6-1973

Initial Feasible Solutions of Three Dimensional Transportation Problems

Mamoru Aiga
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Mamoru Aiga entitled "Initial Feasible Solutions of Three Dimensional Transportation Problems." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Randall Cline, Major Professor

We have read this thesis and recommend its acceptance:

Robert M. Aiken, Gordon Sherman

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

May 16, 1973

To the Graduate Council:

I am submitting herewith a thesis written by Mamoru Aiga entitled "Initial Feasible Solutions of Three Dimensional Transportation Problems." I recommend that it be accepted for nine quarter hours of credit in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Randall E. Elkins
Major Professor

we have read this thesis
and recommend its acceptance:

Robert M. Aiken

Gordon R. Sherman

Accepted for the Council:

Vice Chancellor for
Graduate Studies and Research

147

INITIAL FEASIBLE SOLUTIONS OF THREE DIMENSIONAL TRANSPORTATION PROBLEMS

A Thesis

Presented to

the Graduate Council of

The University of Tennesee

In Partial Fullfillment

of the Requirements for the Degree

Master of Science

---

by

Mamoru Aiga

June 1973

ACKNOWLEDGEMENTS

The author wishes to express sincere gratitude to Dr. Randall Cline for his patient guidance in the preparation of this paper and to Dr. Robert Aiken and Dr. Gordon Sherman for their services on the Master's degree committee. Appreciation is also expressed to University Computing Center of the University of Tennessee for providing the author the invaluable opportunity and financial support to study at the University and to make this accomplishment possible.

ABSTRACT

The purpose of this paper is to investigate methods to obtain initial feasible solutions of three dimensional transportation problems. Schell's procedure was tested on various randomly generated problems, and it was determined that this algorithm did not always yield an initial feasible solution. Thus a modified Schell procedure was developed.

Computer programs were written to compare the modified Schell procedure with Phase I of Simplex method. It was concluded , from cases tested, that the modified Schell procedure requires much less computing time and generally gives a feasible solution closer to the optimum solution.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

Although the Simplex method provides a general algorithm for solving LP (Linear Programming) problems, various techniques have been developed for classes of problems with special structure. Perhaps the most famous of these is the Hitchcock-Koopmans transportation problem where it is required to distribute some products from m producers (origins) to n consumers (destinations), subject to minimum total shipping cost. This well known problem is such that it can be solved directly using the "stepping stone" algorithm as applied to any initial basic feasible solution which again may be constructed in several ways (see, for example Hadley (1)). In a recent numerical study, McWilliams (2), compared several of the methods for obtaining initial basic feasible solutions of randomly generated transportation problems. It is the purpose of this thesis to examine the question of obtaining initial basic feasible solutions to a class of LP problems obtained by generalizing the Hitchcock-Koopmans transportation problem. This class of problems, to be called " three dimensional transportation problems ", has been considered by Schell (3), Cline and Pyle (4), and others.

Following the general usage of notation in (1),(4), capital Latin letters will designate matrices and small Latin letters will designate column vectors. (Thus, if A is

m by n, x is an n-tuple and b is an m-tuple, Ax=b is simply a system of m linear equations in n unknowns.) The symbol 0 will be used to denote both zero and the null matrix, where the size of the null matrix is determined by the contex. x will designate a row vector and $x \geq 0$ implies that every element in vector x is greater than or equal to zero. Then a general LP problem (in equality form) is to determine x such that

$$Ax=b$$

$$x \geq 0$$

$$\min(\max) \quad z = c^T x.$$

As indicated in the next section the Hitchcock-Koopmans transportation problem and the three dimensional transportation problem to be considered herein are obtained by special choices of the matrix A.

II. BACKGROUND

Two dimensional transportation problem

Let m and n be any positive integers. Then the two dimensional transportation problem can be formulated as follows: A product is available in known quantiites at each of m origins. It is required that given quantities of the product be shipped to each of n destinations, where the cost of shipping from any origin to any destination is known. The problem is to determine the shipping schedule which minimizes the total cost of shipping. To now formulate the problem mathematically, let a_i be the quantity of the product available at origin i, and let b_j be the quantity of the product required at destination j. Also, let the cost of shipping one unit from origin i to destination j be c_{ij}. Then if x_{ij} denotes the number of units to be shipped from origin i to destination j, we want to minimize the total shipping cost

$$z = \sum_{i,j} c_{ij}x_{ij} , \qquad (1)$$

subject to the constraints

$$\sum_{j=1}^{n} x_{ij} = a_i > 0, \qquad i=1,.....,m, \qquad (2)$$

$$\sum_{i=1}^{m} x_{ij} = b_j > 0, \qquad j=1,.....,n, \qquad (3)$$

$$x_{ij} \geq 0, \qquad \begin{matrix} i=1,.....,m, \\ j=1,.....,n, \end{matrix} \qquad (4)$$

and

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \qquad (5)$$

In case equality in equation (5) does not hold , we only have to add a pseudo origin or a pseudo destination which requires the number of units which is the difference of (See, for example, Hadley (1)).

A two dimensional transportation problem is usually written in tableau form. The tableau of an m x n two dimensional transportation problem is shown in Figure 1.

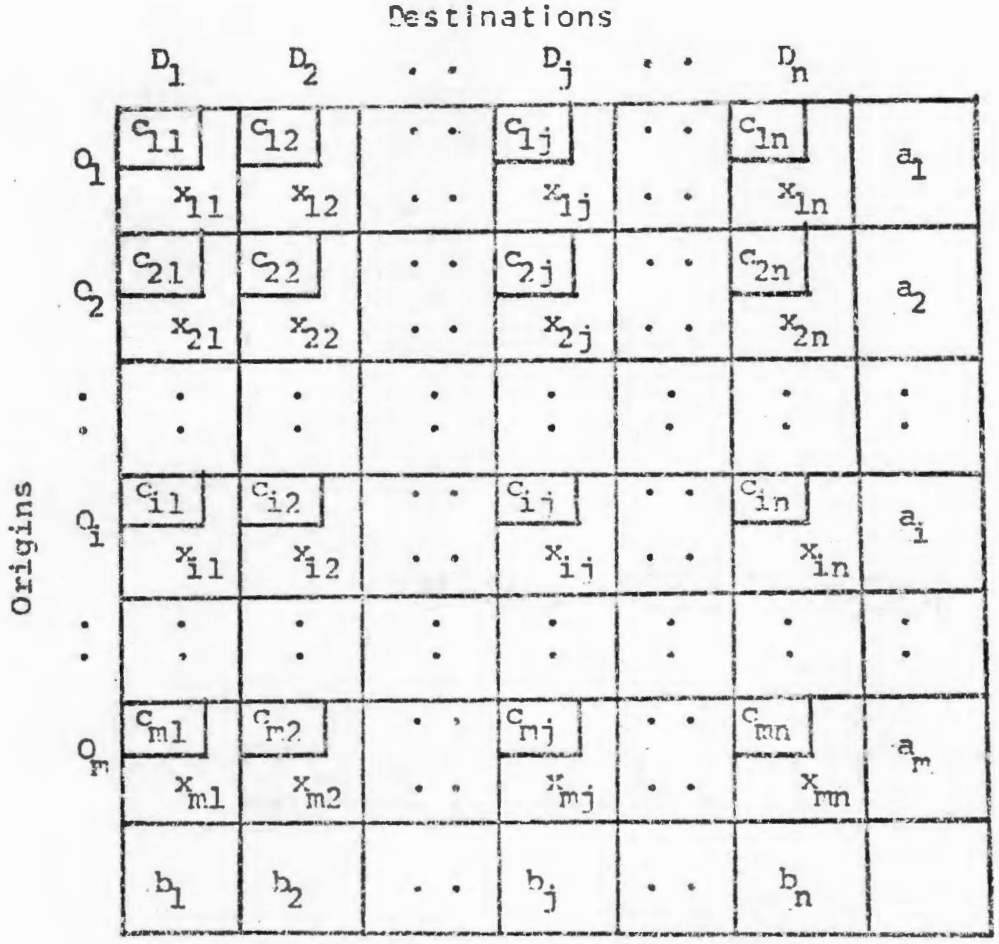


Figure 1. Table of a two dimensional transportation problem.

To formulate the general m x n two dimensional transportation problem as a LP problem, we let

$$T_{m,n} = \begin{bmatrix} I_n & I_n & \cdots & I_n & I_n \\ e_n^T & 0 & \cdots & 0 & 0 \\ 0 & e_n^T & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & e_n^T & 0 \\ 0 & 0 & \cdots & 0 & e_n^T \end{bmatrix}$$

where I_n is n x n identity matrix, e_n^T is a row vector such that every element is 1 and the number of submatrices I_n is m. Also let

$$b = (b_1, b_2, \ldots, b_n, a_1, a_2, \ldots, a_m)^T,$$

$$c = (c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots, c_{m1}, \ldots, c_{mn})^T.$$

Then the conditions in (2),(3) and (4) can be written as

$$T_{m,n}\, x = b$$

and

$$x \geq 0.$$

where

$$x = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{m1}, \ldots, x_{mn})^T$$

corresponding to the x_{ij} in the tableau form.

Moreover, z in (1) can be written as the inner product

$$z = c^T x = (x, c).$$

Three dimensional transportation problem

Let l,m and n be any positive integers. Then the three dimensional transportation problem can be formulated as follows: Assume l kinds of products are available in known quantities at each of m origins. It is required that given quantities of the products be shipped to n destinations, where the cost of shipping any kind of product from any origin to any destination is known. The problem is to determine the shipping schedule which minimizes the total shipping cost. To now formulate the problem mathematically, let a_{ik} be the quantity of product k available at origin i, let b_{jk} be the quantity of prolet duct k required at destination j and let d_{ij} be the total quantity of every kind of product to be shipped from origin i to destination j. Also let the cost of shipping one unit of product k from origin i to destination j be c_{ijk}. Then if x_{ijk} denotes the number of units of product k to be shipped from origin i to destination j, we want to minimize total shipping cost

$$z = \sum_{i,j,k} c_{ijk}x_{ijk} , \qquad (6)$$

subject to the constraints

$$\sum_{j=1}^{m} x_{ijk} = a_{ik} \geq 0, \qquad \begin{matrix} i=1,.....,l, \\ k=1,.....,n, \end{matrix} \qquad (7)$$

$$\sum_{i=1}^{l} x_{ijk} = b_{jk} \geq 0, \qquad \begin{matrix} j=1,.....,m, \\ k=1,.....,n, \end{matrix} \qquad (8)$$

$$\sum_{i=1}^{l} x_{ijk} = b_{jk} > 0 \qquad j=1,.....,m, \quad k=1,.....,n, \qquad (8)$$

$$\sum_{k=1}^{n} x_{ijk} = d_{ij} > 0 \qquad i=1,.....,l, \qquad (9)$$

$$\sum_{k=1}^{n} a_{ik} = \sum_{j=1}^{m} d_{ij} \qquad (10)$$

$$\sum_{i=1}^{l} d_{ij} = \sum_{k=1}^{n} b_{jk} \qquad (11)$$

$$\sum_{j=1}^{m} b_{jk} = \sum_{i=1}^{l} a_{ik} \qquad (12)$$

$$x_{ijk} \geq 0 \qquad (13)$$

(Schell (3) also suggested various alternatives which can be considered as three dimensinal problems (for example, ellmination of constraints (9),(10) and (11)). In this thesis, however only the case in which the constraints (7) to (13) are all included is considered as this is the most obvious direct extension to the two dimensional transportaion problem.) A tableau form of a three dimensional transportation problem is shown in Figure 2.

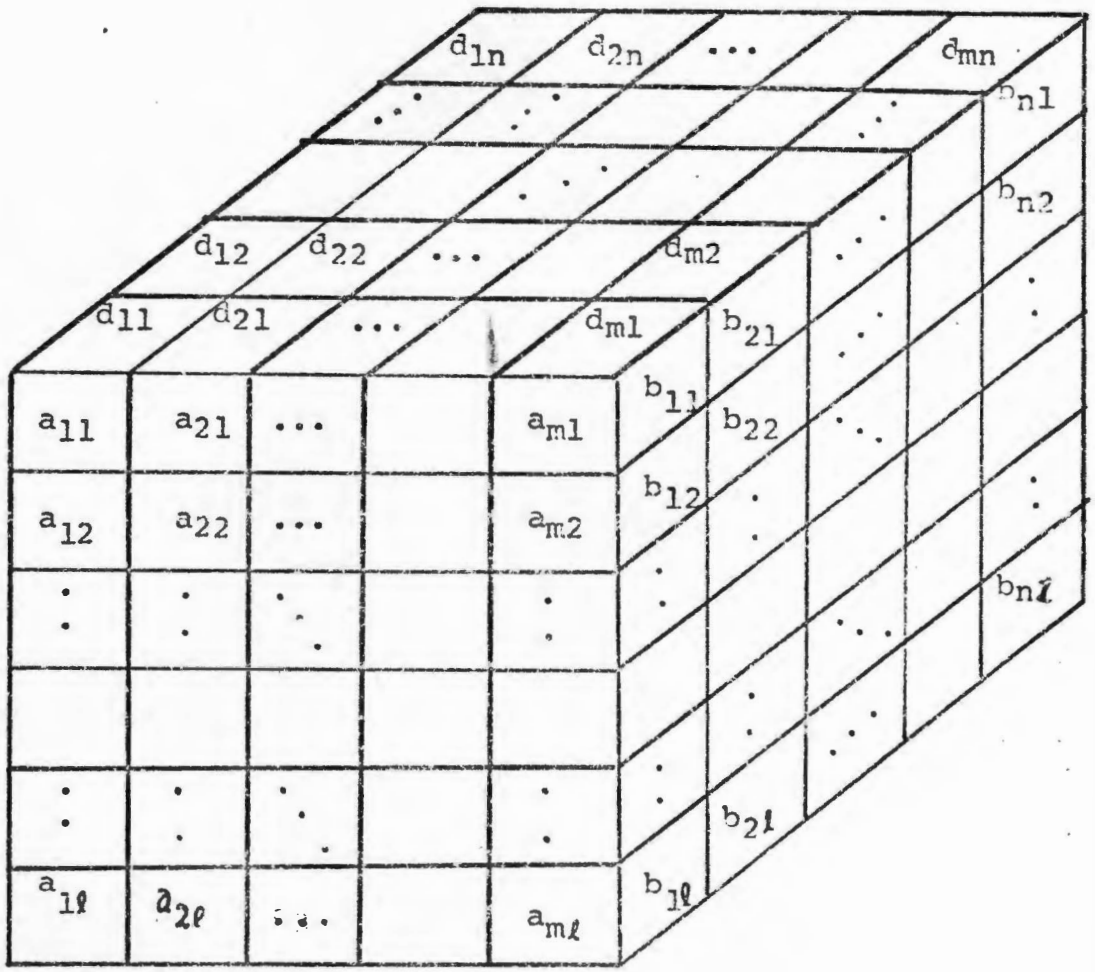


Figure 2. A tableau form of a ℓ x m x n three dimensional transportation problem.

As shown in Figure 3, the tableau of a three dimensional transportation problem can be viewed as slices which are two dimensional tableaus, subject to the condition that corresponding elements from each tableau sum to a final quantity. This decomposition of a three dimensional tableau into slices is illusrated in Figure 3 for the special case l=3,m=2 and n=4.

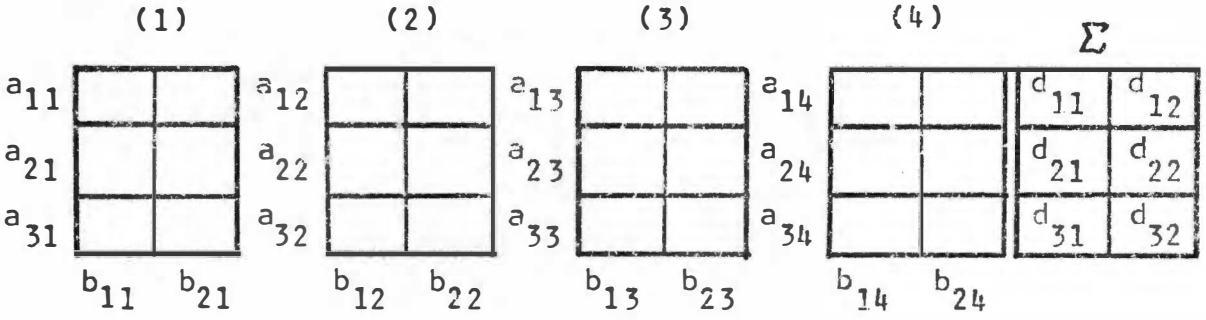


Figure 3: The tableau of a 3 x 2 x 4 three dimensional transportation problem.

To formulate the general l x m x n three dimensional transportation problem mathematically, we let

$$T_{l,m,n} = \begin{bmatrix} I_{lm} & I_{lm} & \cdot & \cdot & \cdot & I_{lm} & I_{lm} \\ T_{l,m} & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ 0 & T_{l,m} & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & T_{l,m} & 0 \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & T_{l,m} \end{bmatrix}$$

and let $b = (\bar{d}^T, \bar{b}^T, \bar{a}^T)^T$, $c = (\bar{c}_1^T, \bar{c}_2^T, \ldots, \bar{c}_n^T)^T$

where

$$\bar{a} = (a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, \ldots, a_{1n})^T,$$
$$\bar{b} = (b_{11}, b_{12}, \ldots, b_{1n}, b_{21}, \ldots, b_{mn})^T,$$
$$\bar{d} = (d_{11}, d_{12}, \ldots, d_{1m}, d_{21}, \ldots, d_{1m})^T,$$
$$\bar{c}_1 = (c_{111}, c_{121}, \ldots, c_{1m1}, c_{211}, \ldots, c_{1m1})^T,$$
$$\bar{c}_2 = (c_{112}, c_{122}, \ldots, c_{1m2}, c_{212}, \ldots, c_{1m2})^T,$$
$$\cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot$$
$$\bar{c}_n = (c_{11n}, c_{12n}, \ldots, c_{1mn}, c_{21n}, \ldots, c_{1mn})^T.$$

Then the conditions in (7),(8),(9) and (10) can be written as

$$T_{1,m,n} x = b$$

and

$$x \geq 0,$$

where

$$x = (x_1^T, x_2^T, \ldots\ldots, x_n^T)^T$$

with

$$x_1 = (x_{111}, x_{121}, \ldots, x_{1m1}, x_{211}, \ldots, x_{1m1})^T,$$
$$x_2 = (x_{112}, x_{122}, \ldots, x_{1m2}, x_{212}, \ldots, x_{1m2})^T,$$
$$\cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot$$
$$x_n = (x_{11n}, x_{12n}, \ldots, x_{1mn}, x_{21n}, \ldots, x_{1mn})^T,$$

corresponding to the x_{ijk} in the tableau form.

Moreover z in (6) can be written as the inner product

$$z = c^T x = (x, c).$$

III. METHODS FOR OBTAINING INITIAL FEASIBLE SOLUTIONS

As described in the introduction, there are algorithms to find an optimum solution for two dimensional transportation problem given a basic feasible solution. We also have various algorithms (1) to find initial basic feasible solutions. These algorithms are developed utilizing the following algorithm (4).

Algorithm

Given that x_{ij} is the variable to be given a value, make it as large as possible, consistent with row and column totals, i.e., set

$$x_{ij} = \text{Min } (a_i, b_j)$$

Case1: If $a_i < b_j$, then all the other variables in the ith row are to be given the value zero and designated as nonbasic. Next, delete the jth row, reduce the value of b_j to $(b_j - a_i)$, and proceed in the same manner to evaluate a variable in the reduced array composed of the m-1 rows and n columns remaining.

Case2: If $a_i > b_j$, then the jth columns is to be deleted and a_i replaced be $a_i - b_j$.

Case3: If $a_i = b_j$, then delete either the row or the column but not both. If several columns, but only one row remain in the reduced array, then drop jth column and conversely, if several rows and one column, drop the ith row.

This rule will select as many variables for the basic set as there are rows plus columns, less one, m+n-1, since on the last step, when one row and one column remain, both must be dropped after the last variable is assigend.

Various algorithms such as "north west corner rule","row minima" "column minima" (1), etc., are simply different methods for making the sequence of assignments. As noted in the introduction, McWilliams (2) tested certain of these methods in her thesis.

An approach for obtaining initial basic feasible solutions for three dimensional transportation problem was suggested by Schell (3) and can be described as follows:

(1) Let $m_{ijk} = MIN(a_{ik}, b_{jk}, d_{ij})$.
If $(d_{ijk} - \sum_{P=1, P\neq k}^{n} m_{ijk}) > 0$ then assign the difference to the cell (i,j,k) as a lower limit value.
Reduce the amount of d_{ij} by the lower limit value. Repeat this step for all slices.

(2) Construct a feasible solution to the two dimensional transportation problem of the first slice.

(3) Remove first slice from three dimensional tableau reducing planar sums (a_{ik}, b_{jk}, d_{ij}) by appropriate amounts and repeat steps (1) through (3) with the reduced three dimensional tableau until only one slice remains in three dimensional tableau.
The entries for the last slice are the reduced planar entries and they complete the feasible solution.

(A general flow diagram of this procedure is shown in Appendix 1.)

It should be noted that the above procedure described by Schell is somewhat ambiguous since it does not specify precisely how to construct the feasible solution to the first slice. Although there are various algorithms which always give feasible solutions to the two dimensional transportation problem, it is not clear whether these algorithms will always give feasible solutions to the slice of the three dimensional transoportation problem, for there is the third constraint which will limit the maximum amount to be assigned to cells. This difficulty is illustrated by the following example.

Given a 4 x 4 x 4 three dimensional transportation problem which can be shown to have feasible solutions, Schell's procedure was applied. "Matrix minima (1)" was used to attempt to obtain feasible solution for slices. Figure 4 shows the slices of this problem before amounts are assigned to cells. Figure 5 shows a feasible solution obtained in the first slice.

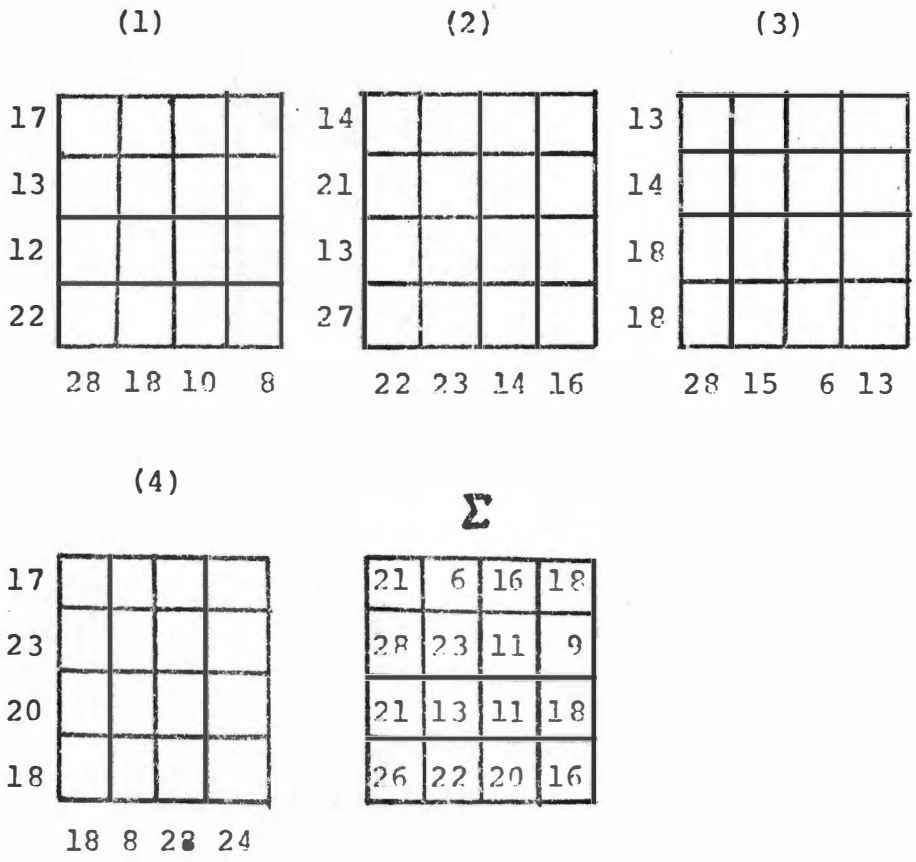


Figure 4. Slices before values are assigned.

The feasible solution of the first slice was obtained with the following assignment order.

	(i,j,k)	amount
1	(3,1,1)	12
2	(1,3,1)	10
3	(1,4,1)	7
4	(4,1,1)	16
5	(4,2,1)	6
6	(2,2,1)	12
7	(2,4,1)	1

(1)

17			10	7
13		12		1
12	12			
22	10	6		
	28	18	10	8

Figure 5. The feasible solution obtained in the first slice.

After eliminating the first slice, the problem is reduced to a 4 x 4 x 3 three dimensional problem as shown in Figure 6.

(2)

14				
21				
13				
27				
	22	23	14	16

(3)

13				
14				
18				
17				
	28	15	6	13

(4)

17				
23				
20				
18				
	18	8	28	24

Σ

21	6	6	11
28	11	11	8
9	13	11	18
10	16	20	16

Figure 6. Slices after eliminating the first slice.

Then the procedure gave the following assignment order and reported that the procedure would not work. (Observe Figure 7.)

	(i,j,k)	amount	
1	(1,2,2)	6	
2	(1,3,2)	6	
3	(2,4,2)	8	
4	(1,4,2)	2	
5	(3,3,2)	8	
6	(4,1,2)	10	
7	(2,2,2)	11	
8	(2,1,2)	2	
9	(3,1,2)	10	* reported not assignable.

(2)

14		6	6	2
21	2	11		8
13	10		8	
27	10			
	22	23	14	16

Figure 7: Assignments for the second slice.
The cell (3,1,2) is not assignable.

Notice that to satisfy b_{12} =22, amount 10 must be assigned to cell (3,1,2), however 5 is the maximum amount assignable to the same cell because of the limit of a_{32} =13 - 8 =5. Here we have encountered the problem that we cannot just employ the same procedure which is used for two dimensional transportation problem to obtain the initial basic feasible solution for the three dimensional transportation problem.

Suppose we used some other scheme and obtained the feasible solution for the second slice as shown in Figure 8.

(2)

14		6	6	2
21	3	10		8
13	9		4	
27	10	7	4	6
	22	23	14	16

Figure 8: The feasible solution for the second slice.

Then the problem is reduced to 4 x 4 x 2 three dimensional transportation problem as shown in Figure 9. (Note that cells which contain a dash must be assigned the value zero since corresponding summation of slices is zero.)

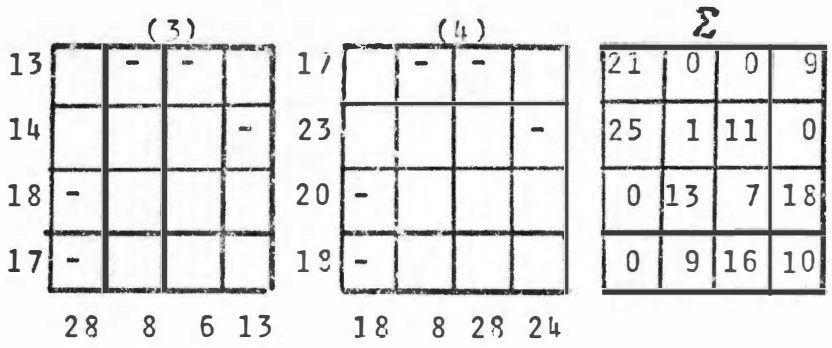


Figure 9. Slices after eliminating two slices.

Observing the third slice , we see at once that the third slice cannot have a feasible solution. Notice that b_{13} =28 and the total amount which is assignable to that column is $a_{13} + a_{23} = 13 + 14 = 27$. In other words, we can not have a feasible solution for the third slice, although we started with a problem which has at least one feasible solution. This type of difficulty certainly necessitates some sort of changes in Schell's procedure. To modify the Schell procedure, the author used a computer program which permitted a variety of choices of assignment orders in each slice. A general flow chart of this modified procedure is shown in Appendix B.

The modified Schell procedure for obtaining an initial feasible solution can be described as follows:

(1) Set k = 0.

(2) Set k = k + 1.

(a) If k is less than 1 then stop.

(b) Otherwise go to (3).

(3) Let $M_{ijp} = MIN\,(a_{ip}, b_{jp}, d_{ij})$.

(a) If $(d_{ij} - \sum_{p=k}^{n} M_{ijp}) > 0$ then assign the difference to cell (i,j,p) as a lower limit value.

Go to (4).

(b) Otherwise go to (4).

(4) Examine kth slice.

(a) If there is no solution in kth slice, then the cell which was assigned first in (k-1)th slice should not be considered as a first assignment cell.

Set k = k - 2.

Go to (2).

(b) Otherwise go to (5).

(5) Find a cell which can be considered as a first assignment cell in kth slice.

(a) If all cells are prohibited as first assignment, then the cell which was assigned first in (k-1)th slice should not be considered as a first assignment cell.

Delete all designations of nonassignable cells in

kth slice.

Set k = k - 2.

Go to (2).

(b) If not all cells are prohibited then try to obtain a feasible solution in kth slice using "Matrix minima".

Start assigning values from the minimum cost cell which is not prohibited.

Go to (6).

(6) (a) If a solution is found, subtract the amounts assigned to kth slice from slice totals (d_{ij}).

(a1) If k<n-1 then go to (2).

(a2) If k=n-1 then go to (7).

(b) If a solution is not found, then the cell which was assigned first in kth slice should not be considered as a first assignment cell.

Go to (5).

(7) Assign remaining slice total (d_{ij}) to nth slice.

Go to (8).

(8) An initial feasible solution was found.

Stop.

An example for obtaining an initial feasible solution using the modified Schell procedure is shown in the following pages.

A basic feasible solution of the same problem was obtained in the following manner using a modification of Schell's procedure. The first feasible solution , which is shown in Figure 10 , is exactly the same result which was obtained previously.

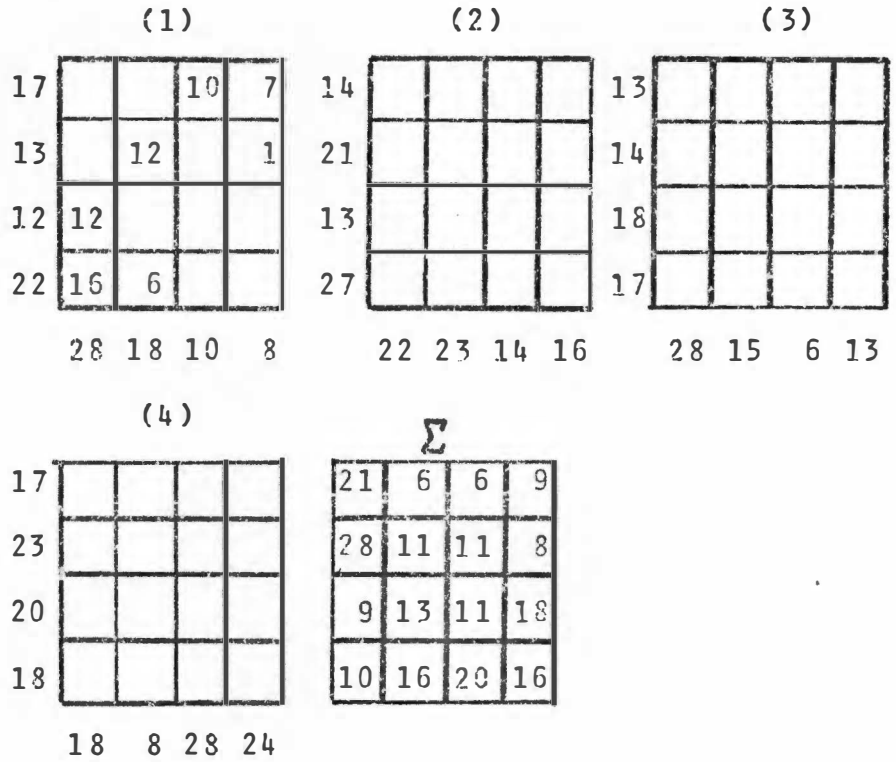


Figure 10: Feasible solution with first slice assigned.

Next the procedure tried the following assignment order to obtain a feasible solution for the second slice.

assignment order	cell (i,j,k)	amount
1	(1,2,2)	6
2	(1,3,2)	6
3	(2,4,2)	8

4	(1,4,2)	2
5	(3,3,2)	8
6	(4,1,2)	10
7	(2,2,2)	11
8	(2,1,2)	2
9	(3,1,2)	10

At the nineth assignment, the procedure found that the assignment order was improper - which indicated that the cell (1,2,2) should not be used as the first assignment cell. Then the procedure tried the following assignment order.

assignment order	cell (i,j,k)	amount
1	(1,3,2)	6
2	(1,2,2)	6
3	(2,4,2)	8
4	(1,4,2)	2
5	(3,3,2)	8
6	(4,1,2)	10
7	(2,2,2)	11
8	(2,1,2)	2
9	(3,1,2)	10

Again at the nineth assignment the procedure found that the assignment order was improper and the cell (1,3,2) should not be used as the first assignning cell. The procedure tried several additional assignment orders and found the following assignment order to obtain the feasible solution for the second slice shown in Figure 11. (Note that cells which contain an asterisk could not be used as the first assignment cells.)

assignment order	cell (i,j,k)	amount
1	(3,1,2)	9
2	(1,2,2)	6
3	(1,3,2)	6
4	(2,4,2)	8
5	(1,4,2)	2
6	(3,3,2)	4
7	(4,4,2)	6
8	(4,1,2)	10
9	(2,1,2)	3
10	(2,2,2)	10
11	(4,2,2)	7
12	(4,3,2)	4

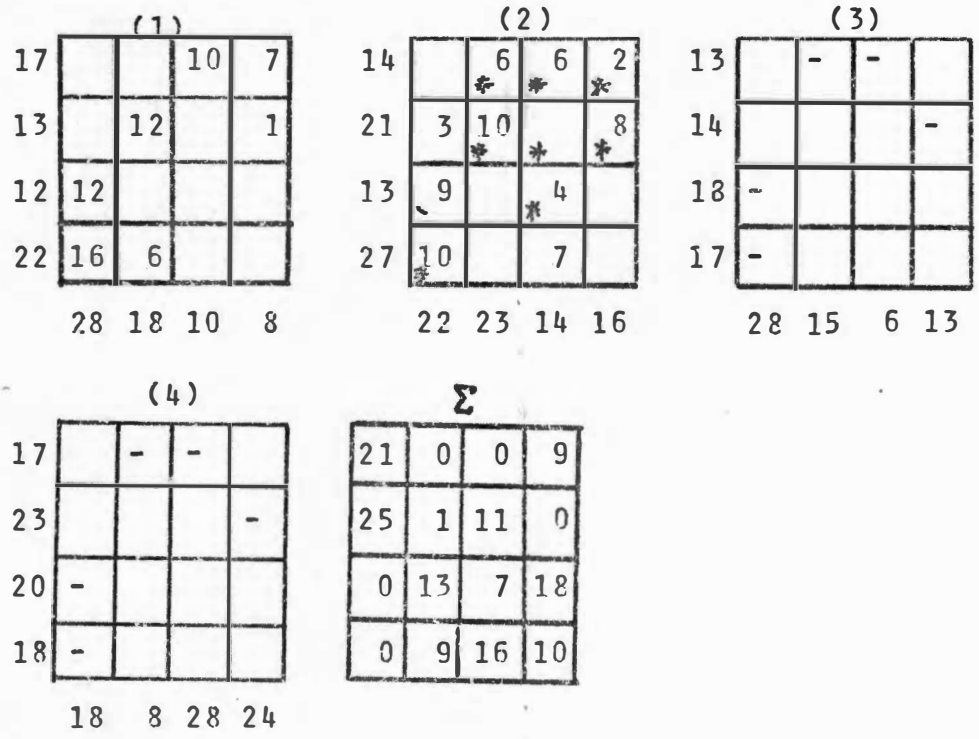


Figure 11. Slices after values were assigned to two slices.

The procedure examined the third slice and decided that the third slice could not have a feasible solution. Thus the procedure decided to alter the feasible solution for the second slice and obtained the feasible solution as shown in Figure 12.

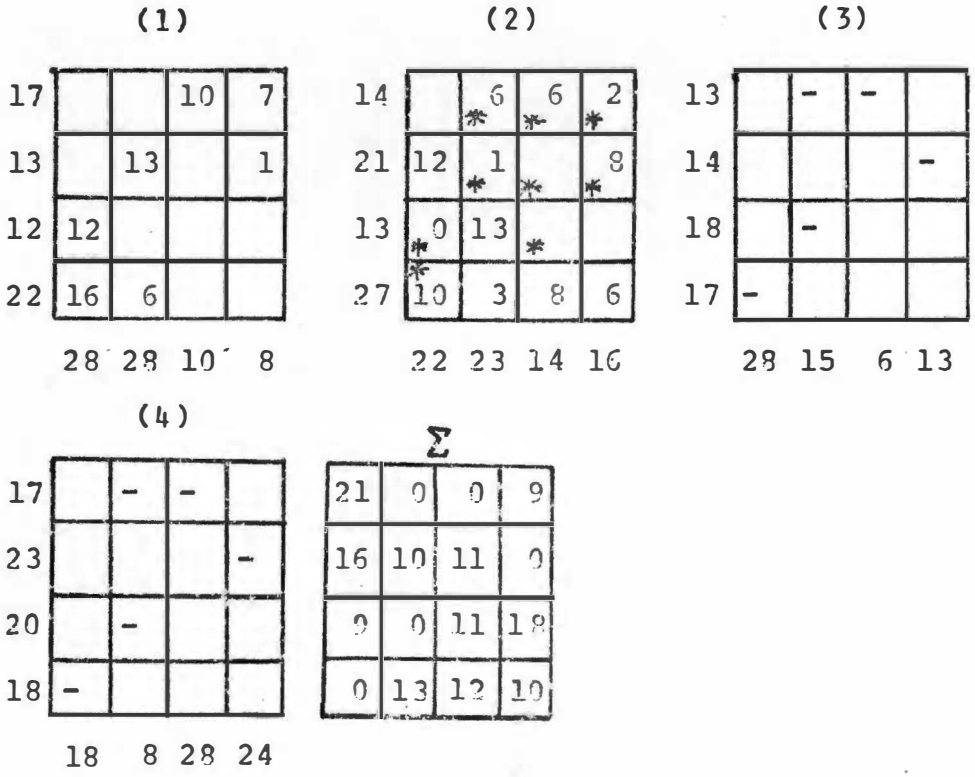


Figure 12. Feasible solutions in two slices.

The procedure then assigned the third slice and obtained the feasible solution shown in Figure 13.

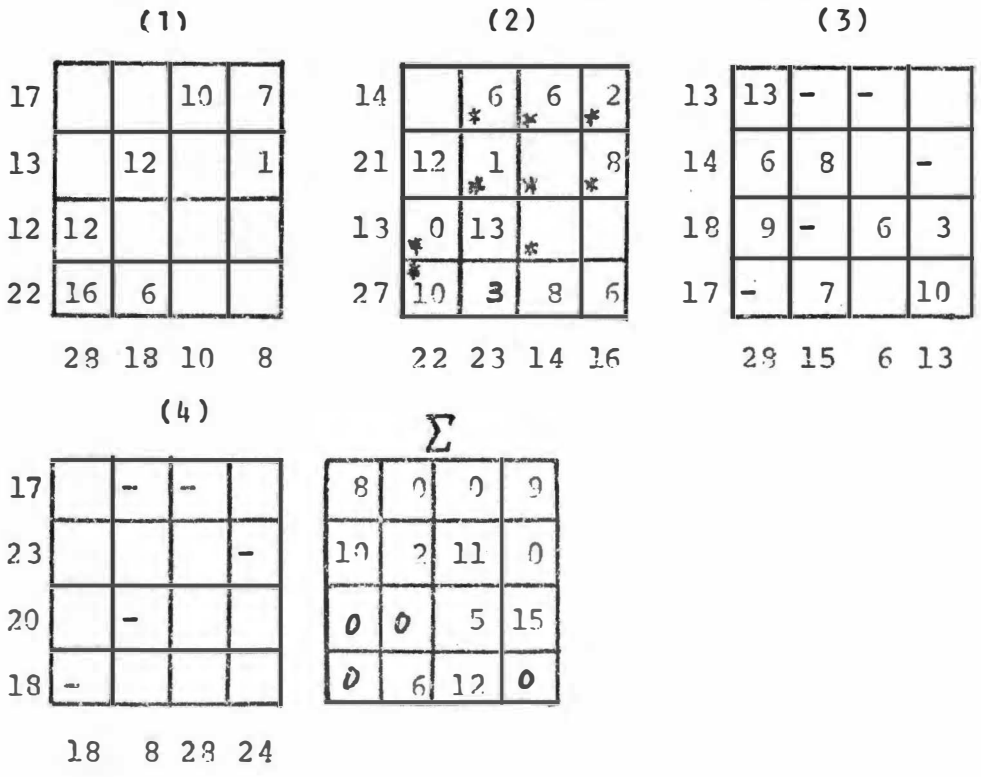


Figure 13. Feasible solutions in three slices.

Finally, the feasible solution of the last slice was obtained by simply assigning the remaining d to the fourth slice Figure 14 shows the basic feasible solution obtained by the procedure.

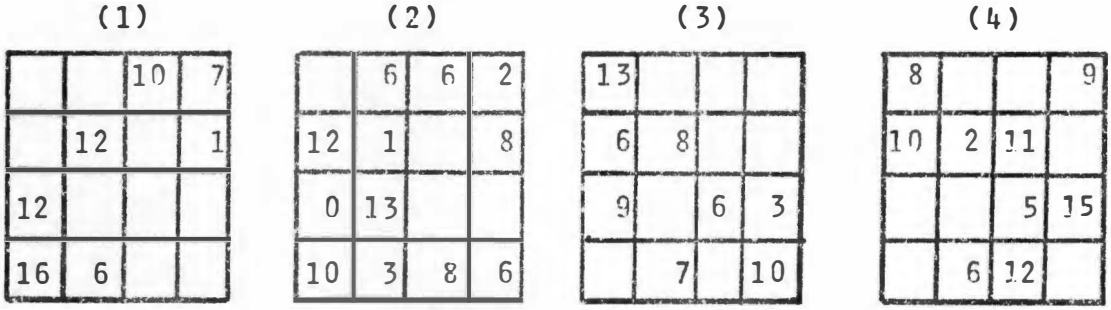


Figure 14. Initial feasible solution.

The difficulties observed in the example can be explained in the following manner: When assigning amounts to any slice k ($k \leq n-1$), any algorithm described in Hadley (1) and McWilliams (4) will give a feasible solution to the slice as long as $d_{ij} \geq$ MIN (a_{ik}, b_{jk}) for all i and j since MIN (a_{ik}, b_{jk}, d_{ij}) = MIN (a_{ik}, b_{jk}) guarantees that assignment order and amounts to be assigned to each cell will be exactly the same as those obtained by existing algorithms for two dimensional transportation problem. However, when $d_{ij} <$ MIN (a_{ik}, b_{jk}) for one or more pairs of indices, i and j, the amount which can be assigned to such a cell (i,j,k) is restricted and the algorithm for the two dimensional transportation problem must be modified.

IV NUMERICAL EXPERIMENTS

Numerical experiments for obtaining initial basic feasible solutions were conducted using a LP code and the modified Schell method. The size of the problems was limited to 4 x 4 x 4. Problems which had at least one feasible solution were obtained by first generating random numbers for each cell. Row totals, column totals and slice totals were obtained by summing the numbers in cells in the row direction, column direction and slice direction respectively. The cost entries and numbers to obtain row,column and slice totals were selected from a uniform distribution of integer values in the range $0 \leq c_{ijk} \leq 9$ and $0 \leq x_{ijk} \leq 9$.

Ten problems were examined. Among the ten problems which were examined, the modified Schell procedure found initial feasible solution to all problems, although the direct application of Schell's procedure found feasible solutions to two out of ten problems. Both the LP procedure and the modified Schell procedure were written in PL1 language for IBM 360 MODEL 65 digital computer. To obtain the initial basic feasible solution by LP procedure, the Two Phase method (1) was used.

Computing time for obtaining initial feasible solutions by the modified Schell procedure averaged 10 seconds compared to approximately 2 minutes by Phase I.

Both optimum solutions (maximum cost and minimum cost) were also obtained using the LP procedure. After matrix generation, the maximizing and minimizing vectors, x' and y', respectively, were computed along with related maximum and minimum values of the objective functions ((x',c) and (y'c) respectively). With this information about a particular problem, initial feasible solutions obtained by Phase I and the modified Schell procedure were compared using the ratio

$$P = \frac{(z,c) - (y',c)}{(x',c) - (v',c)} \quad .$$

This ratio, which was also used by McWilliams is the measure of the portion of the range of the objective function covered by the initial solution. TABLE I shows the result of this experiment.

TABLE I

PORTION OF THE RANGE OF THE OBJECTIVE FUNCTION COVERED BY THE INITIAL SOLUTION OBTAINED BY THE MODIFIED SCHELL'S PROCEDURE AND BY LP PROCEDURE

PROBLEM	P PROCEDURE A	PROCEDURE B
1	0.052	0.492
2	0.082	0.355
3	0.383	0.362
4	0.031	0.457
5	0.403	0.564
6	0.125	0.716
7	0.213	0.271
8	0.159	0.397
9	0.094	0.378
10	0.150	0.470

$$P = \frac{(z,c) - (y',c)}{(x',c) - (y',c)}$$

PROCEDURE A: The modified Schell procedure
PROCEDURE B: LP procedure

V. AREAS FOR FURTHER STUDY

The following problems are areas for further study.

1. Use of methods other than "matrix minima" in assigning values in cells, for example, methods used by McWilliams (2).

2. Investigation of "stepping stone method" type algorithm for three dimensional transportation problems.

3. Attempt to find physical problems which can be formulated as three dimensional transportation problems.

4. Examination of k dimensional transportation problems ($k>3$).

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Hadley, George, Linear Programming, Addison-Wesley Publishing Co., 1962.

2. McWilliams, Gloria J., Master's Thesis, University of Texas at Austin, August, 1970.

3. Scheil, Emil d., "Distribution of a product by several properties," Proceedings of the Second Symposium in Linear programming, Vol. 2, National Bureau of Standards and Directorate of Management Analysis, DCS/Comptroller, USAF, Washington, D. C. 1955.

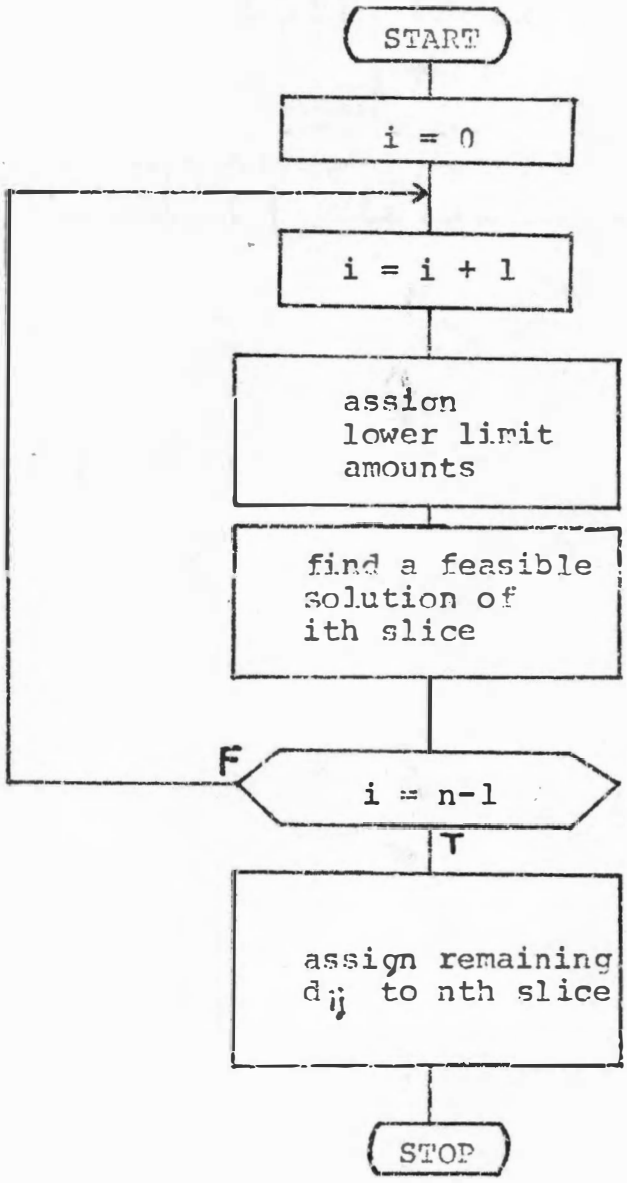
4. Cline, R. E. and L. D. Pyle, "The generalised Inverse in Linear Programming - An intersection Projection Method and the Solution of a Class of Structured Linear Programming Problems," SIAM J. Appl. Math., 24(1973), pp.338-351.

5. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, 1963.

APPENDIXES

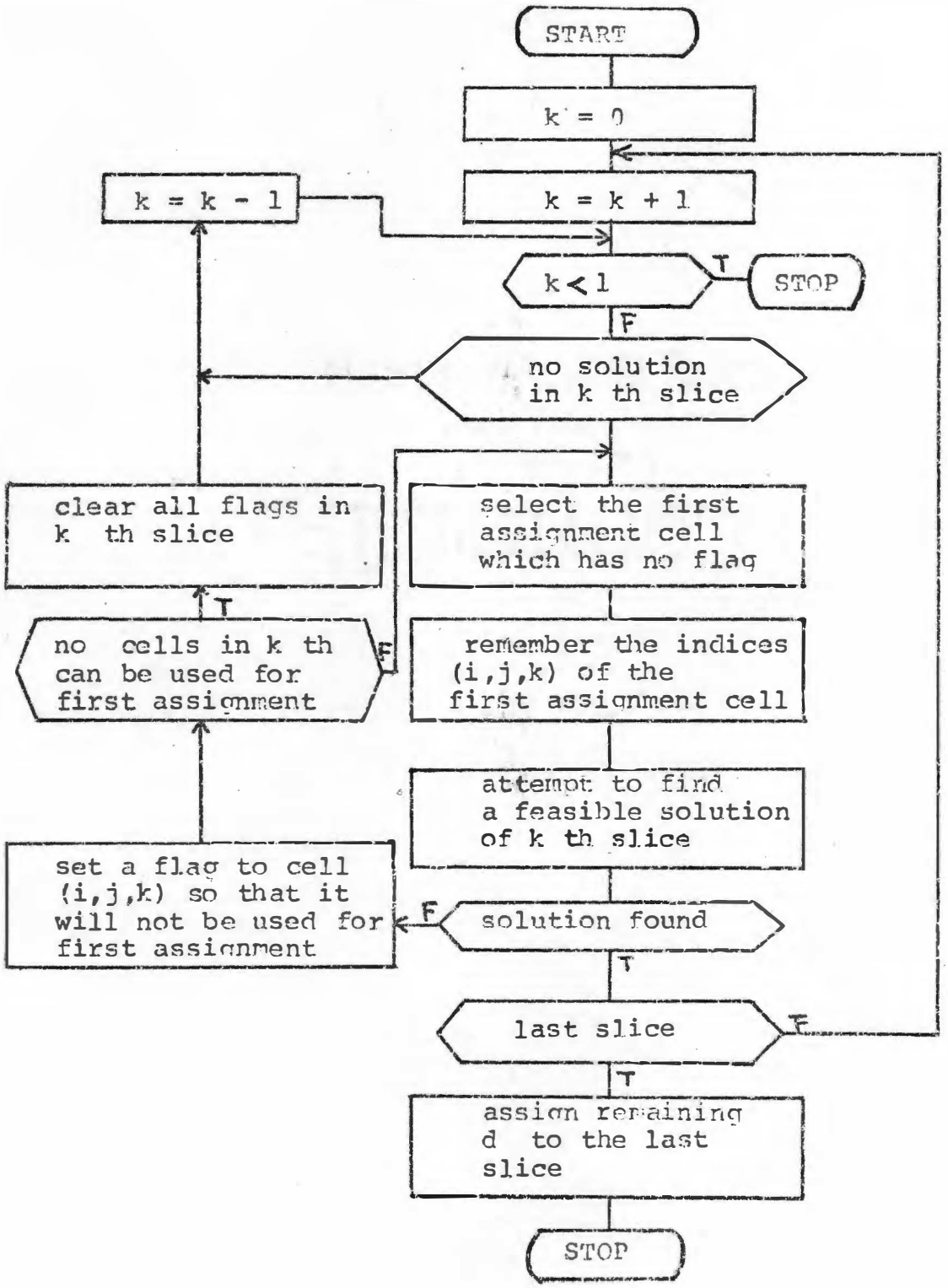
Appendix A

A GENERAL FLOW CHART OF SCHELL'S PROCEDURE



APPENDIX B.

A GENERAL FLOW CHART OF A MODIFIED SCHELL PROCEDURE



APPENDIX C

COMPUTER PROGRAM FOR A MODIFIED SCHELL PROCEDURE

```
SETDUP: PROCEDURE OPTIONS(MAIN);
   DECLARE AMOUNT FIXED;
   DECLARE TEMP FIXED (15,0);
/*************************************************************/
/*                                                           */
/* RANDOM NUMBER GENERATING PROCEDURE                        */
      RANDU: PROCEDURE (IX,IY,YFL);
/*                                                           */
/*************************************************************/
      DECLARE IX BINARY FIXED (31,0);
      DECLARE IY BINARY FIXED (31,0);
/*    DECLARE MK FIXED (15,0);                               */
 S:   IY=IX*65539;
      IF IY < 0 THEN GO TO A; ELSE GO TO B;
 A:   IY=IY+2147483647 +1;
 B:   YFL=IY;
      YFL=YFL*.4656613E-9;
    END;
/*************************************************************/
/*                                                           */
/* THIS PROCEDURE WILL PRINT TABLEAU IN TWO DIMENSINAL       */
/* FIGURE (SLICES).                                          */
PRINT_TABLE: PROCEDURE (MATRIX,NO_OF_ROW,NO_OF_COLUMN);
/*                                                           */
/*************************************************************/
   DECLARE MATRIX (NO_OF_ROW,NO_OF_COLUMN) FIXED;
        DO I=1 TO NO_OF_ROW;
              PUT SKIP (1);
                 DO J=1 TO NO_OF_COLUMN;
                      PUT EDIT ('--------')(A);
                 END;
                 PUT SKIP(1);
                 DO J=1 TO NO_OF_COLUMN + 1;
                      PUT EDIT ('I')(A);
                      PUT EDIT ('       ')(A);
                 END;
                 PUT SKIP (1);
                 DO J=1 TO NO_OF_COLUMN;
                      PUT EDIT ('I')(A);
                      PUT EDIT ('  ')(A);
                      PUT EDIT(MATRIX(I,J))(F(2));
                      PUT EDIT('  ')(A);
                 END;
                 PUT EDIT('I  ',I)(A,F(1));
                 PUT SKIP;
                 DO J=1 TO NO_OF_COLUMN +1;
                      PUT EDIT ('I')(A);
                      PUT EDIT('       ')(A);
                 END;
                 PUT EDIT ('--------')(A);
           END;
```

```
            PUT SKIP(1);
            DO J=1 TO NO_OF_COLUMN;
            END;
            PUT SKIP(1);
            DO J=1 TO NO_OF_COLUMN;
                 PUT EDIT('   ',J,'   ')(A,F(1),A);
            END;
        END;
/*                                                              */
/* THIS PART WILL GENERATE SIZE OF PROBLEM.  L=4; M=4;          */
/* AND N=4; WILL FORCE TO GENERATE 4 X 4 X 4 X MATRIX.          */
/*                                                              */
/* L,M AND N ARE NUMBER OF DIMENSION OF RWO, COLUMN AND         */
/* SLICE RESPECTIVELY.                                          */
      DECLARE IX BINARY FIXED (31,0);
      DECLARE IY BINARY FIXED (31,0);
      DECLARE ALL_DONE BIT(1);
 ON FIXEDOVERFLOW;
        IX=46857;
   INC: IX = IX +2;
      CALL RANDU(IX,IY,YFL);
      L=YFL*10;
      IF L <= 1 THEN GO TO INC;
      CALL RANDU(IY,IX,YFL);
      M=YFL*10;
      IF M <= 1 THEN GO TO INC;
      CALL RANDU (IX,IY,YFL);
      N=YFL*10;
      IF N <= 1 THEN GO TO INC;
        L=4; M=4; N=4;
BEGIN;
      DECLARE THREEX (L,M,N) FIXED;
      DECLARE THREE (L,M,N) FIXED;
      DECLARE COST (L,M,N) FIXED;
      DECLARE ROW_SUM (M,N) FIXED;
      DECLARE COLUMN_SUM (L,N) FIXED;
      DECLARE SLICE_SUM (L,M) FIXED;
      DECLARE VALUE FIXED;
      DECLARE WORK_MATRIX (L,M) FIXED;
   DECLARE CURRENT_TRANSACTION (L,M) FIXED (15,0);
   DECLARE ONE_SPACE_ALREADY_FOUND BIT(1);
   DECLARE ONE_SPACE_IN_ROW_REMAINED BIT(1);
   DECLARE ONE_SPACE_IN_COLUMN_REMAINED BIT (1);
   DECLARE ONE_SPACE_IN_SLICE_REMAINED BIT (1);
   DECLARE ASSIGNED_CELL (L,M,N) BIT (1);
   DECLARE ASSIGNABLE BIT (1);
   DECLARE DANGER_CELL (L,M) BIT(1);
   DECLARE ZERO FIXED;
   DECLARE SW1 BIT (1);
   DECLARE SW2 BIT (1);
   DECLARE SW3 BIT (1);
```

```
   DECLARE SW5 BIT (1);
   DECLARE SW6 BIT (1);
   DECLARE PROHIBIT_CELL (L,M,N) BIT(1);
   DECLARE  RESTART_SLICE FIXED;
   DECLARE IP (N) FIXED;
   DECLARE JP(N) FIXED;
   BEGIN;
/*************************************************************/
/*                                                           */
/* THIS PROCEDURE TRY TO ASSIGN VALUE IN TO CELL (I,J,K) */
/* IF IT IS ASSIGNABLE IT IWLL ASSIGN AMOUNT "AMOUNT"    */
/* IN TO CELL (I,J,K) AND SUBTRACT AMOUNT FROM PLANNAR   */
/* SUMS.                                                     */
/* IF IT IS NOT ASSIGNABLE IT WILL SEND A SIGNAL BACK    */
/* TO CALLING PROCEDURE "NOT ASSIGNABLE".                    */
   JUST_ASSIGN: PROCEDURE (I,J,K);
/*                                                           */
/*************************************************************/
        ASSIGNABLE ='1'B;
        THREEX (I,J,K) = THREEX(I,J,K) + AMOUNT;
        ROW_SUM (J,K) = ROW_SUM (J,K) - AMOUNT;
        COLUMN_SUM(I,K) = CCLUMN_SUM (I,K) - AMOUNT;
        SLICE_SUM (I,J) = SLICE_SUM (I,J) - AMOUNT;
             CURRENT_TRANSACTION (I,J) =
                  CURRENT_TRANSACTION + AMOUNT;
        IF ROW_SUM (J,K) < 0 |     COLUMN_SUM (I,K) < 0
                | SLICE_SUM (I,J) <0 THEN DO;
             ASSIGNABLE ='0'B;
             GO TO JUST_ASSIGN_END;
        END;
   JUST_ASSIGN_END: END;
/*************************************************************/
/*                                                           */
/* THIS PROCEDURE WILL FIND THE CELLS SUCH THAT IT IS THE*/
/* ONLY CELL LEFT TO BE UNASSIGNED IN A PARTICULAR ROW,  */
/* COLUMN AND SLICE.                                         */
   FINAL_ASSIGN: PROCEDURE;
/*                                                           */
/*************************************************************/
PUT SKIP;
PUT LIST('FINAL');
             ASSIGNABLE='1'B;
        SW2='1'B;
        SW3='1'B;
   A:
             K3= MK;
        SW1 = '0'B;
        DO J3 = 1 TO M;
        DO;
             ONE_SPACE_IN_ROW_REMAINED = '0'B;
             ONE_SPACE_ALREADY_FOUND = '0'B;
```

```
            DO I3 = 1 TO L;
            IF ¬ ASSIGNED_CELL (I3,J3,K3)
                  THEN DO;
                        IF ONE_SPACE_ALREADY_FOUND THEN
                           GO TO NEXT_ROW;
                        ONE_SPACE_ALREADY_FOUND = '1'B;
                        IF ¬ ONE_SPACE_IN_ROW_REMAINED
                        THEN DO;
                              ONE_SPACE_IN_ROW_REMAINED = '1'B;
                              LI=I3;
                        END;
                        ELSE ONE_SPACE_IN_ROW_REMAINED = '0'B;
                  END;
            END;
            IF ONE_SPACE_IN_ROW_REMAINED
            THEN DO;
                  ASSIGNED_CELL (LI,J3,K3) = '1'B;
                  AMOUNT =ROW_SUM(J3,K3);
                  CALL JUST_ASSIGN (LI,J3,K3);
                  IF ¬ ASSIGNABLE THEN GO TO FINAL_ASSIGN_END;
                  SW1='1'B;
            END;
NEXT_ROW:   END;
      END;
      IF ¬ SW1 & ¬ SW2 & ¬ SW3 THEN GO TO FINAL_ASSIGN_END;
      SW2 = '0'B;
      DO I3= 1 TO L;
      DO;
            ONE_SPACE_ALREADY_FOUND = '0'B;
            ONE_SPACE_IN_COLUMN_REMAINED = '0'B;
            DO J3= 1 TO M;
            IF ¬ ASSIGNED_CELL (I3,J3,K3)
            THEN DO;
                  IF ONE_SPACE_ALREADY_FOUND THEN
                      GO TO NEXT_COLUMN;
                  ONE_SPACE_ALREADY_FOUND = '1'B;
                  IF ¬ ONE_SPACE_IN_COLUMN_REMAINED
                  THEN DO;
                        ONE_SPACE_IN_COLUMN_REMAINED = '1'B;
                        LJ=J3;
                  END;
            ELSE ONE_SPACE_IN_COLUMN_REMAINED = '0'B;
            END;
            END;
            IF ONE_SPACE_IN_COLUMN_REMAINED
            THEN DO;
                  ASSIGNED_CELL (I3,LJ,K3) = '1'B;
                  AMOUNT = COLUMN_SUM (I3,K3);
                  CALL JUST_ASSIGN (I3,LJ,K3);
                  IF ¬ ASSIGNABLE THEN GO TO FINAL_ASSIGN_END;
                  SW2='1'B;
```

```
            END;
   NEXT_COLUMN: END;
        END;
        IF ¬ SW1 & ¬ SW2 & ¬ SW3 THEN GO TO FINAL_ASSIGN_END;
        SW3 = '0'B;
        IF ¬ SW1 & ¬ SW2 & ¬ SW3 THEN GO TO FINAL_ASSIGN_END;
        GO TO A;
   FINAL_ASSIGN_END: END;
/*************************************************************/
/*                                                           */
/* THIS PROCEDURE SET  A FLAG "ASSIGNED_CELL" TO DESIGNA-*/
/* TE THAT VALUE IS ALREADY ASSIGNED TO THAT CELL, OR    */
/* THE CELL HAS BEEN DECIDED AS A NONBASIC VARIABLE.     */
   ZERO_SUM: PROCEDURE;
/*                                                           */
/*************************************************************/
ASSIGNABLE ='1'B;
        K1 = MK;
        DO J1 = 1 TO M;
        DO;
             IF ROW_SUM (J1,K1) = 0 THEN
             DO;
                  DO I1= 1 TO L;
                       ASSIGNED_CELL (I1,J1,K1) = '1'B;
                  END;
                       CALL FINAL_ASSIGN;
                       IF ¬ ASSIGNABLE THEN GO TO ZERO_SUM_END;
             END;
        END;
        END;
        DO I1 = 1 TO L;
        DO;
             IF COLUMN_SUM (I1,K1) = 0 THEN
             DO;
                  DO J1= 1 TO M;
                       ASSIGNED_CELL (I1,J1,K1) = '1'B;
                  END;
                       CALL FINAL_ASSIGN;
                       IF ¬ ASSIGNABLE THEN GO TO ZERO_SUM_END;
             END;
        END;
        END;
        DO I1 = 1 TO L;
        DO J1= 1 TO M;
             IF SLICE_SUM (I1,J1) = 0 THEN
             DO;
                  DO K1= 1 TO N;
                       ASSIGNED_CELL (I1,J1,K1) = '1'B;
                  END;
                       CALL FINAL_ASSIGN;
                       IF NOT ASSIGNABLE THEN
```

```
                        GO TO ZERO_SUM_END;
                END;
        END;
        END;
   ZERO_SUM_END: END;
/*****************************************************/
/*                                                   */
/* THIS PROCEDURE ASSIGN LOWER LIMIT VALUES TO CELLS.  */
   LIMIT_ASSIGN: PROCEDURE (I,J,K);
/*                                                   */
/*****************************************************/
PUT SKIP;
PUT LIST ('LIMIT_ASSIGN');
        ASSIGNABLE = '1'B;
        THREEX (I,J,K) = THREEX (I,J,K) + AMOUNT;
        ROW_SUM (J,K) = ROW_SUM (J,K) - AMOUNT;
        COLUMN_SUM (I,K) = COLUMN_SUM (I,K) - AMOUNT;
        SLICE_SUM (I,J) = SLICE_SUM (I,J) - AMOUNT;
        IF ROW_SUM (I,J) < 0 | COLUMN_SUM (I,K) <0 |
             SLICE_SUM (I,J) < 0 THEN ASSIGNABLE  = '0'B;
   END;
/*****************************************************/
/*                                                   */
/* THIS  RECURSIVE PROCEDURE  WILL GO BACK TO (K-1)TH  */
/* SLICE IN CASE THERE IS NO SOLUTION IN KTH SLICE.    */
/* IP,JP ARE STACKS WHICH CONTAINS THE HISTORY OF ASSIGN-*/
/* MENT.  IN OTHER WORDS IP,JP CONTAINS WHICH CELL IN  */
/* (K-1)TH SLICE WAS FIRSTLY ASSIGNED.  WE KNOW THAT   */
/* ASSIGNMENT CELL.  BECAUSE IT CAUSED NO SOLUTION IN  */
/*K TH SLICE.                                          */
/* ALSO, THERE ARE MANY CLEARING WORK SUCH AS RESETTING */
/* ROW_SUM,COLUMN_SUM AND SLICE_SUM TO K-1 SLICE STAGE.  */
/*                                                   */
   NEXT_TRY: PROCEDURE (NO) RECURSIVE;
/*                                                   */
/*****************************************************/
        DECLARE I FIXED;
        DECLARE J FIXED;
        DECLARE K FIXED;
        DECLARE ROW (M) FIXED;
        DECLARE COLUMN (L) FIXED;
        DECLARE SLICE (L,M) FIXED;
PUT LIST('NEXT_TRY');
PUT SKIP;
        RESTART_SLICE = NO -1;
        NO_1 = NO - 1;
        IF NO_1 = 0 THEN DO; PUT LIST('NO GOOD');
                  GO TO DEAD;
                     END;
        PUT EDIT ('N=',NO_1)(A,F(1));
        PROHIBIT_CELL (IP(NO_1),JP(NO_1),NO_1) ='1'B;
```

```
DO I= 1 TO L;
COLUMN (I) = 0;
END;
DO J= 1 TO M;
ROW (J) = 0;
END;
DO I= 1 TO L;
DO J= 1 TO M;
SLICE (I,J) = 0;
END;
END;
DO J= 1 TO M;
DO I= 1 TO L;
ROW (J) = ROW (J) + THREEX (I,J,NO_1);
END;
END;
DO I= 1 TO L;
DO J= 1 TO M;
COLUMN (I) = COLUMN (I) + THREEX (I,J,NO_1)
END;
END;
DO I= 1 TO L;
DO J= 1 TO M;
SLICE (I,J) = SLICE (I,J) + THREEX (I,J,NO_1);
     SLICE(I,J) = SLICE (I,J) + SLICE_SUM (I,J);
END;
END;
DO I= 1 TO L;
     IF COLUMN (I) = 0 THEN DO;
     DO J= 1 TO M;
          PROHIBIT_CELL (I,J,NO_1) = '1'B;
     END;
END;
END;
DO J= 1 TO M;
    IF ROW (J) = 0 THEN
     DO;
      DO I= 1 TO L;
           PROHIBIT_CELL (I,J,NO_1) = '1'B;
     END;
    END;
END;
DO I= 1 TO L;
DO J= 1 TO M;
        PROHIBIT_CELL(I,J,NO_1) = '1'B;
     IF SLICE (I,J) = 0 THEN
END;
END;
DO I= 1 TO L;
DO J = 1 TO M;
     IF ¬ PROHIBIT_CELL (I,J,NO_1) THEN GO TO PEND;
```

```
          END;
          END;
          CALL NEXT_TRY (NO_1);
     PEND:
          DO K = NO_1 TO N-1;
          DO I= 1 TO L;
          DO J= 1 TO M;
               ROW_SUM (J,K) = ROW_SUM (J,K) + THREEX (I,J,K);
               COLUMN_SUM (I,K) = COLUMN_SUM (I,K) + THREEX (I,J,K);
               SLICE_SUM (I,J) = SLICE_SUM (I,J) + THREEX (I,J,K);
               THREEX(I,J,K) =0;
               ASSIGNED_CELL (I,J,K) = '0'B;
          END;
          END;
          END;
          DO K= NO TO N-1;
          DO I= 1 TO L;
          DO J= 1 TO M;
               PROHIBIT_CELL (I,J,K) = '0'B;
          END;
          END;
          END;
     END;
          DO I26= 1 TO 10;
 /* ZERO CLEAR OF THREEX */
 DO I=1 TO L;
     DO J= 1 TO M;
          DO K=1 TO N;
               THREEX(I,J,K)=0;
 END; END; END;
/* ASSIGN VALUE */
       DO I= 1 TO L;
       DO J=1 TO M;
       DO K=1 TO N;
       CALL RANDU (IX,IY,YFL);
       VALUE= 10 * YFL;
       THREE (I,J,K) = VALUE;
       CALL RANDU (IY,IX,YFL);
       VALUE= YFL * 10;
       COST (I,J,K) = VALUE;
END; END; END;
  GO TO COSTY;
IF I26 ¬=10 THEN GO TO TEND;
/* SET ROW SUM,COLUMN SUM AND SLICE SUM TO ZERO */
       DO J=1 TO M;
       DO K=1 TO N;
       ROW_SUM(J,K) =0;
 END; END;
       DO I= 1 TO L;
       DO K=1 TO N;
       COLUMN_SUM (I,K) =0;
```

```
 END; END;
      DO I=1 TO L;
      DO J=1 TO M;
      SLICE_SUM (I,J) = 0;
 END; END;
/* ROW SUM */
      DO J=1 TO M;
      DO K=1 TO N;
      DO I=1 TO L;
      ROW_SUM(J,K)= THREE(I,J,K) + ROW_SUM(J,K);
END; END; END;
/* COLUMN SUM */
      DO I=1 TO L;
      DO K=1 TO N;
      DO J=1 TO M;
      COLUMN_SUM (I,K) = THREE (I,J,K) + COLUMN_SUM (I,K);
END; END; END;
/* SLICE SUM */
      DO I=1 TO L;
      DO J=1 TO M;
      DO K=1 TO N;
      SLICE_SUM (I,J) = THREE (I,J,K) +SLICE_SUM(I,J);
END; END; END;
         PUT PAGE;
         PUT LIST ('ORIGINAL ROW_SUM');
         PUT SKIP(1);
         PUT LIST('ROW SUM ( J , K )');
         PUT SKIP(3);
         CALL PRINT_TABLE(ROW_SUM,M,N);
         PUT PAGE;
         PUT LIST ('ORIGINAL COLUMN SUM');
         PUT SKIP (1);
         PUT LIST('COLUMN SUM ( I , K )');
         PUT SKIP(3);
         CALL PRINT_TABLE(CCLUMN_SUM,L,N);
         PUT PAGE;
         PUT LIST ('ORIGINAL SLICE SUM');
         PUT SKIP (1);
         PUT LIST('SLICE SUM ( I , J )');
         PUT SKIP(3);
         CALL PRINT_TABLE(SLICE_SUM,L,M);
   FLAG_INITIAL_SET:
         DO I=1 TO L;
         DO J= 1 TO M;
         DO K=1 TO N;
              ASSIGNED_CELL (I,J,K) = '0'B;
         END; END; END;
         ZERO =0;
         DO I= 1 TO L;
         DO J= 1 TO M;
         DO K= 1 TO N;
```

```
                PROHIBIT_CELL (I,J,K) = '0'B;
          END; END; END;
          DO KK = 1 TO N-1;
     JUMP4:
          MK=KK;
     PUT DATA ( MK );
/*************************************************************/
/*                                                           */
/* THIS IS A CHECKING STEP TO FIND OUT WHETHER K TH          */
/* SLICE HAS SOLUTION OR NOT.  IF THERE IS NOT GO TO         */
/* THE PROCEDURE NEXT_TRY.                                   */
/*                                                           */
/*************************************************************/
          DO J=1 TO M;
          TEMP=0;
          DO I=1 TO L;
               IF ASSIGNED_CELL(I,J,MK) THEN GO TO JUMP1;
               TEMP= TEMP + COLUMN_SUM (I,MK);
          JUMP1: END;
          IF ROW_SUM (J,MK) > TEMP THEN GO TO GOBACK1;
          END;
/* ANOTHER CHECK */
          DO I= 1 TO L;
          TEMP= 0;
          DO J= 1 TO M;
               IF ASSIGNED_CELL (I,J,MK) THEN GO TO JUMP2;
               TEMP = TEMP + ROW_SUM (J,MK);
          JUMP2: END;
          IF COLUMN_SUM (I,MK) > TEMP THEN GO TO GOBACK1;
          END;
/* SLICE CHECK OK */
          GO TO JUMP3;
     GOBACK1:
          MK2=MK;
          CALL NEXT_TRY(MK2);
          KK= RESTART_SLICE;
          GO TO JUMP4;
     JUMP3:
     JUMP5:
               SW5='0'B;
/*************************************************************/
/*                                                           */
/* IN THIS PROGRAM DANGER CELL IS AFFECTING NONE.            */
/*                                                           */
/*************************************************************/
/* DANGER CELL FLAG RESET */
          DO I4 = 1 TO L;
          DO J4= 1 TO M;
          DO K5 = MK TO N;
               ASSIGNED_CELL (I4,J4,K5) = '0'B;
          END;
```

```
                CURRENT_TRANSACTION (I4,J4)=0;
                DANGER_CELL (I4,J4) = '0'B;
                IF MIN (ROW_SUM(J4,MK),COLUMN_SUM(I4,MK),
                        SLICE_SUM (I4,J4)) = SLICE_SUM (I4,J4)
                THEN DANGER_CELL (I4,J4)='1'B;
           END; END;
/*************************************************************/
/*                                                           */
/* THIS PART OF PROGRAM IS FINDING LOWER LIMIT VALUES        */
/*                                                           */
/*************************************************************/
           DO K8 = MK TO N-1;
           DO I=1 TO L;
           DO J=1 TO M;
                TEMP=0;
           DO K=1 TO N;
                IF K= K8 THEN GO TO END5;
                TEMP=TEMP + MIN (ROW_SUM(J,K),COLUMN_SUM (I,K),
                                 SLICE_SUM (I,J));
      END5: END;
/* CALCULATE LOWER LIMIT OF SLICE MK */
                LOWER_LIMIT = SLICE_SUM (I,J) - TEMP;
/* ASSIGN AMOUNT TO THE CELL WHICH HAS LOWER_LIMIT >0 */
                IF LOWER_LIMIT > 0 THEN
                DO ;
                     AMOUNT = LOWER_LIMIT;
                     MM= K8;
                CALL LIMIT_ASSIGN (I,J,MM);
                IF ¬ ASSIGNABLE THEN GO TO T;
                END;
           END;
           END;
           END;
           CALL ZERO_SUM;
                IF ¬ ASSIGNABLE THEN GO TO T;
           CALL FINAL_ASSIGN;
                IF ¬ ASSIGNABLE THEN GO TO T;
/*************************************************************/
/* THIS PART IS FINDING SEQUENCE OF ASSIGNMENT BY            */
/* "MATRIX MINIMA".                                          */
/*                                                           */
/*************************************************************/
MINIMUM_COST_METHOD:
           K=MK;
         MINIMUM_COST=9999;
         ALL_DONE = '1'B;
            DO I=1 TO L;
            DO J= 1 TO M;
           IF PROHIBIT_CELL (I,J,MK) & ¬ SW5 THEN GO TO END6;
           DO;
                IF MINIMUM_COST > COST (I,J,K)
```

```
                    & ¬ ASSIGNED_CELL (I,J,K) THEN
                 DO ;
                    IF ROW_SUM (J,K) > 0 & COLUMN_SUM (I,K) > 0
                       & SLICE_SUM (I,J) > 0
                    THEN DO ;
         ALL_DONE='0'B;
                          MINIMUM_COST = COST (I,J,K) ;
                          MI =I;    MJ=J;
                       END;
                 END;
          END;
   END6:
          END;
          END;
       IF ¬ ALL_DONE THEN
          DO;
         MIJK= MIN (ROW_SUM (MJ,MK),COLUMN_SUM (MI,MK),
                    SLICE_SUM (MI,MJ));
         AMOUNT = MIJK;
              MM=MK;
              CALL JUST_ASSIGN (MI,MJ,MM);
              IF ¬ ASSIGNABLE THEN GO TO T;
              CALL FINAL_ASSIGN;
              IF ¬ ASSIGNABLE THEN GO TO T;
              CALL ZERO_SUM;
              IF ¬ ASSIGNABLE THEN GO TO T;
              IF SW5 THEN GO TO END7;
              MI1= MI;
              MJ1 = MJ;
              SW5 = '1'B;
/*                                                              */
/* ASSIGN MIN_SUM TO CELL (MI,MJ,MK)                            */
/*                                                              */
/* CHECK SUCCEEDED OR NO GOOD                                   */
   END7:
              GO TO MINIMUM_COST_METHOD;
         END;
              IP(MK) = MI1;
              JP(MK) = MJ1;
          END;
         DO J= 1 TO M;
         DO K= 1 TO N-1;
         IF ROW_SUM (J,K) > 0 THEN GO TO LIVE;
         END;
         END;
         GO TO ALL_COMPLETED;
LIVE:
              MK2 = MK;
              CALL NEXT_TRY (MK2);
              KK= RESTART_SLICE ;
              GO TO JUMP4;
```

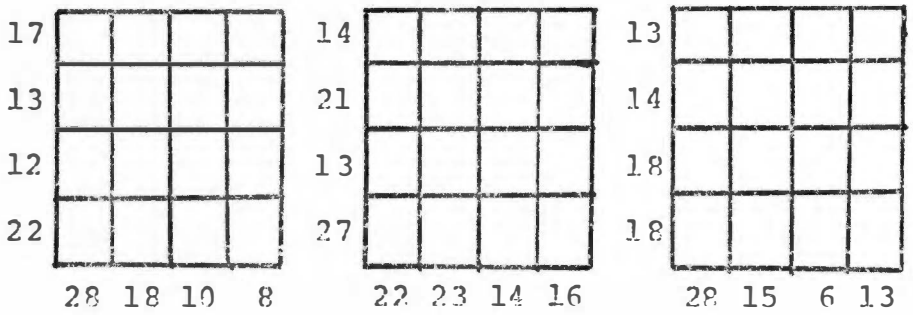
```
   ALL_COMPLETED:
        DO I=1 TO L;
        DO J=1 TO M;
             AMOUNT = SLICE_SUM(I,J);
             CALL JUST_ASSIGN (I,J,N);
        END; END;
        GO TO DEAD;
T:
/*************************************************************/
/*                                                           */
/* THIS PART WILL SET A FLAG "PROHIBIT_CELL" TO CELLS        */
/* WHICH WAS A FIRST ASSIGNMENT CELLS AND IT WAS FOUND       */
/* THAT ASSIGNMENTS SEQUENCE IS NOT PROPER.                  */
/*                                                           */
/*************************************************************/
        IF ¬ALL_DONE THEN DO;
             PROHIBIT_CELL (MI1,MJ1,MK)='1'B;
/*************************************************************/
/*                                                           */
/* THIS PART WILL DO HOUSE KEEPING TO  TRY ANOTHER           */
/* ASSIGNMENTS SEQUENCE IN K TH SLICE.                       */
/*                                                           */
/*************************************************************/
        DO I5= 1 TO L;
        DO J5 = 1 TO M;
             ROW_SUM(J5,MK) = ROW_SUM (J5,MK)
                            + CURRENT_TRANSACTION ( I5,J5);
             COLUMN_SUM (I5,MK) = COLUMN_SUM (I5,MK)
                            + CURRENT_TRANSACTION (I5,J5);
             SLICE_SUM (I5,J5) = SLICE_SUM (I5,J5)
                            + CURRENT_TRANSACTION (I5,J5);
             THREEX (I5,J5,MK) = THREEX (I5,J5,MK)
                                - CURRENT_TRANSACTION (I5,J5);
        END;
        END;
             GO TO JUMP5;
        END;
DEAD:
PUT DATA (PROHIBIT_CELL);
/*************************************************************/
/*                                                           */
/* LSISTING RESULTS                                          */
/*                                                           */
/*************************************************************/
        PUT PAGE;
        PUT LIST ('          ROW_SUM');
        PUT SKIP(1);
        PUT LIST('ROW SUM ( J , K )');
        PUT SKIP(3);
        CALL PRINT_TABLE(ROW_SUM,M,N);
        PUT PAGE;
```

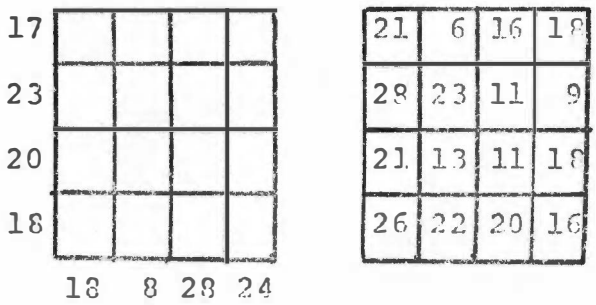
```
          PUT LIST ('          COLUMN SUM');
          PUT SKIP (1);
          PUT LIST('COLUMN SUM ( I , K )');
          PUT SKIP(3);
          CALL PRINT_TABLE(CCLUMN_SUM,L,N);
          PUT PAGE;
          PUT LIST ('          SLICE SUM');
          PUT SKIP (1);
          PUT LIST('SLICE SUM ( I , J )');
          PUT SKIP(3);
          CALL PRINT_TABLE(SLICE_SUM,L,M);
          DO K=1 TO N;
               PUT PAGE;
               DO I= 1 TO L;
               DO J=1 TO M;
                    WORK_MATRIX(I,J) =THREEX(I,J,K);
               END;
               END;
               PUT EDIT ('RESULTED MATRIX OF TYPE***',K)(A,F(1));
               PUT SKIP(2);
               CALL PRINT_TABLE(WORK_MATRIX,L,M);
          END;
TTT:
/*****************************************************************/
/*                                                               */
/* CALCULATING TOTAL SHIPPING COST .                             */
/*                                                               */
/*****************************************************************/
COSTY:
DO K=1 TO N;
      PUT PAGE;
DO I=1 TO L;
  DO J=1 TO M;
    WORK_MATRIX (I,J) = COST (I,J,K);
  END; END;
 CALL PRINT_TABLE (WORK_MATRIX,L,M);
          END;
          MONEY =0;
          DO I= 1 TO L;
          DO J= 1 TO M;
          DO K= 1 TO N;
               MONEY = MONEY + COST(I,J,K) * THREEX(I,J,K) ;
          END;
          END;
          END;
          PUT EDIT ('TOTAL COST BY MY METHOD ******',MONEY)(A,F(4));
TEND:
   END;
 END;
END;
END;
```

APPENDIX D

NUMERICAL EXAMPLES

A numerical example of initial feasible solutions obtained by LP procedure and by the modified method are shown. Also, both optimum (maximum cost and minimum cost) solutions obtained by the LP procedure are shown.





Slices before assignments.

7	5	4	4
9	6	5	8
2	3	4	5
4	4	7	9

7	0	0	3
9	4	5	0
5	5	3	6
3	8	5	8

6	0	8	8
7	3	3	1
8	5	1	4
5	4	4	1

7	4	8	4
8	2	3	7
4	8	9	4
3	1	8	6

Cost matrix

16	1		
	13		
	4		8
12		10	

			14
18	1		2
4		9	
	22	5	

5	5		3
10	4		
1	6	1	10
12		5	

		16	1
	5	11	7
16	3	1	
2			16

Total cost = 1498

Initicial feasible solution obtained by LP procedure

		10	7
	12		1
12			
16	6		

	6	6	2
	12	1	
0	13		
10	3	8	6

13			
8		6	8
9		6	3
	7		10

8			9
10	2	11	
		5	15
	6	12	

Total cost = 1273

Initial feasible solution obtained by the modified method.

11	6		
4	9		
1	3		8
12		10	

10			4
10		11	
2	1		10
	22	3	2

			13
	14		
17	1		
11		6	

		16	1
14			9
1	8	11	
3		1	14

Total cost = 1816

Optimum solution (maximum cost)

		9	8
13			
12			
3	18	1	

	6	7	1
	12		9
	5	5	3
22		2	3

13			
11	3		
4	8	6	
	4		13

8			9
4	8	11	
5			15
1		17	

Total cost = 1174

Optimum solution (minimum cost)

VITA

Mamoru Aiga was born in Yokohama, Japan, on January 17, 1947. He attended elementary schools in that city and was graduated from Tsurumi Technical High School in 1965. The following April he entered Meiji University, and in March, 1969, he received a Bachelor of Science degree in Electrical Engineering. The following April he was employed by Com-Stute-Inc.. In September , 1970, he accepted a graduate assistantship at The University of Tennessee and began study toward a Master's degree. He received the Master of Science with a major in Computer Science in June 1973.