cdot \Delta X = 331 - (1)(1)$$
$$= 330$$

The new values of (U_i, V_j) are shown in Table 5.13.

Here all $\bar{C}_{ij} - C_{ij} \leqslant 0$ and this last solution is a minimum feasible solution.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i	U_i
S_1	2 / (30)	1 / (20)	3 / -2	3 / -4	2 / -1	5 / -7	50	2
S_2	3 / 0	2 / (30)	2 / (10)	4 / -4	3 / -1	4 / -5	40	3
S_3	3 / 2	5 / -1	4 / (9)	2 / (40)	4 / 0	1 / (11)	60	5
S_4	4 / -1	2 / 0	2 / (1)	1 / 1	2 / (30)	2 / -3	31	2
D_j	30	50	20	40	30	11	181	
V_j	0	-1	-1	-3	0	-4		

TABLE 5.7 : Combined Table after Second Iteration

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	$30-\Delta X$	$20+\Delta X$					50
S_2		$30-\Delta X$	$10+\Delta X$				40
S_3			$9-\Delta X$	40		11	60
S_4			1		30		31
D_j	30	50	20	40	30	11	181

TABLE 5.8 : Introduction of ΔX

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	21	29					50
S_2		21	19				40
S_3				40		11	60
S_4			1		30		31
D_j	30	50	20	40	30	11	181

TABLE 5.9 : Shipment Matrix after Third Iteration

	D_1	D_2	D_3	D_4	D_5	D_6	S_i	U_i
S_1	2 / (21)	1 / (29)	3 / -2	3 / -2	2 / -1	5 / -5	50	2
S_2	3 / 0	2 / (21)	2 / (19)	4 / -2	3 / -1	4 / -3	40	3
S_3	3 / (9)	5 / -3	4 / -2	2 / (40)	4 / -2	1 / (11)	60	3
S_4	4 / -1	2 / 0	2 / (1)	1 / 1	2 / (30)	2 / -1	31	3
D_j	30	50	20	40	30	11	181	
V_j	0	-1	-1	-1	-1	-2		

TABLE 5.10: Combined Table after Third Iteration

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	21- ΔX	29+ ΔX					50
S_2		21- ΔX	19+ΔX				40
S_3	9 X			40- ΔX		11	60
S_4			1-ΔX	ΔX	30		31
D_j	30	50	20	40	30	11	181

TABLE 5.11: Introduction of ΔX

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	20	30					50
S_2		20	20				40
S_3	10			39		11	60
S_4				1	30		31
D_j	30	50	20	40	30	11	181

TABLE 5.12: Shipment Matrix after Fourth Iteration

	D_1	D_2	D_3	D_4	D_5	D_6	S_i	U_i
S_1	2 / (20)	1 / (30)	3 / -2	3 / -2	2 / 0	5 / -5	50	2
S_2	3 / 0	2 / (20)	2 / (20)	4 / -2	3 / 0	4 / -3	40	3
S_3	3 / (10)	5 / -3	4 / -2	2 / (39)	4 / -1	1 / (11)	60	3
S_4	4 / -2	2 / -1	2 / -1	1 / (1)	2 / (30)	2 / -2	31	2
D_j	30	50	20	40	30	11	181	
V_j	0	-1		-1	0	-2		

TABLE 5.13: Combined Table after Fourth Iteration

The stepping-stone algorithm permits the solution of transportation problems that are too large to lend themselves to direct application of the simplex method. For example, on a computer, it is not difficult to solve a transportation problem involving 1,000 destinations and 50 origins by means of the stepping-stone algorithm or some of its variants.

The stepping-stone algorithm has another property which is very important when a digital computer is to be used: it requires only the arithmetic operations of addition and subtraction. A digital computer can be made to operate in fixed-point arithmetic so that no rounding off errors occur in addition or subtraction. Thus, the stepping-stone algorithm makes it possible to avoid the problem of rounding-off errors which limit the size of the problems that can be solved by the simplex method. A transportation problem may require an arbitrarily large number of iterations, and no loss of accuracy due to rounding-off will occur if fixed-point arithmetic is used.

CHAPTER 6

DETERMINATION OF AN INITIAL BASIC FEASIBLE SOLUTION

The Northwest-Corner rule for determining an initial basic feasible solution to a transportation problem has already been discussed. Now some other methods which often yield a result much closer to an optimal solution than that obtained by the Northwest-Corner rule will be presented. As suggested before, it is worth while to spend some time finding a good initial solution because it can considerably reduce the total number of iterations required to reach an optimal solution.

Most of the methods for determining an initial basic feasible solution assign a positive value to one variable and, at the same time, satisfy either a row or column constraint at each step. Any procedure for determining a feasible solution which assigns a positive value to one variable and satisfies either a row or column constraint at each step will automatically yield a basic feasible solution, and in the absence of degeneracy, the resulting cells will form a basic tree. Such a technique cannot give more than $m + n - 1$ positive variables since, after $m + n - 1$ steps, $m + n - 1$ of the constraints will be satisfied, and the remaining constraint is automatically satisfied[16]. All methods described below make some use of the costs.

(1) COLUMN MINIMA.

Beginning with column 1 of the Table 2.5, choose the minimum cost in this column. Suppose that it occurs in row r. Then set $X_{r1} = \min(S_r, D_1)$. If $X_{r1} = D_1$, cross off column 1 and move to column 2. If $X_{r1} = S_r$, cross off row r from the table, and choose the next lowest cost in column 1. Assume that it occurs in row s. Set $X_{s1} = \min(S_s, D_1 - S_r)$. Continue in this way until the requirement at the first destination is satisfied. If the minimum cost is not unique, select any one of the minima. When the requirement of column 1 is satisfied, cross off column 1 and repeat the above procedure for column 2. Continue until the requirement of column n is satisfied.

In the event that a row constraint and a column constraint, say column k, are satisfied simultaneously, cross off only the row. Then move to the cell in column k having the next lowest cost. Assign a value of zero to this cell and assume it to be in the basic solution. Now cross off column k and move to column k + 1. This will yield a degenerate basic feasible solution.

If this procedure is used to obtain an initial basic solution for the example solved in Table 5.1, Table 6.1 is obtained, provided that in columns 2 and 3 the row with the lowest index is chosen when the minimum cost is not unique.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	2 / 30	1 / 20	3	3	2	5	50
S_2	3	2 / 30	2 / 10	4	3	4	40
S_3	3	5	4	2 / 19	4 / 30	1 / 11	60
S_4	4	2	2 / 10	1 / 21	2	2	31
D_j	30	50	20	40	30	11	181

TABLE 6.1: Example of Column Minima.

(2) ROW MINIMA:

Beginning with row 1, choose the minimum cost in this row. Suppose that it occurs in column r. Set $X_{1r} = \min(S_1, D_r)$. If $X_{1r} = S_1$, cross off row 1 and move to row 2. If $X_{1r} = D_r$, cross off column r and determine the next lowest cost in row 1. Assume it occurs in column s. Set $X_{1s} = \min(S_1 - D_r, D_s)$. Continue in this way until the first row constraint is satisfied. When the requirement of the first row is satisfied, cross off row 1 and repeat the above procedure for row 2. Continue until the row constraint m is satisfied. Whenever a minimum cost is not unique, make an arbitrary choice among the minima.

In the event that a row constraint, say row k, and a column constraint are satisfied simultaneously, cross off only the column. Then find the next lowest cost in row k, and insert this cell into the solution at a zero level. Then cross off row k, and move on to row k + 1.

If this technique is used to obtain a first solution for the problem of Table 6.1, Table 6.2 results. Note that degeneracy appears. According to the rule, either cell (1,1) or (1,5) could have been added at a zero level. Cell (1,1) is chosen.

(3) MATRIX MINIMA.

Determine the smallest cost in the entire table. Suppose this occurs for cell (i,j). Set $X_{ij} = \min(S_i, D_j)$. Then cross off either

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	2 (0)	1 (50)	3	3	2	5	50
S_2	3 (20)	2	2 (20)	4	3	4	40
S_3	3 (9)	5	4	2 (40)	4	1 (11)	60
S_4	4 (1)	2	2	1	2 (30)	2	31
D_j	30	50	20	40	30	11	181

TABLE 6.2: Example of Row Minima.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	2 / 0	1 / 50	3	3	2	5	50
S_2	3 / 20	2	2 / 20	4	3	4	40
S_3	3 / 10	5	4	2 / 9	4 / 30	1 / 11	60
S_4	4	2	2	1 / 31	2	2	31
D_j	30	50	20	40	30	11	181

<u>TABLE 6.3</u>: Example of Matrix Minima.

row i or column j, depending on which requirement is satisfied. If $X_{ij} = S_j$, decrease D_j by S_i, and if $X_{ij} = D_j$, decrease S_i by D_j. Repeat the process for the resulting table. Whenever the minimum cost is not unique, make an arbitrary choice among the minima. If a row and a column constraint are satisfied simultaneously, cross off only the row or the column, not both.

This method yields the initial solution (shown in Table 6.3) for the example under consideration if, in the absence of a unique minimum, the cell for which $i + j$ is smallest is chosen. Here again degeneracy appears.

(4) VOGEL'S METHOD.

This technique has been suggested by Vogel[23]. For each row, find the lowest cost C_{ij} and the next lowest cost C_{it} in that row. Compute $C_{it} - C_{ij}$. In this way, m numbers are obtained. Proceed in the same way for each of the columns and obtain n more numbers. Choose the largest of these $m + n$ differences. Suppose that the largest of these numbers was associated with the difference in column j. Let cell (i,j) contain the lowest cost in column j. Then set $X_{ij} = \min(S_i, D_j)$. Cross off either row i or column j, depending on which requirement is satisfied, and repeat the whole process for the resulting table. When the maximum difference is not unique, an arbitrary choice can be made, and if a row and a column constraint are satisfied simultaneously, cross off only the row or the column, not both.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i	
S_1	2 / 30	1 / 20	3	3	2	5	50	1
S_2	3	2 / 30	2 / 10	4	3	4	40	1
S_3	3	5	4	2 / 40	4 / 9	1 / 11	60	1
S_4	4	2	2 / 10	1	2 / 21	2	31	1
D_j	30	50	20	40	30	11	181	
	1	1	1	1	1	1		

TABLE 6.4: Example of Vogel's Method.

For the above example, this method yields the solution shown in Table 6.4. It is convenient to list the row differences in a column to the right of the table and the column differences in a row at the bottom of the table. The differences shown in the difference row and column are those for the first step, i.e., those which are to be used in selecting the first basis cell. This is the worst possible case; every difference has the same value. The tie is resolved by choosing the cell with the smallest value of $i + j$. At each step, a new set of differences must be computed.

CHOICE OF METHOD.

Unfortunately, there is no easy means at present for determining an initial basic feasible solution which would lead to the smallest number of iterations[16]. It would be necessary to solve the problem in each case.

CHAPTER 7

DEGENERACY AND THE TRANSPORTATION PROBLEM

A feasible solution to a transportation problem is degenerate if less than $m + n - 1$ of the X_{ij} are positive[8]. Degeneracy may be encountered in the process of determining the initial basic feasible solution or at some subsequent iteration. From the practical point of view, degeneracy does not cause any difficulties. No transportation problem has ever been known to cycle [16]. The degeneracy problem can be eliminated by the use of a perturbation method as in the case of simplex method. However, a much simpler perturbation method can be used.

DEGENERACY DUE TO INSUFFICIENT POSITIVE X_{ij}'s:

If h is the number of $X_{ij} > 0$ and if the method used to provide an initial solution yields a feasible solution with $h < m + n - 1$ degeneracy occurs. The cells associated with the positive X_{ij}, do not form a basic tree. To obtain an initial basic solution and a basic tree in the table, $m + n - 1 - h$ additional cells at a zero level must be added. Choose the cells to be added such that the resulting $m + n - 1$ cells form a basic tree. Enter a zero into the cells added and circle these zeros to indicate that they are part of the initial basic solution. In manual computations,

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	20	5					25
S_2		25	0	X			25
S_3			40	10			50
S_4				40	0		40
S_5					10	20	30
D_j	20	30	40	50	10	20	170

TABLE 7.1: Degeneracy where $h < m + n - 1$ of Positive X_{ij}

it is very easy to choose cells which will yield a basic tree. Having determined a basic solution, proceed in the usual way. In this case a variable can enter and leave the solution at a zero level.

Consider the Table 7.1. Costs are omitted since they are not relevant to the discussion. The northwest-corner rule to find an initial feasible solution results in Table 7.1. Here there are only 8 positive X_{ij}, although $m + n - 1 = 10$. It will be noted that the set of cells corresponding to the positive X_{ij}, is not connected.

Two more cells are needed to obtain a basic tree. Adding cells (2,3) and (4,5) (dashed circles), a basic tree results. The value zero is entered into these cells, giving a basic (degenerate) feasible solution.

If the $\bar{C}_{ij} - C_{ij}$ are such that cell (2,4) should appear in the basic tree at the next iteration, X_{24} will enter the basic solution at a zero level. The values of the variables in the basic solution remain unchanged. Only cell (2,3) is replaced by cell (2,4) to obtain a new basic tree.

DEGENERACY DUE TO TIE OF VARIABLES:

Degeneracy can also appear at some later iteration if there is a tie for the variable to leave the basic solution. Choose arbitrarily any one of the tied variables as the variable to leave the basis. At the next stage, the variables that were tied with the removed variable will be at a zero level. However, keep these variables in the basic solution (i.e.,

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	(30)	(20)					50
S_2		(30)	(10)				40
S_3			(10)	(40)	(10)		60
S_4					(20)	(10)	30
D_j	30	50	20	40	30	10	180

TABLE 7.2 : Initial Basic Feasible Solution (with tie of Variables)

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1	30	20					50
S_2		30	10				40
S_3			10	40		10	60
S_4					30	0	30
D_j	30	50	20	40	30	10	180

TABLE 7.3 : Solution of Table 7.2, Degeneracy occurred

they remain circled) and proceed as usual.

Suppose that in the problem in Table 5.1 the availability at origin is changed from 4 to 30 and the requirement at destination 6 to 10. For this new problem, the northwest-corner rule yields the initial basic feasible solution shown in Table 7.2.

The $\overline{C}_{ij} - C_{ij}$ are the same as in Table 5.1, so X_{36} enters the basic solution at the next stage. Now, there is a tie for the variable to be removed. Either X_{35} or X_{46} can be replaced by X_{36}. If X_{35} is arbitrarily replaced, the new basic solution is that shown in Table 7.3. One of the basic variables is now zero, and degeneracy has appeared. Again, the $\overline{C}_{ij} - C_{ij}$ are the same as in Table 5.4. If X_{43} is chosen to enter the basis, it enters at a zero level.

Suppose that at step k, a row and a column constraint are satisfied simultaneously, this means that degeneracy has appeared. Now, imagine that either the row or column constraint satisfied at step k is perturbed by increasing its requirement by Δ. Then continue the process of finding the initial solution. Hadley(16) has shown that the resulting solution will be basic, and when Δ's are set to zero, a degenerate basic solution is found. It is really unnecessary to introduce Δ's explicitly. It eliminates the necessity of introducing numerical values of Δ's, when problem is solved with digital computers. Thus cells at zero level can be automatically added.

CHAPTER 8

MODIFIED STEPPING STONE ALGORITHM FOR DIGITAL COMPUTERS

Associated with each basic feasible solution is a basis whose elements include all positions in the shipment matrix where X_{ij} does not equal zero. The following procedure may be used to form an initial basic feasible solution and the corresponding basis with which to start the stepping stone method.

Examine the entries in the first row of the cost matrix and select the entry having the smallest cost. Include this position in the shipment matrix as an element of the basis, with its value equal to the smaller of the supply for its row and the demand for its column. Decrease the supply and demand by this amount. If a positive supply is left, drop the column whose demand was just satisfied from further consideration and again look for the lowest cost in this row. If a positive demand is left, insert an element in the same column but the next row, and again check whether there is supply or demand left over, and proceed as above. This is the row minima method described in Chapter 6.

This process yields in general a basis containing exactly m + n - 1 elements with no loops[12]. An example of a shipment matrix and a basic feasible solution generated in this manner from given supply and demand data is shown in Table 8.2 and 8.3.

DATA STORAGE IN THE COMPUTER:

The input data required to specify a transportation problem are:

(1) cost matrix C_{ij}: (mn) values,

(2) supplies and demands S_i and D_j: (m + n) values.

After the formation of the initial basic solution, the data required in the performance of the iterations are:

(1) shipment matrix X_{ij}: (m + n - 1) values,

(2) dual variables U_i and V_j: (m + n) values,

(3) cost matrix C_{ij}: (mn) values.

The largest volume of data that must be handled is the cost data. However, it is only necessary to have access to this data sequentially. Therefore, it may be recorded on some slow access, high capacity storage medium like magnetic tape or magnetic drum.

MAJOR COMPUTER ROUTINES:

The block diagram of the major computer routines required for the transportation problem is shown in figure 8.1.

Part 1 and 8:

The input and output blocks are not of interest becuase they involve no unusual ideas.

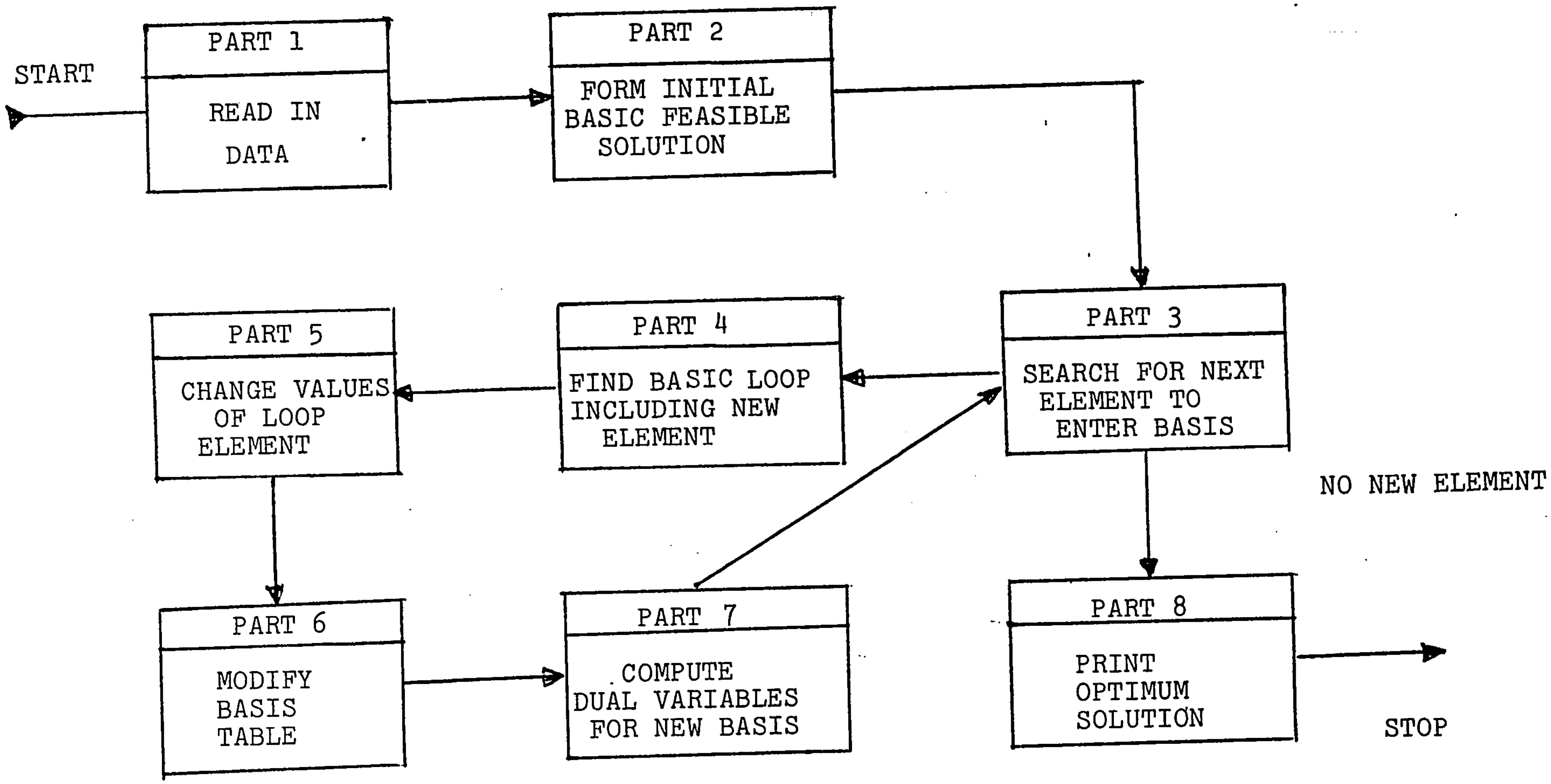


Fig. 8.1: MAJOR ROUTINES

1. The flow charts are given in Appendix A except for Part 1 and 8 which are user dependent.

2. The program listing is given in Appendix B.

	D_1	D_2	D_3	D_4	D_5	D_6	S_i
S_1		2					2
S_2				4	5	6	15
S_3	X						7
S_4	1	8	3		9		21
S_5						10	10
D_j	1	10	3	11	14	16	55

Table 8.2: Shipment Matrix

t	I_t	K_t	J_t	L_t
1	4	8	1	1
2	1	2	2	8
3	4	1	3	3
4	2	6	4	7
5	2	4	5	9
6	2	5	6	10
7	3	7	4	4
8	4	9	2	2
9	4	3	5	5
10	5	10	6	6

Table 8.3: Basis Table

Part 2:

The "Form initial basic feasible solution" block sets up a feasible solution in the manner described above, computes K and L values for the basis table, and calculates the initial values of the dual variables U_i and V_j.

Part 3:

This block searches through the cost data in sequence and selects a position in the shipment matrix for which $U_i + V_j - C_{ij}$ is positive as the next element to be added to the basis. If no such position is found, the present basis is an optimum solution and the results are printed. The amount of cost data which is examined for each iteration by this block has a very important effect on the overall time requirement for solving a problem.

Part 4 and 5:

The next two blocks perform the tasks of finding the basic loop including the new element, finding the element that limits the size of the new element and hence drops from the basis, and changing the values of the elements in the loop.

Part 6:

The "Modify basis table" block makes the necessary changes in the K and L values of the basis table to take care of the new element. It also shifts an entry in the table, if required, to maintain the ordering of the first n entries.

Part 7:

Block seven computes the U's and V's for the new basis. The details of each routine (except Part 1 and 8) are given in Appendix A and a program listing in Appendix B. Details for Part 1 and 8 (input/output blocks) are not given as they are user dependent.

DETAILS OF THE PROCEDURE:

From the discussion of the stepping stone method it is seen that the shipment matrix must be referred essentially at random. Therefore, in order to facilitate high speed operation, the shipment matrix must be stored in the high speed memory of the computer. Since, at most m + n - 1 of the positions of the shipment matrix can contain non-zero X_{ij}'s, a basic feasible solution may be stored as a table of m + n - 1 entries giving the row, column and value of each element of the basis.

In carrying out the processes of the stepping stone method, basic paths between elements must be traced. This requires much referring on the part of the computer between basic elements either in the same column or the same row of the shipment matrix. In order to allow the computer to proceed as rapidly as possible in tracing the paths, the following information may be stored for each entry in the basis table.

(1) Data informing the computer where to find entries for the other basic elements in the same row of the shipment matrix.

(2) Data informing the computer where to find entries for the other basic elements in the same column of the shipment matrix.

One convenient way of representing this information is shown in Table 8.3. For entry number t of the table, the value of K_t gives the entry number of another basic element in the same row; the value of L_t gives the entry number of a basic element in the same column of the shipment matrix. The table is arranged so that by jumping from entry to entry as directed by K (or L) values, all basic elements in a particular row (column) of the shipment matrix will be encountered. Also, for ease in finding the entry for a basic element in a particular column, the first n entries of the basis table are for elements in columns of the shipment matrix corresponding to their entry number. These features of the basis table give the program to be described its high speed. However, the arrangement has the slight disadvantage that as the basis changes with each iteration, the new basis table must be calculated.

The numbers in the shipment matrix are the entry numbers of the elements in the basis table and not the values of the elements. Each point on the graph is identified by the entry number of the element it represents in the basis table. The graph starts with the element in the basis table whose entry number is the column number of the element being introduced. This element will always be in the same column of the shipment matrix as the new element, i.e., column 1 in the example. A move from left to right in the graph correcponds to moving from one element in a given row of the shipment matrix to the element in the same row designated by the K value of the first in the basis table. Similarly, a move down in the graph corresponds to moving to the element in the same column of the shipment matrix designated by the L value in the basis table.

In searching for the loop involving the new element, start from an element in the same column as the new element. Hence a basic path from the starting element to an element in the same row as the new element must be found. This is done by searching the branches of the graph sucessively until an element in the correct row is found. Sufficient information must be remembered during the search so that the path may be identified once it is found. The search is carried out as follows: First, a path is traced through the tree without turning at any of the branch points as shown by the arrow labelled 1 in figure 8.4. While tracing this path the branch points encountered are as in Table 8.5 (b) to provide starting points for searching the remaining branches of the graph. Entry b in the branch point table gives the basis table entry number of the branch point G_b and an indicator B_b telling whether the branch point was encountered while moving across a row ($B_b = 0$) or down a column ($B_b = 1$). A separate tabulation, Table 8.5 (a) is kept of those elements (branch points and corner points) which could belong to the desired loop. Entry a in the search table gives the basis table entry number F_a of the element, and tells whether it is a corner point ($A_a = 0$) or branch point ($A_a = 1$) of the graph.

Since an entry in row 3 was not found while searching this path, the last entry in the branch point table is examined and the search is continued by tracing the path indicated by the arrow labelled 2 in figure 8.4. The number of this branch point is recorded in the search table and

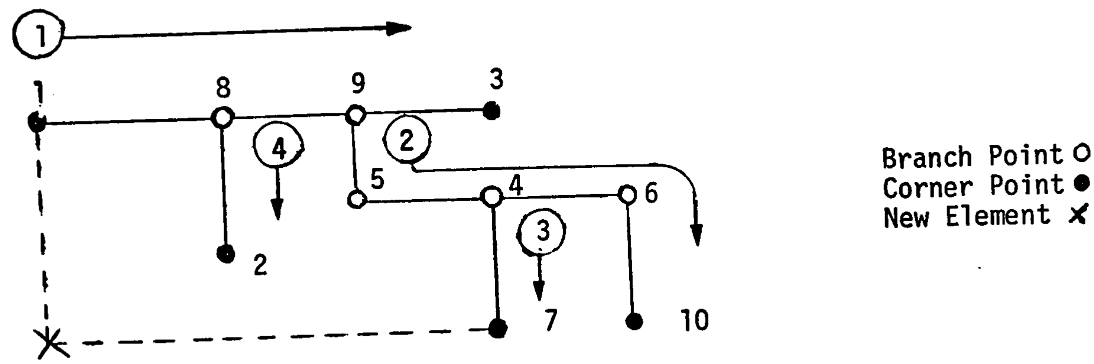


Fig. 8.4: Sequence of Searching the Branches of a Tree

a	F_a	A_a
1	1	0
2	8	1
3	9	1

(a) Search Table

b	G_b	B_b
1	8	0
2	9	0

(b) Branch Point Table

Table 8.5: First Stage of the Search for a Loop

identified by A_a = 2 to denote that a new path is being traced as shown in Table 8.6. The search is continued as indicated in figure 8.4 until element 7 is reached which is in row 3. Since element 7 is an element of the desired loop, it is entered in the search table. The state of the tables is then as in Table 8.7, and the search is terminated.

The decisions as to whether a particular element t is a branch point or a corner point, and what basic element is to be examined next, may be easily made by examining K and L values in the basis table. For instance, suppose we have just moved to element t from another element in the same row. Let h be the entry number of the first element that was examined in this row. Then,

(a) if $K_t = h$ and $L_t = t$, the end of this path has been reached.

(b) if $k_t = h$ and $L_t \neq t$, element is a corner point; examine element number L_t next.

(c) if $K_t \neq h$ and $L_t = t$, element t is neither a branch or corner point; examine element number K_t next.

(d) if $K_t \neq h$ and $L_t \neq t$, element is a branch point; examine element number K_t next.

Completely analogous rules apply for moving an element in the same column of the shipment matrix.

The determination of the elements of the basic loop from the entries in the search table may be accomplished by considering the entries

a	F_a	A_a
1	1	0
2	8	1
3	9	1
4	9	2
5	5	0
6	4	1
7	6	0

(a) Search Table

b	G_b	B_b
1	8	0
2	Ø	Ø
2	4	0

(b) Branch Point Table

Table 8.6: Second Stage of Search

a	F_a	A_a
1	1	0
2	8	1
3	9	1
4	9	2
5	5	0
6	4	1
7	6	0
8	4	2
9	7	0

(a) Search Table

b	G_b	B_b
1	8	0

(b) Branch Point Table

Table 8.7: Completion of Search

of the search table in sequence starting with the last and applying the following rules:

(1) The last element in the search table is the first element of the loop.

(2) If entry a of the search table has $A_a = 0$, element F_a is in the loop.

(3) If entry a of the search table has $A_a = 1$, it is ignored except as noted below.

(4) If entry a of the search table has $A_a = 2$, all entries are ignored until an entry b is found with $A_b = 1$, $F_b = F_a$. Element F_a is a member of the loop.

Essentially the identical search technique is used in calculating the new values of the dual variables. The difference is that the object of the search is to examine all of the elements in a particular sub-tree of the basis rather than to find an element in a particular row.

When the logical search procedure is used on a high speed computer, the time required to carry out the operations of one iteration becomes short compared to the time required to search through the entire cost data in finding the new element to be put in the basis. Thus, the total time for solution might be cut down if the time spent searching the cost data per iteration were shortened at the expense of a larger number of iterations.

The average time required to perform an iteration using the method described here depends linearly on the size of the problem, m + n. This is a consequence of the fact that in searching for a loop, each entry in the basis table is considered no more than once. Since experience has shown that the number of iterations required to solve a problem is roughly proportional to m + n, the overall time requirements should increase as the square of the size of the problem. All this indicates the feasibility of solving extremely large transportation type problems in a economical amount of time.

The example of Chapter 5 was solved by this technique. The results are shown below.

```
NUMBER OF ROWS    -M =    4
NUMBER OF COLS    -N =    6

**D**    30    50    20    40    30    11
**S**    50    40    60    31
```

ITERATION 1 *** NUMBER OF ROWS -M = 4 NUMBER OF COLS -N = 6

**** BASIS TABLE ****

**KT*	*I(KT)*	*K(KT)*	*J(KT)*	*LS(KT)*+	*X(KT)*	*F(KT)*	*A
1	4	9	1	0	30	0	
2	1	5	2	0	50	0	
3	2	7	3	2	10	0	
4	3	6	4	3	39	0	
5	1	2	5	1	0	0	
6	3	8	6	0	11	0	
7	2	3	5	0	30	0	
8	3	4	3	0	10	0	
9	4	1	4	0	1	0	

**** SHIPMENT MATRIX ****

*	50*+	0+	50+	0+	0+	0+	0+
*	40*+	0+	0+	10+	0+	30+	0+
*	60*+	0+	0+	10+	39+	0+	11+
*	31*+	30+	0+	0+	1+	0+	0+
*	181*+	30+	50+	20+	40+	30+	11+

**** COST MATRIX ****

*	0*+	0+	1+	0+	0+	2+	0+
*	1*+	0+	0+	2+	0+	3+	0+
*	3*+	0+	0+	4+	2+	0+	1+
*	2*+	4+	0+	0+	1+	0+	0+
*	410*+	2+	1+	1+	-1+	2+	-2+

TOTAL COST 410

**** ELEMENT MATRIX ****

*	1*+	0+	2+	0+	0+	5+	0+
*	2*+	0+	0+	3+	0+	7+	0+
*	3*+	0+	0+	8+	4+	0+	6+
*	4*+	1+	0+	0+	9+	0+	0+

ITERATION 2 *** NUMBER OF ROWS -M = 4 NUMBER OF COLS -N = 6

**** BASIS TABLE ****

**KT*	*I(KT)*	*K(KT)*	*J(KT)*	*LS(KT)*	*X(KT)*	*F(KT)*	*A
1	3	6	1	0	30	1	
2	1	5	2	0	50	9	
3	2	7	3	2	10	4	
4	3	1	4	3	9	0	
5	1	2	5	1	0	0	
6	3	8	6	0	11	0	
7	2	3	5	0	30	0	
8	3	4	3	0	10	0	
9	4	9	4	0	31	0	

ELEMENTS ADDED I2 = 3 J2 = 1
ELEMENTS DROPPED I1 = 4 J1 = 1 IH1 = 1

**** SHIPMENT MATRIX ****

*	50*+	0+	50+	0+	0+	0+	0+
*	40*+	0+	0+	10+	0+	30+	0+
*	60*+	30+	0+	10+	9+	0+	11+
*	31*+	0+	0+	0+	31+	0+	0+
*	181*+	30+	50+	20+	40+	30+	11+

**** COST MATRIX ****

*	-2*+	0+	1+	0+	0+	2+	0+
*	-1*+	0+	0+	2+	0+	3+	0+
*	1*+	3+	0+	4+	2+	0+	1+
*	0*+	0+	0+	0+	1+	0+	0+
*	350*+	2+	3+	3+	1+	4+	0+

TOTAL COST 350

**** ELEMENT MATRIX ****

*	1*+	0+	2+	0+	0+	5+	0+
*	2*+	0+	0+	3+	0+	7+	0+
*	3*+	1+	0+	8+	4+	0+	6+
*	4*+	0+	0+	0+	9+	0+	0+

ITERATION 3 *** NUMBER OF ROWS -M = 4 NUMBER OF COLS -N = 6

**** BASIS TABLE ****

**KT*	*I(KT)*	*K(KT)*	*J(KT)*	*LS(KT)*	*Y(KT)*	*F(KT)*	*A
1	3	6	1	0	30	5	
2	1	5	2	0	50	7	
3	2	7	3	0	20	3	
4	3	1	4	3	19	8	
5	1	2	5	2	0	4	
6	3	4	6	0	11	4	
7	2	3	5	0	20	9	
8	4	9	5	1	10	0	
9	4	8	4	0	21	0	

ELEMENTS ADDED I2 = 4 J2 = 5
ELEMENTS DROPPED I1 = 3 J1 = 3 IH1 = 8

**** SHIPMENT MATRIX ****

*	50*+	0+	50+	0+	0+	0+	0+
*	40*+	0+	0+	20+	0+	20+	0+
*	60*+	30+	0+	0+	19+	0+	11+
*	31*+	0+	0+	0+	21+	10+	0+
*	181*+	30+	50+	20+	40+	30+	11+

**** COST MATRIX ****

*	-2*+	0+	1+	0+	0+	2+	0+
*	-1*+	0+	0+	2+	0+	3+	0+
*	-1*+	3+	0+	0+	2+	0+	1+
*	-2*+	0+	0+	0+	1+	2+	0+
*	330*+	4+	3+	3+	3+	4+	2+

TOTAL COST 330

**** ELEMENT MATRIX ****

*	1*+	0+	2+	0+	0+	5+	0+
*	2*+	0+	0+	3+	0+	7+	0+
*	3*+	1+	0+	0+	4+	0+	6+
*	4*+	0+	0+	0+	9+	8+	0+

CHAPTER 9

INEQUALITIES IN THE CONSTRAINTS OF TRANSPORTATION PROBLEM

The discussion so far is based upon the equations 2.2 through 2.5, which assume that total quantity shipped is equal to total quantity required. However, it is possible to have inequalities for the constraints. If these inequalities could be replaced by equalities then the problem could be solved as outlined earlier. Two such cases are considered below.

1. Total availability is greater than the total demand.

Consider the following equations:

$$\sum_{j=1}^{n-1} X_{ij} \leqslant S_i, \qquad i = 1,\ldots, m \qquad (9.1)$$

$$\sum_{i=1}^{m} X_{ij} = D_j, \qquad j = 1,\ldots, n-1 \qquad (9.2)$$

$$X_{ij} \geqslant 0, \qquad \text{all } i, j, \qquad (9.3)$$

$$\min Z = \sum_{i,j} C_{ij} X_{ij} \tag{9.4}$$

The first m constraints now contain a $\leqslant$ sign rather than an equality sign. Physically, this simply means that more units may be available at the origins than are required at the destinations.

The inequalities can be converted to equalities by the addition of m slack variables. These slack variables may be written as X_{in}, for $i = 1,\ldots,m$.

Then the constraints become

$$\sum_{j=1}^{n-1} X_{ij} + X_{in} = S_i, \qquad i = 1,\ldots, m \tag{9.5}$$

$$\sum_{i=1}^{m} X_{ij} = D_j, \qquad j = 1,\ldots, n-1 \tag{9.6}$$

Sum (9.5) over i and subtract from the result the sum of (9.6) over j.

This gives

$$\sum_{i=1}^{m} X_{in} = \sum_{i=1}^{m} S_i - \sum_{j=1}^{n-1} D_j = D_n \tag{9.7}$$

Thus the total slack, i.e. the sum of the slack variables, remains constant and is the difference, denoted by D_n, between the origin availabilities and the destination requirements. To construct the table, simply add one more column, i.e., an additional destination for the slack. Intuitively, this approach is to be expected since the units not shipped can be considered to be shipped to origins at no cost. That is, the cost C_{in} associated with the slack variable X_{in} is zero.

2. Total demand is greater than total availability.

Consider the following problem:

$$\sum_{j=1}^{n} X_{ij} = S_i, \qquad i = 1,\ldots, m-1 \qquad (9.8)$$

$$\sum_{i=1}^{m-1} X_{ij} \geqslant D_j, \qquad j = 1,\ldots, n \qquad (9.9)$$

$$X_{ij} \geqslant 0, \qquad \text{all } i, j \qquad (9.10)$$

$$\max Z = \sum_{i,j} C_{ij} X_{ij}. \qquad (9.11)$$

Here, $\sum_j D_j > \sum_i S_i$

Introduce the surplus variables X_{mj}, for $j = 1,\ldots, n$. Now

$$-\sum_{j=1}^{n} X_{mj} = \sum_{j=1}^{n} D_j - \sum_{i=1}^{m-1} S_i = S_m \leqslant 0. \qquad (9.12)$$

The constraints of equation (9.8) through (9.10) can therefore be converted into the set of equations:

$$\sum_{j=1}^{n} X_{ij} = S_i, \qquad i = 1,\ldots, m-1 \qquad (9.13)$$

$$-\sum_{j=1}^{n} X_{mj} = S_m, \qquad (9.14)$$

$$\sum_{i=1}^{m-1} X_{ij} - X_{mj} = D_j, \qquad j = 1,\ldots, n \qquad (9.15)$$

The computational method is precisely the same as before except that use $\bar{C}_{mj} = -U_m - V_j$ for computing $\bar{C}_{mj} - C_{mj}$. This follows immediately from the dual. To solve this problem, add one more row to the table, i.e., an additional origin containing the negative of the total surplus. Here the additional cost elements $C_{m+1,j}$ are assumed to be zero.

Thus, provided all costs are positive in the optiaml solution to a problem of the form:

$$\sum_{j=1}^{m} X_{ij} \leqslant S_i, \qquad i = 1,\ldots, m \qquad (9.16)$$

$$\sum_{i=1}^{m} X_{ij} \geqslant D_j, \qquad j = 1, \ldots, n \qquad (9.17)$$

$$X_{ij} \geqslant 0, \qquad \text{all } i,j \qquad (9.18)$$

$$\text{Max or min } Z = \sum_{i,j} C_{ij} X_{ij}, \qquad (9.19)$$

strict equalities will hold

(a) in the destination constraints if Z is to be minimized and

(b) in the origin constraints if Z is to be maximized.

Physically, this means that if costs are minimized, no more will be shipped than necessary, and if Z is maximized, as much will be shipped as possible.

CHAPTER 10

OPERATING EXPERIENCE

The size of problem that can be solved by the method outlined in this paper is limited by the size of basis table and the amount of cost data which would fit into the fast access cores. However, the cost data may be stored on the magnetic drum or disk.

In many practical transportation situations, it is known at the start that many of the total of mn possible shipping routes are absurd. In fact, costs may not be known for many of these routes because it is certain that they would be inefficient. One would not ship to a customer in Vancouver from Montreal warehouse if there were a warehouse in Victoria, (B.C.). For problems involving a large number of cost elements, only the essential costs may be stored, all other costs not specified in the data being assumed infinite.

Virtually all applications of the transportation problem require the shipping of only whole units of the item being considered. Hence every basic feasible solution has integral values. Any optimal basic solution to transportation problem will have the property that all positive X_{ij} will be integers [12].

This integrality property is peculiar to the transportation problem. In general, an optimal solution to an arbitrary linear programming problem may not have integral values for the variables. In fact, for the variables to be integers, a linear programming problem usually becomes a nonlinear programming problem. Since an integral number of units is required at each destination, an integral number of units will be shipped.

Three different methods of selecting the element for the next iteration were tried. The results are tabulated in Table 10.1. The test data was stored in the main memory and execution times were obtained for each method for the same test data.

METHOD 1: BEST POSITION IN COST MATRIX.

According to usual practice with the stepping stone method, the entire cost matrix is examined at each iteration and the position is selected that gives the greatest incremental cost, $\Delta C = U_i + V_j - C_{ij}$. A new element is placed at this position in the shipment matrix for the next iteration.

When the logical search procedure is used on a high speed computer, the time required to carry out the operations of one iteration becomes short compared to the time required to search through the entire cost data in finding the new element to be put in the basis. Thus, the total time for solution might be cut down if the time spent searching the cost data per iteration were shortened at the expense of a larger number of iterations.

METHOD 2: FIRST WHICH WOULD REDUCE TOTAL COST.

The cost matrix is scanned row by row and the first position for which $U_i + V_j - C_{ij}$ is positive is selected. After each iteration, the search is resumed where it was broken off.

METHOD 3: BEST IN ROW OF COST MATRIX.

A complete row of the cost matrix is examined and the position in this row with the greatest incremental cost $\Delta C = U_i + V_j - C_{ij}$ is chosen. For the next iteration the next row is examined in the same way.

COMPARISON OF METHODS:

The method of searching one row at a time gives the best results. Naturally, with a different computer, a different compromise between number of iterations and time spent in searching the cost data will be optimum.

All the 3 methods were tried for a problem with 30 supplies, 260 demands and 7800 non-infinite costs on CDC 3500 computer. The results are compared with the usual rule in Table 10.1. Where,

$$\text{Ratio of timings} = \frac{\text{solution time of a method}}{\text{solution time of method 3 for 30 x 260 problem}}$$

Method of Selecting New Element	Number of Iterations	No. of Examinations of Cost Matrix	Ratio of Timings*
Best position in cost matrix	509	509	2
First which would reduce total cost	2200	29	2.6
Best in row of cost matrix	672	43	1

Table 10 .1: Comparison of Methods for a 30 by 260 Problem

* Obtained by dividing solution timings by the solution time of third method.

Solution Time Ratio for	Problem Size m + n								
	10	50	100	150	190	250	290	350	390
Method 1 / Method 3	2.033	2.026	2.016	2.008	2.006	2.002	2.000	2.001	2.001
Method 2 / Method 3	2.624	2.611	2.609	2.606	2.603	2.601	2.600	2.602	2.601
Method 3 / Method 3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

TABLE 10.2: Comparison of Solution Time Ratios

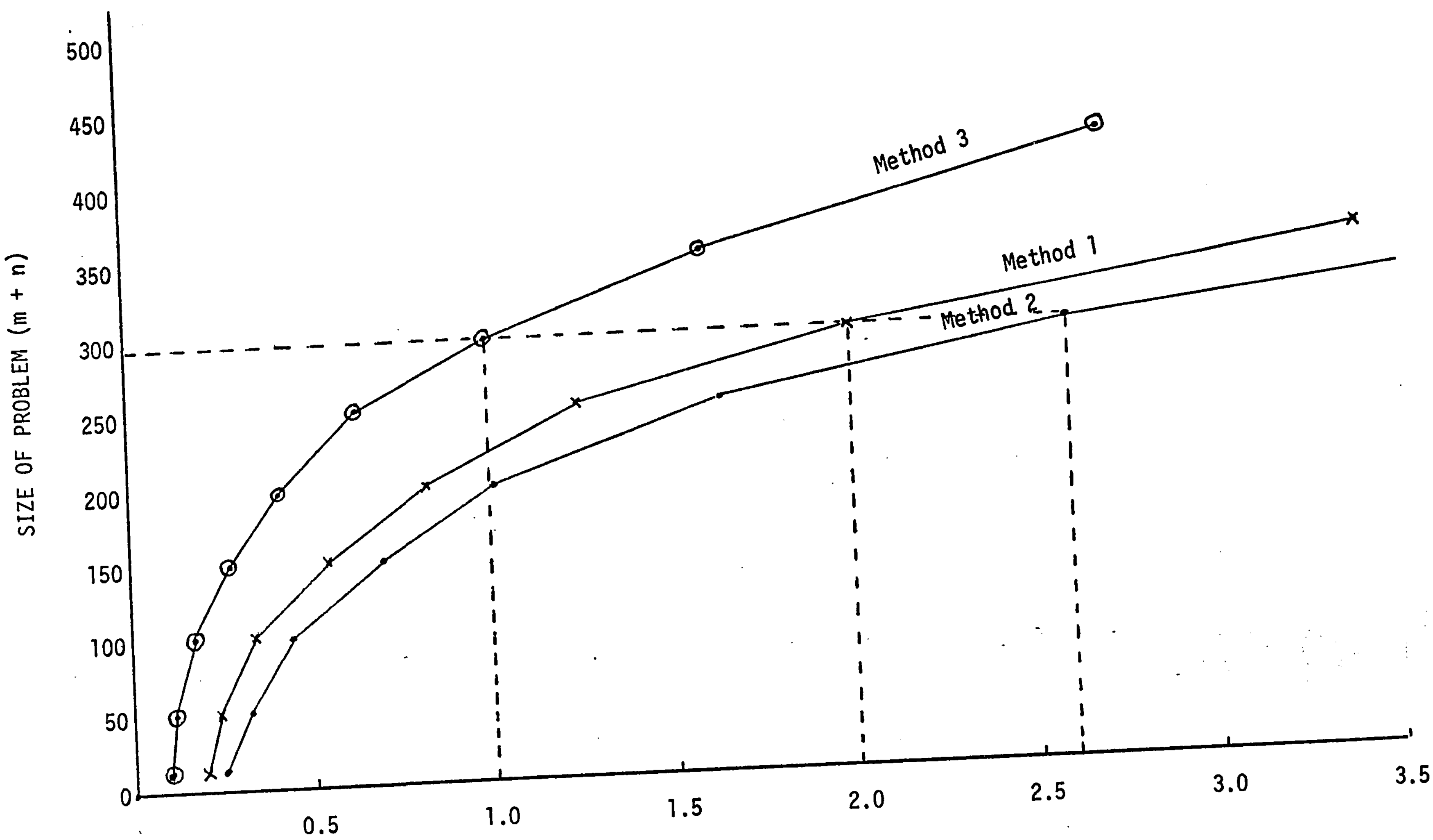


Fig. 10.3: Solution Time Ratio as a Funtion of Problem Size

COMPARISON OF SOLUTION TIME RATIOS.

A second set of tests were performed using a variable set of data by all the above three methods. The results are shown in Table 10.2 and Fig. 10.3. It should be noted that these results vary only in the second decimal place.

CONCLUSION

The stepping stone method described in this report was succesfully implemented and tested. The results show consistently that the mehtod presented in this report considerably reduces the solution time.

BIBLIOGRAPHY

1. Bellman, R., "On the Theory of Dynamic Programming - A Warehousing Problems", Management Science 2, 3, 1956.

2. Charnes, A. and W.W. Cooper, "The Stepping Stone Method of Explaning Linear Programming Calculations in Transportation Problems", Management Science, 1, 1, 1954.

3. Charnes, A. and W.W. Cooper, "Management Models and Industrial Applications of Linear Programming", Management Science, 4, 1, 1957.

4. Charnes, A., W.W. Cooper and A. Henderson, "An Introduction to Linear Programming", John Wiley, New York, 1953.

5. Dantzig, G.B., "Linear Programming", Princeton University, Press, N.J., 1963.

6. Dennis, J.B., "A High Speed Computer Technique for the Transportation Problem", Journal Association of Computer Machinery, April 1958.

7. Dickson, J.C. and F.P. Frederick, "A Decision Rule for Improved Efficiency in Solving Linear Programming Problems with the Simplex Algorithm", Communications of the ACM, Vol. 3, No. 9, 1960.

8. Eisemann, "Simplified Treatment of Degeneracy in Transportation Problems", Quarterly of Applied Mathematics, XIV, 4, 1957.

9. Ferguson, A.R. and G.B. Dantzig, "The Allocation of Aircraft to Routes - An Example of Linear Programming under Uncertain Demand", Management Science, 3, 1, 1956.

10. Flood, M.M., "Application to Transportation Theory to Scheduling a Military Tanker Fleet", Journal of Operations Research Society, Vol. 2, No. 2, May 1954.

11. Ford, L.R. and D.R. Fulkerson, "Solving the Transportation Problem", RAND Report RM - 1736., The RAND Corporation, Santa Monica, Calif., 1956.

12. Gauss, S.I., "Linear Programming", McGraw Hill Co., New York, 1969.

13. Gauss, S.I., "An Illustrated Guide to Linear Programming", McGraw Hill Book Co., New York, 1969.

14. Gauss, S.I., "Advances in Computers", Vol. 2, Academic Press, New York, 1961.

15. Goode, H.W., "The Application of a Highspeed Computer to the Definition and Solution of the Vehicular Traffic Problem", Journal of the Operations Research Society, Vol. 5, No. 6, Dec. 1957.

16. Hadley, G., Linear Programming, Addison-Wesley Publishing Co., 1963.

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

18. Hitchcock, F.L., "The Distribution of a Product from Several Sources to Numerous Lucalities", Journal of Mathematics and Physics, Vol. 20, 1941.

19. Koopmans, T.C., "Activity Analysis of Production and Allocation", Cowles Commission Monograph 13, John Wiley & Sons Inc., New York, 1951.

20. Koopmans, T.C., "Optimum Utilization of the Transportation System", Econometrica, Vol. XVII, 1949.

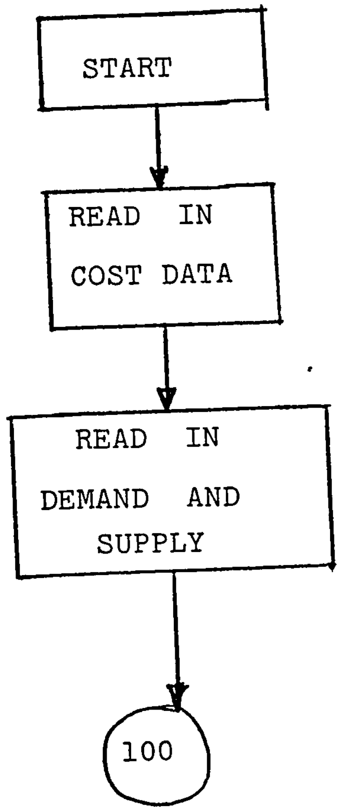
21. Kuhn, H.W. and A.W. Tucker, "Linear Inequalities and Related Systems", Princeton University Press, Princeton, N.J., 1956.

22. Orden, A., "The Transhipment Problem", Management Science 2, 3, 1956.

23. Reinfeld, N.V. and W.R. Vogel, "Mathematical Programming", Englewood Cliffs, Prentice Hall, 1958.

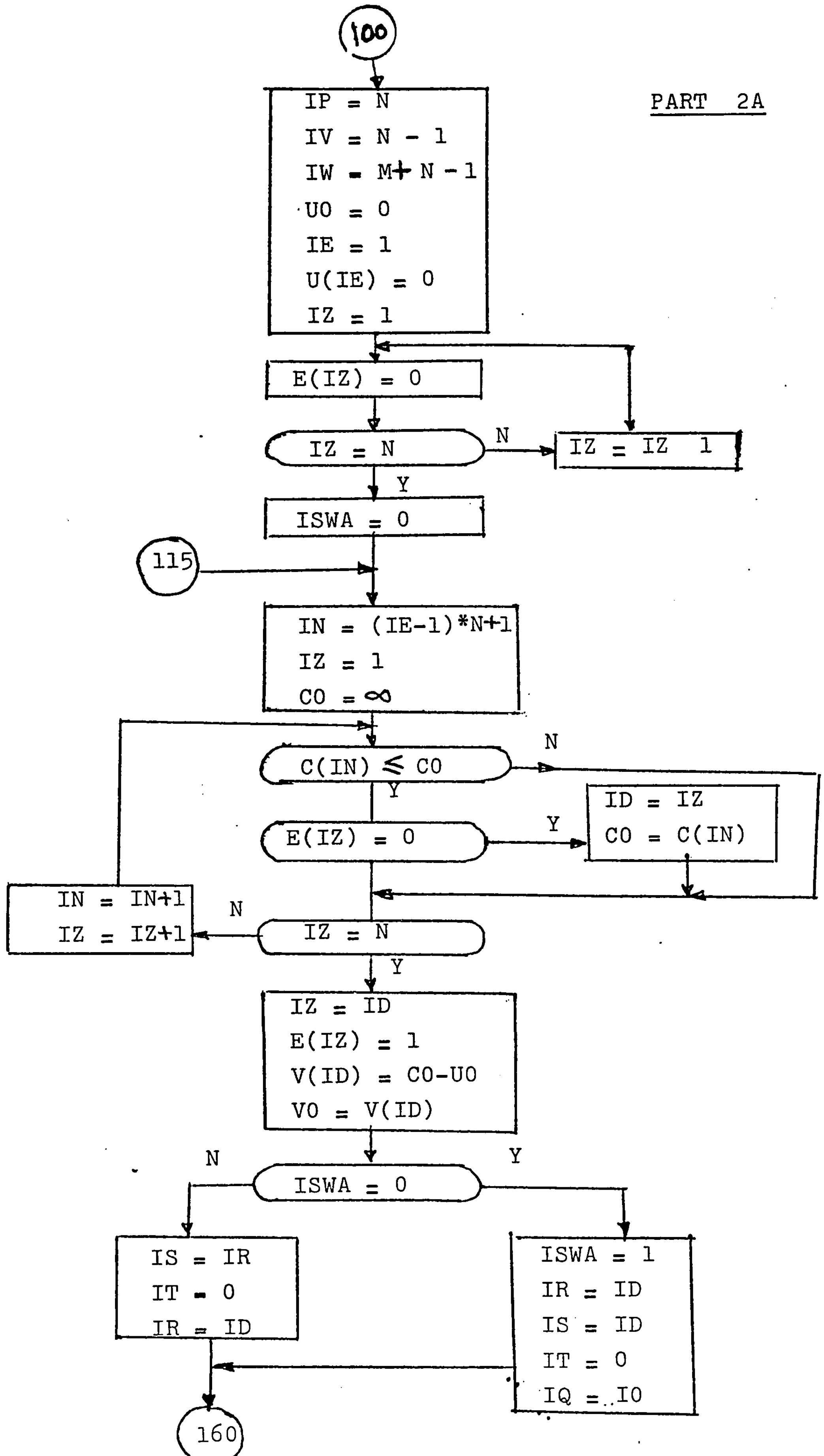
APPENDIX A

FLOW CHARTS



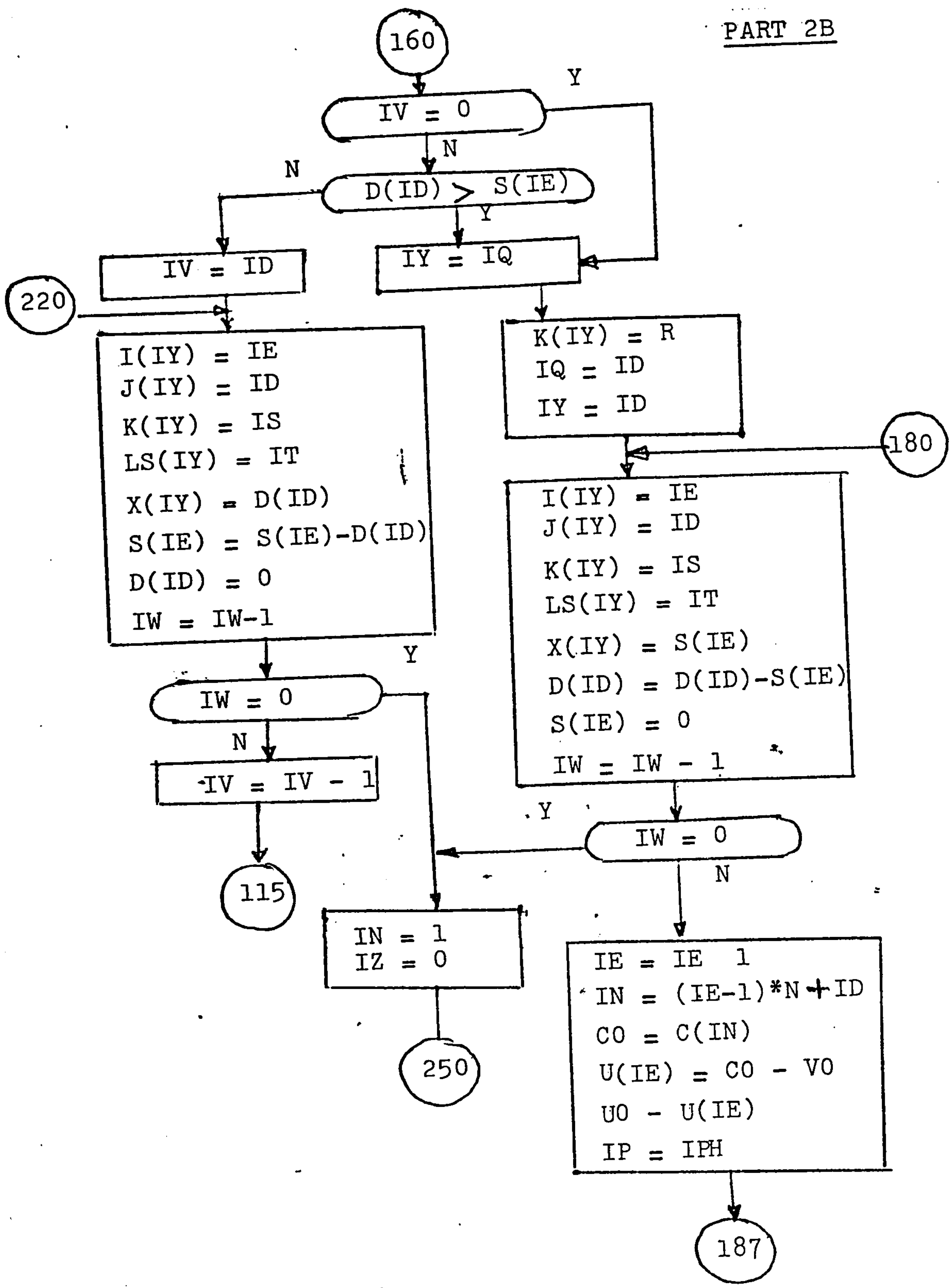
Note: The input/output routines are user dependent and hence the above flow chart is block diagram. However, it is coded for the computer program to enable testing.

PART 2A



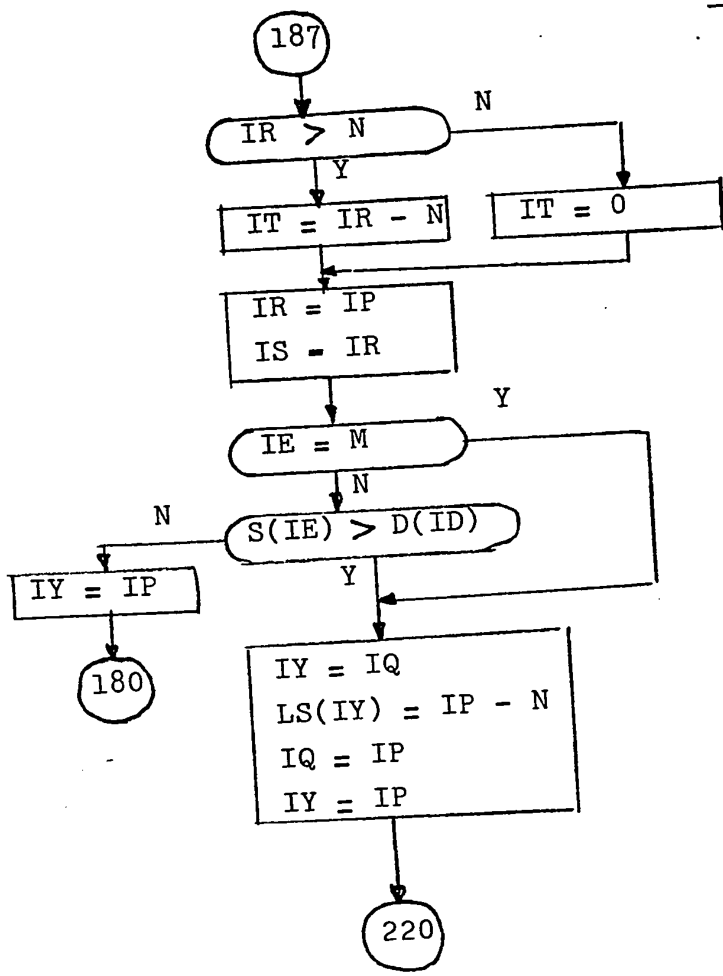
100
IP = N
IV = N - 1
IW = M+N-1
U0 = 0
IE = 1
U(IE) = 0
IZ = 1
E(IZ) = 0
IZ = N
N
IZ = IZ 1
Y
ISWA = 0
115
IN = (IE-1)*N+1
IZ = 1
CO = ∞
C(IN) ≤ CO
N
Y
E(IZ) = 0
Y
ID = IZ
CO = C(IN)
IN = IN+1
IZ = IZ+1
N
IZ = N
Y
IZ = ID
E(IZ) = 1
V(ID) = CO-U0
V0 = V(ID)
N
ISWA = 0
Y
IS = IR
IT = 0
IR = ID
ISWA = 1
IR = ID
IS = ID
IT = 0
IQ = IO
160

PART 2B



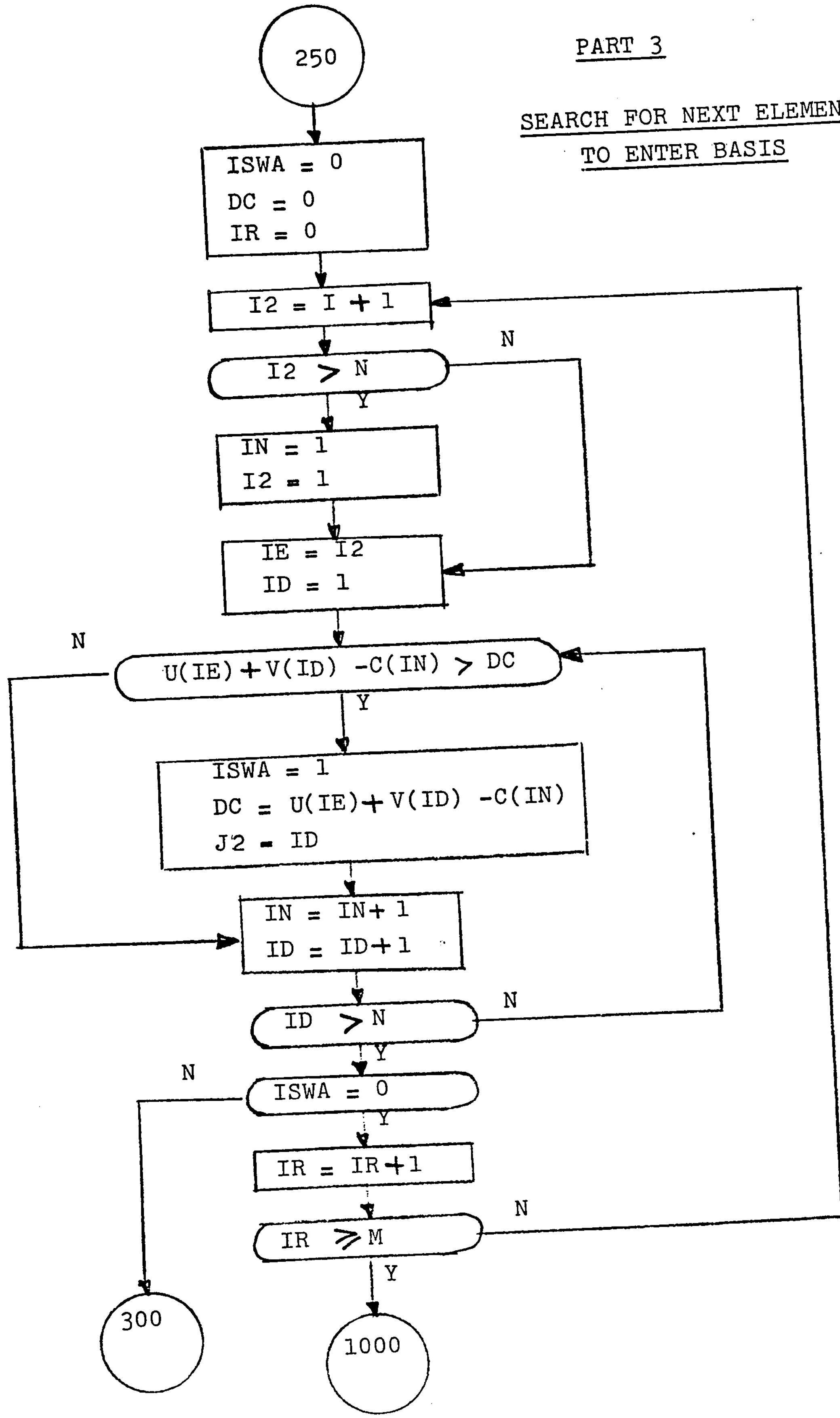
160
IV = 0
Y
N
D(ID) > S(IE)
N
Y
IV = ID
IY = IQ
220
I(IY) = IE
J(IY) = ID
K(IY) = IS
LS(IY) = IT
X(IY) = D(ID)
S(IE) = S(IE)-D(ID)
D(ID) = 0
IW = IW-1
K(IY) = R
IQ = ID
IY = ID
180
I(IY) = IE
J(IY) = ID
K(IY) = IS
LS(IY) = IT
X(IY) = S(IE)
D(ID) = D(ID)-S(IE)
S(IE) = 0
IW = IW - 1
IW = 0
Y
N
IV = IV - 1
115
IN = 1
IZ = 0
250
IW = 0
Y
N
IE = IE 1
IN = (IE-1)*N+ID
CO = C(IN)
U(IE) = CO - VO
UO - U(IE)
IP = IPH
187

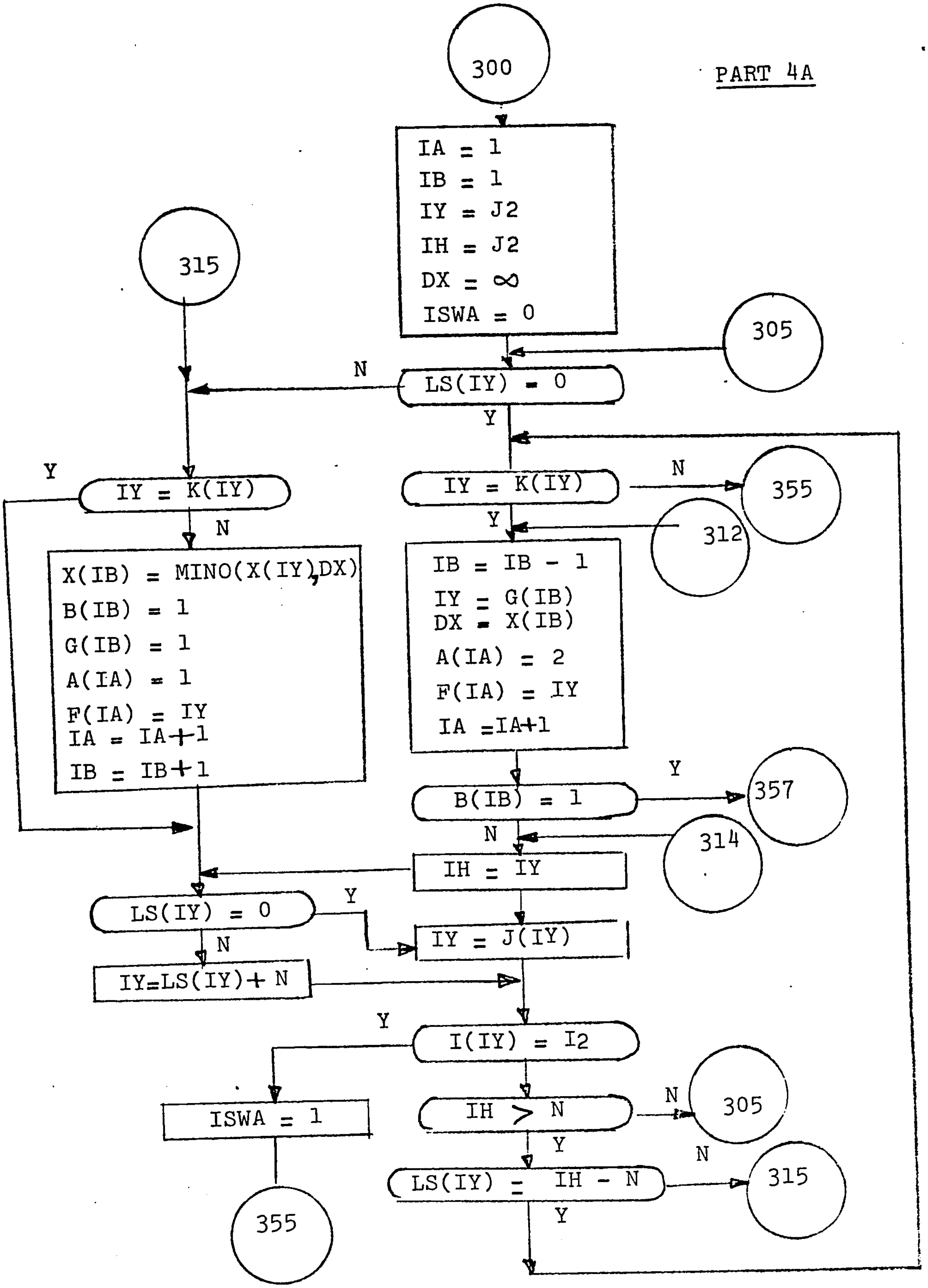
PART 2C


187
IR > N
N
Y
IT = IR - N
IT = 0
IR = IP
IS = IR
Y
IE = M
N
N
S(IE) > D(ID)
Y
IY = IP
180
IY = IQ
LS(IY) = IP - N
IQ = IP
IY = IP
220

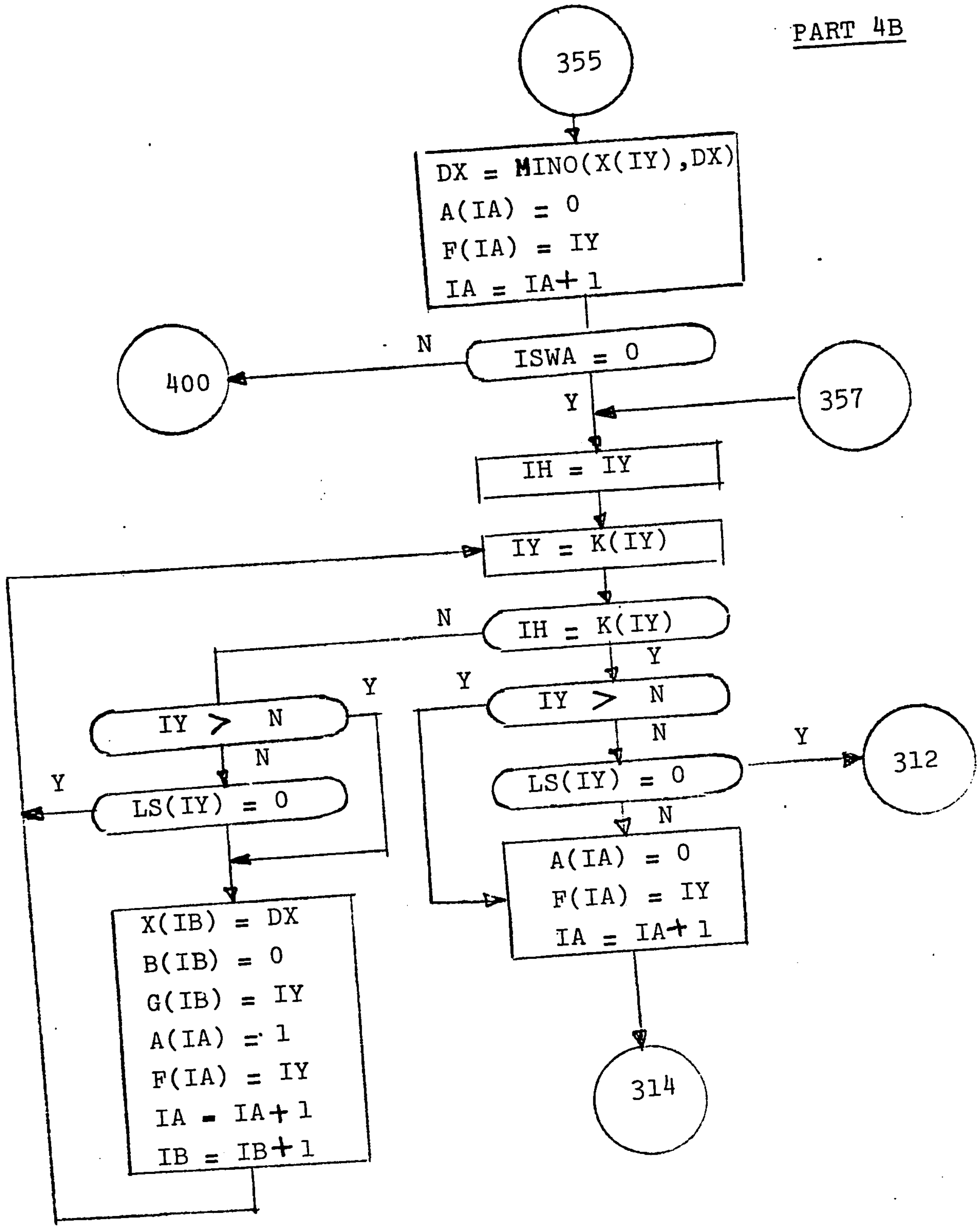
PART 3

SEARCH FOR NEXT ELEMENT TO ENTER BASIS




PART 4A
300
IA = 1
IB = 1
IY = J2
IH = J2
DX = ∞
ISWA = 0
305
LS(IY) = 0
N
Y
315
IY = K(IY)
Y
N
X(IB) = MINO(X(IY),DX)
B(IB) = 1
G(IB) = 1
A(IA) = 1
F(IA) = IY
IA = IA+1
IB = IB+1
IY = K(IY)
N
355
Y
312
IB = IB - 1
IY = G(IB)
DX = X(IB)
A(IA) = 2
F(IA) = IY
IA =IA+1
B(IB) = 1
Y
357
N
314
IH = IY
LS(IY) = 0
Y
N
IY = J(IY)
IY=LS(IY)+ N
I(IY) = I2
Y
ISWA = 1
355
IH > N
N
305
Y
LS(IY) = IH - N
N
315
Y

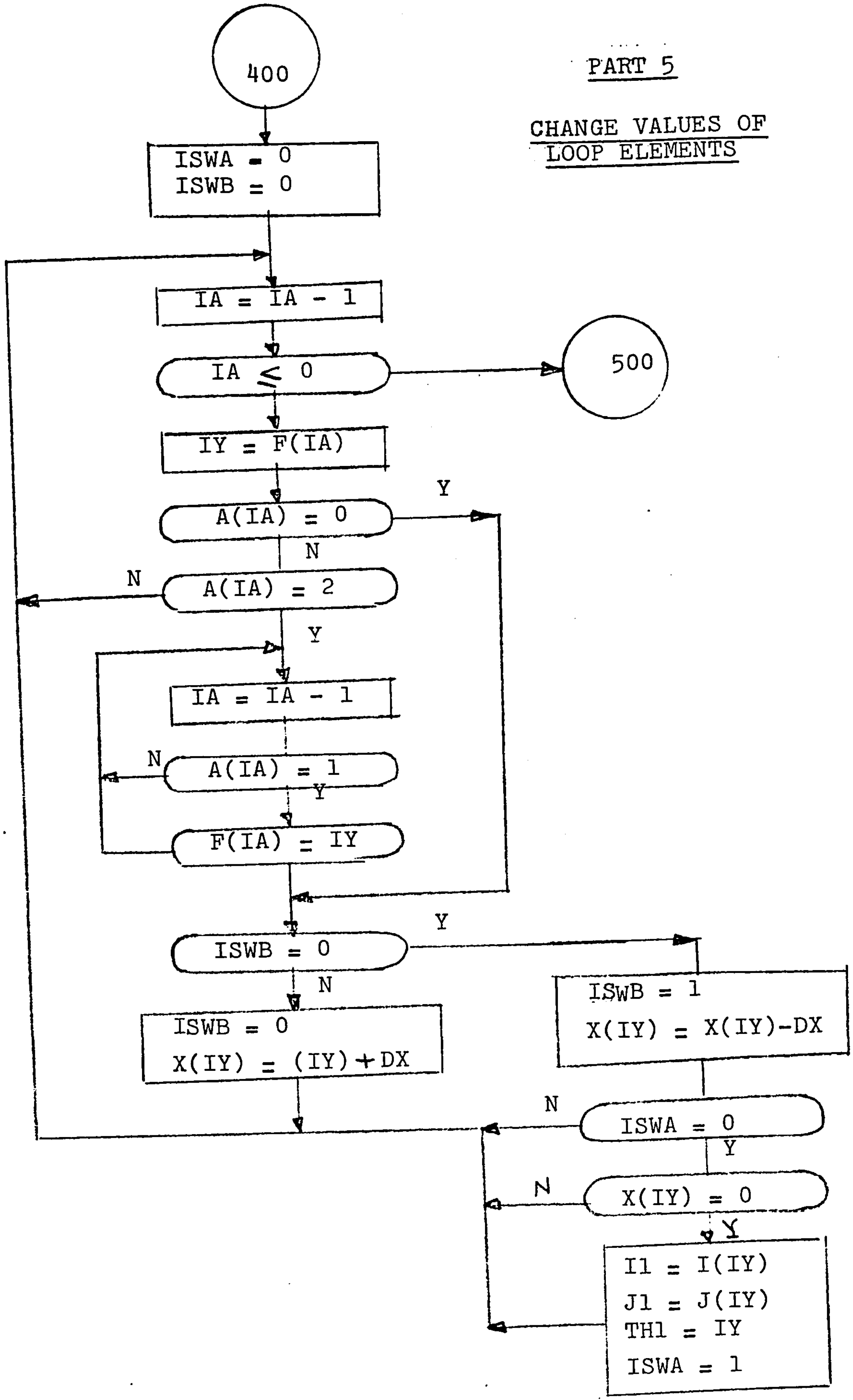
PART 4B



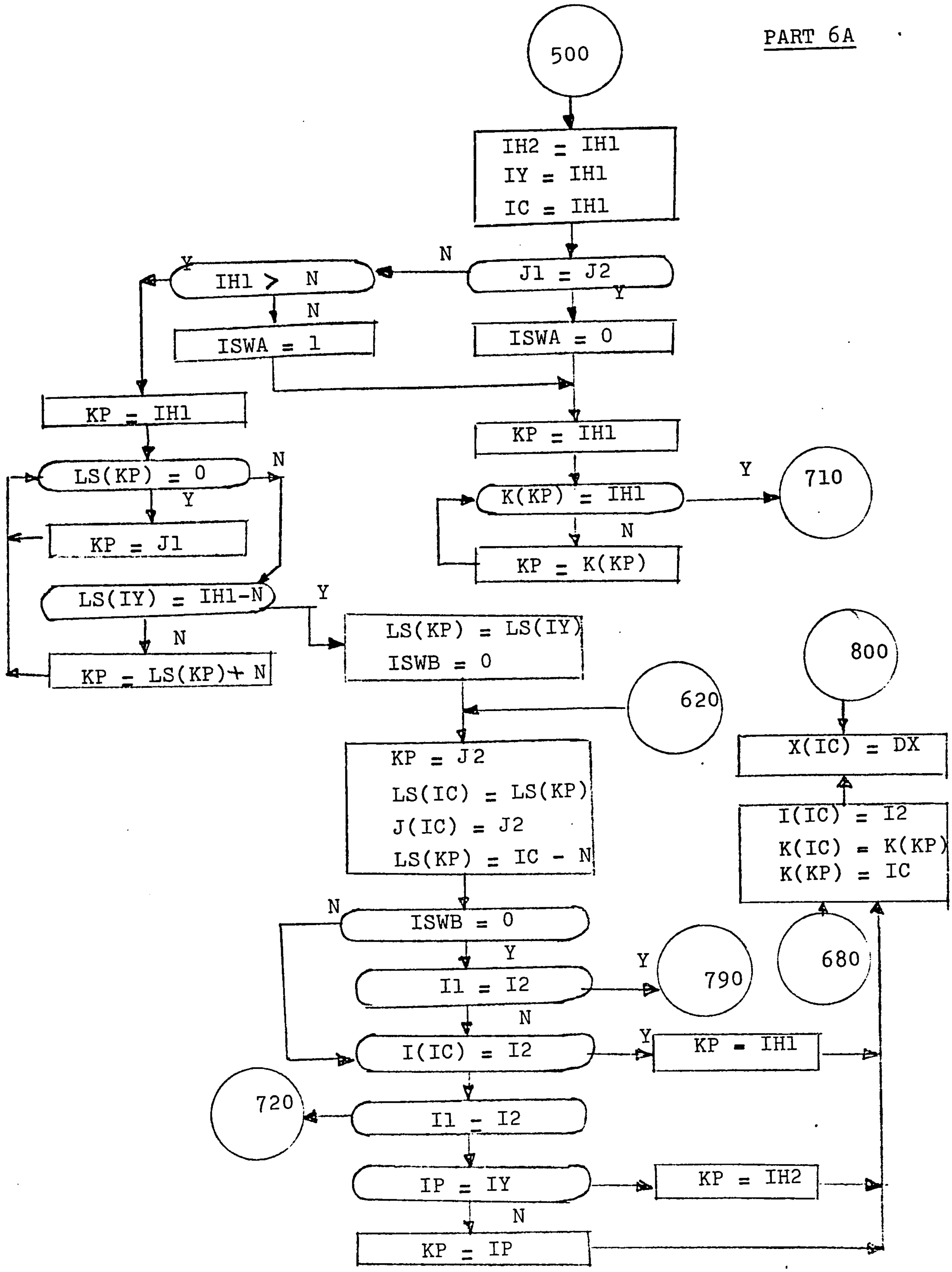
355
DX = MINO(X(IY),DX)
A(IA) = 0
F(IA) = IY
IA = IA+1
N
ISWA = 0
400
Y
357
IH = IY
IY = K(IY)
N
IH = K(IY)
Y
Y
IY > N
Y
IY > N
N
N
Y
312
Y
LS(IY) = 0
LS(IY) = 0
N
A(IA) = 0
F(IA) = IY
IA = IA+1
X(IB) = DX
B(IB) = 0
G(IB) = IY
A(IA) = 1
F(IA) = IY
IA = IA+1
IB = IB+1
314

PART 5

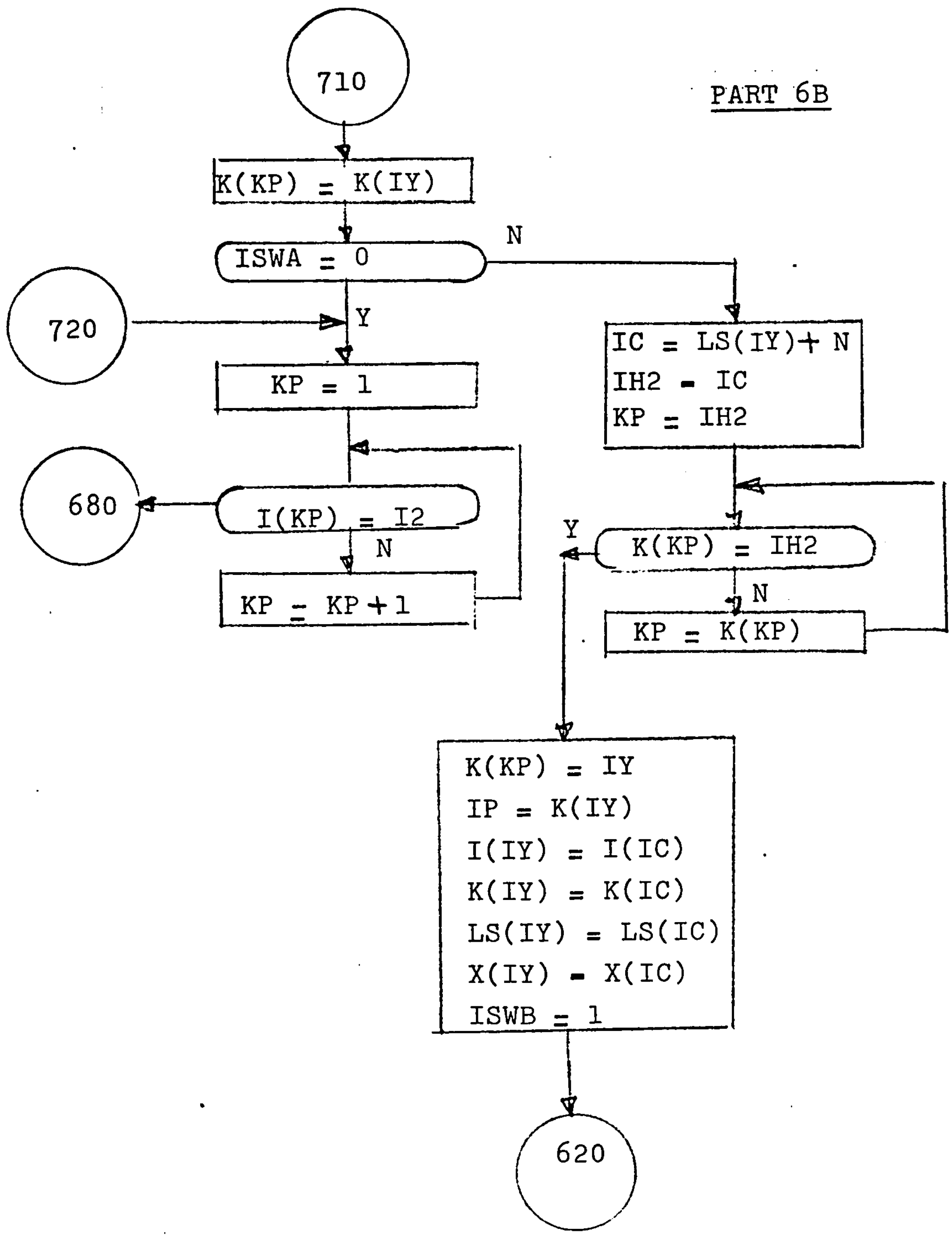
CHANGE VALUES OF LOOP ELEMENTS



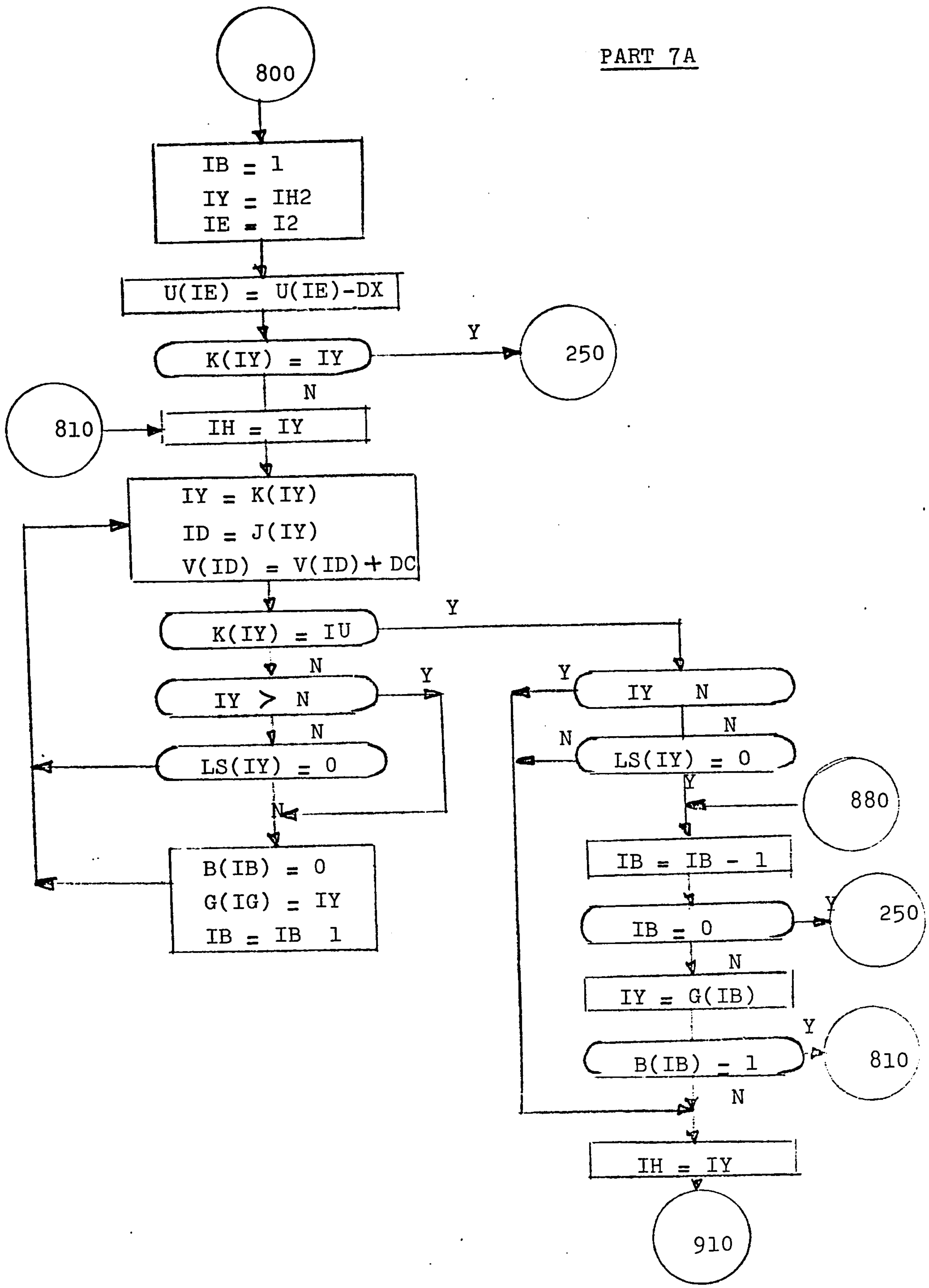
PART 6A



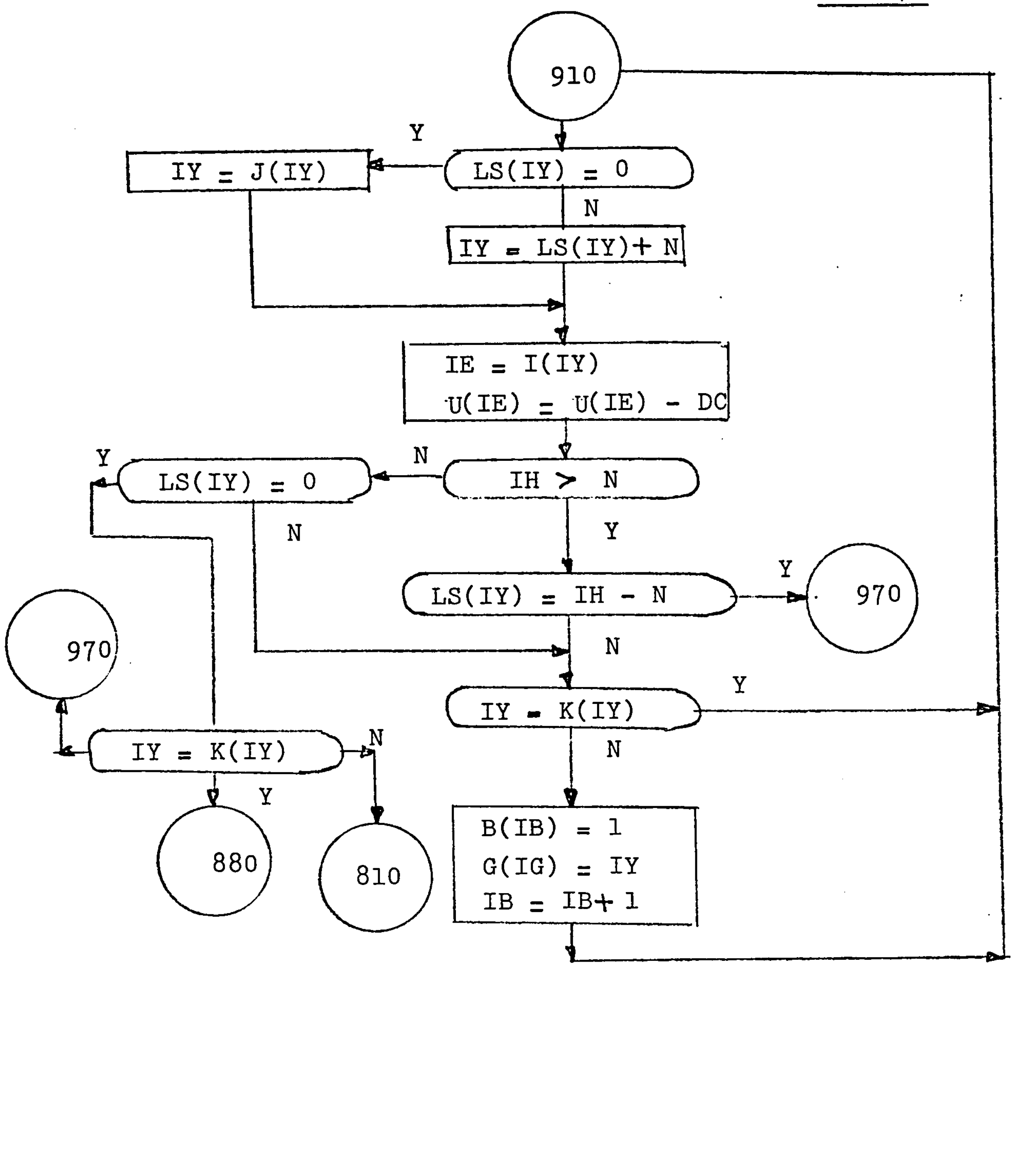
500
IH2 = IH1
IY = IH1
IC = IH1
J1 = J2
N
Y
IH1 > N
Y
N
ISWA = 1
ISWA = 0
KP = IH1
KP = IH1
LS(KP) = 0
N
Y
KP = J1
K(KP) = IH1
Y
710
N
KP = K(KP)
LS(IY) = IH1-N
Y
N
KP = LS(KP)+ N
LS(KP) = LS(IY)
ISWB = 0
620
800
KP = J2
LS(IC) = LS(KP)
J(IC) = J2
LS(KP) = IC - N
X(IC) = DX
I(IC) = I2
K(IC) = K(KP)
K(KP) = IC
ISWB = 0
N
Y
I1 = I2
Y
790
680
N
I(IC) = I2
Y
KP = IH1
720
I1 = I2
IP = IY
KP = IH2
N
KP = IP

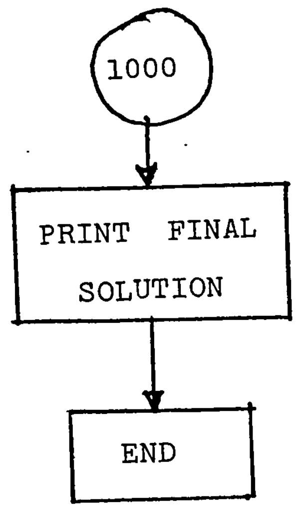

PART 6B
710
K(KP) = K(IY)
ISWA = 0
N
Y
720
KP = 1
IC = LS(IY)+ N
IH2 = IC
KP = IH2
680
I(KP) = I2
N
Y
K(KP) = IH2
N
KP = KP + 1
KP = K(KP)
K(KP) = IY
IP = K(IY)
I(IY) = I(IC)
K(IY) = K(IC)
LS(IY) = LS(IC)
X(IY) = X(IC)
ISWB = 1
620

PART 7A


800
IB = 1
IY = IH2
IE = I2
U(IE) = U(IE)-DX
K(IY) = IY
Y
250
N
810
IH = IY
IY = K(IY)
ID = J(IY)
V(ID) = V(ID)+DC
K(IY) = IU
Y
N
IY > N
Y
N
LS(IY) = 0
N
B(IB) = 0
G(IG) = IY
IB = IB 1
IY N
Y
N
LS(IY) = 0
N
Y
880
IB = IB - 1
IB = 0
Y
250
N
IY = G(IB)
B(IB) = 1
Y
810
N
IH = IY
910

PART 7B


910
Y
IY = J(IY)
LS(IY) = 0
N
IY = LS(IY)+ N
IE = I(IY)
U(IE) = U(IE) - DC
Y
LS(IY) = 0
N
IH > N
N
Y
Y
LS(IY) = IH - N
970
970
N
Y
IY = K(IY)
IY = K(IY)
N
N
Y
880
810
B(IB) = 1
G(IG) = IY
IB = IB+ 1



Note: The input/output routines are user dependent and hence the above flow chart is block diagram. However, it is coded for the computer program to enable testing.

APPENDIX B

PROGRAM LISTING

```
      SUBROUTINE GXPOSE (A,B,C,D,E,F,G,I,J,K,LS,M,N,S,U,V,X)
C
C ---------------------------------------------------------------------
C  *** R  GUPTA ***   TRANSPORTATION PROBLEM ***
C ---------------------------------------------------------------------
C
C   AUTHOR   R. GUPTA
C
C   DATE WRITTEN    MARCH 10,1970
C
C----------------------------------------------------------------------
C
C
C PROBLEM.
C
C   MINIMIZE COST C = SUN OVER I,J OF (C(I,J)*X(I,J))
C      SUBJECT TO THE CONSTRANTS
C      1.  SUM OVER J OF X(I,J) = S(I)
C
C      2.  SUM OVER I OF X(I,J) = D(J)
C      3.  X(I,J)  .GE. 0
C
C   NOTE: DATA MUST HAVE SUM OVER I OF S(I)= SUM OVER J OF D(J)
C
C----------------------------------------------------------------------
C
C
C PARAMETERS.
C
C
C     M      - NUMBER OF SOURCE (PLANTS), S(I)                        ROW
C     N      - NUMBER OF DISTRIBUTION POINTS (CUSTOMERS), D(J)        COL.
C     D(J)   - DEMAND OF COMMODITY BY CUSTOMER J               ** MAX N
C     S(I)   - SUPPLY OF COMMODITY AT PLANT    I               ** MAX M
C     I      - VARIABLE FOR M (VALUE BETWEEN 1 AND M)          **
C     J      - VARIABLE FOR N (VALUE BETWEEN 1 AND N)
C     DC     - INCREMENTAL COST = U(I)+V(J)-C(I,J)
C     U(IE)  - DUAL VARIABLE FOR ROW IE OF COST MATRIX         ** MAX M
C     V(ID)  - DUAL VARIABLE FOR COL ID OF COST MATRIX         ** MAX N
C     K(IY)  - NUMBER OF AN ELEMENT IN THE SAME ROW OF     ** MAX M+N-1
C                SHIPMENT MATRIX AS ELEMENT IY
C     L(IY)  - NUMBER OF AN ELEMENT IN THE SAME COL OF     ** MAX M+N-1
C                SHIPMENT MATRIX AS ELEMENT IY
C     I(IY)  - ROW OF BASIS ELEMENT                        ** MAX M+N-1
C     J(IY)  - COL OF BASIS ELEMENT                        ** MAX M+N-1
C     LS(IY)= 0 IF L(IY) .LE. N                            ** MAX M+N-1
C            = L(IY)-N OTHERWISE
C     F(IA)  - ELEMENT NUMBER OF BRANCH OR CORNER POINT     ** MAX 2*M+N-2
C                ENCOUNTERED DURING SEARCH
C     A(IA)  - IDENTIFIES F(IA) ENCOUNTERED DURIN SEARCH AS** MAX 2*M+N-2
C                A(IA)=0 - CORNER POINT
C                     =1 - BRANCH POINT
C                     =2 - BRANCH POINT FROM WHICH A NEW
C                          PATH WAS SEARCHED
C     G(IB)  - ELEMENT NUMBER OF BRANCH POINT ENCOUNTERED DURING**MAX M-1
```

```
C                     SEARCH
C      B(IB) - INDICATES WHETHER BRANCH POINT WAS FOUND DURING   **MAX M-1
C                ROW SEARCH - B(IB)=0    OR
C                COL SEARCH - B(IB)=1.
C      X(IB) - VALUE OF SMALLEST ALTERNATE CORNER OR BRANCH       **MAX M-1
C                ELEMENT OF THE PATH FROM THE STARTING POINT
C                OF A TREE TO AND INCLUDING ELEMENT G(IB).
C      C(IN) - UNIT COST FROM PLANT I TO CUSTOMER J               **MAX M*N
C      IN    - INDEX FOR COST,C(IN) = J+N*(I-1)
C
C      E(IZ) - DENOTES WHETHER AN ENTRY IN THE BASIS TABLE HAS    **MAX N
C                BEEN MADE (E(IZ)=1) OR
C                HAS NOT BEEN MADE (E(IZ)=0)
C                FOR COL IZ OF SHIPPING MATRIX
C                (USED ONLY IN FORMING INITIAL BASIS).
C      C0    - COST FOR CURRENT BASIS ELEMENT
C      U0    - VALUE OF U FOR ROW OF CURRENT ELEMENT
C      V0    - VALUE OF U FOR COL OF CURRENT ELEMENT
C      IH1   - BASIS TABLE ENTRY  NUMBER OF ELEMENT DROPPED FROM
C                BASIS IN A PARTICULAR ITERATION.
C      I1    - ROW          ENTRY  NUMBER OF ELEMENT DROPPED FROM
C                BASIS IN A PARTICULAR ITERATION
C      J1    - COL          ENTRY NUMBER OF ELEMENT DROPPED FROM
C                BASIS IN A PARTICULAR ITERATION.
C      IP    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IQ    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IR    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IS    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IT    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IU    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IV    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IW    - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C      IH2   - BASIS TABLE ENTRY NUMBER OF ELEMENT ADDED TO BASIS
C                IN THIS ITERATION.
C      I2    - ROW          ENTRY NUMBER OF ELEMENT ADDED TO BASIS
C                IN THIS ITERATION
C      J2    - COL          ENTRY NUMBER OF ELEMENT ADDED TO BASIS
C                IN THIS ITERATION
C      IH    - BASIS TABLE ENTRY NUMBER OF FIRST ELEMENT EXAMINED
C                IN A PARTICULAR ROW OR COL OF THE SHIPMENT MATRIX
C                DURING LOGICAL SEARCH.
C      IPALL = 3   PRINT  EVERYTHING
C            = 4   PRINT  FINAL ITERATION ONLY
C      KAUNT - NUMBER OF ITERATION
C
C ---------------------------------------------------------------------
C
C
C
       COMMON    IPALL,KOST,KSUMS
C                                         *****  SIZE   M*N    *****
       INTEGER   C(1)
C
C                                         *****  SIZE    M     *****
       INTEGER S(1),U(1)
```

```
      C
      C                                               *****  SIZE     N      *****
            INTEGER D(1),V(1),E(1)
      C
      C                                               *****  SIZE  M+N-1     *****
            INTEGER I(1),J(1),K(1),LS(1),X(1)
      C
      C                                               *****  SIZE    M-1     *****
            INTEGER G(1),B(1)
      C
      C                                               *****  SIZE 2*M+N-2    *****
            INTEGER A(1),F(1)
      C
            INTEGER   DC,DX,C0,U0,V0
      C
      C
            KR    = 60
            KW    = 61
            KSUMS = 0
            KSUMD = 0
      C
      C ===========================================================================
      C
      C * PART 1 *  INPUT SECTION  -  READ IN THE DATA *
      C
      C ===========================================================================
      C
      C
            DO 20 IK=1,M
            F(IK)=0
            A(IK)=0
            B(IK)=0
            G(IK)=0
            KSUMS = KSUMS + S(IK)
         20 CONTINUE
      C
      C
      C ===========================================================================
      C
      C * PART 2 * FORM INITIAL BASIC FEASIBLE SOLUTION **
      C
      C ===========================================================================
      C
            IP    = N
            IV    = N-1
            MN1   = M+N-1
            IW    = MN1
            MN2   = M*2+N-2
            M1    = M-1
            U0    = 0
            INF   = 2**20
            IE    = 1
            KAUNT = 0
      C
            U(IE)= 0
```

```
      C
            DO 100 IZ=1,N
            KSUMD = KSUMD + D(IZ)
        100 E(IZ)= 0
      C
            IF (KSUMS-KSUMD) 103,105,103
        103 WRITE (KW,104) KSUMS,KSUMD
        104 FORMAT (1H ,10HSUM OF S =,I6,3X,10HSUM OF D =,I6/)
            GO TO 2000
      C
        105 CONTINUE
            ISWA = 0
      C
        115 IN    = (IE-1)*N+1
            IZ    = 1
            C0   = INF
        120 IF(C(IN)-C0) 125,125,135
        125 IF(E(IZ))135,130,135
        130 ID    = IZ
            C0    = C(IN)
        135 IF(IZ-N) 140,145,140
        140 IN    = IN+1
            IZ    = IZ+1
            GO TO 120
      C
        145 IZ    = ID
            E(IZ)= 1
            V(ID)= C0-U0
            V0    = V(ID)
      C
            IF(ISWA) 155,150,155
        150 ISWA = 1
            IR    = ID
            IS    = ID
            IT    = 0
            IQ    = ID
            GO TO 160
      C
        155 IS    = IR
            IT    = 0
            IR = ID
      C
        160 IF(IV) 165,175,165
        165 IF(D(ID)-S(IE)) 170,170,175
        170 IY    = ID
            GO TO 220
      C
        175 IY    = IQ
            K(IY)= IR
      C
            IQ    = ID
            IY    = ID
      C
        180 I(IY)= IE
            K(IY)= IS
```

```
      J(IY)= ID
      LS(IY)=IT
      X(IY)= S(IE)
      D(ID)=D(ID)-S(IE)
      S(IE)= 0
      IW   = IW-1
C
C
      IF(IW) 185,225,185
  185 IE   = IE+1
      IN   = (IE-1)*N+ID
      C0   = C(IN)
      U(IE)= C0-V0
      U0   = U(IE)
      IP   = IP+1
      IF(IR-N) 195,195,190
  190 IT   = IR-N
      GO TO 200
C
  195 IT   = 0
  200 IR   = IP
      IS=IR
      IF(IE-M) 205,215,205
  205 IF(S(IE)-D(ID)) 210,210,215
  210 IY   = IP
      GO TO 180
C
  215 IY   = IQ
      LS(IY)= IP-N
C
      IQ   = IP
      IY   = IP
C
  220 I(IY) = IE
      J(IY) = ID
      K(IY) = IS
      LS(IY)= IT
      X(IY) = D(ID)
      S(IE) = S(IE)-D(ID)
      D(ID) = 0
      IW    = IW-1
C
      IF(IW) 230,225,230
  225 IN    = 1
      I2    = 0
      GO TO 250
C
  230 IV    = IV-1
      GO TO 115
C
  250 CONTINUE
C
      KAUNT = KAUNT+1
      IGO = 1
      IF (IPALL-3) 8253,1002,8253
```

```
 8253 CONTINUE
C
C ======================================================================
C
C * PART 3 * SEARCH FOR NEXT ELEMENT TO ENTER BASIS
C
C ======================================================================
C
      ISWA = 0
      DC   = 0
      IR   = 0
  255 I2   = I2+1
      IF(I2-M) 265,265,260
  260 IN   = 1
      I2   = 1
  265 IE   = I2
      ID   = 1
  270 IDC = U(IE)+V(ID)-C(IN)
      IF( IDC-DC ) 280,280,275
  275 ISWA = 1
      DC = IDC
      J2   = ID
  280 CONTINUE
C
      IN   = IN+1
      ID   = ID+1
      IF(ID-N) 270,270,285
  285 IF(ISWA) 300,295,300
  295 IR   = IR+1
      IF(IR-M) 255,1000,1000
C
  300 CONTINUE
      DO 301 IJ=1,MN1
C
  301 E(IJ) = X(IJ)
C
C ======================================================================
C
C * PART 4 * FIND BASIC LOOP INCLUDING NEW ELEMENTS *
C
C ======================================================================
C
      IA   = 1
      IB   = 1
      IY   = J2
      IH   = J2
      DX   = INF
      ISWA = 0
C
  305 IF(LS(IY)) 315,310,315
  310 IF(IY-K(IY)) 355,312,355
  311 CONTINUE
C
      WRITE (KW,8311)
 8311 FORMAT (1H ,15H* PART 4 * IB=0 /)
```

```
      GO TO 2000
C
  312 IB    = IB-1
C
      IF (IB) 311,311,313
  313 IY    = G(IB)
C
      DX    = E(IB)
      A(IA)= 2
      F(IA)= IY
      IA    = IA+1
      IF(B(IB)-1) 314,357,314
  314 IH    = IY
C
      GO TO 325
C
  315 CONTINUE
      IF(IY-K(IY)) 320,325,320
C
  320 CONTINUE
C
      E(IB)=MIN0(X(IY),DX)
      B(IB)= 1
      G(IB)= IY
      A(IA)= 1
      F(IA)= IY
      IA    = IA+1
      IB    = IB+1
  325 IF(LS(IY)) 335,330,335
  330 IY    = J(IY)
      GO TO 340
C
  335 IY    = LS(IY)+N
  340 IF(I(IY)-I2) 345,342,345
  342 ISWA = 1
      GO TO 355
C
  345 CONTINUE
      IF(IH-N) 305,305,350
  350 IF(LS(IY)-(IH-N)) 315,310,315
  355 DX = MIN0(X(IY),DX)
C
      A(IA)= 0
      F(IA)= IY
      IA    = IA+1
      IF(ISWA) 400,357,400
C
  357 IH    = IY
  360 IY    = K(IY)
      IF(IH-K(IY)) 380,365,380
  365 IF(IY-N) 370,370,375
  370 IF(LS(IY)) 375,312,375
  375 A(IA)= 0
      F(IA)= IY
      IA    = IA+1
```

```
      GO TO 314
C
  380 IF(IY-N) 385,385,390
  385 IF(LS(IY)) 390,360,390
C
  390 CONTINUE
C
      E(IB)=DX
      B(IB)= 0
      G(IB)= IY
      A(IA)= 1
      F(IA)= IY
      IA   = IA+1
      IB   = IB+1
      GO TO 360
C
  400 CONTINUE
C
C ======================================================================
C
C * PART 5 *   CHANGE   VALUES   OF   LOOP   ELEMENTS *
C
C ======================================================================
C
      ISWA = 0
      ISWB = 0
C
  405 IA   = IA-1
      IF(IA) 500,500,410
  410 IY   = F(IA)
      IF(A(IA)) 420,450,420
  420 IF(A(IA)-2) 405,430,405
  430 IA   = IA-1
      IF(IA) 500,500,435
  435 IF(A(IA)-1) 430,440,430
  440 IF(F(IA)-IY) 430,450,430
  450 IF(ISWB) 460,470,460
  460 ISWB = 0
      X(IY)= X(IY)+DX
      GO TO 405
C
  470 ISWB = 1
      X(IY)= X(IY)-DX
      IF(ISWA) 405,480,405
  480 IF(X(IY)) 405,490,405
C
  490 I1    = I(IY)
      J1    = J(IY)
      IH1   = IY
      ISWA  = 1
      GO TO 405
C
  500 CONTINUE
C
C ======================================================================
```

```
C
C * PART 6 *    * MODIFY   BASIS   TABLE *
C
C ===========================================================================
C
      IH2   = IH1
      IY    = IH1
      IC    = IH1
C
      IF(J1-J2) 550,510,550
  510 ISWA = 0
  520 KP   = IH1
  530 IF(K(KP)-IH1) 540,710,540
  540 KP   = K(KP)
      GO TO 530
C
  550 IF(IH1-N) 560,560,570
  560 ISWA = 1
      GO TO 520
C
  570 KP   = IH1
  580 IF(LS(KP)) 605,590,605
  590 KP   = J1
      GO TO 580
C
  600 KP   = LS(KP)+N
      GO TO 580
C
  605 IF(LS(KP)-(IH1-N)) 600,610,600
  610 LS(KP)= LS(IY)
      ISWB  = 0
  620 KP    = J2
      LS(IC)= LS(KP)
      J(IC) = J2
      LS(KP)= IC-N
      IF(ISWB) 640,630,640
C
  630 IF(I1-I2) 510,790,510
  640 IF(I(IC)-I2) 650,690,650
  650 IF(I1-I2) 720,660,720
  660 IF(IP-IY) 670,700,670
  670 KP    = IP
  680 I(IC) = I2
      K(IC) = K(KP)
      K(KP) = IC
      GO TO 790
C
  690 KP    = IH1
      GO TO 680
C
  700 KP    = IH2
      GO TO 680
C
  710 K(KP) = K(IY)
      IF(ISWA) 750,720,750
```

```
  720 KP     = 1
  730 IF(I(KP)-I2) 740,680,740
  740 KP     = KP+1
      GO TO 730
C
  750 IC     = LS(IY)+N
      IH2    = IC
      KP     = IH2
  760 IF(K(KP)-IH2) 770,780,770
  770 KP     = K(KP)
      GO TO 760
C
  780 K(KP) = IY
      K(IY) = K(IC)
      I(IY) = I(IC)
      IP    = K(IY)
      LS(IY)= LS(IC)
      X(IY) = X(IC)
      ISWB  = 1
      GO TO 620
C
  790 X(IC) = DX
      GO TO 800
C
C ==========================================================================
C
C * PART 7 *   COMPUTE  DUAL  VARIABLES   FOR  NEW  BASIS *
C
C ==========================================================================
C
  800 CONTINUE
      IB     = 1
      IY     = IH2
      IE     = I2
C
      U(IE) = U(IE)-DC
      IF(K(IY)-IY) 810,250,810
C
  810 IH     = IY
  820 IY     = K(IY)
      ID     = J(IY)
      V(ID) = V(ID)+DC
      IF(K(IY)-IH) 830,860,830
  830 IF(IY-N) 840,840,850
  840 IF(LS(IY)) 850,820,850
  850 B(IB) = 0
      G(IB) = IY
      IB     = IB+1
      GO TO 820
C
  860 IF(IY-N) 870,870,900
  870 IF(LS(IY)) 900,880,900
  880 IB     = IB-1
      IF(IB) 890,250,890
  890 IY     = G(IB)
```

```
      IF(B(IB)-1) 900,810,900
  900 IH    = IY
  910 IF(LS(IY)) 920,930,920
  920 IY=LS(IY)+N
      GO TO 940
C
  930 IY    = J(IY)
  940 IE    = I(IY)
      U(IE) = U(IE)-DC
      IF(IH-N) 950,950,960
  950 IF(LS(IY)) 980,970,980
  960 IF(LS(IY)-(IH-N))980,970,980
  970 IF(IY-K(IY)) 810,880,810
  980 IF(IY-K(IY)) 990,910,990
  990 B(IB) = 1
      G(IB) = IY
      IB    = IB+1
      GO TO 910
C
 1000 CONTINUE
      IGO = 4
C
C =======================================================================
C
C * PART 8 *          ** PRINT  FINAL  SOLUTION **
C
C =======================================================================
C
C
 1002 CONTINUE
C
C        *****   CALCULATE S AND D   *****
C
      DO 1003 IJ=1,N
 1003 D(IJ)=0
C
      DO 1004 IJ=1,M
 1004 S(IJ)=0
C
      DO 1060 KT=1,MN1
      IJ=J(KT)
      D(IJ)=D(IJ)+X(KT)
      IJ=I(KT)
      S(IJ)=S(IJ)+X(KT)
 1060 CONTINUE
C
      KSUMS = 0
      KSUMD = 0
C
      DO 1075 IJ=1,N
 1075 KSUMD = KSUMD+D(IJ)
C
      DO 1080 IJ=1,M
 1080 KSUMS = KSUMS + S(IJ)
C
```

```
C         ** TOTAL S = TOTAL D   IF NOT ERROR CANCEL JOB **
C
      IF (KSUMS-KSUMD) 103,1090,103
 1090 CONTINUE
C
      IF (IPALL-3) 2000,1093,1092
 1092 IF(IPALL-5) 1093,2000,2000
 1093 CONTINUE
C
      WRITE (KW,1095) KAUNT,M,N
 1095 FORMAT(1H1,14H***  ITERATION,  I4,5H  *** ,
     1         22H NUMBER OF ROWS    -M =,I4,22H NUMBER OF COLS   -N =I4/)
C
C         *****   BASIS  TABLE   *****
C
      WRITE (KW,1100)
 1100 FORMAT (12X, 22H**** BASIS  TABLE ****  )
C
      WRITE (KW,1110)
 1110 FORMAT( 1H ,131(1H-))
      WRITE (KW,1120)
 1120 FORMAT ( 83H        ***KT*....*I(KT)*...*K(KT)*...*J(KT)*....*LS(KT)*
     1+..*X(KT)*..*F(KT)*..*A(KT)X        )
      WRITE (KW,1110)
C
      DO 1130 KT=1,MN1
      WRITE(KW,1125) KT,I(KT),K(KT),J(KT),LS(KT),X(KT),F(KT),A(KT)
 1125 FORMAT (1H ,12I10)
 1130 CONTINUE
      WRITE (KW,1110)
C
      WRITE(KW,1131) (U(IP),IP=1,M)
 1131 FORMAT (6H **U**,25I5)
      WRITE(KW,1132) (V(IP),IP=1,N)
 1132 FORMAT (6H **V**,25I5)
 1134 FORMAT (6H **F**,25I5)
 1135 FORMAT (6H **A**,25I5)
      WRITE(KW,1136) (B(IP),IP=1,M1)
 1136 FORMAT (6H **B**,25I5)
      WRITE(KW,1137) (G(IP),IP=1,M1)
 1137 FORMAT (6H **G**,25I5)
C
C
      WRITE (KW,1160) I2,J2,I1,J1,IH1
 1160 FORMAT (/21H ELEMENTS ADDED  I2 =,  I4,6H  J2 =, I4/
     122H ELEMENTS DROPPED I1 =,I4,5H J1 =,I4,7H IH1 = ,I4)
C
C         *****  SHIPMENT  MATRIX  *****
C
      KSUMS = 0
      WRITE (KW,1180)
 1180 FORMAT(/12X,25H**** SHIPMENT MATRIX ****)
      WRITE (KW,1110)
C
      DO 1300 IROW=1,M
```

```
      KSUMS=KSUMS+S(IROW)
C
      DO 1220 KZ=1,N
 1220 E(KZ) = 0
C
      DO 1240 KZ=1,MN1
      IF(IROW-I(KZ)) 1240,1230,1240
 1230 NCOL = J(KZ)
      E(NCOL)=X(KZ)
 1240 CONTINUE
C
      WRITE(KW,1250) S(IROW),(   E(IJ),IJ=1,N)
 1250 FORMAT(1H ,1H*,I5,1H*,24(1H+,I4))
 1300 CONTINUE
      WRITE (KW,1110)
      WRITE (KW,1250) KSUMS,(D(IJ),IJ=1,N)
      WRITE (KW,1110)
C
C        *****  COST  MATRIX  *****
      KOST = 0
      WRITE (KW,1420)
 1420 FORMAT(/12X, 22H**** COST  MATRIX ****)
      WRITE (KW,1110)
C
      DO 1500 IROW=1,M
      DO 1430 KZ=1,N
 1430 E(KZ) = 0
C
      DO 1470 KZ=1,MN1
      IF (IROW-I(KZ)) 1470,1450,1470
 1450 NCOL = J(KZ)
      IJ= (IROW-1)*N+NCOL
      KOST = KOST+X(KZ)*C(IJ)
      E(NCOL) = C(IJ)
 1470 CONTINUE
C
      WRITE(KW,1250) U(IROW),(E(IJ),IJ=1,N)
 1500 CONTINUE
C
      WRITE (KW,1110)
      WRITE(KW,1250) KOST,(V(IJ),IJ=1,N)
      WRITE (KW,1110)
      WRITE(KW,1510) KOST
 1510 FORMAT (11H TOTAL COST,I8/)
C
C
C        *****  ELEMENT  MATRIX   *****
C
      WRITE (KW,1620)
 1620 FORMAT(/12X,24H**** ELEMENT MATRIX ****)
      WRITE (KW,1110)
      DO 1700 IROW = 1,M
C
      DO 1630 KZ =1,N
 1630 E(KZ) = 0
```

```
      C
            DO 1670 KZ = 1,MN1
            IF (IROW-I(KZ)) 1670,1650,1670
       1650 NCOL = J(KZ)
            E(NCOL) = KZ
       1670 CONTINUE
      C
            WRITE (KW,1250) IROW,(E(IJ),IJ=1,N)
       1700 CONTINUE
      C
            WRITE (KW,1110)
      C
      C
            IF (IGO-1) 2000,8253,2000
       2000 CONTINUE
      C
            RETURN
            END
```

```
      PROGRAM GUPTA (INPUT,OUTPUT,TAPE60=INPUT,TAPE61=OUTPUT)
C
C
      COMMON     IPALL,KOST,KSUMS
      INTEGER    C(900),S(30),U(30),D(30),V(30),E(59),F(88),A(88)
      INTEGER    I(59),J(59),K(59),LS(59),X(59),G(29),B(29)
C
C
      KR    = 60
      KW    = 61
      IBLANK = 0
      IPALL = 3
C
C     ****  READ SIZE OF MATRIX ***
C
      DO 2000 IPT=1,5
      READ(KR,10) M,N
   10 FORMAT (16I5)
      WRITE(KW,15) M,N
C
      MN1  = M+N-1
      IP1   = 1
C
C   ****    READ  COST MATRIX ***
      DO 20 IK=1,M
      IP2   = IP1+N-1
      READ(KR,10) (C(IP),IP=IP1,IP2)
      WRITE (KW,16)(C(IP),IP=IP1,IP2)
   15 FORMAT(22H1NUMBER OF ROWS    -M =,I4/22H NUMBER OF COLS    -N =I4/)
   16 FORMAT (1H ,12I10)
   20 IP1   = IP1+N
      WRITE(KW,15) M,N
C    ***  READ SUPPLY AND DEMAND   ***
      READ  (KR,10)  (D(IP),IP=1,N)
      WRITE (KW,32)  (D(IP),IP=1,N)
      READ  (KR,10)  (S(IP),IP=1,M)
      WRITE (KW,33)  (S(IP),IP=1,M)
C
   32 FORMAT (6H **D**,25I5)
   33 FORMAT (6H **S**,25I5)
C
C    ***  GO TO TRANSPORTATION  SUBROUTINE  ***
C
      CALL         GXPOSE (A,B,C,D,E,F,G,I,J,K,LS,M,N,S,U,V,X)
C
 2000 CONTINUE
C
      STOP
      END
```