SOLVING THE MATRIX BALANCING PROBLEM
An Algorithm Based on a Modification of the Transportation Problem

by
Julianne M. Labbiento

Submitted in partial fulfillment of the requirements
for the Degree of
Master of Science
in the
Mathematics
Program

3/11/94
Advisor Date

3/14/94
Dean of the Graduate School Date

YOUNGSTOWN STATE UNIVERSITY
March, 1994

ABSTRACT

SOLVING THE MATRIX BALANCING PROBLEM
An Algorithm Based on a Modification of the Transportation Problem

by

Julianne M. Labbiento

Master of Science in Mathematics

Youngstown State University, 1994

The classical matrix balancing problem has many applications in areas such as economics and statistics. Generally, solutions to instances of this problem are found by minimizing either a squared or entropy objective function. This paper introduces a new procedure for solving the problem derived from a weighted least absolute value objective function.

An efficient algorithm, incorporating the ideas of piecewise linear programming with the traditional transportation algorithm, is developed and presented. A simple example is illustrated and a computer program, written in FORTRAN, is given. Finally, special features of the program are discussed.

ACKNOWLEDGMENTS

I wish to express my gratitude to my thesis advisor, *Dr. Nathan P. Ritchey*, who inspired me to successfully complete this thesis.

I would also like to thank *Dr. Anita Burris* and *Dr. R. Bruce Mattingly* for their contributions and helpful suggestions on the improvement of this thesis.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Intent of Paper

This paper will introduce a new method for solving the matrix balancing problem and others of its type. The method is based on the transportation algorithm from the field of operations research. The transportation problem is a linear programming problem whose solution, with slight modifications, can be used to solve the matrix balancing problem. This modified transportation method will be developed, presented, and illustrated.

1.2 Outline of Chapters

Chapter 2 will explore the matrix balancing problem. Several applications will be presented in the areas of business and mathematics. The problem will be formally defined mathematically.

In Chapter 3, the modified transportation algorithm will be developed. Beginning with the traditional transportation problem, the primal and dual linear programs will be stated and discussed. Complementary slackness conditions and their importance in the determination of the dual solution will also be addressed. Observations will be made

regarding the matrix balancing problem. Comparisons will be drawn between these observations and the transportation problem that will eventually lead to the definition of the modified transportation problem. The associated primal and dual linear programs and complementary slackness conditions for this modified transportation problem will be developed and presented.

The modified transportation algorithm is presented in Chapter 4, where an example solving a matrix balancing problem using the algorithm is also illustrated. A version of the Northwest Corner Method for finding the initial basic feasible solution is explained and presented in algorithmic form.

Chapter 5 will discuss the implementation of the *Modified Transportation Algorithm* as a computer program written in FORTRAN. The program is given in the Appendix. Specific features of the program will be highlighted in detail in this chapter, including establishing an initial basic feasible solution using the *Minimal Cost Northwest Corner Algorithm* described in Chapter 4, finding a dual solution and verifying its feasibility, locating the unique cycle on which to pivot, and updating the current feasible solution accordingly. A method for identifying the cycle using ideas from graph theory will be discussed. Finally, topics for further research will be presented.

Chapter 2

The Matrix Balancing Problem

2.1 Applications

The matrix balancing problem is one that arises in many fields such as economics, statistics, and demography. Contingency tables are used in insurance, science, and research when probabilities must be assigned to a finite two-dimensional array of items. This m x n contingency table, while finite in size, may be extremely large. Determining the probabilities for each cell in the table by directly sampling the general population may not be feasible due to monetary, time, and other constraints.

However, deriving these probabilities from a representative sample of the population may yield results that contradict previously existing information. There may be reason to believe that recently collected probabilities are inconsistent with the existing probabilities. Perhaps the existing data is obviously outdated in comparison to the recently collected data. Or, suppose the sample of the population is not as representative as intended. Using the techniques for solving a matrix balancing problem, incorrect cell probabilities must be updated to reflect the new information [Schneider, p. 442].

Another incidence of the use of the matrix balancing problem occurs in the area of demography. Demography is the study of the size, density, and distribution of human population. A particular topic of interest in demography is interregional migration.

It is important to economists, agricultural analysts, and industrial analysts, to estimate interregional migration patterns in the United States on a regular basis. However, matrices detailing flow and other migration characteristics become available through the general census of the population only once each decade. Intermediary migration estimates must be derived from the existing data and currently known migration trends using matrix balancing techniques in order to revise out-of-date interregional migration patterns [Schneider, p.442].

2.2 Mathematical Definition

The general definition of the matrix balancing problem is defined as follows:

Given an m x n matrix P with elements p_{ij}, *and vectors* $R = [\, r_1, \ldots, r_n \,]$ *and* $S = [\, s_1, \ldots, s_m \,]$, *find an m x n matrix Q with elements* q_{ij}, *which satisfies the following conditions :*

i) $\sum_{i=1}^{m} q_{ij} = r_j, \; j=1,\ldots,n;$ *and*

ii) $\sum_{j=1}^{n} q_{ij} = s_i, \; i=1,\ldots,m$

and which is "as close as possible" to matrix P.

The phrase "as close as possible" allows the interpretation of a well-balanced matrix to vary from problem to problem. For example, one situation may require that elements along the main diagonal of the balanced matrix equal the elements along the main diagonal of the original matrix. Any deviation from this form would be interpreted as an "unbalanced" and, therefore, unacceptable matrix. The introduction of additional constraints may be necessary to achieve the "closeness" desired.

The procedures introduced in this paper will use a weighted least absolute value function to ensure closeness. By utilizing this type of function, a natural interpretation of the matrix balancing problem as a transportation problem will occur. By slightly modifying the traditional transportation algorithm, the matrix balancing problem can be solved.

Chapter 3

The Modified Transportation Problem

3.1 The Transportation Problem

Historically, the transportation problem can be stated as a shipping problem. *Given m sources and n destinations, their supply and demand requirements, and related shipping costs, how many units should be shipped from each of the sources to each of the destinations in order to minimize the total cost of shipping ?* While total supply and total demand are required to be nonnegative, they need not be equal. Dummy variables can be introduced into the problem to force equality. By allocating prohibitively high costs to all routes associated with this dummy variable, a shipment of zero units along those routes is guaranteed in the optimal solution. For simplicity, it is assumed throughout the remainder of this paper that equality of total supply and demand does hold.

The transportation problem can be expressed as a linear program. In matrix and vector form, the primal and dual linear programming problems of the dual pair are given as follows:

minimize	$\mathbf{C^TX}$	*maximize*	$\mathbf{a^Tu + b^Tv}$
subject to :	$\mathbf{Ax = a}$	*subject to :*	$\mathbf{u + v \leq C}$
	$\mathbf{Bx = b}$		**u** free
	$\mathbf{x \geq 0}$		**v** free

In the less general form, the primal linear program for the transportation problem is as follows :

$$\textit{min} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij}$$

subject to:

$$\sum_{i=1}^{m} x_{ij} = r_j, \quad j=1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = s_i, \quad i=1,\ldots,m$$

$$x_{ij} \geq 0, \qquad i=1,\ldots,m;\ j=1,\ldots,n$$

where :

c_{ij}= *cost associated with transporting 1 unit from i to j*

x_{ij}= *number of units transported from i to j*

r_j= *total supply for j*

s_i= *total demand for i*

By solving this linear program, a total shipping cost can be found. The dual of the transportation problem is then used to determine whether the current solution is optimal, or whether it can be improved upon. The dual problem is as follows :

$$max \quad \sum_{i=1}^{m} u_i s_i + \sum_{j=1}^{n} v_j r_j$$

subject to:

$$u_i + v_j \leq c_{ij}$$

$$u_i , v_j \ \ free, \quad i=1,...,m; j=1,...,n$$

where :

u_i= *dual variable associated with row i*

v_j= *dual variable associated with column j*

Duality relations exist between the two linear programs of the transportation problem. If **x** is a feasible vector for the primal linear program and **(u,v)** is a feasible vector for the dual linear program, then $\mathbf{C^TX \geq a^Tu+b^Tv}$. If one of the problems has an optimal solution, then both have optimal solutions. Further, if both have feasible vectors, then both have optimal vectors [Ecker, p.110]. In the case of the transportation problem, an optimal solution will always exist.

The complementary slackness conditions are used to find a solution for the dual of the transportation problem. These conditions are :

$$i) \quad x_{ij} (u_i + v_j - c_{ij})=0, \quad i=1,...,m; j=1,...,n$$

$$ii) \quad u_i (\sum_{j=1}^{n} x_{ij} - s_i)=0, \quad i=1,...,m$$

$$iii) \quad v_j (\sum_{i=1}^{m} x_{ij} - r_j)=0, \quad j=1,...,n$$

If the solution found for the dual problem is also feasible, then the feasible solution found for the primal problem is optimal.

The transportation problem is easily solved using the transportation algorithm and tableau. The tableau stores the supplies, demands, and per unit costs.

Algorithm : *The Transportation Algorithm [Ecker, p. 176]*

Step 1 : *Find an initial basic feasible solution* $\mathbf{x}$.

Step 2 : *For the current solution* $\mathbf{x}$ *and the current cost coefficients* c_{ij}, *find a dual vector* $(\mathbf{u},\mathbf{v})$ *such that* $u_i + v_j = c_{ij}$ *for each* (i,j) *with* x_{ij} *basic and calculate the adjusted cost coefficients* $c_{ij} - u_i - v_j$ *for each* (i,j)

Step 3 : *If each adjusted cost coefficient is nonnegative, stop; the current* $\mathbf{x}$ *is optimal.*

Otherwise, pick a position with a negative adjusted cost and find the unique loop starting at that position with all other positions in the loop being those associated with basic variables.

Step 4 : *Shift as mush as possible around the loop to obtain a new basic feasible solution* $\mathbf{x}$ *and return to step 2 with the adjusted costs as the current costs.*

The optimal solution will be unique with two exceptions. First, and least significant, is degeneracy. A second, more interesting case occurs when the sum of the dual variables for a nonbasic cell equals either the positive or negative cost coefficient. The sum of the dual variables minus the associated cost coefficient represents the marginal cost for that cell, the per-unit increase in the total cost if that cell is brought into the basis.

If the sum is zero, then bringing this cell into the basis will result in another optimal solution.

3.2 A Comparison : The Matrix Balancing Problem vs. The Transportation Problem

Consider the matrix balancing problem as a matrix equation $\Delta = P - Q$, where P is the given matrix, Q is the desired balanced matrix, and Δ is a transformation matrix. Thus, the elements of Δ are a measure of the individual error for each element in Q. Define the ij^{th} element of Δ to be $e_{ij} = p_{ij}-q_{ij}$. Since e_{ij} represents the difference between two elements, it can be either nonnegative or negative. Define e_{ij}^{+} and e_{ij}^{-} as follows :

Let

$$e_{ij}^{+} = \begin{cases} e_{ij}, \text{ if } e_{ij} \geq 0 \\ 0, \text{ otherwise} \end{cases}$$

and

$$e_{ij}^{-} = \begin{cases} -e_{ij}, \text{ if } e_{ij} \leq 0 \\ 0, \text{ otherwise} \end{cases}$$

The row and column totals for the Δ matrix indicate how much error is incurred for a particular row or column.

For each element of P, the cost of altering that element must be determined. This cost will then be associated with the elements of Δ. If the error is nonnegative, a positive cost per unit will be applied; if the error is negative, a negative cost per unit will be used. Define the costs c_{ij}^{+} and c_{ij}^{-} in this way.

Let

$$c_{ij}^{+} = \textit{the cost of altering } p_{ij} \textit{ if } e_{ij} \geq 0$$

and

$$c_{ij}^{-} = \textit{the cost of altering } p_{ij} \textit{ if } e_{ij} \leq 0$$

Viewing the matrix balancing problem in this manner allows for a natural progression to a linear programming model that closely resembles the traditional transportation problem.

3.3 The Modified Transportation Problem

Using the ideas defined in the previous section, formulate the modified transportation linear program as follows :

$$min \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}^{+}e_{ij}^{+}+c_{ij}^{-}e_{ij}^{-})$$

subject to :

$$\sum_{i=1}^{m}(e_{ij}^{+}-e_{ij}^{-})=\overline{r}_j$$

$$\sum_{j=1}^{n}(e_{ij}^{+}-e_{ij}^{-})=\overline{s}_i$$

$$e_{ij}^{+}\geq 0,\quad e_{ij}^{-}\geq 0,\quad i=1,\ldots,m;\ j=1,\ldots,n$$

The dual of this modified transportation problem is as follows :

$$max \sum_{i=1}^{m}u_i e_{ij}^{+}+\sum_{j=1}^{n}v_j e_{ij}^{-}$$

subject to :

$$c_{ij}^{-}\leq u_i+v_j\leq c_{ij}^{+}$$

$$c_{ij}^{+}\geq 0,\quad c_{ij}^{-}<0,\quad i=1,\ldots,m;\ j=1,\ldots,n$$

Again, complementary slackness conditions are used to find the dual vectors **u** and **v**. Since the modified transportation problem allows for both positive and negative values in the optimal solution, the complementary slackness conditions have also been slightly changed. These modified conditions are :

i) $e_{ij}^{+}(u_i+v_j-c_{ij}^{+})=0,$ $i=1,\ldots,m;\ j=1,\ldots,n$

ii) $e_{ij}^{-}(u_i+v_j-c_{ij}^{-})=0,$ $i=1,\ldots,m;\ j=1,\ldots,n$

iii) $u_i\left[\sum_{i=1}^{m}(e_{ij}^{+}-e_{ij}^{-})-\overline{r}_j\right]=0,$ $j=1,\ldots,n$

iv) $v_j\left[\sum_{j=1}^{n}(e_{ij}^{+}-e_{ij}^{-})-\overline{s}_i\right]=0,$ $i=1,\ldots,m$

One special characteristic of the modified transportation problem is that, unlike the transportation problem, there is no positive value constraint set on any of the general variables. That is, the row totals, column totals, costs, and ultimately, the optimal solution, can have negative values. This difference is most significant, as it allows for a wider range of problems to be solved using the basic method of the transportation algorithm. Thus, more real-life applications can be modeled using this modified version.

Chapter 4

An Illustration of the Method

4.1 The Modified Transportation Algorithm

The modified transportation problem can be solved using the transportation tableau. The modified tableau is created by recording both the positive and negative cost associated with each element of the Δ matrix in the corresponding cell of the tableau, and noting the row and column totals of Δ as usual. The two costs for each cell can be interpreted as endpoints of a closed interval. A zero cost can be interpreted as the upper (positive) or lower (negative) endpoint of the interval depending on the other cost. An initial basic feasible solution can be found using an updated version of the Northwest Corner Method [Ecker, p. 166]. This Minimal Cost Northwest Corner Method introduces the cheaper of the two options for each cell under consideration into the solution.

Algorithm : *Minimal Cost Northwest Corner Method*
for finding an initial feasible solution

Step 1 : *Consider the available cell in the most Northwest corner of the tableau. If the row or column total is positive, apply the positive cost; if it is negative,*

apply the negative cost. If the row total times its associated cost is less than the column total times its associated cost, enter the row total amount into the cell. Decrease the row total and column total by this amount. If the updated row and column total are both zero, enter a zero into the next cell to the right.

Otherwise, enter the column total amount into the cell. Decrease the row total and column total by this amount. If the updated row and column totals are both zero, enter a zero in to cell beneath the current one.

If the number of filled cells equals m+n-1, then stop.

Step 2 : *If the row amount was entered in Step1, move down one row and go to Step 1.*

Otherwise, move to the right one column and go to Step 1.

The Modified Transportation Algorithm is then used to find the optimal solution to the problem, the optimal Δ matrix.

Algorithm : *Modified Transportation Method for finding the optimal Δ matrix*

Step 1 : *Find an initial basic feasible solution* $\mathbf{e}$.

Step 2 : *For the current solution* $\mathbf{e}$ *and cost coefficients* c_{ij}^{+} *and* c_{ij}^{-}, *find a dual vector* $(\mathbf{u},\mathbf{v})$ *such that for* e_{ij} *basic :*

1) $u_i + v_j = c_{ij}^{+}$ *if* $e_{ij} = e_{ij}^{+}$, *and*

2) $u_i + v_j = c_{ij}^{-}$ *if* $e_{ij} = e_{ij}^{-}$

Step 3 : *Calculate the adjusted cost coefficients by :*

1) $u_i + v_j - c_{ij}^{+}$ *if* $u_i + v_j \geq c_{ij}^{+}$, *and*

2) $c_{ij}^{-} - u_i - v_j$ *if* $u_i + v_j \leq c_{ij}^{-}$

Step 4 : *If each adjusted cost coefficient lies in the closed interval* $\left[c_{ij}^{-}, c_{ij}^{+}\right]$, *stop; the current solution* **e** *is optimal.*

Otherwise, choose the cell with the largest absolute value cost coefficient. Find a unique loop beginning with that cell and having all basic cells as components.

Step 5 : *If a cell in the loop holds a positive value prior to the shift, it must retain a positive value after completion of the shift. Similarly, negativity of a negative cell must be maintained throughout the shift.*

If $u_i + v_j < c_{ij}^{-}$ *for the cell with the largest absolute value adjusted cost coefficient, then shift the smallest negative amount possible around the loop to obtain a new basic feasible solution,* **e**.

If $u_i + v_j > c_{ij}^{+}$ *for the cell with the largest absolute value adjusted cost coefficient, then shift the largest positive amount possible around the loop to obtain a new feasible solution* **e**.

Return to Step 2.

4.2 An Example

The following example will illustrate the Modified Transportation Method for solving the matrix balancing problem.

Given the matrix P, vectors R and S, and cost matrices C^+ and C^-, as shown, find a matrix Q that is as close as possible to P, as determined by the costs given in C^+ and C^-.

$$P = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 2 & 8 & 3 & 1 \\ 4 & 5 & 7 & 0 \end{bmatrix},$$

$$C^+ = \begin{bmatrix} 2 & 1 & 4 & 5 \\ 6 & 5 & 3 & 2 \\ 1 & 3 & 4 & 1 \end{bmatrix}, \qquad C^- = \begin{bmatrix} -3 & -4 & -2 & -1 \\ -1 & -3 & -4 & -2 \\ -4 & -2 & -6 & -5 \end{bmatrix}$$

$$R = [9 \quad 8 \quad 18 \quad 3] \quad , \qquad S = [6 \quad 19 \quad 13]$$

Using the Minimal Cost Northwest Corner Algorithm to find an initial feasible solution, the initial modified transportation tableau is as follows. Note that the costs associated with each cell are recorded in the upper right corner of the cell.

-2^{2}_{-3}	6^{1}_{-4}	$^{4}_{-2}$	$^{5}_{-1}$	4
$^{6}_{-1}$	1^{5}_{-3}	-4^{3}_{-4}	-2^{2}_{-2}	−5
$^{1}_{-4}$	$^{3}_{-2}$	$^{4}_{-6}$	3^{1}_{-5}	3
-2	7	-4	1	

Using the complementary slackness conditions, dual variables can be found. Arbitrarily, let

$u_1 = 0$. The remaining dual variables are recorded above and beside the tableau. To check feasibility of the duals, the sum of the dual variables associated with a cell must lie within the closed interval indicated by the two cost coefficients for that cell. The analysis of the dual variables for the nonbasic cells of the initial solution is given.

	$v_1 = -3$	$v_2 = 1$	$v_3 = -8$	$v_4 = -6$	
$u_1 = 0$	-2^{2}_{-3}	6^{1}_{-4}	$^{4}_{-2}$	$^{5}_{-1}$	4
$u_2 = 4$	$^{6}_{-1}$	1^{5}_{-3}	-4^{3}_{-4}	-2^{2}_{-2}	−5
$u_3 = 7$	$^{1}_{-4}$	$^{3}_{-2}$	$^{4}_{-6}$	3^{1}_{-5}	3
	-2	7	-4	1	

$$u_1 + v_3 = -8 \notin [-2,4] \qquad u_3 + v_1 = 4 \notin [-4,1]$$
$$u_1 + v_4 = -6 \notin [-1,5] \qquad u_3 + v_2 = 8 \notin [-2,3]$$
$$u_2 + v_1 = 1 \in [-1,6] \qquad u_3 + v_3 = -1 \in [-6,4]$$

The total cost of this initial feasible solution is

$$z = (-2)(-3)+(6)(1)+(1)(5)+(-4)(-4)+(-2)(-2)+(3)(1) = 40$$

But, because the dual constraint has not been satisfied for all of the dual variables, this cannot be the optimal solution. The cell that most violates the cost coefficient interval is cell (1,3). Since the sum the dual variables lies outside the interval on the negative side, a negative value should be introduced into that cell. The unique cycle about that cell is formed by cells (1,3), (2,3), (2,2), and (1,2). To add a negative value into cell (1,3), that same negative amount must be subtracted from cell (2,3), added to cell (2,2), and subtracted from cell (1,3). The smallest negative amount that can be shifted around this cycle is -1, the value found in cell (2,2). Shifting -1 units around this cycle will maintain a feasible solution for the primal problem, since the row totals and column totals are unchanged. Once the new feasible solution has been found, a new corresponding dual solution must be obtained and evaluated.

	$v_1 = -3$	$v_2 = 1$	$v_3 = -2$	$v_4 = 0$	
$u_1 = 0$	-2^{2}_{-3}	7^{1}_{-4}	-1^{4}_{-2}	$^{5}_{-1}$	4
$u_2 = -2$	$^{6}_{-1}$	$^{5}_{-3}$	-3^{3}_{-4}	-2^{2}_{-2}	−5
$u_3 = 1$	$^{1}_{-4}$	$^{3}_{-2}$	$^{4}_{-6}$	3^{1}_{-5}	3
	-2	7	-4	1	

$u_1 + v_4 = 0 \in [-1,5]$ $\quad u_3 + v_1 = -2 \in [-4,1]$

$u_2 + v_1 = -5 \notin [-1,6]$ $\quad u_3 + v_2 = 2 \in [-2,3]$

$u_2 + v_2 = -1 \in [-3,5]$ $\quad u_3 + v_3 = -1 \in [-6,4]$

z = (-2)(-3)+(7)(1)+(-1)(-2)+(-3)(-4)+(-2)(-2)+(3)(1) = 34

Updating the basic feasible solution by pivoting on cell (2,1) yields the following tableau:

	$v_1 = 1$	$v_2 = 1$	$v_3 = -2$	$v_4 = 0$	
$u_1 = 0$	$^{2}_{-3}$	7^{1}_{-4}	-3^{4}_{-2}	$^{5}_{-1}$	4
$u_2 = -2$	-2^{6}_{-1}	$^{5}_{-3}$	-1^{3}_{-4}	-2^{2}_{-2}	−5
$u_3 = 1$	$^{1}_{-4}$	$^{3}_{-2}$	$^{4}_{-6}$	3^{1}_{-5}	3
	-2	7	-4	1	

$u_1 + v_1 = 1 \in [-3,2]$ $\quad$ $u_3 + v_1 = 2 \notin [-4,1]$

$u_1 + v_4 = 0 \in [-1,5]$ $\quad$ $u_3 + v_2 = 2 \in [-2,3]$

$u_2 + v_2 = -1 \in [-3,5]$ $\quad$ $u_3 + v_3 = -1 \in [-6,4]$

$$z = (7)(1)+(-3)(-2)+(-2)(-1)+(-1)(-4)+(-2)(-2)+(3)(1) = 26$$

The constraints for the dual of the modified transportation problem are violated most for cell (3,1). Because the sum of the dual variables lies outside the closed interval on the positive side, a positive amount must be introduced into this cell. The largest positive amount that can be shifted around the cycle formed on this cell is 2. After pivoting on this cell and calculating the dual variables, analysis of those variables shows that this solution to the dual problem is feasible. Therefore, the corresponding solution to the primal problem is the optimal solution.

	$v_1 = 1$	$v_2 = 1$	$v_3 = -2$	$v_4 = 1$	
$u_1 = 0$	$\frac{2}{-3}$	$7^{1/-4}$	$-3^{4/-2}$	$\frac{5}{-1}$	4
$u_2 = -2$	$-4^{6/-1}$	$\frac{5}{-3}$	$-1^{3/-4}$	$\frac{2}{-2}$	−5
$u_3 = 0$	$-2^{1/-4}$	$\frac{3}{-2}$	$\frac{4}{-6}$	$1^{1/-5}$	3
	-2	7	-4	1	

$u_1 + v_1 = 1 \in [-3,2]$ $\quad$ $u_2 + v_4 = -1 \in [-2,2]$

$u_1 + v_4 = 1 \in [-1,5]$ $\quad$ $u_3 + v_2 = 1 \in [-2,3]$

$u_2 + v_2 = -1 \in [-3,5]$ $\quad$ $u_3 + v_3 = -2 \in [-6,4]$

$$z = (7)(1)+(-3)(-2)+(-4)(-1)+(-1)(-4)+(2)(1)+(1)(1) = 24$$

Thus, the optimal Δ matrix and balanced matrix Q are :

$$\Delta = \begin{bmatrix} 0 & 7 & -3 & 0 \\ -4 & 0 & -1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & -5 & 7 & 3 \\ 6 & 8 & 4 & 1 \\ 2 & 5 & 7 & -1 \end{bmatrix}$$

Chapter 5

Computer Implementation of the Method

5.1 Motivation

Modifying the transportation problem to allow for negativity of variables and dual costs for one entity introduces a new method for solving many problems of today. While the previous example illustrated the ease of solving a simple 3 x 4 matrix balancing problem using this method, obviously one would not want to solve a larger problem by manipulating the tableaus by hand.

The program found in Appendix I implements the ideas set forth in this paper. Written in FORTRAN, it allows the user to read in a file containing the dimensions of a matrix, a cost array, and supply and demand requirements. Using the method of the modified transportation problem, an optimal solution is found and sent to a file. No previous knowledge of the method is required.

5.2 Overview of the FORTRAN Program

The FORTRAN program, MTRANS, performs the basic procedure for solving the modified transportation problem as described in Chapters 3 and 4. Upon reading the data from the input file, a test to check the equality of the row totals and column totals is performed. If equality does not exist, a dummy row or column is added to the problem. Cells in this dummy row or column have positive cost coefficient 199998 and negative cost coefficient -199998 thereby guaranteeing that they will not be represented in the optimal solution. The numbers 199998 and -199998 were chosen to represent extremely large, and most likely prohibitive, costs.

Establishing an Initial Basic Feasible Solution

The initial feasible solution is found using the Minimal Cost Northwest Corner Algorithm. Consider cell (1,1). If the row total for row 1 is positive, multiply it by the positive cost coefficient for that cell; if it is negative, apply the negative cost coefficient. Apply costs to the column total for column 1 in the same manner. Input the minimum of the two values in cell (1,1). If the row value is used, no other cells in row 1 will be able to

be filled without violating the constraint that cell totals in a row sum to the row total. A flag is turned on to indicate that row 1 is full. Move down one row and consider the northwest-most available cell. Repeat the procedure. If the column value is used as input for the cell, a similar action results. Move to the right one column and repeat the procedure. Continue until m + n - 1 cells have been filled.

In the event of a tie between the row and column totals, fill the cell with the appropriate value, and place a zero in an adjoining cell. If the row total is positive, place a zero in the next cell to the right; otherwise, enter a zero in the cell directly beneath the current one. Refer to the *Minimal Cost Northwest Corner Algorithm* on page 18 for more details on this method.

In addition to the cell amount, which is recorded in the array A(i,j), another value is associated with each cell. There is a ternary decision function T applied to each cell. This function is primarily used to identify whether a cell should be interpreted as basic or nonbasic if there is a zero value in it. Initially, both A(i,j) and T(i,j) equal zero for all i and j. Then as the initial feasible solution is found,

$$T(i,j) = \begin{cases} 1, & \text{if } \{cell(i,j) \text{ is basic, } A(i,j) = 0, \text{ and } c_{ij}^{+} \text{ applies} \\ -1, & \text{if } \{cell(i,j) \text{ is basic, } A(i,j) = 0, \text{ and } c_{ij}^{-} \text{ applies} \\ 0, & \text{otherwise} \end{cases}$$

Any basic cells with A(i,j)=0 in the initial basic feasible solution will have T(i,j)=1 associated with them; that is, for this solution, the positive cost will be applied to the zeroes in those cells. The use of the T(i,j) array may not seem important at this point in the problem. However, it will be necessary to keep track of the "positivity" or "negativity" of a cell. Associating this "T-value" with each cell will allow the positivity or negativity of a cell to be known at a glance.

Finally, a parent array is created to identify a tree structure associated with the initial basic feasible solution. The use of this tree is explained in detail later when finding the pivoting cycle. The initial parent array is built as cells are filled, with row one being the root of the tree; thus, the parent of row one is zero. The basic premise used here is that if the cell in row i, column j is basic, then either i is the parent of j, or vice versa.

Finding a Dual Solution

Once an initial basic feasible solution has been established, a corresponding dual solution must be found. Arbitrarily, u_1 is set to equal zero. All other dual variables are equated to 199998. Systematically stepping through the A(i,j) array in a southeasterly direction, dual variables are calculated, if possible, at each basic cell. If only one of the dual variables associated with a cell equals 199998, then the other dual variable for the cell can be determined. If both variables equal 199998, nothing can be done; continue through the A(i,j) array to the next basic cell.

If the dual variable can be found, consider A(i,j) for the cell. If A(i,j) is positive, or if it is zero with T(i,j) =1, then the unknown dual variable is found by

$$u_i + v_j - c_{ij}^{+} = 0.$$

Recall that if A(i,j)=0 and T(i,j) $\neq$ 0, then the cell is basic. If A(i,j) is negative, or if A(i,j)=0 with T(i,j) = -1, then calculate the required dual variable using

$$c_{ij}^{-} - u_i - v_j = 0.$$

If all of the dual variables have not been calculated once A(m,n) has been checked, return to A(1,1) and systematically move through the array again. Update dual variables wherever possible. Continue this procedure until a dual solution has been determined.

Checking the Feasibility of the Dual Solution

After a dual solution has been found, it must be checked for feasibility. Recall that the dual solution is feasible if for all i and j,

$$c_{ij}^{-} \leq u_i + v_j \leq c_{ij}^{+}.$$

If the sum of the dual variables lies within the closed interval determined by the cost coefficients for the cell, a counter **opt** is increased by one. After checking each cell, if opt = m x n, the dual solution is feasible and, therefore, the primal solution is optimal. This optimal solution is written to a file, along with the optimal total cost.

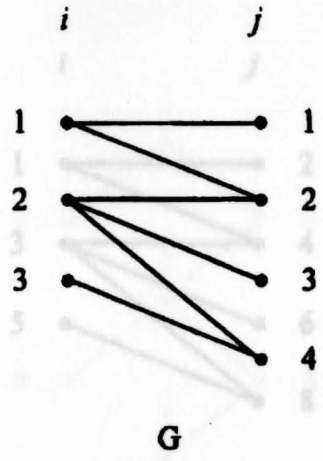
If, however, the number of optimal cells is not equal to m x n, the pivot cell must be located. Running through the non-optimal cells, find the cell whose dual variables produced the largest deviation from the interval. If the largest deviation occurs when

$$u_i + v_j - c_{ij}^{+} \geq 0,$$

a switch called **pos** is set to 1. This indicates that a positive value will enter the basis in this cell. If the largest difference occurs on the negative side of the interval, **neg** is set equal to 1 and a negative value will enter the basis in this cell. Record the pivot cell as A(i1,j1).

Locating the Cycle

One interesting feature of this program is the way the cycle about the pivot cell is found. All feasible solutions for the primal problem can be interpreted as a matching on a bipartite graph, G (V_1,V_2,E); let V_1={ i | i = 1,...,m }, V_2 = { j | j = 1,...,n }, and E={(i,j)| cell(i,j) is basic}. Thus, for the example in Chapter 4, the bipartite graph associated with the initial basic feasible solution **x** is :

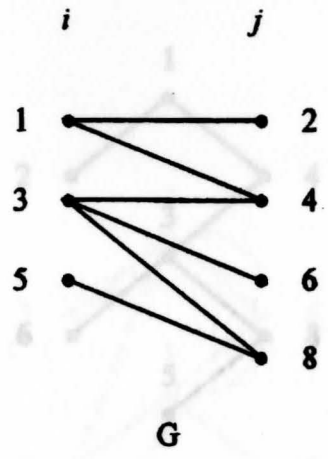


G

A simple function allows the rows, i, to be identified by odd numbers, and the columns, j, to be identified as even numbers.

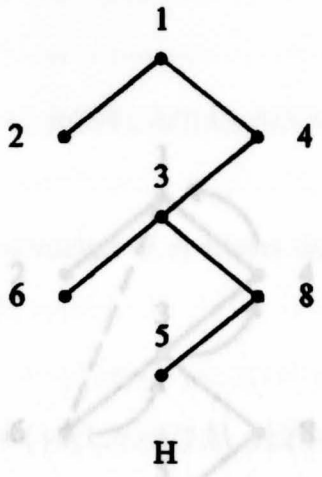
$$f(x)=\begin{cases}2x-1, \text{ if } x \text{ is a row, and}\\ 2x, \text{ if } x \text{ is a column}\end{cases}$$

The use of this function avoids confusion and misinterpretation of the rows and columns of the tableau as they are manipulated.

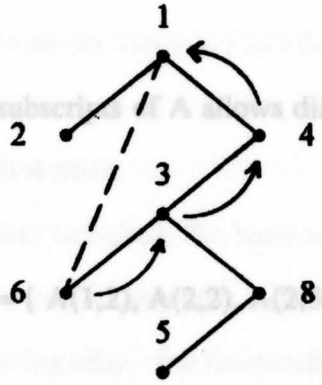


G

It is most advantageous now to restructure the bipartite graph G into the tree H. The root of this tree will be i1. For all initial basic feasible solutions i1=1. Hence, the tree H for the example is :



Recall the parent array introduced when finding the initial basic feasible solution. By maintaining this parent array throughout the program, the cycle can be found rather easily [Balinski, p. 171]. For the example, it was determined that cell(1,3) was to enter the basis on the first pivot. Using the function *f(x)* to convert the rows and columns to unique numbers, an edge should be added between nodes 1 and 6 on the tree. There is a unique path from node 1 to node 6; adding edge (1,6) would complete the cycle. To find this path, start at node 1. Step back through the parent array until either the root or node 6 is reached, recording each node visited. If node 6 is visited first, the path has been found. If the root is reached first, continue searching through the parent array, starting this time with node 6. Again, record any visited node. Once a previously visited node, Y, is reached, the unique path will be the union of the two searches minus any nodes "above" Y in the parent array. That is, any generations prior to Y are not included in the unique path.



H

$$\text{cycle} = \{\ 1,4,3,6\ \}$$

For convenience the cycle of length z is recorded as an array of nodes, with **cycr(1)**= i1 and **cycr(z)**=j1. The array cycr(k) represents a cycle of nodes. It must be converted into a cycle of edges, since edges represent basic cells in the tableau. The cycle of edges is also held in an array, **cyc(k)**, which is of length p. Recall the array cycr(k) :

$$\text{cycr(k)} = \{\ 1,4,3,6\ \}$$

The elements of cyc(k) can be found by taking consecutive pairs of elements of cycr(k), ordering them with the odd number coming first, and listing them as coordinates of the

array A. Append the edge entering the basis onto the end of the cyc(k) array as cyc(p). Hence,

$$cyc(k) = \{ A(1,4), A(3,4), A(3,6), A(1,6) \}$$

Applying $f^{-1}(x)$ to the subscripts of A allows direct reference of the cycle and the tableau.

$$cyc(k) = \{ A(1,2), A(2,2), A(2,3), A(1,3) \}$$

This tree structure requires updating each time the basic feasible solution is updated, but offers a quick way of locating the unique cycle about the pivot cell.

Updating the Cycle

When the feasibility of the dual variables was verified, it was determined whether a positive or negative value was to enter the basis in A(i1,j1). Since the pivot value will be added to even elements in the cycle and subtracted from the odd elements, the pivot value is determined by checking the odd and even cycles separately. The last element of the cycle, the entering edge, is excluded from this evaluation. The pivot value will be the largest possible amount that can be shifted around the cycle while maintaining positive values in previously positive cells, and negative values in those cells already negative.

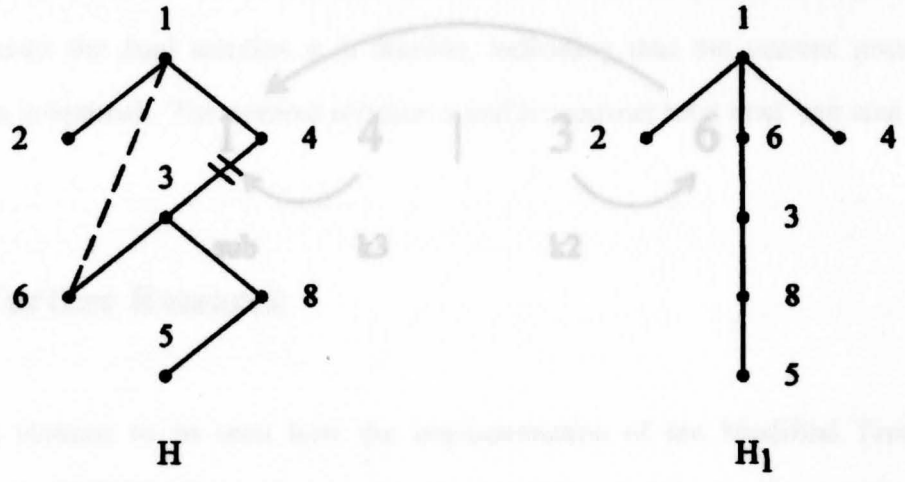
If a positive value is to be brought into the basis, a positive number will be added to even elements in the cycle. The best pivot candidate will be the largest negative value in those cells. Next, the odd cycle must be evaluated. The pivot candidate from this cycle will be the minimum of the positive values in the cells. Finally, comparing the two

candidates, the pivot value will be the absolute value of the minimum of them. This amount is shifted around the declared cycle.

If a negative number is to enter the basis, a similar comparison is made. The pivot amount will be the opposite of the absolute value of the minimum of the two candidates. Again, this amount is shifted around the cycle to form the new basic feasible solution.

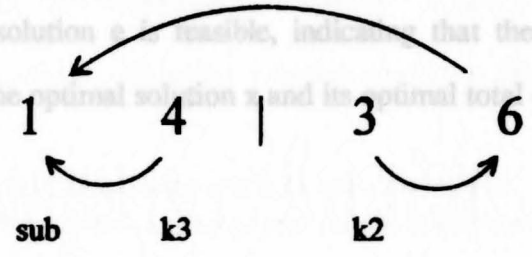
Updating the Tree Structure

Once the shift of units has occurred, the basis supports a new cell structure. The parent array must be updated to reflect this new information. Using the tree H, add the entering edge and delete the leaving edge. For the example, cell(1,3) entered the basis and cell(2,2) left; that is, edge (1,6) and edge (3,4) on the tree are updated to create tree H_1. The root of the tree will remain the same. Thus, the graph H_1 appears as follows:



Using the right and left paths found when locating the cycle, **cycr(k)** and **cycl(k)**, it is possible to find the depth of the nodes in the cycle. It must be determined whether the root of the tree is an element in the cycle. If is it not, a subroot of the cycle must be found. This subroot will be the node in the cycle that sits closest to the root of the tree. By design, the right and left paths end at the top of the tree. Thus, by checking the depth of last elements in the arrays cycr(k) and cycl(k), the subroot can be determined.

The parent array can now be updated based on the elements that changed in the cycle. Using the graph of H, let k2 designate the odd node associated with the edge that is leaving the basis. Let k3 designate the even node associated with that edge. Recall that i1 and j1 indicate the odd and even nodes of the entering edge, respectively. Finally, label the subroot of the tree as sub. Listing the nodes of the cycle, beginning with i1 and ending with j1, labeling k2, k3, and sub, indicating the exiting edge by a vertical line between the nodes, and letting an arrow indicate the parent of a node, some observations can be made :



1) The flow indicated by the arrows is interrupted only twice: at the break and at the sub.

2) Arrows only flow into the sub. This is expected, since the sub is either the root whose parent is always zero, or the sub is a subroot whose parent should remain unchanged within the cycle.

3) Arrows only flow away from the break.

4) Arrows never flow into k2 or k3, since they cannot be parents to any node in the cycle.

By considering the locations of k2, k3, and sub, a unique pattern of flow exists for each combination of locations. By determining the pattern, the parent array can be methodically updated.

Recording the Optimal Solution

The parent array has been updated to reflect the new tree structure and the new basic feasible solution. Now, the dual problem must also be checked. Repeat the entire process until the dual solution **e** is feasible, indicating that the current primal feasible solution **x** is optimal. The optimal solution **x** and its optimal total cost are sent to a file as output.

5.3 Further Research

It remains to be seen how the implementation of the Modified Transportation Problem in FORTRAN compares to other existing programs for matrix balancing problems. Pertaining to this program, improvements can be made specifically in the method used to find the initial basic feasible solution. The Minimal Cost Northwest

Corner Method, while effective, may not necessarily find the best starting solution. Vogel's Approximation Method for finding an initial basic feasible solution for the transportation problem is known to yield much better results [Ecker, p. 180]. In the future, a modified version of Vogel's Approximation Method may prove to be equally as successful in conjunction with the Modified Transportation Algorithm.

Another topic for future research lies in determining the complexity of calculating the dual variables in very large problems. The mapping of the primal basic feasible solution onto the modified transportation tableau may yield a particular pattern of basic and nonbasic cells. When this pattern occurs in large problems, the associated dual solution is rather difficult to find using the method outlined in the program. Determining the complexity of calculating the dual variables for patterns such as those may enable the program to be updated to handle such situations correctly. Alternately, studies may show that a new method for calculating the dual variables is required.

BIBLIOGRAPHY

Bacharach, Michael. *Biproportional Matrices and Input-Output Change.* London : Cambridge University Press, 1970.

Balinski, M.L., and J. Gonzalez. "Maximum Matchings in Bipartite Graphs via Strong Spanning Trees." *Networks 21* (1991). 165-179.

Ecker, Joseph G., and Michael Kupferschmid. *Introduction to Operations Research.* Malabar, Florida : Krieger Publishing Company, 1991.

Higham, Nicholas J. "Matrix Nearness Problems and Applications." *Applications of Matrix Theory* (1989) : 1-27.

Intriligator, Michael D. *Mathematical Optimization and Economic Theory.* Englewood Cliffs, New Jersey : Prentice-Hall Inc., 1971.

Kruse, Robert L. *Data Structures and Program Design.* Englewood Cliffs, New Jersey : Prentice-Hall, Inc., 1984.

Roberts, Fred S. *Applied Combinatorics.* Englewood Cliffs, New Jersey : Prentice-Hall, Inc., 1984.

Schneider, Michael H., and Stavros A. Zenios. "A Comparative Study of Algorithms for Matrix Balancing." *Operations Research 38*, no. 3 (May-June 1990) : 439-455.

Sedgewick, Robert. *Algorithms.* Reading, Massachusetts : Addison-Wesley Publishing Co., Inc., 1983.

Tarjan, Robert Endre. *Data Structures and Network Algorithms.* Philadelphia, Pennsylvania : Society for Industrial and Applied Mathematics, 1983.

Yan, Chiou-Shuang. *Introduction to Input-Output Economics.* New York : Holt, Rinehart, Winston, 1969.

APPENDIX

```
      Program MTRANS
      IMPLICIT INTEGER (A-Z)

C     This program solves matrix balancing problems using the Modified Transportation
C     Method.  Input is read from a file MTRANS*.IN and output is sent to a file
C     MTRANS.OUT.

C     The parameters of this program are :

C          PARAMETER                DESCRIPTION

C               M1          Number of rows in a normal transportation tableau
C               N1          Number of columns in a normal transportation tableau
C               XTRA(I)     Supply vector
C               Y(J)        Demand vector
C               CTRAP(I,J)  Array of positive costs associated with the matrix
C               CTRAM(I,J)  Array of negative costs associated with the matrix
C               U(I)        Dual vector associated with rows
C               V(J)        Dual vector associated with columns
C               A(I,J)      Array of cell values
C               T(I,J)      Array used to distinguish basic and nonbasic cells in certain
C                           situations
C               PARENT(K)   Array associated with the tree structure; used to find the
C                           unique cycle when pivoting
C               BOOL(K)     Boolean array used in locating the cycle
C               CYCR(K)     Intermediate arrays used in locating the cycle
C               CYCL(K)
C               CYC(K)      Array of cells appearing in the cycle
C               II(K),JJ(K) Coordinates of element CYC(K)
C               TOTCOST     Total cost of the final solution

C     To use this program to solve a problem :
C          Read in the dimensions of the matrix : M1,N1.
C          Read in the positive cost coefficients, CTRAP(I,J)
C          Read in the negative cost coefficients, CTRAM(I,J)
C          Read in the available supplies, XTRA(I)
C          Read in the market demands, Y(J)


      DIMENSION CTRAP(101,101), CTRAM(101,101)
      DIMENSION U(101), V(101), XTRA(101), Y(101)
      DIMENSION A(101,101), T(101,101), PARENT(101)
      DIMENSION C(101), R(101), MAX(I,J)
      DIMENSION CYCR(101),CYCL(101), II(101), JJ(101)
      DIMENSION CYC(101), BOOL(206)
```

```
      OPEN(6,FILE='MTRANS.OUT',STATUS='OLD')
      OPEN(5,FILE='MTRANS.IN',STATUS='OLD')
      READ(5,*) M1,N1
      DO 9992 I=1,M1
         READ(5,*)(CTRAP(I,J),J=1,N1)
9992  CONTINUE
      DO 9993 I=1,M1
         READ(5,*)(CTRAM(I,J),J=1,N1)
9993  CONTINUE
      READ(5,*)(XTRA(I),I=1,M1)
      READ(5,*)(Y(J),J=1,N1)

      INFIN1=1999998
      INFIN2=1999999
      M=M1+1
      N=N1+1
      N2=M+N-1
      N3=N2+1
      S1=0
      S2=0
      NUMPVT=0
      DO 495 I=1,M1
         CTRAP(I,N)=0
         CTRAM(I,N)=-INFIN2
495   CONTINUE
      DO 486 J=1,N
         CTRAP(M,J)=0
         CTRAM(M,J)=-INFIN2
486   CONTINUE
      DO 500 I=1,M1
         S1=S1+XTRA(I)
500   CONTINUE
      DO 518 J=1,N1
         S2=S2+Y(J)
518   CONTINUE
      IF(S1.GT.S2)GO TO 1510
      IF(S1.EQ.S2)THEN
         M=M1
         N=N1
         N2=M+N-1
         N3=N2+1
         GO TO 1511
      ENDIF
      XTRA(M) = S2 - S1
      Y(N) =0
      GO TO 1511
1510  Y(N)=S1-S2
      XTRA(M) = 0
```

```
1511  IPRINT=0

C  ***************************************************************
C  FIND STARTING SOLUTION
C  Finds a starting solution using the Minimum Cost Northwest Corner Algorithm
C  ***************************************************************
      NUMBC=0
      DO 4 K=1,M*N
         PARENT(K)=INFIN1
4     CONTINUE
      DO 5 I=1,M
         DO 6 J=1,N
            T(I,J)=0
6        CONTINUE
5     CONTINUE
      I2=0
      J2=0
      PARENT(1)=0
      ROOT=1
      DO 15 I=1,M
         IF (R(I).NE.0) GOTO 15
         DO 20 J=1,N
            IF (C(J).NE.0) GOTO 20
            IF (R(I).NE.0) GOTO 15
            IF (NUMBC.EQ.M+N-3) THEN
               J1=J
               I1=I
               GOTO 35
            ENDIF
            IF (I.EQ.M.AND.Y(J).NE.0) THEN
               A(I,J)=Y(J)
               XTRA(I)=XTRA(I)-Y(J)
               Y(J)=0
               C(J)=1
               GOTO 19
            ELSE IF (J.EQ.N.AND.XTRA(I).NE.0) THEN
               A(I,J)=XTRA(I)
               Y(J)=Y(J)-XTRA(I)
               XTRA(I)=0
               R(I)=1
               GOTO 19
            ENDIF
            IF (Y(J).LT.0.AND.XTRA(I).LT.0) THEN
               IF (Y(J).GT.XTRA(I)) THEN
                  A(I,J)=Y(J)
                  XTRA(I)=XTRA(I)-Y(J)
                  Y(J)=0
                  C(J)=1
```

```
            GOTO 19
        ELSE
            A(I,J)=XTRA(I)
            Y(J)=Y(J)-XTRA(I)
            XTRA(I)=0
            R(I)=1
            IF(Y(J).EQ.XTRA(I)) THEN
              IF(I+1.LE.M) THEN
                A(I+1,J)=0
                C(J)=1
                IF (PARENT(2*(I+1)-1).EQ.INFIN1) THEN
                    PARENT(2*(I+1)-1)=2*J
                ELSE
                    PARENT(2*J)=2*(I+1)-1
                ENDIF
                T(I+1,J)=1
                NUMBC=NUMBC+1
              ELSE
                A(I,J+1)=0
                T(I,J+1)=1
                IF (PARENT(2*(J+1)).EQ.INFIN1) THEN
                    PARENT(2*(J+1))=2*I-1
                ELSE
                    PARENT(2*I-1)=2*(J+1)
                ENDIF
                NUMBC=NUMBC+1
              ENDIF
            ENDIF
            GOTO 19
        ENDIF
    ELSE IF (Y(J).GE.0.AND.XTRA(I).LT.0) THEN
        IF (Y(J)*CTRAP(I,J).GE.XTRA(I)*CTRAM(I,J)) THEN
            A(I,J)=XTRA(I)
            Y(J)=Y(J)-XTRA(I)
            XTRA(I)=0
            R(I)=1
            GOTO 19
        ELSE
            A(I,J)=Y(J)
            XTRA(I)=XTRA(I)-Y(J)
            Y(J)=0
            C(J)=1
            GOTO 19
        ENDIF
    ELSE IF (Y(J).LT.0.AND.XTRA(I).GE.0) THEN
        IF (Y(J)*CTRAM(I,J).GE.XTRA(I)*CTRAP(I,J)) THEN
            A(I,J)=XTRA(I)
            Y(J)=Y(J)-XTRA(I)
```

```
            XTRA(I)=0
            R(I)=1
            GOTO 19
         ELSE
            A(I,J)=Y(J)
            XTRA(I)=XTRA(I)-Y(J)
            Y(J)=0
            C(J)=1
            GOTO 19
         ENDIF
      ELSE
         IF (Y(J).GE.XTRA(I)) THEN
            A(I,J)=XTRA(I)
            Y(J)=Y(J)-XTRA(I)
            XTRA(I)=0
            R(I)=1
            IF(Y(J).EQ.XTRA(I)) THEN
               IF(J+1.LE.N) THEN
                  A(I,J+1)=0
                  R(I)=1
                  C(J)=1
                  T(I,J+1)=1
                  IF (PARENT(2*(J+1)).EQ.INFIN1) THEN
                     PARENT(2*(J+1))=2*I-1
                  ELSE
                     PARENT(2*I-1)=2*(J+1)
                  ENDIF
                  NUMBC=NUMBC+1
               ELSE
                  A(I+1,J)=0
                  T(I+1,J)=1
                  C(J)=1
                  R(I)=1
                  IF (PARENT(2*(I+1)-1).EQ.INFIN1) THEN
                     PARENT(2*(I+1)-1)=2*J
                  ELSE
                     PARENT(2*J)=2*(I+1)-1
                  ENDIF
                  NUMBC=NUMBC+1
               ENDIF
            ENDIF
            GOTO 19
         ELSE
            A(I,J)=Y(J)
            XTRA(I)=XTRA(I)-Y(J)
            Y(J)=0
            C(J)=1
            GOTO 19
```

```
            ENDIF
          ENDIF
          NUMBC = NUMBC+1
          IF(I.EQ.M-1.AND.Y(J).NE.0.AND.NUMBC.NE.M+N-3) THEN
            A(I+1,J)=Y(J)
            XTRA(I+1)=XTRA(I+1)-Y(J)
            C(J)=1
            R(I)=1
          ENDIF
          IF (J.EQ.N-1.AND.XTRA(I).NE.0.AND.NUMBC.NE.M+N-3) THEN
            A(I,J+1)=XTRA(I)
            Y(J+1)=Y(J+1)-XTRA(I)
            R(I)=1
            C(J)=1
          ENDIF
19        IF (PARENT(2*I-1).EQ.INFIN1) THEN
            PARENT(2*I-1)=2*J
          ENDIF
          IF (PARENT(2*J).EQ.INFIN1) THEN
            PARENT(2*J)=2*I-1
          ENDIF
20        CONTINUE
15      CONTINUE
C     ------------------------------------
C     Place Last Two Basic Cells
C     ------------------------------------
35      IF (I1.EQ.M) THEN
          A(I1,J1)=Y(J1)
          XTRA(I1)=XTRA(I1)-Y(J1)
          Y(J1)=0
          A(I1,J1+1)=XTRA(I1)
          IF (PARENT(2*J1).EQ.INFIN1) THEN
            PARENT(2*J1)=2*I1-1
          ENDIF
          IF (PARENT(2*I1-1).EQ.INFIN1) THEN
            PARENT(2*I1-1)=2*J1
          ENDIF
          IF (PARENT(2*I1+1).EQ.INFIN1) THEN
            PARENT(2*I1+1)=2*J1
          ENDIF
          IF (PARENT(2*(J1+1)).EQ.INFIN1) THEN
            PARENT(2*(J1+1))=2*I1-1
          ENDIF
        ELSE IF (J1.EQ.N) THEN
          A(I1,J1)=XTRA(I1)
          Y(J1)=Y(J1)-XTRA(I1)
          XTRA(I1)=0
          A(I1+1,J1)=Y(J1)
```

```
            IF (PARENT(2*(J1-1)).EQ.INFIN1) THEN
               PARENT(2*(J1-1))=2*I1-1
            ENDIF
            IF (PARENT(2*J1).EQ.INFIN1) THEN
               PARENT(2*J1)=2*I1-1
            ENDIF
            IF (PARENT(2*I1+1).EQ.INFIN1) THEN
               PARENT(2*I1+1)=2*J1
            ENDIF
         ENDIF
C  *********************************************************
C  Find the Dual Variables
C  *********************************************************
200      U(1)=0
         DO 50 I=2,M
            DO 55 J=1,N
               U(I)=INFIN1
               V(J)=INFIN1
55          CONTINUE
50       CONTINUE
         DO 59 TIME=1,M+N
            DO 60 I=1,M
               DO 65 J=1,N
                  IF (U(I).NE.INFIN1.AND.V(J).NE.INFIN1) GOTO 65
                  IF (U(I).EQ.INFIN1.AND.V(J).EQ.INFIN1) GOTO 65
                  IF (A(I,J).GT.0) THEN
                     IF (U(I).EQ.INFIN1) THEN
                        U(I)=CTRAP(I,J)-V(J)
                     ELSE
                        V(J)=CTRAP(I,J)-U(I)
                     ENDIF
                  ELSE IF (A(I,J).LT.0) THEN
                     IF (U(I).EQ.INFIN1) THEN
                        U(I)=CTRAM(I,J)-V(J)
                     ELSE
                        V(J)=CTRAM(I,J)-U(I)
                     ENDIF
                  ELSE IF (A(I,J).EQ.0.AND.T(I,J).NE.0) THEN
                     IF (T(I,J).EQ.1) THEN
                        IF (U(I).EQ.INFIN1) THEN
                           U(I)=CTRAP(I,J)-V(J)
                        ELSE
                           V(J)=CTRAP(I,J)-U(I)
                        ENDIF
                     ELSE IF (T(I,J).EQ.-1) THEN
                        IF (U(I).EQ.INFIN1) THEN
                           U(I)=CTRAM(I,J)-V(J)
```

```
                    ELSE
                        V(J)=CTRAM(I,J)-U(I)
                    ENDIF
                ENDIF
            ENDIF
65          CONTINUE
60        CONTINUE
59      CONTINUE
C  *****************************************
C  Check Feasibility of Dual Variables
C  *****************************************
        OPT=0
        MX=0
        DO 80 I=1,M
          DO 85 J=1,N
            IF (CTRAM(I,J).LE.U(I)+V(J).AND.U(I)+V(J).LE.CTRAP(I,J)) THEN
              OPT=OPT+1
              GOTO 85
            ELSE IF (U(I)+V(J).LT.0) THEN
              MAX(I,J)=CTRAM(I,J)-U(I)-V(J)
              IF (MAX(I,J).GT.MX) THEN
                MX=MAX(I,J)
                I1=I
                J1=J
                POS=0
                NEG=1
              ENDIF
            ELSE IF (U(I)+V(J).GT.0) THEN
              MAX(I,J)=U(I)+V(J)-CTRAP(I,J)
              IF (MAX(I,J).GT.MX) THEN
                MX=MAX(I,J)
                I1=I
                J1=J
                POS=1
                NEG=0
              ENDIF
            ENDIF
85        CONTINUE
80      CONTINUE
        IF (OPT.EQ.M*N) then
          WRITE(6,*)'THE OPTIMAL SOLUTION IS :'
          DO 301 I=1,M
            WRITE(6,*)(A(I,J),J=1,N)
301       CONTINUE
C  ----------------------------------------
C  Calculate Total Cost
c  ----------------------------------------
```

```
      TOTCOST=0
      DO 310 I=1,M
         DO 315 J=1,N
            IF (A(I,J).LT.0) THEN
               TOTCOST=TOTCOST+A(I,J)*CTRAM(I,J)
            ELSE
               TOTCOST=TOTCOST+A(I,J)*CTRAP(I,J)
            ENDIF
315      CONTINUE
310      CONTINUE
         WRITE(6,*)'THE OPTIMAL TOTAL COST IS :', TOTCOST
999      STOP
      ENDIF

C  ***********************************************
C  Find Cycle Using Tree Structure
C  ***********************************************
      DO 861 K=1,P
         CYCR(K)=0
861   CONTINUE
      P=0
      DO 862 K=1,T2
         CYCL(K)=0
862   CONTINUE
      T2=0
      DO 86 NODE=1,M*N
         BOOL(NODE)=0
86    CONTINUE
      Z=1
      CYCR(Z)=2*I1-1
      NODE=CYCR(Z)
      BOOL(NODE)=1
87    IF (NODE.NE.2*J1) THEN
         IF (NODE.NE.ROOT) THEN
            CYCR(Z+1)=PARENT(CYCR(Z))
            NODE=CYCR(Z+1)
            BOOL(NODE)=1
            Z=Z+1
            GOTO 87
         ELSE
            P=Z
            T2=0
            GOTO 88
         ENDIF
      ELSE
         P=Z
         GOTO 89
      ENDIF
```

```
88      T1=1
        CYCL(T1)=2*J1
        NODE=CYCL(T1)
90      BOOL(NODE)=1
        CYCL(T1+1)=PARENT(CYCL(T1))
        IF (BOOL(CYCL(T1+1)).NE.1) THEN
          NODE=CYCL(T1+1)
          T1=T1+1
          GOTO 90
        ELSE
          GOTO 91
        ENDIF
91      T2=T1
        LAST=CYCL(T1+1)
        DO 98 K=1,M*N
          IF (PARENT(LAST).EQ.0) GOTO 95
          IF (BOOL(PARENT(LAST)).NE.0) THEN
            BOOL(PARENT(LAST))=0
            LAST=PARENT(LAST)
          ENDIF
98      CONTINUE
95      RTEND=Z
        DO 97 K=1,RTEND
          IF (BOOL(CYCR(K)).EQ.0) THEN
            Z=Z-1
          ENDIF
97      CONTINUE
        DO 92 K=Z+1,Z+T2
          CYCR(K)=CYCL(T1)
          T1=T1-1
92      CONTINUE
        P=Z+T2
89      DO 93 K=1,P
          IF (MOD(K,2).NE.0) THEN
            CYC(K)=A(0.5*CYCR(K)+0.5,0.5*CYCR(K+1))
            II(K)=0.5*CYCR(K)+0.5
            JJ(K)=CYCR(K+1)*0.5
          ELSE
            IF (K.EQ.P) THEN
              CYC(P)=A(I1,J1)
              II(P)=I1
              JJ(P)=J1
            ELSE
              CYC(K)=A(0.5*CYCR(K+1)+0.5,0.5*CYCR(K))
              II(K)=0.5*CYCR(K+1)+0.5
              JJ(K)=0.5*CYCR(K)
            ENDIF
          ENDIF
```

```
93      CONTINUE

C  ****************************************
C  Pivot Around the Cycle
C  ****************************************
        MIN=INFIN1
        MX=(-1)*INFIN1
        IF(POS.EQ.1) THEN
           K1=0
           DO 150 K=1,P-1,2
              IF (CYC(K).GT.0.OR.T(II(K),JJ(K)).EQ.1) THEN
                 IF (CYC(K).LE.MIN) THEN
                    MIN=CYC(K)
                    K1=K
                 ENDIF
              ENDIF
150        CONTINUE
           K2=0
           DO 151 K=2,P-2,2
              IF (CYC(K).LT.0.OR.T(II(K),JJ(K)).EQ.-1) THEN
                 IF (CYC(K).GE.MX) THEN
                    MX=CYC(K)
                    K2=K
                 ENDIF
              ENDIF
151        CONTINUE
           IF (K2.EQ.0) THEN
              CYC(P)=CYC(K1)
           ELSE IF (K1.EQ.0) THEN
              CYC(P)=(-1)*CYC(K2)
              K1=K2
           ELSE
              IF (ABS(CYC(K2)).LT.CYC(K1)) THEN
                 CYC(P)=ABS(CYC(K2))
                 K1=K2
              ELSE
                 CYC(P)=CYC(K1)
              ENDIF
           ENDIF
154     ELSE
           K1=0
           DO 152 K=1,P-1,2
              IF (CYC(K).LT.0.OR.T(II(K),JJ(K)).EQ.-1) THEN
                 IF (CYC(K).GE.MX) THEN
                    MX=CYC(K)
                    K1=K
                 ENDIF
              ENDIF
```

```
152       CONTINUE
          K2=0
          DO 153 K=2,P-2,2
            IF (CYC(K).GT.0.OR.T(II(K),JJ(K)).EQ.1) THEN
              IF (CYC(K).LE.MIN) THEN
                MIN=CYC(K)
                K2=K
              ENDIF
            ENDIF
153       CONTINUE
          IF (K2.EQ.0) THEN
            CYC(P)=CYC(K1)
          ELSE IF (K1.EQ.0) THEN
            CYC(P)=(-1)*CYC(K2)
            K1=K2
          ELSE
            IF (ABS(CYC(K1)).LT.CYC(K2)) THEN
              CYC(P)=CYC(K1)
            ELSE
              CYC(P) =(-1)*CYC(K2)
              K1=K2
            ENDIF
          ENDIF
        ENDIF
        VALUE=CYC(p)
        AMT=K1
        IF (T(II(AMT),JJ(AMT)).EQ.1) THEN
          T(II(P),JJ(P))=1
        ELSE IF (T(II(AMT),JJ(AMT)).EQ.-1) THEN
          T(II(P),JJ(P))=-1
        ENDIF
        IF (POS.EQ.1) THEN
          DO 154 K=1,P-1
            IF (CYC(K).GE.0) THEN
              T(II(K),JJ(K))=1
            ELSE
              T(II(K),JJ(K))=-1
            ENDIF
            CYC(K)=CYC(K)+(-1)**K*VALUE
154       CONTINUE
        ELSE
          DO 155 K=1,P-1
            IF (CYC(K).GE.0) THEN
              T(II(K),JJ(K))=1
            ELSE
              T(II(K),JJ(K))=-1
            ENDIF
            CYC(K)=CYC(K)+(-1)**K*VALUE
```

```
155         CONTINUE
        ENDIF
        CYC(K1)=0
        T(II(AMT),JJ(AMT))=0
        I2=II(AMT)
        J2=JJ(AMT)
        DO 160 K=1,P
            A(II(K),JJ(K))=CYC(K)
            IF (A(II(K),JJ(K)).NE.0) THEN
                T(II(K),JJ(K))=0
            ENDIF
160     CONTINUE

C   ************************************
C   Update Tree Structure
C   ************************************
        RTCT=0
        DO 1541 K=1,P
            IF (CYCR(K).EQ.2*II(K1)-1) THEN
                K2=K
            ELSE IF (CYCR(K).EQ.2*JJ(K1)) THEN
                K3=K
            ENDIF
1541    CONTINUE
C   ------------------------------------
C   Check if Root is in Cycle
C   ------------------------------------
        RTCOUNT=0
        DO 1543 COUNT=1,P
            IF (CYCR(COUNT).EQ.ROOT) THEN
                RTCOUNT=COUNT
            ENDIF
1543    CONTINUE
        IF (RTCOUNT.NE.0) THEN
            SUBROOT=ROOT
            SUB=RTCOUNT
        ELSE
            IF (T2.EQ.0) THEN
                SUBROOT=CYCR(Z)
                SUB=Z
            ELSE IF (Z.EQ.0) THEN
                SUBROOT=CYCL(T2)
                SUB=1
            ELSE
                TOPRT=CYCR(Z)
                TOPLT=CYCL(T2)
                DEPTHRT=0
                DEPTHLT=0
```

```
910         IF (PARENT(TOPRT).NE.ROOT) THEN
               DEPTHRT=DEPTHRT+1
               TOPRT=PARENT(TOPRT)
               GOTO 910
            ENDIF
911         IF (PARENT(TOPLT).NE.ROOT) THEN
               DEPTHLT=DEPTHLT+1
               TOPLT=PARENT(TOPLT)
               GOTO 911
            ENDIF
            IF (DEPTHRT.LT.DEPTHLT) THEN
               SUBROOT=CYCR(Z)
               SUB=Z
            ELSE
               SUBROOT=CYCL(T2)
               SUB=Z+1
            ENDIF
         ENDIF
      ENDIF
      IF (K2.EQ.1) THEN
         IF (SUB.EQ.K2) THEN
            DO 1561 NUM=K3,P-1
               PARENT(CYCR(NUM))=CYCR(NUM+1)
1561        CONTINUE
            PARENT(CYCR(P))=CYCR(1)
         ELSE
            DO 1562 NUM=K3,SUB-1
               PARENT(CYCR(NUM))=CYCR(NUM+1)
1562        CONTINUE
            DO 1563 NUM=SUB+1,P
               PARENT(CYCR(NUM))=CYCR(NUM-1)
1563        CONTINUE
            PARENT(CYCR(1))=CYCR(P)
         ENDIF
      ELSE IF (K3.EQ.P) THEN
         IF (SUB.EQ.K3) THEN
            DO 1564 NUM=2,K2
               PARENT(CYCR(NUM))=CYCR(NUM-1)
1564        CONTINUE
            PARENT(CYCR(1))=CYCR(P)
         ELSE
            DO 1565 NUM=1,SUB-1
               PARENT(CYCR(NUM))=CYCR(NUM+1)
1565        CONTINUE
            DO 1566 NUM=SUB+1,K2
               PARENT(CYCR(NUM))=CYCR(NUM-1)
1566        CONTINUE
            PARENT(CYCR(P))=CYCR(1)
```

```
      ENDIF
    ELSE IF (K2.LT.K3) THEN
      IF (SUB.EQ.1) THEN
        DO 1567 NUM=2,K2
          PARENT(CYCR(NUM))=CYCR(NUM-1)
1567    CONTINUE
        DO 1568 NUM=K3,P-1
          PARENT(CYCR(NUM))=CYCR(NUM+1)
1568    CONTINUE
        PARENT(CYCR(P))=CYCR(1)
      ELSE IF (SUB.EQ.P) THEN
        DO 1569 NUM=2,K2
          PARENT(CYCR(NUM))=CYCR(NUM-1)
1569    CONTINUE
        DO 1570 NUM=K3,P-1
          PARENT(CYCR(NUM))=CYCR(NUM+1)
1570    CONTINUE
        PARENT(CYCR(1))=CYCR(P)
      ELSE IF (SUB.EQ.K2) THEN
        DO 1571 NUM=1,K2-1
          PARENT(CYCR(NUM))=CYCR(NUM+1)
1571    CONTINUE
        DO 1572 NUM=K3,P-1
          PARENT(CYCR(NUM))=CYCR(NUM+1)
1572    CONTINUE
        PARENT(CYCR(P))=CYCR(1)
      ELSE IF (SUB.EQ.K3) THEN
        DO 1573 NUM=2,K2
          PARENT(CYCR(NUM))=CYCR(NUM-1)
1573    CONTINUE
        DO 1574 NUM=K3+1,P
          PARENT(CYCR(NUM))=CYCR(NUM-1)
1574    CONTINUE
        PARENT(CYCR(1))=CYCR(P)
      ELSE IF (SUB.LT.K2) THEN
        DO 1575 NUM=1,SUB-1
          PARENT(CYCR(NUM))=CYCR(NUM+1)
1575    CONTINUE
        DO 1576 NUM=SUB+1,K2
          PARENT(CYCR(NUM))=CYCR(NUM-1)
1576    CONTINUE
        DO 1577 NUM=K3,P-1
          PARENT(CYCR(NUM))=CYCR(NUM+1)
1577    CONTINUE
        PARENT(CYCR(P))=CYCR(1)
      ELSE
        DO 1578 NUM=2,K2
          PARENT(CYCR(NUM))=CYCR(NUM-1)
```

```
1578          CONTINUE
              DO 1579 NUM=K3,SUB-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1579          CONTINUE
              DO 1580 NUM=SUB+1,P
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1580          CONTINUE
              PARENT(CYCR(1))=CYCR(P)
           ENDIF
        ELSE
           IF (SUB.EQ.1) THEN
              DO 1581 NUM=2,K3
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1581          CONTINUE
              DO 1582 NUM=K2,P-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1582          CONTINUE
              PARENT(CYCR(P))=CYCR(1)
           ELSE IF (SUB.EQ.P) THEN
              DO 1583 NUM=2,K3
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1583          CONTINUE
              DO 1584 NUM=K2,P-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1584          CONTINUE
              PARENT(CYCR(1))=CYCR(P)
           ELSE IF (SUB.EQ.K3) THEN
              DO 1585 NUM=1,K3-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1585          CONTINUE
              DO 1586 NUM=K2,P-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1586          CONTINUE
              PARENT(CYCR(P))=CYCR(1)
           ELSE IF (SUB.EQ.K2) THEN
              DO 1587 NUM=2,K3
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1587          CONTINUE
              DO 1588 NUM=K2+1,P
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1588          CONTINUE
              PARENT(CYCR(1))=CYCR(P)
           ELSE IF (SUB.LT.K3) THEN
              DO 1589 NUM=1,SUB-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1589          CONTINUE
              DO 1590 NUM=SUB+1,K3
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
```

```
1590          CONTINUE
              DO 1591 NUM=K2,P-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1591          CONTINUE
              PARENT(CYCR(P))=CYCR(1)
           ELSE
              DO 1592 NUM=2,K3
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1592          CONTINUE
              DO 1593 NUM=K2,SUB-1
                 PARENT(CYCR(NUM))=CYCR(NUM+1)
1593          CONTINUE
              DO 1594 NUM=SUB+1,P
                 PARENT(CYCR(NUM))=CYCR(NUM-1)
1594          CONTINUE
              PARENT(CYCR(1))=CYCR(P)
           ENDIF
        ENDIF

        GOTO 200

        END
```