

FIXED-CHARGE TRANSPORTATION PROBLEM:  
A GROUP THEORETIC APPROACH

A THESIS

Presented to

The Faculty of the Division of Graduate  
Studies and Research

by

Jeffery Lynn Kennington

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Industrial and Systems Engineering

Georgia Institute of Technology

March, 1973

FIXED-CHARGE TRANSPORTATION PROBLEM:

A GROUP THEORETIC APPROACH

Approved:

*1/22/73*

\_\_\_\_\_  
V. E. Unger, Chairman

*[Handwritten signature]*

\_\_\_\_\_  
D. J. Jarvis

*[Handwritten signature]*

\_\_\_\_\_  
C. M. Shetty

Date approved by Chairman: 2/26/73

## ACKNOWLEDGMENTS

I wish to express my sincere appreciation to Ed Unger who introduced me to the group theoretic approach to integer programming. His performance as a thesis advisor has exceeded my highest expectations. His example as a teacher, researcher, consultant, and advisor shall be my pattern.

Many of the ideas which appear in this thesis were first introduced to me by Warren Langley. The numerous discussions we had were invaluable in leading to the successful completion of this research.

Special thanks are also due to Drs. Fyffe, Jarvis, Shetty, and Smythe who served on the reading committee.

The author also wishes to acknowledge I. Heller who provided the original proofs to propositions 12, 13, 14, and 15.

## TABLE OF CONTENTS

|  | Page |
|--|------|
| ACKNOWLEDGMENTS . . . . .                                | ii   |
| LIST OF TABLES . . . . .                                 | v    |
| LIST OF ILLUSTRATIONS . . . . .                          |      |
| SUMMARY . . . . .  | vii  |
| Chapter  |      |
| I.    INTRODUCTION . . . . .                             | 1    |
| II.   LITERATURE SURVEY . . . . .                        | 7    |
| Heuristic Approaches                                     |      |
| Exact Approaches   |      |
| III.  A GROUP THEORETIC ALGORITHM . . . . .              | 23   |
| Introduction   |      |
| Framework for Integer Programming Algorithm              |      |
| A Group Theoretic Algorithm for the                      |      |
| Pure Integer Programming Problem                         |      |
| Network Formulation of the Relaxed                       |      |
| Fixed-Charge Transportation Problem                      |      |
| Generation of the Group Problem from the Network         |      |
| Network Characterization of an Optimal Basis             |      |
| Algorithm for Determining the Group                      |      |
| Problem Objective Function (ALG-2)                       |      |
| Simplification of Penalty Calculation                    |      |
| Group Theoretic Algorithm for the                        |      |
| Fixed-Charge Transportation Problem                      |      |
| IV.   COMPUTATIONAL EXPERIENCE . . . . .                 | 72   |
| V.   UNIMODULAR STRUCTURE IN THE GROUP PROBLEM . . . . . | 78   |
| Introduction   |      |
| Properties of Unimodular Matrices and Sets               |      |
| The Group Problem Associated with the                    |      |
| Fixed-Charge Transportation Problem                      |      |
| The Multiparametric Integer Programming Problem          |      |
| VI.  CONCLUSIONS AND RECOMMENDATIONS . . . . .           | 99   |

| Appendix                              | Page |
|---------------------------------------|------|
| A. FORTRAN CODE FOR (ALG-3) . . . . . | 102  |
| BIBLIOGRAPHY . . . . .                | 123  |
| VITA . . . . .                        | 126  |

## LIST OF TABLES

| Table |   | Page |
|-------|---|------|
| 1.    | Arc Costs for $(T^*)$ . . . . .   | 45   |
| 2.    | Summary of Test Problems. . . . .   | 75   |
| 3.    | Summary of Computational Experience . . . . .   | 76   |
| 4.    | Average Computational Times as a Function<br>of Problem Size and Cost Structure . . . . . | 77   |

## LIST OF ILLUSTRATIONS

| Figure |   | Page |
|--------|---|------|
| 1.     | General Branch-and-Bound Algorithm for Solving Integer Programming Problems . . . . . | 26   |
| 2.     | Illustration of the Up and Down Penalties . . . . .                                   | 32   |
| 3.     | Network Generated by a $3 \times 2$ Fixed-Charge Transportation Problem . . . . .     | 40   |
| 4.     | General Network Representation of (FCTP <sub>R</sub> ) . . . . .                      | 42   |
| 5.     | Network for an $m \times n$ Transportation Problem . . . . .                          | 44   |
| 6.     | Solution Tree for Example 5 . . . . .   | 71   |

## SUMMARY

A branch-and-bound algorithm for the fixed-charge transportation problem has been developed. The procedure was coded for the Univac 1108 and over 50 test problems have been solved. Further, it is shown that the group problem associated with the fixed-charge transportation problem may be viewed as a multiparametric integer programming problem having a totally unimodular constraint matrix.

## CHAPTER I

## INTRODUCTION

Since the development of the simplex method for solving linear programming problems by George Dantzig in 1947, researchers in all areas of applied mathematics have been drawn to the interesting field of mathematical programming. One particular area of mathematical programming which is, at present, of prime interest to operations researchers, industrial engineers, and applied mathematicians, is that of discrete or integer programming. The best explanation for this interest is that numerous practical problems can be formulated as integer linear programming problems. Scheduling, capital budgeting, resource allocation, and distribution problems are but a few examples of where integer programming problems arise.

One particularly interesting integer programming problem is the *fixed-charge transportation problem*. The simplicity of problem statement and difficulty of solution makes this problem of great interest to the mathematical programming theoretician. In addition, the numerous applications in the area of distribution makes practical solution techniques of considerable interest to the operations research practitioner. Our objective has been to further develop the theory for this problem with the aim of obtaining a practical solution technique which can be used by the practitioner.

The fixed-charge transportation problem can be simply stated in

terms of a distribution problem in which there are  $m$  suppliers (warehouses or factories) and  $n$  customers (destinations). Each of the  $m$  suppliers can ship to any customer at a per unit shipping cost of  $c_{ij}$  (unit cost for shipping from supplier  $i$  to customer  $j$ ) plus a fixed-cost of  $f_{ij}$  assessed for opening this route. If in reality a route from some source  $i$  to some customer  $j$  does not exist, one may set  $f_{ij}$  equal to a large number to exclude this route from consideration. Each supplier,  $i=1, \dots, m$ , has  $S_i$  units of supply and each customer,  $j=1, \dots, n$ , demands  $D_j$  units. The objective is to determine which routes to be opened and the shipment size so that the total cost of meeting demand given the supply constraints is minimized. Throughout the first four chapters of this study it has been assumed that  $\sum_i S_i = \sum_j D_j$ . If this is not the case in some problem, then an additional customer can be added with a demand of  $\sum_i S_i - \sum_j D_j$  and unit and fixed cost from all sources equal zero.

Mathematically this problem may be stated as follows:

$$(1.1) \quad \min \sum_i \sum_j (c_{ij} x_{ij} + f_{ij} \delta_{ij})$$

$$\text{st. } \sum_i x_{ij} = D_j, \quad (j=1, \dots, n)$$

$$\sum_j x_{ij} = S_i, \quad (i=1, \dots, m)$$

$$x_{ij} \geq 0, \quad (i=1, \dots, m; j=1, \dots, n)$$

and

$$\delta_{ij} = \begin{cases} 0, & \text{if } x_{ij} = 0 \\ 1, & \text{if } x_{ij} > 0, \end{cases}$$

where  $x_{ij}$  = the number of units shipped from source  $i$  to customer  $j$ .

This problem can be stated as an integer programming problem by introducing logic variables  $z_{ij}$  as follows:

$$(1.2) \quad \min \sum_i \sum_j (c_{ij} x_{ij} + f_{ij} z_{ij})$$

$$\text{st. } \sum_i x_{ij} = D_j, \quad (\text{all } j)$$

$$\sum_j x_{ij} = S_i, \quad (\text{all } i)$$

$$x_{ij} - \mu_{ij} z_{ij} + s(i,j) = 0, \quad (\text{all } i,j)$$

$$x_{ij}, s(i,j) \geq 0, \quad (\text{all } i,j)$$

$$z_{ij} \in \{0,1\}, \quad (\text{all } i,j)$$

where  $\mu_{ij} = \min(S_i, D_j)$ .

An equivalent but more useful formulation for the work which follows is

$$\begin{aligned}
(\text{FCTP}) \quad & \min \sum_i \sum_j \left[ c_{ij} x_{ij} + [f_{ij}/\mu_{ij}] y_{ij} \right] \\
& \text{st. } \sum_i x_{ij} = D_j, \quad (\text{all } j) \\
& \sum_j x_{ij} = S_i, \quad (\text{all } i) \\
& x_{ij} - y_{ij} + s(i,j) = 0, \quad (\text{all } i,j) \\
& x_{ij}, s(i,j) \geq 0, \quad (\text{all } i,j) \\
& y_{ij} \in \{0, \mu_{ij}\}, \quad (\text{all } i,j).
\end{aligned}$$

Clearly (1.2) and (FCTP) are equivalent and  $y_{ij} = \mu_{ij} z_{ij}$ . Further, a relaxed version of (FCTP) obtained by omitting the integer restriction is

$$\begin{aligned}
(\text{FCTP}_R) \quad & \min \sum_i \sum_j (c_{ij} x_{ij} + \bar{f}_{ij} y_{ij}) \\
& \text{st. } \sum_i x_{ij} = D_j, \quad (\text{all } j) \\
& \sum_j x_{ij} = S_i, \quad (\text{all } i) \\
& x_{ij} - y_{ij} + s(i,j) = 0, \quad (\text{all } i,j) \\
& x_{ij}, y_{ij}, s(i,j) \geq 0, \quad (\text{all } i,j)
\end{aligned}$$

where  $\bar{f}_{ij} = f_{ij}/\mu_{ij}$ .

The principal results of this investigation are presented in Chapters III, IV, and V. It is shown in Chapter III that by adding an intermediate node between each source-destination pair and inserting arcs from the source to the intermediate node and from the intermediate node to both the source and destination, (FCTP) can be formulated as a max flow min cost network problem with side constraints. These side constraints require that certain flows in the network assume the value of either 0 or  $\mu_{ij}$ . The group problem associated with the relaxed version of this max flow min cost problem, which can be obtained directly from the network, is used in a branch-and-bound scheme to solve the original fixed-charge transportation problem. This method has been coded in FORTRAN and tested with randomly generated problems on a Univac 1108. The computational experience is given in Chapter IV. It is shown in Chapter V that the group problem associated with (FCTP) can be viewed as a multiparametric integer programming problem having a totally unimodular constraint matrix.

The notations and conventions used in this study are now presented. Matrices are denoted by upper case Latin letters and the elements of a matrix by a corresponding lower case Latin letter with two subscripts. An upper case Latin letter enclosed by  $\{\cdot\}$  denotes a set whose elements are the columns of the corresponding matrix. Lower case Latin letters with a single subscript denote an element of the vector with the same name. The symbols  $\underline{0}$  and  $\underline{1}$  denote the zero and one column vectors respectively. The notation  $\{B\} \subset \{A\}$  implies that  $\{B\}$  is a

proper subset of  $\{A\}$ . The notation  $\{B\} \subseteq \{A\}$  implies either  $\{B\} \subset \{A\}$  or  $\{B\} = \{A\}$ . Non-negativity is expressed by  $\underline{c} \geq \underline{0}$ .  $A'$  denotes the transposition of matrix  $A$ . Scalars are denoted by lower case Greek letters.  $Z$  will denote the set of integers.

## CHAPTER II

## LITERATURE SURVEY

This chapter surveys the current state of theory and methodology available for the solution of the fixed-charge transportation problem. To facilitate discussion of the progress on this problem, the approaches have been separated into two categories as follows:

- i. Heuristic Approaches, and
- ii. Exact Approaches.

These approaches differ in that the exact approaches guarantee optimality whereas the heuristic approaches do not. The algorithm of this thesis is an exact approach.

Heuristic Approaches

One of the first heuristic approaches was presented by Balinski (2) in 1961. He observed that there exist an optimal solution to the relaxed version of (1.2) (i.e. the problem formed by ignoring the restriction  $z_{ij} \in \{0,1\}$ ) with the property that  $z_{ij} = x_{ij}/\mu_{ij}$ . This observation reduces the relaxed version of (1.2) to the following problem,

$$\begin{aligned}
 (\text{FCTP}_A) \quad & \min \sum_j \sum_i (c_{ij} + f_{ij}/\mu_{ij}) x_{ij} \\
 & \text{st. } \sum_i x_{ij} = D_j, \quad (\text{all } j)
 \end{aligned}$$

$$\sum_j x_{ij} = S_i, \quad (\text{all } i)$$

$$x_{ij} \geq 0, \quad (\text{all } i, j).$$

The problem (FCTP<sub>A</sub>) is simply a transportation problem and can be solved by any of the available algorithms. An approximate solution to (FCTP) is taken as the solution of (FCTP<sub>A</sub>). This heuristic is sensitive to the magnitude of the fixed costs with the heuristic working best when the fixed costs are small.

Kuhn and Baumol (26) worked with a large material distribution problem in the Navy supply system in which the fixed costs for shipments between points were unknown, but were thought to be approximately equal for each source destination pair. Also they made small adjustments in both the supplies and demands to reduce the system cost. They reasoned that for this case, a highly degenerate solution (i.e., use of a minimum number of arcs) was desirable. Consequently, they developed a heuristic which searched for a low unit transportation cost solution while simultaneously adjusting the supplies and demands within a prespecified limit so that a highly degenerate solution could be obtained.

There exist several heuristic methods which are simplex-based. Motivation for this approach arises from the fundamental result of Hirsch and Dantzig (22) that the solution to the general fixed charge problem (FCP) will occur at one of its finitely many extreme points.

$$\begin{aligned}
 \text{(FCP)} \quad & \min \quad z = \underline{c}'\underline{x} + \underline{f}'\underline{\sigma} \\
 & \text{st.} \quad \underline{A}\underline{x} = \underline{b} \\
 & \quad \quad \underline{x} \geq 0,
 \end{aligned}$$

and

$$\sigma_j = \begin{cases} 0 & \text{if } x_j = 0 \\ 1 & \text{if } x_j > 0. \end{cases}$$

It is also well known that  $z$  is a concave function. Consequently there may exist numerous local optima which differ from the global optima. It is this feature of the fixed charge problem which would lead one to suspect that adjacent extreme points methods would fail to give good approximate solutions to (FCP). However, Cooper and Drebes (8) have used this basic philosophy to develop a heuristic approach which appears to be fairly successful in practice. They begin with the linear programming solution and change the basis using certain heuristic rules in an attempt to obtain a better total cost. Their results indicate that their method produced optimal solutions in well over 90 per cent of the several hundred problems investigated and very close to optimal (a few per cent) in the remaining cases. Their test problems ranged in size from 50 to 450 variables. Cooper and Olson (9) extended the original work of Cooper and Drebes by adding a search procedure once a local optima was found. The basic difficulty in their work was the

determination of when to stop the search.

Denzler (10) also used an extreme point technique to obtain an approximate solution to (FCTP). His method used the simplex technique while also taking into account the fixed charges associated with the vector entering and the vector leaving the basis at each iteration. It differs from the simplex theory only in that a modified criterion is used to select the entering vector. Once an extreme point is found which is superior to any of its neighbors, the method saves this point and begins a random walk over the extreme points in an attempt to locate an extreme point superior to the best one found thus far. The algorithm terminates once a random walk has gone L steps without improving the solution.

In 1969 Robers and Cooper (31) applied an adjacent extreme point technique to the fixed-charge transportation problem. Their heuristic began with the solution given by Balinski's heuristic and searched all adjacent extreme points in an attempt to obtain a better solution. This continued until a search of the adjacent extreme points failed to yield an improved solution. They solved problems with up to five sources and 35 destinations in approximately one minute on an IBM 7072. Of the 280 test problems, they obtained the optimal solution for all but two.

Steinberg (32) presented three other possible heuristics which could be used to move along adjacent extreme points in an attempt to obtain a good solution. His results also showed that optimal or near optimal solutions could be obtained using his heuristic procedure.

Exact Approaches

In the field of linear integer programming, there are four problems of interest. The first and most restrictive is the *pure 0-1 integer programming problem*. Problem (2.1) is this type

$$\begin{aligned}
 (2.1) \quad & \text{Max: } \underline{c}'\underline{x} \\
 & \text{Subj: } \underline{A}\underline{x} \leq \underline{b} \\
 & x_i = 0 \text{ or } 1 \text{ for all } i.
 \end{aligned}$$

The second is the *mixed 0-1 integer programming problem*. Problem (2.2) is of this type.

$$\begin{aligned}
 (2.2) \quad & \text{Max: } \underline{c}'\underline{x} + \underline{d}'\underline{y} \\
 & \text{Subj: } \underline{A}\underline{x} + \underline{B}\underline{y} \leq \underline{b} \\
 & \underline{y} \geq \underline{0} \\
 & x_i = 0 \text{ or } 1 \text{ for all } i.
 \end{aligned}$$

The third is the *pure integer programming problem*. Problem (2.3) takes this form.

$$\begin{aligned}
 (2.3) \quad & \text{Max: } \underline{c}'\underline{x} \\
 & \text{Subj: } \underline{A}\underline{x} \leq \underline{b}
 \end{aligned}$$

$$\underline{x} \geq \underline{0}$$

$x_i$  is integer for all  $i$ .

The last and most general type of problem is the *mixed integer programming problem*. Problem (2.4) is this type problem.

$$(2.4) \quad \text{Max: } \underline{c}'\underline{x} + \underline{d}'\underline{y}$$

$$\text{Subj: } A\underline{x} + B\underline{y} \leq \underline{b}$$

$$\underline{x} \geq \underline{0}$$

$$\underline{y} \geq \underline{0}$$

$x_i$  is integer for all  $i$ .

For problems (2.1) through (2.4)

$\underline{x}$  is a  $n$ -component column vector

$\underline{y}$  is a  $p$ -component column vector

$\underline{b}$  is a  $m$ -component column vector

$A$  is a  $m \times n$  matrix

$B$  is a  $m \times p$  matrix

$\underline{c}$  is a  $n$ -component column vector, and

$\underline{d}$  is a  $p$ -component column vector.

Hirsh and Dantzig (22) have shown that the solution to (FCP) will lie at an extreme point. Since (FCTP) is a special case of (FCP), the result holds and the solution will lie at an extreme point of the set

$$\left[ \begin{array}{l} \sum_{i=1}^m x_{ij} = D_j, \quad (\text{all } j); \\ \sum_{j=1}^n x_{ij} = S_i, \quad (\text{all } i); \\ x_{ij} \geq 0, \quad (\text{all } i,j) \end{array} \right].$$

It is well known that the extreme points of the above constraint set are integer (see Hadley (20), p. 280). Hence, solution techniques for (2.2) and (2.3) are also applicable for (FCTP). In addition, there are five exact techniques which have been designed specifically for (FCTP).

This integer programming problem can be approached by *cutting plane techniques*. These methods begin by solving the continuous version of the (IP) problem. If the solution is integer, the problem is solved. If this is not the case, then a cutting plane (constraint) is added to the constraint set and the procedure is repeated. The various cutting plane methods differ according to the cut used. Some of the best known are those by Gomory (14,15), Balas (1), and Burdet (5,6,7).

A *branch-and-bound* technique for the pure integer programming problem could also be used for solving this problem. A technique of this type was presented by Land and Doig (27). Frank (11) points out that this method would experience difficulties when the fixed costs are large compared to the variable costs. It is conjectured that the bounds developed at each iteration would be extremely weak for this case and consequently the branch-and-bound procedure would work poorly.

Trotter (35) recently developed an extension to Geoffrion's (13)

work to solve the pure (IP) problem. This procedure when applied to the (FCTP) is severely restricted by the number of integer variables which it can handle. For example, a five source, ten destination (FCTP) will result in 100 integer variables. Hence, the number of integer variables for moderate size problems ( $mn \geq 100$ ) becomes prohibitive for this technique.

Since (FCTP) may be viewed as a mixed integer programming problem, it can be decomposed via Bender's Procedure (4) into a transportation problem and a pure 0-1 integer programming problem. The optimal solution is obtained by iterating between these two problems. It is conjectured that more information is available from the original problem than from the Bender's 0-1 problem and that one will enjoy more success by attacking the (FCTP) directly rather than using the Bender's approach.

The *group theoretic approach* to integer programming as described by Gomory (16), Hu (23), and Gorry and Shapiro (17), could be applied to the (FCTP). This procedure solves the continuous version of the (FCTP), the solution of which is used to define a group problem. The group problem is solved and yields a correction. If the correction yields a feasible solution, the problem is solved. If this is not the case, the second best solution to the group problem is obtained. If this correction yields a feasible solution, the problem is solved. If not the procedure is continued until such time as a feasible correction is obtained. The difficulty of using this approach for (FCTP) is that the group problem developed at each stage has order sufficiently large to render the problem unsolvable by the current techniques.

An exact solution procedure based on a branch and bound approach was presented by Steinberg (32) in 1970. He dealt with the problem in the following form:

$$\begin{aligned}
 (2.5) \quad & \min \quad \underline{c}'\underline{x} + \underline{d}'\underline{y} \\
 & \text{st.} \quad \underline{A}\underline{x} = \underline{b} \\
 & \quad \underline{x} \geq \underline{0} \\
 & \quad y_i = \begin{cases} 0, & \text{if } x_i = 0 \\ 1, & \text{if } x_i > 0. \end{cases}
 \end{aligned}$$

He obtained a feasible solution which gave an incumbent and an upper bound. The branches took the form  $x_i = 0$  and  $x_i > 0$ . Then linear programming was used on the reduced problem

$$\begin{aligned}
 (2.6) \quad & \min \quad \underline{c}'\underline{x} \\
 & \text{st.} \quad \underline{A}\underline{x} = \underline{b} \\
 & \quad \underline{x} \geq 0,
 \end{aligned}$$

along with a penalty associated with the fixed-charges to evaluate a node. He first attempted to fathom; if that failed he branched again in the same manner. The main objection to this procedure is that the fixed-charge information is ignored in the linear programming problem and is then used in an artificial way in an attempt to fathom. It

appears that more information is provided by the solution to the continuous version of the integer programming formulation rather than the solution to (2.6).

In 1967, Murty (30) introduced the idea of ranking extreme points to develop an algorithm for the general fixed-charge problem. Define (FCP) as follows:

$$\begin{aligned}
 \text{(FCP)} \quad & \min \quad \underline{c}'\underline{x} + \underline{f}'\underline{\delta} \\
 & \text{st.} \quad \underline{A}\underline{x} = \underline{b} \\
 & \quad \quad \underline{x} \geq \underline{0} \\
 & \text{where} \quad \delta_i = \begin{cases} 0, & \text{if } x_i = 0 \\ 1, & \text{if } x_i > 0. \end{cases}
 \end{aligned}$$

Consider the related problem

$$\begin{aligned}
 \text{(VC)} \quad & \min \quad \underline{c}'\underline{x} \\
 & \text{st.} \quad \underline{A}\underline{x} = \underline{b} \\
 & \quad \quad \underline{x} \geq \underline{0}.
 \end{aligned}$$

Suppose one could rank the extreme points of (VC) in terms of increasing cost. Denote these extreme points by  $\underline{x}_1^*, \underline{x}_2^*, \underline{x}_3^*, \dots$ . Then  $\underline{c}'\underline{x}_1^* \leq \underline{c}'\underline{x}_2^* \leq \underline{c}'\underline{x}_3^* \leq \dots$ . Let FCMIN be a lower bound on the fixed cost, and BTC be the best total cost solution found so far. Then Murty's algorithm can

be stated as follows:

*Step 1.* Solve (VC) for  $\underline{x}_1^*$ .

Use  $\underline{x}_1^*$  to obtain BTC.

*Step 2.* Determine FCMIN.

*Step 3.* Is  $\underline{c}'\underline{x}_1^* + \text{FCMIN} \geq \text{BTC}$ ?

Yes - Terminate. Solution is current BTC.

No - Go to step 4.

*Step 4.* Increment  $i$  by 1. Find  $\underline{x}_i^*$ . (This involves searching over all adjacent extreme points.) Update BTC if necessary. Go to step 3.

Since the (FCTP) terminates integer once the  $y_{ij}$ 's have been fixed, an optimal solution can be found by enumerating each of the  $2^{mn}$  potential 0-1 strings and selecting the one with the least total cost. Gray (19), using clever devices to eliminate many of the 0-1 strings, developed an algorithm for the (FCTP). The machinery used to eliminate 0-1 strings is as follows:

i. *Upper Bound on Fixed Cost.* Suppose a feasible solution for (FCTP) is available at a cost of  $T_0$ . Let ZMIN denote the optimal cost obtained when the fixed charges are set equal to zero. Then an upper bound on the fixed charges is

$$\text{FMAX} = T_0 - \text{ZMIN}.$$

Any  $\underline{y}$  with fixed cost exceeding FMAX may be ignored.

ii. *Lower Bound on Fixed Cost.* A lower bound FMIN can be found

by considering the supply constraints one at a time. Enough routes must be opened from each source to permit disposing of the supply there. Consider the sources individually to determine the cheapest set of routes out of each that can absorb the supply. These cheapest routes can be used to determine a lower bound,  $FMIN$ . Any  $\underline{y}$  with fixed cost less than  $FMIN$  may be ignored.

iii. *Basic Solution.* Since the solution to (FCTP) will lie at an extreme point, at most  $m + n - 1$  routes need be open.

iv. *Row Feasibility.* Each source must dispose of its entire supply, i.e.

$$\sum_j D_j y_{ij} \geq S_i$$

v. *Single Source.* At least one route must be open to each destination.

vi. *Column Feasibility.* The total supplies potentially available at each destination must be at least equal to the demand, i.e.

$$\sum_i S_i y_{ij} \geq D_j$$

All 0-1 strings which pass these six tests are solved as transportation problems with only the appropriate routes open.

Thompkins (33) was the first to attack the (FCTP) from a group theoretic point of view. Thompkins used the Smith Normal form of the group problem associated with the (FCTP) and showed that the group elements assumed certain properties. The main results of this study may be summarized as follows:

a. Route Capacities Equal or Unequal (i.e.  $\mu_{ij}$  constant for all  $i,j$ ). The group order equals the product of the route capacities associated with the  $z_{ij}$ 's in the optimal LP basis.

b. Route Capacities Equal.

i. The group problem can be decomposed into the direct sum of cyclic subgroups each with order equal to the route capacity.

ii. The group elements may be represented by vectors whose components are from  $\{-1,0,1\}$ .

iii. The group elements corresponding to nonbasic  $y_{ij}$ 's are null.

Thompkins used some of these properties to develop a branch and bound algorithm to solve the group problem for the smallest feasible correction. At this printing no computational results are available.

In 1972 Frank (11) developed three new algorithms for the (FCTP) which were extensions of Murty's (30) original work on ranking extreme points. These algorithms all make use of two problems related to the original fixed-charge transportation problem. The problem (2.7) is the specialization of (FCTP) which occurs when the fixed charges are all zero. Likewise, the problem (2.8) is the specialization which occurs when the unit costs are all zero.

$$(2.7) \quad \begin{aligned} \min. \quad & \sum_i \sum_j c_{ij} x_{ij} \\ \text{st.} \quad & \sum_i x_{ij} = D_j, \quad (\text{all } j) \end{aligned}$$

$$\sum_j x_{ij} = S_i, \quad (\text{all } i)$$

$$x_{ij} \geq 0, \quad (\text{all } i, j).$$

(2.8)

$$\min \sum_i \sum_j \bar{f}_{ij} y_{ij}$$

$$\text{st. } \sum_i x_{ij} = D_j, \quad (\text{all } j)$$

$$\sum_j x_{ij} = S_i, \quad (\text{all } i)$$

$$x_{ij} - y_{ij} + s(i, j) = 0, \quad (\text{all } i, j)$$

$$x_{ij}, s(i, j) \geq 0, \quad (\text{all } i, j)$$

$$y_{ij} \in \{0, \mu_{ij}\}, \quad (\text{all } i, j).$$

For the above two problems, define the following terms.

Let  $VCMIN(I)$  denote the  $i$ th best solution to (2.7),

$FCMIN(J)$  denote the  $j$ th best solution to (2.8),

$\overline{FCMIN}$  denote an approximation of  $FCMIN(1)$ , and

$BTC$  denote the best total cost solution found so far.

Using these definitions, a brief description of each of Frank's algorithms is given below.

Algorithm 1. This algorithm is identical in structure to Murty's original work except that it ranks extreme points with respect to the opposite problem. Recall that Murty's algorithm determined  $\overline{FCMIN}$  and then found  $VCMIN(1), VCMIN(2), \dots, VCMIN(I)$  until  $I$  is found such that  $\overline{FCMIN} + VCMIN(I) \geq BTC$  which terminates the algorithm. Algorithm 1 determines  $VCMIN(1)$  and finds  $FCMIN(1), FCMIN(2), \dots, FCMIN(J)$  until  $J$

is found such that  $FCMIN(J) + VCMIN(1) \geq BTC$  which terminates the algorithm. A variation of the defender algorithm (Bellmore and Ratliff (3)) has been used to obtain  $FCMIN(1), FCMIN(2), \dots, FCMIN(J)$ .

Algorithm 2. This algorithm combines the ideas of Murty's original work with algorithm 1. The procedure may be stated as follows:

*Step 1.* Set  $I = J = 1$ .

*Step 2.* Determine  $FCMIN(J)$  and  $VCMIN(I)$ . Use the solutions to determine BTC.

*Step 3.*  $FCMIN(J) + VCMIN(I) \geq BTC$ ?

Yes - Stop. Solution is given by BTC.

No - Go to step 4.

*Step 4.* Select either I or J to be increased. If I is selected, go to step 5. Otherwise go to step 6.

*Step 5.* Increment I by 1. Find  $VCMIN(I)$ . Update BTC if necessary. Return to step 3.

*Step 6.* Increment J by 1. Find  $FCMIN(J)$ . Update BTC if necessary. Return to step 3.

Algorithm 3. This procedure is essentially Murty's algorithm with  $FCMIN(1)$  substituted for  $\overline{FCMIN}$ . Also a branching mechanism is used in place of a search over adjacent extreme points in order to find  $VCMIN(1), VCMIN(2), \dots, VCMIN(I)$ .

Algorithms 2 and 3 as well as Murty's original algorithm were coded and some 120 test problems were run. Problems in size up to  $4 \times 4$  were run and it appeared that algorithm 3 was superior to the other two

with Murty's original method being superior to algorithm 2.

## CHAPTER III

## A GROUP THEORETIC ALGORITHM

Introduction

The algorithm presented in this chapter is based on a specialization of the ideas of Johnson (25) for the general integer programming problem. The specialization results from expressing the fixed-charge transportation problem as a minimal cost flow problem with side constraints. We have then drawn upon the work of Johnson (26) to obtain the group problem associated with a noninteger solution. The group problem is used to determine the cost to be incurred if the noninteger variables are forced to be integer. These costs are then used to determine the branching variable, and strengthen the fathoming criteria.

Framework for Integer Programming Algorithms

A general framework for solving integer programming (IP) problems has been developed by Geoffrion and Marsten (12). The framework is based on the following concepts:

- i. separation,
- ii. relaxation,
- iii. fathoming criteria, and
- iv. selection procedures.

Each of these concepts will be discussed, and then a rudimentary algorithm will be presented.

### A. Separation

For any optimization problem (P), let  $F(P)$  denote its set of feasible solutions.

#### DEFINITION 1

The problem (P) is said to be *separated* into subproblems  $(P_1), \dots, (P_n)$  iff (i)  $F(P) = F(P_1) \cup \dots \cup F(P_n)$  and (ii)  $F(P_i) \cap F(P_j) = \emptyset$  for all  $i, j$  with  $i \neq j$ .

### B. Relaxation

#### DEFINITION 2

The problem  $(P_R)$  is said to be a relaxation of P iff  $F(P) \subset F(P_R)$ .

### C. Fathoming Criteria

The branch-and-bound approach can be thought of as a simple divide-and-conquer strategy. The division process is accomplished by separation and the resulting problems are placed in the candidate list (CL). One then attempts to solve each of the candidate problems (CP). If this is difficult or impossible for a particular CP, this problem is separated into two new candidate problems. This process continues until either the CP can be solved or information can be obtained which eliminates this CP from further consideration.

#### DEFINITION 3

If a candidate problem (CP) can be discarded without further separation, then (CP) is said to have been *fathomed*.

There are three basic fathoming criteria which can be used to discard a (CP).

FC 1: If  $F(CP_R) = \emptyset$  then (CP) can be fathomed.

FC 2: Let INC denote the objective value of the best solution yet found. Assume all problems are to minimize and  $v(\cdot)$  denotes the optimal objective value of problem ( $\cdot$ ). If  $v(CP_R) \geq INC$ , then (CP) can be fathomed.

FC 3: Let  $\underline{x}^*$  denote the optimal solution to  $(CP_R)$ . If  $\underline{x}^* \in F(CP)$ , then (CP) can be fathomed.

#### D. Selection Procedures

At each iteration of a branch-and-bound algorithm, one must select from the candidate list (CL), a candidate problem (CP) for analysis. Different rules for selecting the next candidate problem lead to different patterns of enumeration. The two rules generally used are *Last-In-First-Out* (LIFO) and *Priority*. Under LIFO, the problem selected is always the last one that was added to the candidate list. Under the Priority Rule, the next problem selected is the one which has the highest priority index. This index is computed from a tag affixed to the candidate problems at the time of creation. One such tag would be a lower bound on the optimal solution.

A general branch-and-bound algorithm for solving integer programming problems is presented in Figure 1. Algorithms differ in the particular machinery used in boxes one through five. The machinery used in this research is based on the group problem associated with the relaxed fixed-charge transportation problem.

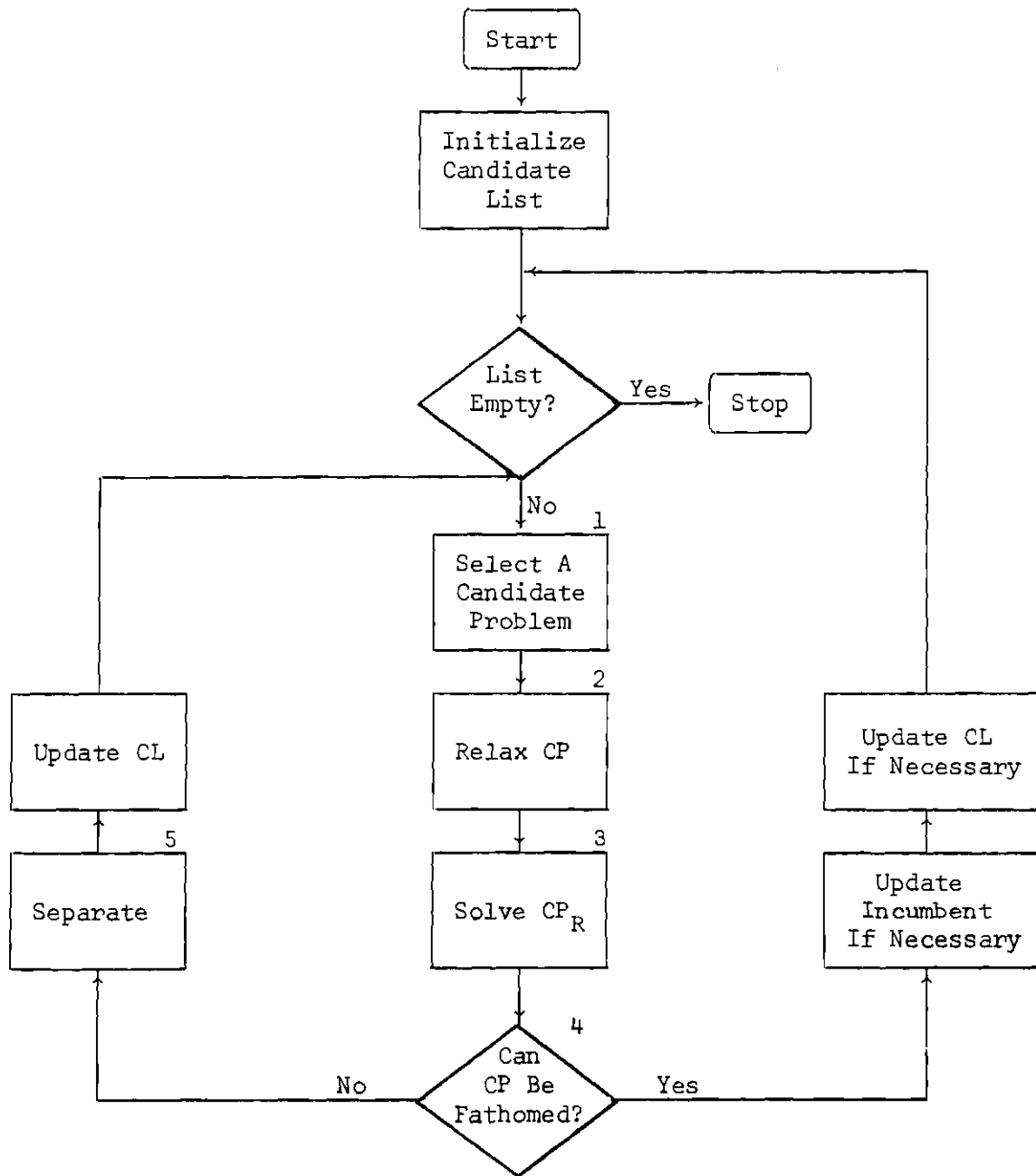


Figure 1. General Branch-and-Bound Algorithm for Solving Integer Programming Problems

A Group Theoretic Algorithm for the  
Pure Integer Programming Problem

As mentioned earlier in Chapter II, the fixed-charge transportation problem (FCTP) may be viewed as a special case of the pure integer programming problem of the form

$$\begin{aligned}
 \text{(IP)} \quad & \min \quad \underline{c}'\underline{x} \\
 & \text{st.} \quad \underline{A}\underline{x} = \underline{b} \\
 & \quad \underline{x} \geq \underline{0} \\
 & \quad \underline{x} = \underline{0} \pmod{\underline{1}}.
 \end{aligned}$$

Clearly any algorithm for (IP) can be used to solve (FCTP). An algorithm for (IP) based on the ideas of Tomlin (34) and Johnson (25) and placed in the framework of Geoffrion and Marsten (12) is now presented.

A. Selection of a Candidate Problem

This algorithm uses the priority rule selection procedure. This rule is preferable to the LIFO rule since the algorithm can choose a candidate problem which holds the best promise for obtaining a new incumbent. The specific rule is to select the CP with the smallest lower bound.

B. Relaxation

The CP will be relaxed by ignoring the integrality restriction.

C. Solving the Relaxed Problem

The CP can be solved by linear programming.

#### D. Fathoming

The fathoming criterion presented earlier can be strengthened by consideration of the group problem associated with (IP). First, a method for obtaining the group problem is given. Let  $B$  denote the optimal LP basis of (IP). Then (IP) is equivalent to

$$\begin{aligned}
 (3.1) \quad & \min \underline{c}'_B \underline{x}_B + \underline{c}'_N \underline{x}_N \\
 & \text{st. } B \underline{x}_B + N \underline{x}_N = \underline{b} \\
 & \underline{x}_B \geq \underline{0} \\
 & \underline{x}_N \geq \underline{0} \\
 & \underline{x}_B = \underline{0} \pmod{\underline{1}} \\
 & \underline{x}_N = \underline{0} \pmod{\underline{1}}
 \end{aligned}$$

where  $\underline{x}_B$  are the variables associated with  $B$  and  $\underline{x}_N$  are the remaining variables. Solving for  $\underline{x}_B$  in terms of  $\underline{x}_N$  yields

$$(3.2) \quad \underline{x}_B = B^{-1} \underline{b} - B^{-1} N \underline{x}_N.$$

Substituting (3.2) into (3.1) yields

$$\begin{aligned}
 (3.3) \quad & \min \underline{c}'_B (B^{-1} \underline{b} - B^{-1} N \underline{x}_N) + \underline{c}'_N \underline{x}_N \\
 & \text{st. } B^{-1} \underline{b} - B^{-1} N \underline{x}_N \geq \underline{0} \qquad (3.3-1)
 \end{aligned}$$

$$\underline{x}_N \geq \underline{0} \quad (3.3-2)$$

$$B^{-1}\underline{b} - B^{-1}N\underline{x}_N = \underline{0} \pmod{\underline{1}} \quad (3.3-3)$$

$$\underline{x}_N = \underline{0} \pmod{\underline{1}} \quad (3.3-4).$$

The group problem is a relaxation of (3.3) obtained by omitting the constraint (3.3-1). Recognizing that  $\underline{c}'_B B^{-1}\underline{b}$  is a scalar and can be omitted from the objective function we obtain

$$(3.4) \quad \begin{aligned} \min \quad & (\underline{c}'_N - \underline{c}'_B B^{-1}N)\underline{x}_N \\ \text{st.} \quad & B^{-1}N\underline{x}_N = B^{-1}\underline{b} \pmod{\underline{1}} \end{aligned} \quad (3.4-1)$$

$$\underline{x}_N \geq \underline{0} \quad (3.4-2)$$

$$\underline{x}_N = \underline{0} \pmod{\underline{1}} \quad (3.4-3).$$

Problem (3.4) is one form of the well-known group problem originally developed by Gomory (16). In general, the group problem takes the following form:

$$(GP) \quad \begin{aligned} \min \quad & \underline{c}'\underline{t} \\ \text{st.} \quad & \bar{A}\underline{t} = \bar{b} \pmod{\underline{1}} \\ & \underline{t} \geq \underline{0} \\ & \underline{t} = \underline{0} \pmod{\underline{1}}. \end{aligned}$$

Suppose  $\bar{b}_1$  corresponds to  $(x_B)_1$  (basic variable 1), and  $\bar{b}_1$  is noninteger. Then  $\bar{b}_1$  can be expressed as  $[\bar{b}_1] + f_1$  where  $[\bar{b}_1]$  is the largest integer less than  $\bar{b}_1$  and  $0 < f_1 < 1$ . Let  $\bar{a}_j'$  denote the  $j$ th row of  $\bar{A}$ . Recall that  $\underline{x}_B = B^{-1}\underline{b} - B^{-1}N\underline{x}_N$  or in terms of the above notation  $\underline{x}_B = \underline{\bar{b}} - \bar{A}\underline{t}$ . Then  $(x_B)_1 = \bar{b}_1 - \bar{a}_1'\underline{t}$ . Requiring  $(x_B)_1$  to assume an integer value less than  $\bar{b}_1$  is equivalent to imposing the following restrictions on  $\underline{t}$ ,

$$\left[ \begin{array}{l} \bar{a}_1'\underline{t} = f_1 \pmod{1} \\ \bar{a}_1'\underline{t} \geq 0 \end{array} \right].$$

Similarly, requiring  $(x_B)_1$  to assume an integer value greater than  $\bar{b}_1$  is equivalent to restricting  $\underline{t}$  as follows:

$$\left[ \begin{array}{l} \bar{a}_1'\underline{t} = (1-f_1) \pmod{1} \\ \bar{a}_1'\underline{t} \leq 0 \end{array} \right].$$

Using the group problem objective function, up and down penalties for forcing the basic variables either up or down can be obtained by solving the following problems.

$$(3.5) \quad \text{DPEN} [(x_B)_1] = \min \underline{c}'\underline{t}$$

$$\text{st. } \bar{a}_1'\underline{t} = f_1 \pmod{1}$$

$$\underline{a}'_1 \underline{t} \geq 0$$

$$\underline{t} \geq 0$$

$$\underline{t} = \underline{0} \pmod{\underline{1}}$$

and

$$(3.6) \quad \text{UPEN} [(x_B)_1] = \min \underline{c}' \underline{t}$$

$$\text{st. } \underline{a}'_1 \underline{t} = (f_1 - 1) \pmod{1}$$

$$-\underline{a}'_1 \underline{t} \geq 0$$

$$\underline{t} \geq \underline{0}$$

$$\underline{t} = \underline{0} \pmod{\underline{1}}.$$

$\text{DPEN} [(x_B)_1]$  is an estimate of the increase in cost which will be incurred if  $(x_B)_1$  is forced to be integer and less than or equal to  $[\bar{b}_1]$ . Conversely,  $\text{UPEN} [(x_B)_1]$  is an estimate of the cost increase which will result if  $(x_B)_1$  is forced to be an integer greater than or equal to  $[\bar{b}_1] + 1$ . This penalty can be determined for each basic variable which attains a noninteger value. These penalties are illustrated in Figure 2.

Define

$$(3.7) \quad \text{PEN}(\text{CP}_R) = \max_{(x_B)_i \neq 0 \pmod{1}} \left[ \min[\text{UPEN}(x_B)_i, \text{DPEN}(x_B)_i] \right]$$

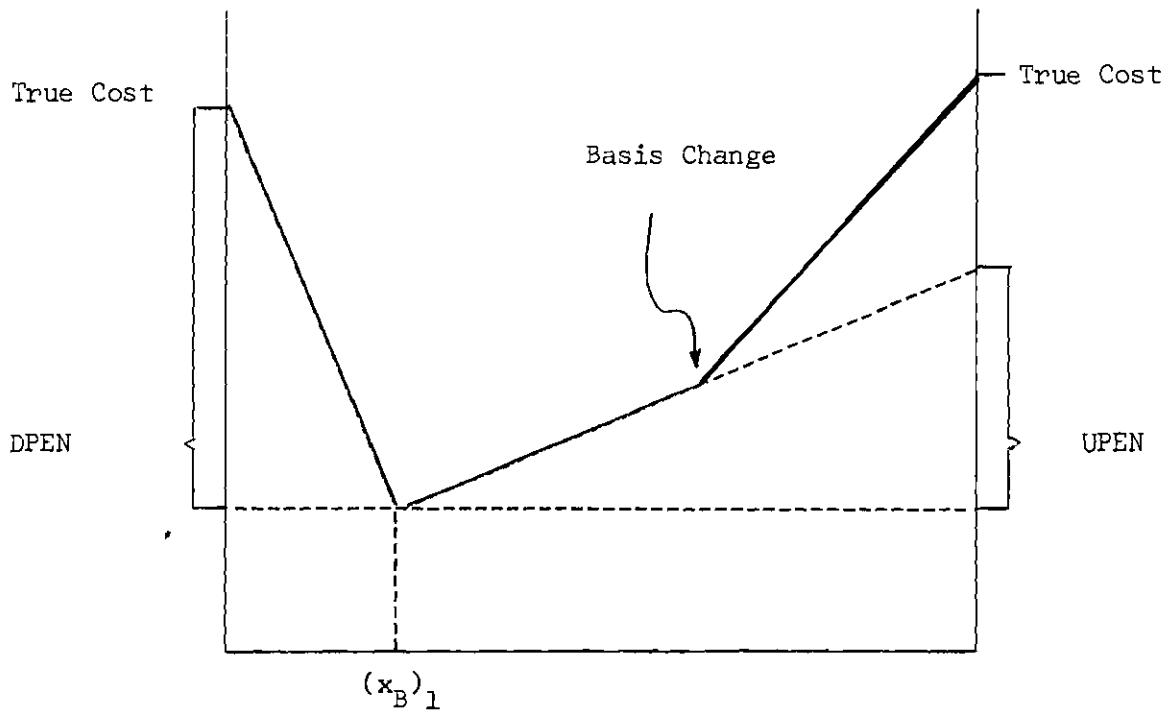


Figure 2. Illustration of the Up and Down Penalties

Since  $v(\text{CP}) \geq v(\text{CP}_R) + \text{PEN}(\text{CP}_R) \geq v(\text{CP}_R)$ , then FC2 can be strengthened to the following: If  $v(\text{CP}_R) + \text{PEN}(\text{CP}_R) \geq \text{INC}$ , then fathom.

#### E. Separation

To determine the separation variable, the penalties developed by inspection of the group problem one row at a time are used.

Let

$$S = \max_{(x_B)_i \neq 0 \pmod{1}} \left[ \max[\text{UPEN}(x_B)_i, \text{DPEN}(x_B)_i] \right].$$

If  $\text{UPEN} [(x_B)_i] = S$ , then  $(x_B)_i$  is the separating variable and a lower bound on the optimal value of the descendent associated with  $(x_B)_i \geq [\bar{b}_i] + 1$  is  $v(\text{CP}_R) + S$ . The lower bound associated with the descendents

of  $(x_B)_i \leq [\bar{b}_i]$  is  $v(\text{CP}_R) + \text{PEN}(\text{CP}_R)$ . If  $\text{DPEN} [(x_B)_i] = S$ , then  $(x_B)_i$  is again the separating variable and a lower bound on the optimal value of the descendent associated with  $(x_B)_i \geq [\bar{b}_i] + 1$  is  $v(\text{CP}_R) + \text{PEN}(\text{CP}_R)$ . Likewise a bound for the descendents of  $(x_B)_i \leq [\bar{b}_i]$  is  $v(\text{CP}_R) + S$ .

#### F. Summary

The above description gives an algorithm which could be used to solve the pure integer programming problem. Since (FCTP) is a special case of (IP), this algorithm will also solve (FCTP). However, due to the special structure of (FCTP) this algorithm can be streamlined. The two main features which allow specialization are:

- i. The relaxed problem is actually a transportation problem and can be solved by a transportation algorithm.
- ii. The group problems (3.5) and (3.6) have special structure and can be solved by inspection.

#### Network Formulation of the Relaxed Fixed-Charge Transportation Problem

This section shows that the relaxed problem ( $\text{FCTP}_R$ ) can be solved as a transportation problem rather than as a LP problem. This greatly enhances the attractiveness of the basic algorithm. The definitions and propositions which follow are used to derive this network formulation.

#### *DEFINITION 4*

A *graph*  $G$  is a finite set  $V$  of vertices (nodes)  $v_1, \dots, v_k$  and a set  $E$  of unordered pairs of vertices,  $e_p = (v_i, v_j)$ , called edges (arcs).

#### *DEFINITION 5*

A *network* is a graph with all edges directed, that is, the

elements of  $E$  are ordered pairs.

*DEFINITION 6*

A node-arc incidence matrix is a matrix with each column having exactly two nonzero entries, one of which is 1, the other -1.

Remark 1

Every network can be associated with a node-arc incidence matrix and vice-versa.

The general minimal cost flow problem for a network  $[N,E]$ , where  $N$  is the set of nodes and  $E$  is the set of arcs, is defined as follows:

$$\begin{aligned}
 \text{(MCFP)} \quad & \min \quad c(x,y)f(x,y) \\
 & \text{st.} \quad f(x,N) - f(N,x) \leq a(x), \quad x \in S \\
 & \quad \quad f(x,N) - f(N,x) = 0, \quad x \in R \\
 & \quad \quad f(x,N) - f(N,x) \leq -b(x), \quad x \in T \\
 & \quad \quad 0 \leq f(x,y) \leq a(x,y), \quad (x,y) \in E
 \end{aligned}$$

where  $c(x,y)$  = the arc cost for  $(x,y)$ ,

$f(x,y)$  = the flow from node  $x$  to node  $y$ ,

$S \subset N$  = the sources,

$T \subset N$  = the sinks,

$R = N - S - T$ ,

$a(x)$  = the supply at source  $x \in S$ ,



follows:

$$\left[ \begin{array}{cccc|cc}
 I & I & \cdots & I & 0 & 0 \\
 \hline
 \underline{1}' & 0' & \cdots & 0' & & \\
 \underline{0}' & \underline{1}' & \cdots & \underline{0}' & & \\
 & & \cdot & & 0 & 0 \\
 & & & \cdot & & \\
 \underline{0}' & \underline{0}' & & \underline{1}' & & \\
 \hline
 & & & I & -I & I
 \end{array} \right] \begin{array}{l} \} \text{ n rows} \\ \\ \\ \} \text{ m rows} \\ \\ \} \text{ m+n rows} \end{array}$$

Step 1 simply changes the signs of the first n rows. The new rows are

$$[-I \quad -I \quad \cdots \quad -I \quad 0 \quad 0].$$

Step 2 changes row n+1 through row n+m. The rows resulting from this transformation are

$$\left[ \begin{array}{cccc|cccc}
 & \underline{1}' & \underline{0}' & \cdots & \underline{0}' & -\underline{1}' & \underline{0}' & \cdots & \underline{0}' \\
 & \underline{0}' & \underline{1}' & \cdots & \underline{0}' & \underline{0}' & -\underline{1}' & \cdots & \underline{0}' \\
 0 & & & \cdot & & & & & \\
 & & & & \cdot & & & & \\
 & \underline{0}' & \underline{0}' & & \underline{1}' & \underline{0}' & \underline{0}' & & -\underline{1}'
 \end{array} \right].$$

The resulting matrix takes the form,





| Node Name |    | Right-Hand Side |
|-----------|----|-----------------|
| 4         | -1 | -1116           |
| 5         | -1 | -505            |
| 1         | 1  | 629             |
| 2         | 1  | 379             |
| 3         | 1  | 613             |
| 6         | 1  | 0               |
| 7         | 1  | 0               |
| 8         | 1  | 0               |
| 9         | 1  | 0               |
| 10        | 1  | 0               |
| 11        | 1  | 0               |

Using the node names 1 through 11 as indicated above, the node-arc incidence matrix defines the network shown in Figure 3. A negative integer near a node indicates a destination with the corresponding demand. A positive integer near a node indicates a source with the corresponding supply. For Figure 3, nodes 4 and 5 are destinations with demands of 1116 and 505, respectively. Nodes 1, 2, and 3 are sources with supplies of 629, 379, and 613, respectively. Also, the flow in each arc corresponds to one of the original variables. This correspondence has been shown in the network of Figure 3. Note that this network is in the form of a minimal cost flow problem and can be solved by any of the algorithms for that problem.

In general, the network formulation for  $(FCTP_T)$  will have  $m$  sources corresponding to the original sources of the fixed-charge transportation problem and  $n$  destinations corresponding to the original destinations. In addition there are  $mn$  intermediate nodes with one

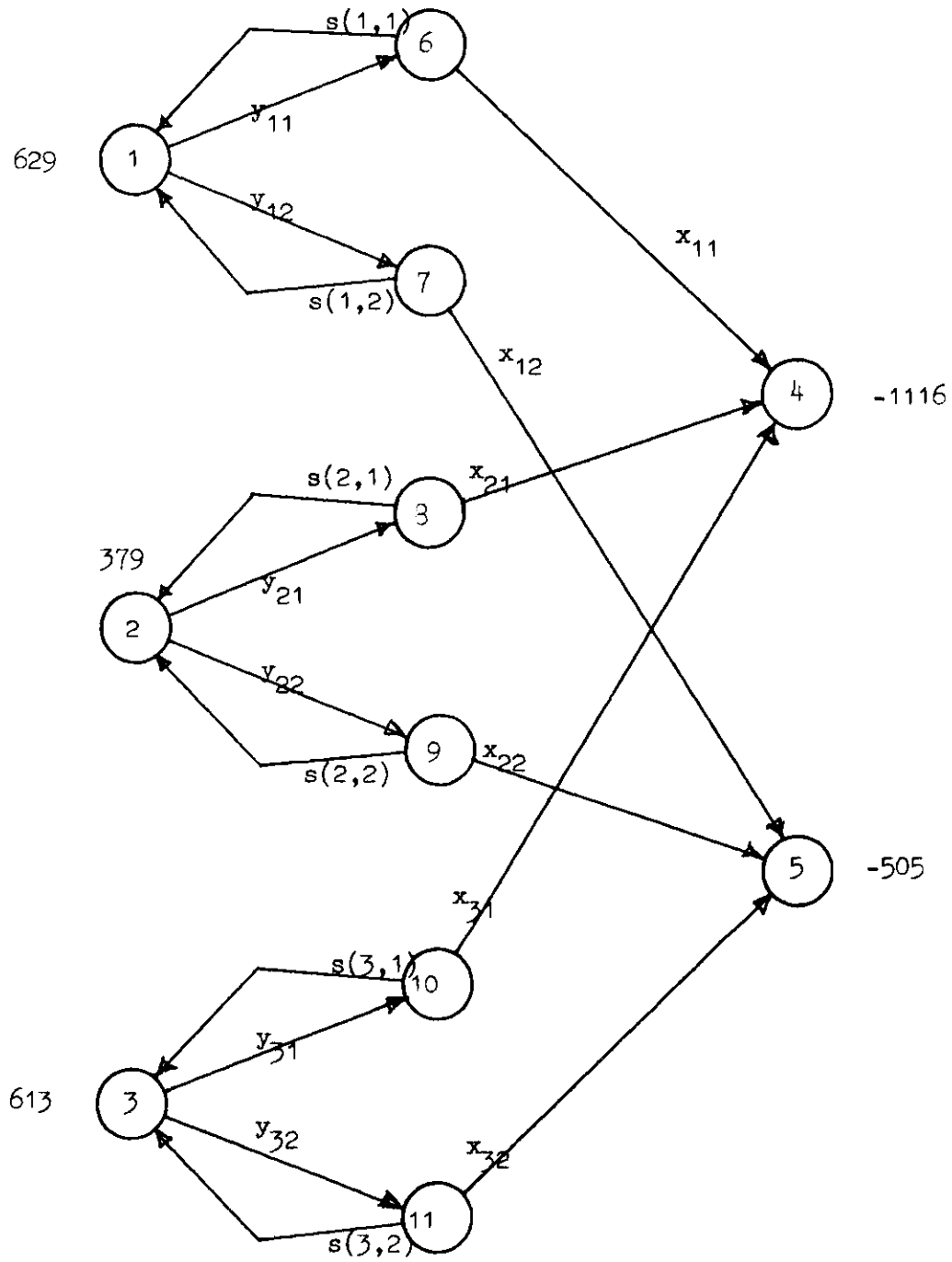


Figure 3. Network Generated by a  $3 \times 2$  Fixed-Charge Transportation Problem

between each source-destination pair. The costs on arcs from the sources to the intermediate nodes will be  $\bar{f}_{ij}$ , the costs on arcs from intermediate nodes to destinations will be  $c_{ij}$ , and the costs on arcs from the intermediate nodes to the sources will be zero. The general form is as shown in Figure 4 and will be called  $(N^*)$ .

It has been shown that  $(FCTP_R)$  is equivalent to  $(FCTP_T)$  which is a max flow min cost problem on the network  $(N^*)$ . It will now be shown that the solution to  $(FCTP_T)$  can be found by solving a transportation problem.

Lemma 1

Suppose  $(\underline{x}^*, \underline{y}^*)$  are optimal for  $(FCTP_R)$ . If  $\bar{f}_{kl} > 0$  for some  $k, l$ , then  $x_{kl}^* = y_{kl}^*$ .

*Proof.* Suppose  $x_{kl}^* \neq y_{kl}^*$ . Then  $y_{kl}^* > x_{kl}^*$ ; otherwise the constraints

$$\begin{cases} x_{kl} - y_{kl} + s(k,l) = 0 \\ x_{kl}, y_{kl}, s(k,l) \geq 0 \end{cases}$$

would be violated. Clearly a less costly feasible solution is  $(\bar{x}, \bar{y})$  formed as follows:  $\bar{x} = \underline{x}^*$

$$\bar{y}_{ij} = \begin{cases} x_{ij}^*, & \text{for } (i,j) = (k,l) \\ y_{ij}^*, & \text{otherwise.} \end{cases}$$

This contradicts optimality of  $(\underline{x}^*, \underline{y}^*)$ . Hence  $x_{kl}^* = y_{kl}^*$ . This completes the proof of Lemma 1.

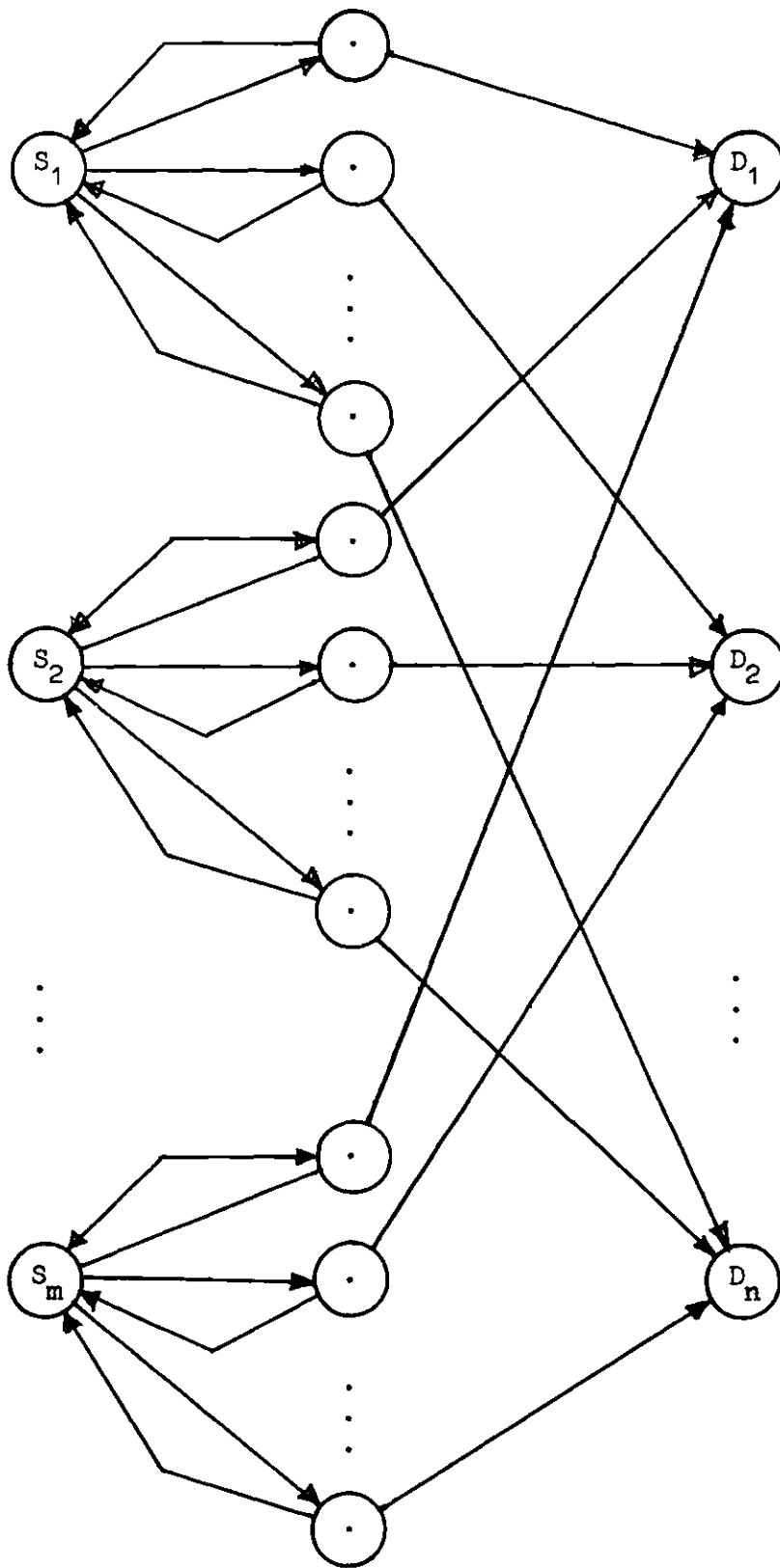


Figure 4. General Network Representation of (FCTP<sub>R</sub>)

Lemma 2

Suppose  $(\underline{x}^*, \underline{y}^*)$  are optimal for  $(FCTP_R)$  with  $\underline{x}^* \neq \underline{y}^*$ . Then there exist an alternative optima  $(\bar{x}, \bar{y})$  with  $\bar{x} = \bar{y}$ .

*Proof.* Let  $k \in \{1, 2, \dots, m\}$ ,  $l \in \{1, 2, \dots, n\}$  such that  $x_{kl}^* \neq y_{kl}^*$ . By Lemma 1,  $x_{kl}^* \neq y_{kl}^*$  implies  $\bar{f}_{kl} \leq 0$ . Since  $\bar{f}_{kl} < 0$  results in an unbounded solution, then  $\bar{f}_{kl} = 0$ . An identical cost feasible solution can be defined as follows:

$$\bar{x} = \underline{x}^*$$

$$\bar{y}_{ij} = \begin{cases} x_{ij}^*, & \text{for } (i, j) = (k, l) \\ y_{ij}^*, & \text{otherwise.} \end{cases}$$

Reapplication of the above eventually results in an alternative optima  $(\bar{x}, \bar{y})$  with  $\bar{x} = \bar{y}$ . This completes the proof of Lemma 2.

Lemmas 1 and 2 taken together imply that there always exist an optima to  $(FCTP_R)$  with  $x_{ij} = y_{ij}$  for all  $i, j$ . Since an optima for  $(FCTP_R)$  is an optima for  $(FCTP_T)$ , the  $y_{ij}$ 's in  $(FCTP_T)$  can be replaced by  $x_{ij}$ 's. The network  $(N^*)$  then reduces to a transportation network with cost on arc  $(S_i, D_j)$  equal to  $c_{ij} + \bar{f}_{ij}$ . That is, the optimal flows in the network  $(N^*)$  can be obtained by solving a transportation problem  $(T^*)$  shown in Figure 5 with the appropriate costs.

Note that in  $(N^*)$  there is a unique path from each source to each destination through one intermediate node. Therefore, the intermediate nodes can be dropped and the cost for the arc  $(S_i, D_j)$  will be  $c_{ij} + \bar{f}_{ij}$ . That is, the optimal flows in the network  $(N^*)$  can be obtained by

solving a transportation problem ( $T^*$ ) shown in Figure 5 with the appropriate costs.

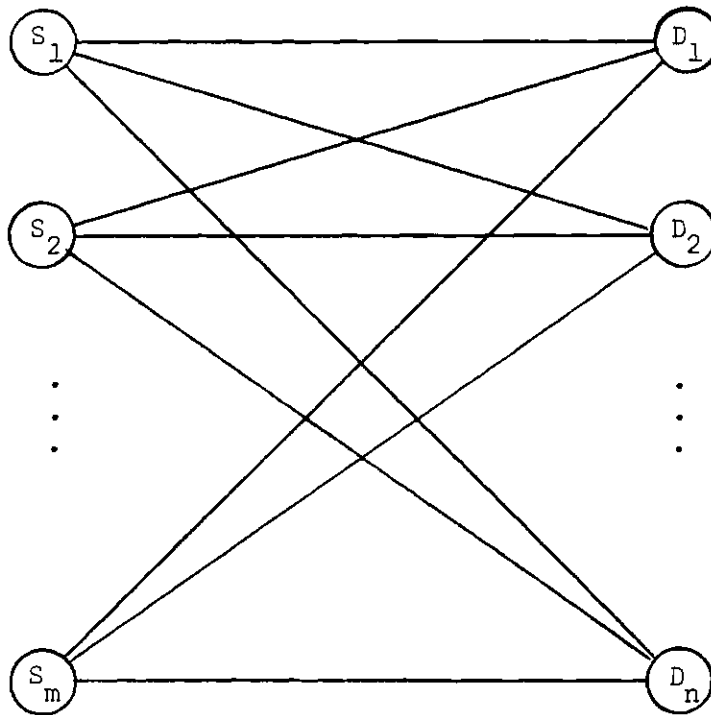


Figure 5. Network for an  $m \times n$  Transportation Problem

Recall that at each iteration of the proposed algorithm (see Figure 1), a relaxed problem must be solved. Since the relaxation used in this algorithm is to ignore the constraints  $y_{ij} \in \{0, u_{ij}\}$ , then there is a choice as to the relaxed problem which may be solved,  $(FCTP_R)$  or  $(FCTP_T)$ . For this work,  $(FCTP_T)$  was chosen because the calculations required to obtain the group problem at each iteration are trivial. This is due to the structure of the constraint matrix associated with  $(FCTP_T)$ , i.e. each column has exactly two nonzero entries a + 1 and

a - 1.

In addition to the case in which the flow in the path from  $S_i$  to  $D_j$  in  $(N^*)$  may assume any value, it must be shown that when  $y_{ij}$  is fixed at either 0 or  $\mu_{ij}$ , that the solution to  $(FCTP_T)$  can still be obtained by solving a transportation problem with the appropriate costs.

*Case 1:* Suppose  $y_{ij} = \mu_{ij}$ .

For the path in  $(N^*)$  from  $S_i$  to  $D_j$  with flow of  $\mu_{ij}$  from  $S_i$  to the intermediate node on this path and flow of  $\alpha$  from the intermediate node to  $D_j$ , the cost associated with this path is  $f_{ij} + \alpha c_{ij}$ . Since the fixed cost is a constant which appears in the objective function it may be ignored in the solution process. Therefore the appropriate cost for this case is  $c_{ij}$  on the arc  $(S_i, D_j)$  in  $(T^*)$ .

*Case 2:* Suppose  $y_{ij} = 0$ .

This implies that there can be no flow from source  $i$  to destination  $j$ . Hence, the arc from  $S_i$  to  $D_j$  can be eliminated from  $(T^*)$ . Table 1 summarizes the costs used on  $(T^*)$  to solve  $(FCTP_T)$  for the three possible states of an arc.

Table 1. Arc Costs for  $(T^*)$

| Status of $y_{ij}$  | Appropriate Cost for $(S_i, D_j)$ in $(T^*)$ |
|---------------------|--|
| free                | $\bar{f}_{ij} + c_{ij}$                      |
| $y_{ij} = 0$        | large positive constant                      |
| $y_{ij} = \mu_{ij}$ | $c_{ij}$                                     |

In addition to solving a transportation problem at each iteration of the algorithm, one must determine certain rows of the group problem associated with  $(FCTP_T)$ . When  $(FCTP_T)$  is solved using linear programming, the group problem appears in the final tableau. However, this is not the case when  $(FCTP_T)$  is solved using a network algorithm. The theory required for obtaining the rows of the group problem is presented in the next section.

#### Generation of the Group Problem from the Network

In the previous section it was shown that  $(FCTP_T)$  is a max flow problem on  $(N^*)$ . Let  $A$  denote the constraint matrix of  $(FCTP_T)$ . Let  $U$  be a  $(mn+m+n) \times (mn+m+n)$  diagonal matrix defined as follows:

$$u_{ij} = \begin{cases} -1, & \text{if the node in } (N^*) \text{ corresponding} \\ & \text{to the } i\text{th row is a destination,} \\ 1, & \text{otherwise.} \end{cases}$$

Then (3.8) below is equivalent to  $(FCTP_T)$  with artificials.

$$(3.8) \quad \begin{aligned} \min \quad & \underline{c}'\underline{x} + \alpha \underline{1}'\underline{s} \\ \text{st.} \quad & \underline{A}\underline{x} + \underline{U}\underline{s} = \underline{b} \\ & \underline{x}, \underline{s} \geq \underline{0} \end{aligned}$$

where  $\underline{x}$  corresponds to flows in  $(N^*)$ ,  
 $\underline{s}$  denotes the artificial vector, and  
 $\alpha$  is a large positive constant.

Solving (FCTP<sub>T</sub>) is equivalent to solving (N\*) is equivalent to solving (3.8). The group problem used in the algorithm will be the one associated with (3.8). However, the rows and objective function of the group problem will be developed from the solution of (N\*). Let  $A_0 = [A, U]$ .

Recall that the group problem takes the form

$$\begin{aligned}
 (3.9) \quad & \min \quad (\underline{c}_N' - \underline{c}_B B^{-1} N) \underline{x}_N \\
 & \text{st.} \quad B^{-1} N \underline{x}_N = B^{-1} \underline{b} \pmod{\underline{1}} \\
 & \quad \underline{x}_N \geq \underline{0} \\
 & \quad \underline{x}_N = \underline{0} \pmod{\underline{1}},
 \end{aligned}$$

where  $B$  is the optimal LP basis of (3.8),

$N$  is the matrix of nonbasic columns of  $A_0$ ,

$\underline{x}_N$  denotes the nonbasic variables,

$\underline{c}_B$  is the cost vector associated with the basic variables, and

$\underline{c}_N$  is the cost vector associated with the nonbasic variables.

To develop (3.9) from the solution of (N\*), the optimal basis,  $B$ , of  $A_0$  must be known.

#### Network Characterization of an Optimal Basis

The definitions and propositions of this section are originally due to Johnson (23). For the purpose of self containment, part of Johnson's work has been repeated here.

*DEFINITION 7*

A *path* in a graph  $C$  is a sequence of vertices and distinct arcs,  $(v_1, e_1, v_2, e_2, \dots, v_{k-1}, e_{k-1}, v_k)$ , such that  $e_i$  is incident to both  $v_i$  and  $v_{i+1}$ .

*DEFINITION 8*

A *simple path* is a path with distinct vertices.

*DEFINITION 9*

A *cycle* is a simple path together with an arc from the beginning to the end of the path.

*DEFINITION 10*

A *connected graph* has at least one path between every pair of vertices.

*DEFINITION 11*

A *tree* is a connected graph with no cycles.

*DEFINITION 12*

A *forest* is a graph consisting of one or more unconnected trees.

*DEFINITION 13*

A *spanning subgraph*  $H$  of  $G$  is a subgraph with the same vertex set as  $G$ .

*DEFINITION 14*

A *spanning forest* of  $G$  is a forest which is a spanning subgraph of  $G$ .

*DEFINITION 15*

The *subgraph generated by a matrix*  $B$  of columns of  $A_0$  called  $F_B$  consist of vertices corresponding to columns of  $U$  in  $B$  and arcs corresponding to columns of  $A$  in  $B$  together with vertices incident to such arcs.

*DEFINITION 16*

A vertex  $v_i$  corresponding to a column of  $U$  in  $B$ , a basis of  $A_0$ , will be called a *root* of the tree in  $F_B$ .

*DEFINITION 17*

A tree with one root is called a *rooted tree*.

*DEFINITION 18*

A forest with each tree having one root is called a *rooted forest*.

*DEFINITION 19*

A *non-singular, triangular matrix* is a square matrix with non-zeros on the main diagonal and all zeros below the main diagonal, or one which can be brought to such a form by swapping rows and swapping columns.

*DEFINITION 20*

The *end* of a tree with a single vertex is that vertex. The *end* of a tree with one or more edges is a vertex touching only one edge of the tree.

Proposition 2 [Johnson (24), Lemma 2]

A tree is either a single vertex or is two disjoint trees joined by a single edge incident to one vertex of one tree and one vertex of the other tree.

*Proof.* Clearly a graph constructed in this way is a tree. It must also be shown that every tree satisfies the condition. If a tree  $T$  has no edge, then it is a single vertex. If  $T$  has an edge, say  $e_1 = (v_1, v_2)$ , then there is no simple path between  $v_1$  and  $v_2$  not using  $e_1$  because if there were,  $T$  would contain a cycle. So if  $e_1$  is removed from  $T$ , the remaining graph has at least two connected components  $T_1$  and  $T_2$  with  $v_1$  in  $T_1$  and  $v_2$  in  $T_2$ . From every vertex  $v$  in  $T$  there are simple paths to  $v_1$  and  $v_2$ ; therefore, there is a simple path to either  $v_1$  or  $v_2$  not containing  $e_1$ . Hence, removal of  $e_1$  causes the remaining graph to have exactly two connected components  $T_1$  and  $T_2$ . They are trees because if either has a cycle, so would  $T$ . This completes the proof of proposition 2.

Proposition 3 [Johnson (24), Lemma 3]

Every tree has at least one end and if it has an edge, it has at least two ends.

*Proof.* The proposition is obviously true for a tree consisting of a single vertex. Suppose the proposition is true for any tree  $T_n$  with  $n$  vertices. Then a tree with  $n + 1$  vertices can be formed by adding a new vertex and an edge incident to this new vertex and some vertex of  $T_n$ . Call the tree formed this way,  $T_{n+1}$ . Since  $T_n$  will have at least one end remaining and the new vertex will be an end,  $T_{n+1}$  will have at

least two ends. By induction on  $n = 1$ , every tree has at least two ends. This completes the proof of proposition 3.

Proposition 4 [Johnson (24), Lemma 4]

A tree with  $m$  vertices has  $m-1$  edges.

*Proof.* For  $m = 1$ , the proposition is clearly true. Suppose the proposition is true for any  $m \geq 1$ . By proposition 2, a tree  $T$  with  $m + 1$  vertices consist of two trees  $T_1, T_2$  and an edge incident to a vertex in each tree. Suppose  $T_1$  has  $m_1$  vertices and  $T_2$  has  $m_2$  vertices with  $m + 1 = m_1 + m_2$ . Then  $T_1$  has  $m_1 - 1$  edges and  $T_2$  has  $m_2 - 1$  edges by the induction hypothesis. Then  $T$  has  $(m_1 - 1) + (m_2 - 1) + 1 = m_1 + m_2 - 1 = m$  edges. By induction on  $m = 1$ , a tree with  $m$  vertices has  $m - 1$  edges. This completes the proof of proposition 4.

Proposition 5 [Johnson (24), Theorem 1]

If  $B$  is a basis of  $A_0$ , then  $F_B$  is a spanning forest.

*Proof.* Suppose  $F_B$  is not a spanning graph. Then some vertex, say  $v_1$ , of  $(N^*)$  does not appear in  $F_B$ . Then row 1 of  $B$  is  $\underline{0}$ . But some column of  $A_0$  has a non-zero entry in row 1, and such a column cannot be written as a linear combination of columns of  $B$ . This contradicts  $B$  a basis of  $A_0$ . Hence  $F_B$  is a spanning graph.

The remainder of the proof consist of showing that  $F_B$  has no cycles. Suppose  $F_B$  has a cycle  $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1} = v_1$ . Then there are  $k$  columns of  $B$ , say  $B^1, \dots, B^k$ , corresponding to  $e_1, \dots, e_k$ .

Let

$$y_j = \begin{cases} 1, & \text{if } e_j = (v_j, v_{j+1}) \\ -1, & \text{if } e_j = (v_{j+1}, v_j). \end{cases}$$

For  $v_i$  not in the cycle,  $\sum_{j=1}^k b_{ij}y_j = 0$  because none of the arcs  $e_1, \dots, e_k$  are incident to  $v_i$  so all  $b_{ij} = 0$  for  $j=1, \dots, k$ . For  $v_i$  in the cycle, there are four cases to consider:

- i.  $e_i = (v_{i-1}, v_i)$ ,  $e_{i+1} = (v_i, v_{i+1})$
- ii.  $e_i = (v_i, v_{i-1})$ ,  $e_{i+1} = (v_i, v_{i+1})$
- iii.  $e_i = (v_{i-1}, v_i)$ ,  $e_{i+1} = (v_{i+1}, v_i)$ , and
- iv.  $e_i = (v_i, v_{i-1})$ ,  $e_{i+1} = (v_{i+1}, v_i)$ .

For case i,  $b_{ij} = \begin{cases} -1, & \text{if } j=i \\ 1, & \text{if } j=i+1 \end{cases}$  and  $b_{ij} = 0$  for all other  $j$ .

With  $y_i = y_{i+1} = 1$ , then  $\sum_{j=1}^k b_{ij}y_j = 0$ . The other three cases are similar and all taken together imply  $\sum_{j=1}^k B^j = 0$ . This contradicts  $B$  having linearly independent columns. Hence,  $F_B$  has no cycles. Therefore,  $F_B$  is a spanning forest. This completes the proof of proposition 5.

Proposition 6 [Johnson (24), Theorem 2]

If  $B$  is a basis of  $A_0$ , then every tree of the forest  $F_B$  has at most one root.

*Proof.* Suppose some tree had two roots  $v_1$  and  $v_k$ . A tree is connected so there is a simple path from  $v_1$  to  $v_k$  in the tree, say  $v_1, e_1, v_2, \dots, v_{k-1}, e_{k-1}, v_k$ . Hence, there are  $k+2$  columns of  $B$ , say  $B^1, \dots, B^{k+2}$ ,

corresponding to  $e_1, e_2, \dots, e_k, v_1, v_k$ , respectively. Let

$$y_{k+1} = \begin{cases} 1, & \text{if } B^{k+1} \text{ has a } -1 \text{ non-zero entry} \\ -1, & \text{if } B^{k+1} \text{ has a } 1 \text{ non-zero entry,} \end{cases}$$

$$y_{k+2} = \begin{cases} 1, & \text{if } B^{k+2} \text{ has a } 1 \text{ non-zero entry} \\ -1, & \text{if } B^{k+2} \text{ has a } -1 \text{ non-zero entry,} \end{cases}$$

$$y_j = \begin{cases} 1, & \text{if } e_j = (v_j, v_{j+1}) \\ -1, & \text{if } e_j = (v_{j+1}, v_j), \quad j=1, \dots, k. \end{cases}$$

Then as in proposition 5,  $\sum_{j=1}^{k+2} y_j B^j = 0$ , contradicting linear independence of the columns of  $B$ . Hence, every tree of the forest  $F_B$  has at most one root. This completes the proof of proposition 6.

Proposition 7 [Johnson (24), Theorem 4]

If  $B$  is a basis of  $A_0$ , then  $F_B$  is a rooted, spanning forest of  $(N^*)$ .

*Proof.* By proposition 5,  $F_B$  is a spanning forest. By proposition 6, every tree of  $F_B$  has at most one root. To complete the proof, it must be shown that every tree in the forest has at least one root. Suppose a tree has no root. Also suppose  $A_0$  has  $k$  rows. Clearly  $\text{rank}(A_0) = k$ . But  $B$  has  $k-1$  columns or less because a tree has one fewer edge than vertices, and no tree in  $F_B$  can have more than one root. This

contradicts  $B$  a basis of  $A_{\circ}$ . Hence,  $F_B$  is a rooted spanning forest of  $(N^*)$ . This completes the proof of proposition 7.

Proposition 8 [Johnson (24), Theorem 3]

If  $F_B$  is a rooted, spanning forest of  $(N^*)$ , then  $B$  is a non-singular, triangular matrix.

*Proof.* By proposition 4,  $B$  will be square. The additional columns of  $U$  for each tree makes  $B$  have as many columns as rows.

The proof is by induction on  $m$ , the number of rows of  $A_{\circ}$ . For  $m = 1$ ,  $B$  is a  $1 \times 1$  non-zero matrix which is non-singular and triangular. Assume the proposition is true for a graph with  $m - 1$  nodes for  $m \geq 2$ . Then  $A_{\circ}$  will have  $m - 1$  rows. Consider a matrix  $A_{\circ}$  having  $m$  rows, that is a graph  $G$  with  $m$  nodes.

If  $B$  has only columns from  $U$ , then  $B$  is diagonal so certainly non-singular and triangular. If  $B$  has a column from  $A$ , then  $F_B$  has an edge, and the tree to which the edge belongs has at least two ends by proposition 3. But the tree has only one root, and hence, there must be a vertex  $v_i$  which is an end of the tree and not a root. Then row  $i$  of  $B$  has only one non-zero entry. Let  $\bar{B}$  be the matrix formed from  $B$  by deleting row  $i$  and the column with non-zero entry in row  $i$ . Let  $\bar{A}_{\circ} = [\bar{A}, \bar{U}]$  denote the matrix formed from  $A_{\circ}$  by deleting row  $i$  and all columns with nonzero entries in row  $i$ , and let  $\bar{G}$  denote the network formed from  $G$  by deleting vertex  $v_i$  and all arcs incident to vertex  $v_i$ . Then  $\bar{A}$  is the node-arc incidence matrix of  $\bar{G}$ , and  $F_{\bar{B}}$  is a spanning forest of  $\bar{G}$ . Furthermore, every tree of  $F_{\bar{B}}$  has exactly one vertex corresponding to  $\bar{U}$  in  $\bar{A}_{\circ}$ . By the induction hypothesis,  $\bar{B}$  is non-singular and triangular.

Therefore  $B$  is non-singular and triangular. By induction on  $m = 1$ , if  $F_B$  is a rooted spanning forest of  $G$ , then  $B$  is a non-singular, triangular matrix. This completes the proof of proposition 8.

Langley (29) developed a primal algorithm using a tree structure for the transportation problem. This method terminates with the rooted spanning forest  $F_B$  where  $B$  corresponds to the optimal basis.

Following the ideas of Langley (28), this optimal basis  $B$  can now be used to find a row of  $B^{-1}N$ . Rearrange  $A_0$  such that  $A_0 = [B, N]$ . Then  $B^{-1}A_0 = [I, B^{-1}N]$ . Let  $\underline{d}_i'$  denote the  $i$ th row of  $B^{-1}N$ . Then the  $i$ th row of  $B^{-1}A_0$  is  $[\underline{e}_i', \underline{d}_i']$ . Since  $B$  is a basis, there exist a vector  $\underline{\lambda}$  such that  $\underline{\lambda}'B = \underline{e}_i'$ . Since  $\underline{\lambda}$  is unique,  $\underline{\lambda}'N = \underline{d}_i'$ . Since  $B$  consist of columns with at most two nonzero entries, a  $+1$  and a  $-1$ ,  $\underline{\lambda}$  can be found by a labeling procedure on  $F_B$ . That is, every column of  $B$  corresponds to either an arc or a root of  $F_B$ . If the  $i$ th column corresponds to an arc, say  $(x, y)$ , then  $\lambda(x) - \lambda(y) = 1$  and the  $\lambda$  for the root of the tree with arc  $(x, y)$  must be zero. If the  $i$ th column corresponds to a root at node  $x$ , then  $\lambda(x) = 1$ . For both cases all other arcs,  $(r, s)$ ,  $\lambda(r) - \lambda(s) = 0$  and all other roots,  $t$ , have  $\lambda(t) = 0$ .

Algorithm for Generating the  $i$ th Row of  $B^{-1}N$  (ALG-1)

*Step 1.* Does the  $i$ th column of  $B$  correspond to a root?

Yes - Denote that node by  $x$ . Go to step 2.

No - Go to step 3.

*Step 2.* Set  $\lambda(x) = 1$ . Starting at the root, determine all other  $\lambda$ 's by the rule  $\lambda(y) - \lambda(z) = 0$  for all arcs  $(y, z)$  in the tree.

Set all other  $\lambda$ 's in all other trees equal to 0. Go to step 5.

*Step 3.* Denote the arc corresponding to the  $i$ th column of  $B$  by  $(x,y)$ .

*Step 4.* Is  $x$  closer to its root than  $y$ ?

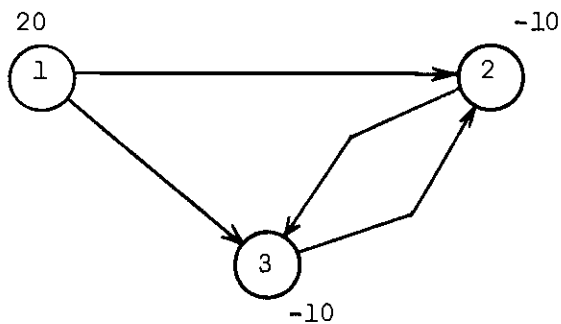
Yes - Let  $\lambda(x) = 0$ ,  $\lambda(y) = -1$ . Starting from both  $x$  and  $y$ , determine all other  $\lambda$ 's by the rule  $\lambda(r) - \lambda(s) = 0$  for all other arcs in the tree. Set all other  $\lambda$ 's in all other trees equal to 0. Go to step 5.

No - Let  $\lambda(x) = 1$ ,  $\lambda(y) = 0$ . Starting from both  $x$  and  $y$ , determine all other  $\lambda$ 's by the rule  $\lambda(r) - \lambda(s) = 0$  for all other arcs in the tree. Set all other  $\lambda$ 's in all other trees equal to 0. Go to step 5.

*Step 5.* For each column of  $N$  corresponding to an arc, say  $(x,y)$ , the entry for that column of the  $i$ th row of  $B^{-1}N$  is simply  $\lambda(x) - \lambda(y)$ . For each column of  $N$  corresponding to a node, say  $z$ , the entry for that column in the  $i$ th row is  $\lambda(z)$ .

### Example 2

Consider the 3 node 4 arc network:



$$c_{12} = 20$$

$$c_{13} = 30$$

$$c_{23} = 5$$

$$c_{32} = 10$$

Note that this network has no slacks, but at least one artificial variable is required to form a basis. The nonbasic artificials are ignored in this example. Suppose arc (2,3) has the restriction that  $f(2,3) \in \{0,15\}$ .

$$A_0 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The optimal flows obtained by ignoring the side restrictions are  $f(1,2) = 20$ ,  $f(2,3) = 10$ . The optimal dual variables are  $\pi(1) = 0$ ,  $\pi(2) = -20$ , and  $\pi(3) = -25$ . The optimality conditions are

$$\begin{aligned} \pi(1) - \pi(2) &= 20 \\ \pi(2) - \pi(3) &= 5 \\ \pi(1) - \pi(3) - c_{13} &\leq 0 \\ \pi(3) - \pi(2) - c_{32} &\leq 0. \end{aligned}$$

Note that

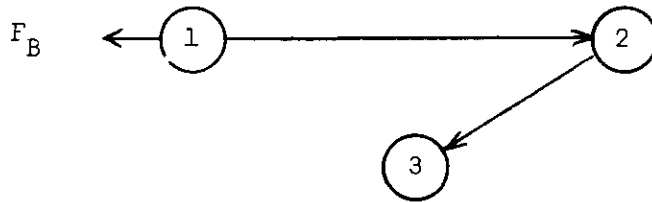
$$B_1 = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

are all optimal bases since each yields a rooted spanning forest of  $(N^*)$ , and there are no slack variables.

Let  $B = B_1$ . Then

$$B^{-1} = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad N = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -1 & 1 \end{bmatrix}.$$

Calculation of Row 1 of  $B^{-1}N$ :



*Step 1.* No

*Step 3.* (1,2)

*Step 4.* Yes,  $\lambda(1) = 0$ ,  $\lambda(2) = -1$ ,  $\lambda(3) = -1$ .

*Step 5.* [1,0]

Calculation of Row 2:

*Step 1.* No

*Step 3.* (2,3)

*Step 4.* Yes,  $\lambda(2) = 0$ ,  $\lambda(3) = -1$ ,  $\lambda(1) = 0$ .

*Step 5.* [1,-1]

Calculation of Row 3:

*Step 1.* Yes

Step 2.  $\lambda(1) = 1, \lambda(2) = 1, \lambda(3) = 1.$

Step 5.  $[0,0]$

Attention is now turned to the objective function of the group problem (3.9). These cost coefficients are  $\underline{c}'_N - \underline{c}'_B B^{-1}N$  where  $B$  is the optimal basis for (3.8). From duality theory, it is known that at optimality, the optimal dual variables are  $\underline{\pi}^* = \underline{c}'_B B^{-1}$ . These dual variables are available from the solution of  $(N^*)$ . Therefore the objective function is simply  $\underline{c}'_N - \underline{\pi}^*N$ .

Algorithm for Determining the Group  
Problem Objective Function (ALG-2)

Denote the columns of  $N$  by  $\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k$ . Denote the costs of the arcs associated with the columns of  $N$  by  $c_1^N, c_2^N, \dots, c_k^N$ . Denote the optimal dual variables by  $\pi^*(x)$  for each node  $x$ .

Step 1. Let  $i = 1$ .

Step 2. Denote the arc associated with  $\underline{n}_i$  by  $e_i = (x,y)$ .

Step 3. The  $i$ th component of the cost vector equals  $\pi^*(y) - \pi^*(x) + c_i^N$ .

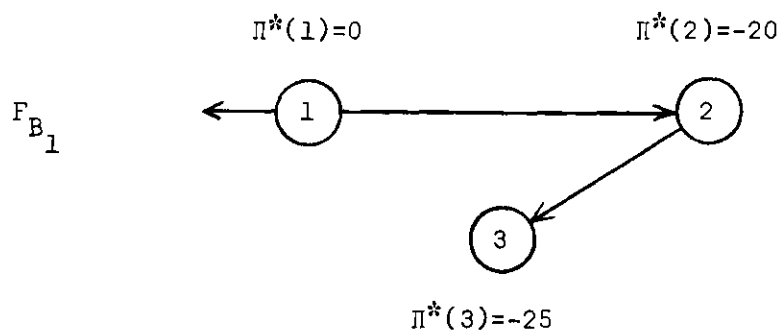
Step 4. Is  $i = k$ ?

Yes - Stop.

No - Increment  $i$  by 1 and return to step 2.

Example 3

Consider the problem of example 2 using  $B_1$  as the optimal basis.



$$N = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -1 & 1 \end{bmatrix} \quad c_1^N = 30, \quad c_2^N = 10.$$

*Step 1.*  $i = 1$ .

*Step 2.*  $e_1 = (1,3)$ .

*Step 3.*  $\Pi^*(3) - \Pi^*(1) + c_1^N = 5$ .

*Step 4.* No.  $i = 2$ .

*Step 2.*  $e_2 = (3,2)$ .

*Step 3.*  $\Pi^*(2) - \Pi^*(3) + c_2^N = 15$ .

*Step 4.* Stop.

Through the use of ALG-1 and ALG-2, the group problem can now be obtained from the solution of  $(N^*)$ .

#### Simplification of Penalty Calculation

At each stage of the algorithm, the relaxed problem is solved by a primal network method which terminates with the optimal basis forest,  $F_B$ . By proposition 8,  $B$  is triangular. Since  $B$  consist of columns of

$A_0$ , the diagonal elements of  $B$  are from  $\{-1,1\}$  and hence  $\text{abs}(|B|) = 1$ , where  $\text{abs}(\cdot)$  denotes the absolute value of the enclosed expression. It will now be shown that the elements of the group problem are from  $\{-1,0,1\}$ .

Proposition 9

If  $B$  is a basis of  $A_0$ , then the elements of  $B^{-1}A_0$  are from  $\{-1,0,1\}$ .

*Proof.* Let  $\underline{d}$  be any column of  $A_0$ . Then  $\underline{d}$  can be expressed as a linear combination of the columns of  $B$ . That is,  $\underline{d} = B\underline{x}$ . By Cramer's Rule the  $i$ th component of  $\underline{x}$  is defined by

$$x_i = \frac{|B^*|}{|B|}$$

where  $B^*$  is formed by replacing the  $i$ th column of  $B$  by  $\underline{d}$ . Since  $B$  is a basis of  $A_0$ , it is a diagonal matrix and hence  $\text{abs}(|B|) = 1$ . Suppose  $B^*$  is a basis of  $A_0$ . By proposition 7,  $F_{B^*}$  is a rooted spanning forest. By proposition 8,  $B^*$  is non-singular and triangular. Then the diagonal elements of  $B^*$  are from  $\{-1,1\}$ . Hence  $\text{abs}(|B^*|) = 1$ . Then  $x_i$  equals either  $-1$  or  $1$ . Suppose  $B^*$  is not a basis of  $A_0$ . Then  $|B^*| = 0$ . Hence  $x_i = 0$ . Therefore, the elements of  $B^{-1}A_0$  are from  $\{-1,0,1\}$ . This completes the proof of proposition 9.

With this additional property, the penalty calculations given in (3.5) and (3.6) simplify to the following:

$$DPEN(y_{ij}) = y_{ij}^* \left[ \min_{k:r'_k(i,j)=1} c_k^G \right]$$

and

$$UPEN(y_{ij}) = (\mu_{ij} - y_{ij}^*) \left[ \min_{k:r'_k(i,j)=-1} c_k^G \right]$$

where  $y_{ij}^*$  is the optimal flow in  $(N^*)$ ,

$r'_k(i,j)$  is the row in the group problem corresponding to variable  $y_{ij}$ , and

$c_k^G$  is the group problem cost coefficient corresponding to the  $k$ th nonbasic variable.

Although these penalties were derived from consideration of the group problem, they reduce to the penalty suggested by Tomlin (34).

#### Group Theoretic Algorithm for the Fixed-Charge Transportation Problem

The theory developed in the previous sections is now incorporated into the general algorithm presented in Figure 1. Let CL denote the candidate list. The elements of CL will be  $mn + 1$  element vectors. The first  $mn$  positions will denote whether or not a  $y_{ij}$  is fixed and if so at what value. The  $mn + 1$ st position will give the lower bound for that partial problem. CP is a vector which defines a candidate problem, and  $CP_R$  is the relaxation of this candidate problem. A -1 as an element of CP indicates that the corresponding  $y$  is free, a 0 indicates that the corresponding  $y$  is fixed at 0 and an  $\alpha > 0$  indicates that the corresponding  $y$  is fixed at  $\alpha$ . INC denotes the cost of the incumbent (the

best solution found so far). Let  $v(\text{CP}_R)$  be the optimal cost of  $\text{CP}_R$ .

Example 4

For a 2 source 3 destination problem, if  $\text{CP} = (-1, 0, 0, 30, -1, -1, 500)$  with  $\text{CP} = (y_{11}, y_{12}, y_{13}, y_{21}, y_{22}, y_{23}, \text{lower bound})$ , then the partial solution will be  $y_{12} = y_{13} = 0$ ,  $y_{21} = 30$  and  $y_{11}, y_{22}, y_{23}$  free to assume any value. The lower bound of this partial problem is 500.

For (ALG-3) below it is assumed that all costs are integer. This permits upward rounding for estimates of candidate problem bounds.

Group Theoretic Algorithm for (FCTP) (ALG-3)

- Step 1.* Define a CP with all variables free.
- Step 2.* Solve  $\text{CP}_R$ .
- Step 3.* Does the solution meet the side restrictions?  
 Yes - Stop. This solution is optimal.  
 No - Go to step 4.
- Step 4.* Increase the flows in  $(N^*)$  so that all side restrictions are met. Let INC denote the cost of this solution.
- Step 5.* Use ALG-2 to obtain the group problem objective function.  
 Denote this cost vector by  $\underline{c}^G$ .
- Step 6.* Define an index set  $S = \{(i,j): y_{ij}^* \notin \{0, \mu_{ij}\} \text{ where } y_{ij}^* \text{ is the optimal flow in } (N^*)\}$ .
- Step 7.* For each  $y_{ij}$  with  $(i,j) \in S$ , use ALG-1 to obtain the corresponding row of the group problem. Denote these rows by

$$\underline{r}'(i,j).$$

*Step 8.* For each  $y_{ij}$  with  $(i,j) \in S$ , define

$$DPEN(y_{ij}) = (y_{ij}^*) \left[ \begin{array}{c} \min \\ k:r'_k(i,j)=1 \\ c_k^G \end{array} \right]$$

and

$$UPEN(y_{ij}) = (u_{ij} - y_{ij}^*) \left[ \begin{array}{c} \min \\ k:r'_k(i,j)=-1 \\ c_k^G \end{array} \right]$$

*Step 9.* Define

$$PEN(CP_R) = \max_{(i,j) \in S} \left[ \min[UPEN(y_{ij}), DPEN(y_{ij})] \right]$$

*Step 10.* Let  $V$  be the smallest integer greater than or equal to  $v(CP_R) + PEN(CP_R)$ .

Is  $V \geq INC$ ?

Yes - Fathom. Go to step 17.

No - Go to step 11.

*Step 11.* Define

$$Q = \max_{(i,j) \in S} \left[ \max[UPEN(y_{ij}), DPEN(y_{ij})] \right].$$

Let  $U$  be the smallest integer greater than or equal to  $Q + v(CP_R)$ .

*Step 12.* Does  $DPEN(y_{kl}) = Q$  for some  $k, l$ ?

Yes - Separate the present CP by fixing  $y_{kl}$ . Let  $y_{kl} = 0$ , with a bound of  $U$  and let  $y_{kl} = \mu_{kl}$  with a bound of  $V$ . Inspect these two candidate problems individually. If the corresponding bound is less than  $INC$ , place this problem in  $CL$ . Otherwise, ignore the problem. Go to step 14.

No - Go to step 13.

*Step 13.* Determine  $(k, l)$  such that  $Q = UPEN(y_{kl})$ . Separate the present

CP by fixing  $y_{kl}$ . Let  $y_{kl} = \mu_{kl}$  with a bound of  $U$ .

Let  $y_{kl} = 0$  with a bound of  $V$ . Inspect these two candidate problems individually. If the corresponding bound is less than  $INC$ , place this problem in  $CL$ . Otherwise, ignore the problem. Go to step 14.

*Step 14.* Select the problem from  $CL$  with the least bound. Denote this problem by  $CP$ . Delete  $CP$  from  $CL$ .

*Step 15.* Solve  $CP_R$ . If no feasible solution exists, go to step 17; otherwise go to step 16.

*Step 16.* Is  $v(CP_R) \geq INC$ ?

Yes - Fathom. Go to step 17.

No - Go to step 18.

*Step 17.* Is  $CL = \emptyset$ ?

Yes - Stop.

No - Go to step 14.

*Step 18.* Does the solution meet the side restrictions?

Yes - Go to step 21.

No - Go to step 19.

*Step 19.* Increase the flow in ( $N^*$ ) so that all side restrictions are met. Let  $z$  denote the cost of this solution.

*Step 20.* Is  $z \geq \text{INC}$ ?

Yes - Go to step 5.

No - Go to step 22.

*Step 21.* Update INC. Delete all problems from CL with bounds greater than INC. Go to step 17.

*Step 22.* Update INC. Delete all problems from CL with bounds greater than INC. Go to step 5.

Proposition 10    *Finiteness*

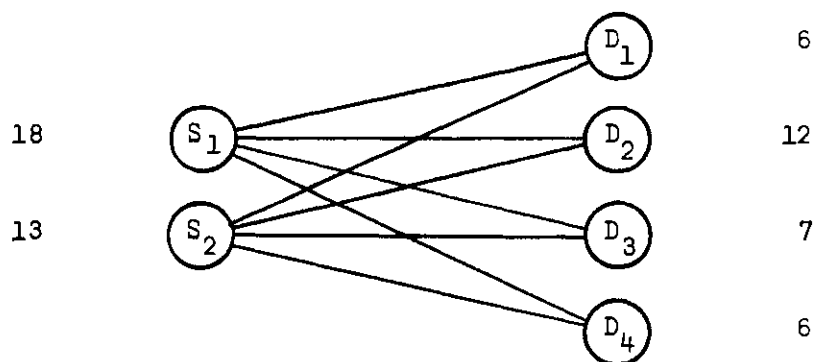
ALG-3 terminates in a finite number of iterations.

*Proof.* For a  $m$  source  $n$  destination fixed-charge transportation problem there will be  $mn$  side constraints of the form  $y_{ij} \in \{0, \mu_{ij}\}$ . The solution tree will have a single starting node with all  $mn$  variables free. The first level will consist of two nodes with one of the variables fixed at 0 and  $\mu_{ij}$ . The second level consists of four nodes, etc. In general, the solution tree will have  $\sum_{i=0}^{mn} 2^i$  nodes or possible candidate problems. Since each iteration through the algorithm disposes of one or more of these candidate problems and since each candidate problem can be considered only once, the algorithm can run at most  $\sum_{i=0}^{mn} 2^i$  iterations. Hence ALG-3 is finite. This completes the proof of proposition 10.

Proposition 11 *Optimality*

If (FCTP) has a feasible solution, then ALG-3 terminates with an optimal solution as the incumbent.

*Proof.* Since the variables associated with the fixed cost in (FCTP) are from the set  $\{0, u_{ij}\}$ , there are only  $2^{mn}$  possible solutions to (FCTP). Each of these possible solutions appears as one of the  $\sum_{i=0}^{mn} 2^i$  candidate problems. Also each of these candidate problems must be compared to the incumbent. The first comparison of an optimal CP will result in the incumbent being replaced by this CP. No other comparison after this will change the incumbent. Hence, if a feasible solution exists, ALG-3 terminates with the optimal solution as the incumbent. This completes the proof of proposition 11.

Example 5

$c_{11} = 2$

$c_{21} = 7$

$f_{11} = 56$

$f_{21} = 16$

$c_{12} = 6$

$c_{22} = 4$

$f_{12} = 10$

$f_{22} = 18$

$c_{13} = 3$

$c_{23} = 8$

$f_{13} = 13$

$f_{23} = 19$

$c_{14} = 0$

$c_{24} = 0$

$f_{14} = 0$

$f_{24} = 0$

Step 1.  $CP = [-1, -1, -1, -1, -1, -1, -1, -1, -\infty]$ .

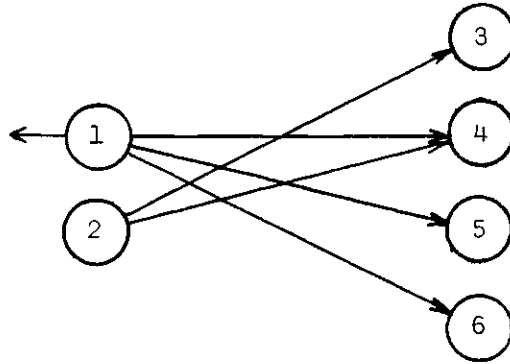
Step 2.  $x_{11} = 0, x_{12} = 5, x_{13} = 7, x_{14} = 6.$

$x_{21} = 6, x_{22} = 7, x_{23} = 0, x_{24} = 0$

$\Pi(1) = 0, \Pi(2) = -13.33, \Pi(3) = -11.00$

$\Pi(4) = -68.33, \Pi(5) = -48.57, \Pi(6) = 0.$

$F_B$ :



Step 3. No.

Step 4. 178.

Step 5.  $\underline{c}^G = [0.33, 7.19, 1.33, 9.33, 0.83, 1.86, 0, 2.67, 1.50, 2.71, 0].$

Step 6.  $S = \{(1,2), (2,2)\}.$

Step 7.  $r^{(1,2)} = [1, -1, -1, 0, -1, 0, 0, 0, 0, 0, 0].$

$r^{(2,2)} = [-1, 1, 1, 0, 0, 0, 0, 0, -1, 0, 0].$

Step 8.  $DPEN(y_{12}) = 1.67 \quad UPEN(y_{12}) = 5.83.$

$DPEN(y_{22}) = 9.33 \quad UPEN(y_{22}) = 1.67$

Step 9.  $PEN(CP_R) = 1.67.$

Step 10.  $167 \geq 178?$  No.

Step 11.  $Q = 9.33$ .

Step 12. Yes,  $(k,1) = (2,2)$ .

$$CL = \{(-1,-1,-1,-1,-1,0,-1,-1,174), \\ (-1,-1,-1,-1,-1,12,-1,-1,167)\}.$$

Step 14.  $CP = (-1,-1,-1,-1,-1,12,-1,-1,167)$ .

Step 15.  $x_{11} = 5, x_{12} = 0, x_{13} = 7, x_{14} = 6$ .

$$x_{21} = 1, x_{22} = 12, x_{23} = 0, x_{24} = 0.$$

$$\Pi(1) = 0, \Pi(2) = -1.67, \Pi(3) = -11.33$$

$$\Pi(4) = -5.67, \Pi(5) = -4.86, \Pi(6) = 0.$$

Step 16. No.

Step 18. No.

Step 19.  $z = 189$ .

Step 20. Yes.

Step 5.  $\underline{c}^G = [1.17, 7.52, 1.67, 9.33, 0.83, 1.86, 0, 2.67, 2.71, 0]$

Step 6.  $S = \{(1,1), (2,1)\}$ .

Step 7.  $r^{(1,1)} = [1, -1, -1, -1, 0, 0, 0, 0, 0, 0]$ .

$$r^{(2,1)} = [-1, 1, 1, 0, 0, 0, 0, -1, 0, 0].$$

Step 8.  $DPEN(y_{11}) = 5.83 \quad UPEN(y_{11}) = 1.67$

$$DPEN(y_{21}) = 1.67 \quad UPEN(y_{21}) = 5.83$$

Step 9.  $PEN(CP_R) = 1.67$ .

Step 10.  $168 \geq 178?$  No.

Step 11.  $Q = 5.83$ .

Step 12. Yes,  $(k,1) = (1,1)$ .

$$CL = \{(-1,-1,-1,-1,-1,0,-1,-1,174), \\ (0,-1,-1,-1,-1,12,-1,-1,172), \\ (6,-1,-1,-1,-1,12,-1,-1,168)\}.$$

Step 14.  $CP = (6,-1,-1,-1,-1,12,-1,-1,168)$ .

$$\begin{aligned} \text{Step 15. } x_{11} &= 6, & x_{12} &= 0, & x_{13} &= 7, & x_{14} &= 5 \\ x_{21} &= 0, & x_{22} &= 12, & x_{23} &= 0, & x_{24} &= 1 \\ \Pi(1) &= 0, & \Pi(2) &= 0, & \Pi(3) &= -2 \\ \Pi(4) &= -4, & \Pi(5) &= 04.86, & \Pi(6) &= 0. \end{aligned}$$

Step 16.  $168 \geq 178?$  No.

Step 18. No.

Step 19.  $z = 168$ .

Step 20. No.

Step 21.  $INC = 168$ ,  $CL = \emptyset$ .

Step 17. Yes. STOP!

Optimal Solution

$$\begin{array}{ll} x_{11} = 6 & x_{21} = 0 \\ x_{12} = 0 & x_{22} = 12 \\ x_{13} = 7 & x_{23} = 0 \\ x_{14} = 5 & x_{24} = 1 \end{array}$$

Total Cost = 168

The solution tree is as shown in Figure 6. The numbers in parentheses near a node are the estimates of the cost incurred for that partial problem. The second number is the LP cost.

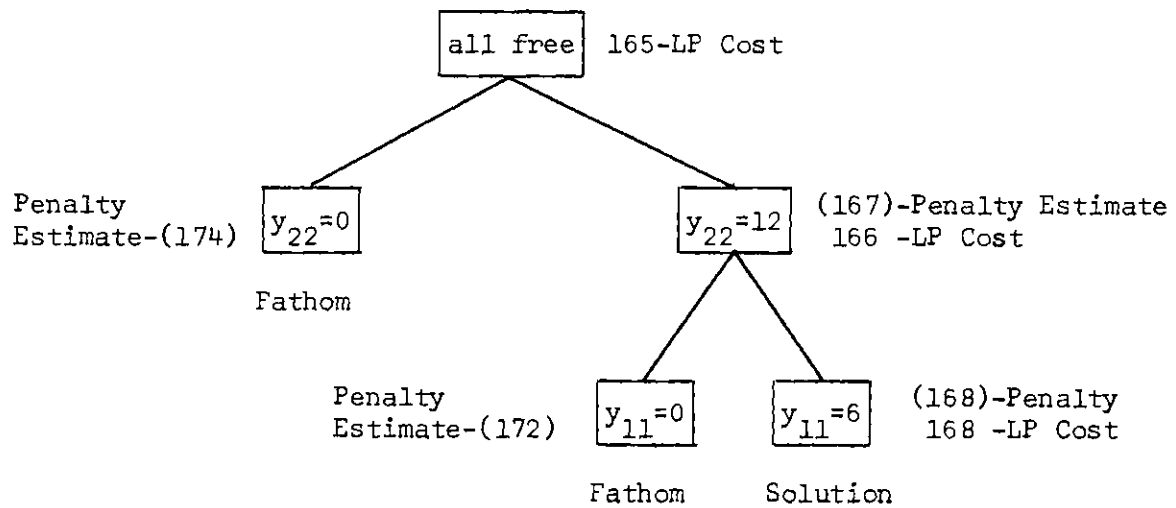


Figure 6. Solution Tree for Example 5

## CHAPTER IV

## COMPUTATIONAL EXPERIENCE

A computer code was developed for ALG-3 and was used to obtain the computational experience reported in Table 3. The code is written entirely in FORTRAN for the Univac 1108. The object program and data are all held in core with all data in fixed point mode. The program is designed to solve problems whose product,  $mn$ , is less than or equal to 100. It also has the option of using either PRIORITY or LIFO as the selection rule. The data has been packed in the candidate list and this list has a limitation of 5161 active nodes. The total number of words required for the code and data is 54,500. A listing of the code may be found in Appendix A.

The transportation code, NETWRK, was developed by Langley (28). It is a primal procedure which follows the simplex operations except that these operations are carried out on a network. This code terminates with a forest which corresponds to the optimal basis and is used directly to compute the required rows of the group problem. This code solves  $6 \times 9$  transportation problems in approximately 0.05 seconds.

Table 2 presents a summary of the problems used in this study. Since in most actual problems  $\sum_i S_i > \sum_j D_j$ , the  $n$ th destination for each problem is a dummy with unit and fixed costs equal zero. The  $S_i$ ,  $D_j$ ,  $c_{ij}$ , and  $f_{ij}$  were randomly generated according to a uniform distribution with predetermined upper and lower limits, also given in the table.

For each problem size, (4×5), (6×9), (7×11), and (5×20), three sets of problems were generated. In one set the magnitude of the fixed and variable costs were identical. The fixed cost dominated in another set and the variable cost dominated in the third.

The computational experience is presented in Table 3. The number of iterations is the average number of transportation problems solved for the problems in the respective sets. The average time excludes read-in and print-out but includes all set-up. The LIFO rule was not used after set 7 because it was estimated that the computational times would be excessive. A LIFO iteration takes slightly less computation than a PRIORITY iteration. Hence, when the average number of iterations is the same for both rules, LIFO requires slightly less time. As the problem sizes increase, PRIORITY significantly reduces the number of iterations required to solve the problem.

Table 4 gives a breakdown on the average computational time for the PRIORITY rule by problem size and cost structure. The order by cost structure is identical for the first three groups, but is changed in the (5×20) problems.

From Tables 3 and 4 it appears that ALG-3 works best for problems in which the unit cost dominates. The algorithm also works well for problems in which the relative magnitude of the two costs are the same. The discrepancy between set 8 and 11 is more difficult to explain. It is conjectured that in addition to the relative magnitude of the costs, the magnitude of  $|n-m|$  is also significant, with problems having large  $|n-m|$  easier to solve. It is believed that for a problem with  $|n-m|$  large, the initial solution of the relaxed problem saturates many of the

arcs. That is, the full fixed-charge is incurred naturally and consequently this solution is a good starting point for the algorithm. For problems with  $|n-m|$  small the converse appears to be true. Additional computational experience is needed with  $|n-m|$  varied while  $mn$  is held constant. In addition to the problems which appear in Table 3, Gray's (18) Test Problems 8 and 9 were solved in 0.5 and 12 seconds, respectively. Gray's program running on a B-5500 solved these two problems in 54.3 minutes and 25.1 minutes, respectively. Frank (11) solved 30 randomly generated  $4 \times 4$  problems on a CDC 6600 in an average time of 6.2 seconds per problem. ALG-3 solved 15 randomly generated  $4 \times 5$  problems in an average time of 0.22 seconds.

The objective of the computational experience has been to demonstrate that the theory developed in Chapter III holds promise for routinely solving moderately sized fixed-charge transportation problems (i.e.  $50 \leq mn \leq 100$ ). Even though there are many questions unanswered about this general approach, the computational times on the 54 test problems demonstrate that this approach is attractive for problems within the range tested.

Table 2. Summary of Test Problems

| Set | Size       | Parameter | Range    | Set | Size        | Parameter | Range  |
|-----|------------|-----------|----------|-----|-------------|-----------|--------|
| 1   | m=2<br>n=4 | $S_i$     | 5- 50    | 8   | m=7<br>n=11 | $S_i$     | 20-100 |
|     |            | $D_j$     | 5- 50    |     |             | $D_j$     | 10- 60 |
|     | $c_{ij}$   | 2- 10     | $c_{ij}$ |     |             | 2- 12     |        |
|     | $f_{ij}$   | 10- 60    | $f_{ij}$ |     |             | 20- 40    |        |
| 2   | m=4<br>n=5 | $S_i$     | 5- 50    | 9   | m=7<br>n=11 | $S_i$     | 20-100 |
|     |            | $D_j$     | 5- 50    |     |             | $D_j$     | 10- 60 |
|     | $c_{ij}$   | 2- 10     | $c_{ij}$ |     |             | 20- 40    |        |
|     | $f_{ij}$   | 10- 60    | $f_{ij}$ |     |             | 20- 40    |        |
| 3   | m=4<br>n=5 | $S_i$     | 5- 50    | 10  | m=7<br>n=11 | $S_i$     | 20-100 |
|     |            | $D_j$     | 5- 50    |     |             | $D_j$     | 10- 60 |
|     | $c_{ij}$   | 10- 60    | $c_{ij}$ |     |             | 20- 40    |        |
|     | $f_{ij}$   | 10- 60    | $f_{ij}$ |     |             | 2- 12     |        |
| 4   | m=4<br>n=5 | $S_i$     | 5- 50    | 11  | m=5<br>n=20 | $S_i$     | 60-100 |
|     |            | $D_j$     | 5- 50    |     |             | $D_j$     | 10- 25 |
|     | $c_{ij}$   | 10- 60    | $c_{ij}$ |     |             | 1- 5      |        |
|     | $f_{ij}$   | 2- 10     | $f_{ij}$ |     |             | 100-200   |        |
| 5   | m=6<br>n=9 | $S_i$     | 10-100   | 12  | m=5<br>n=20 | $S_i$     | 60-100 |
|     |            | $D_j$     | 5- 60    |     |             | $D_j$     | 10- 25 |
|     | $c_{ij}$   | 2- 12     | $c_{ij}$ |     |             | 100-200   |        |
|     | $f_{ij}$   | 100-200   | $f_{ij}$ |     |             | 100-200   |        |
| 6   | m=6<br>m=9 | $S_i$     | 10-100   | 13  | m=5<br>n=20 | $S_i$     | 60-100 |
|     |            | $D_j$     | 5- 60    |     |             | $D_j$     | 10- 25 |
|     | $c_{ij}$   | 100-200   | $c_{ij}$ |     |             | 100-200   |        |
|     | $f_{ij}$   | 100-200   | $f_{ij}$ |     |             | 10- 20    |        |
| 7   | m=6<br>n=9 | $S_i$     | 10-100   |     |             |           |        |
|     |            | $D_j$     | 5- 60    |     |             |           |        |
|     | $c_{ij}$   | 100-200   |          |     |             |           |        |
|     | $f_{ij}$   | 2- 12     |          |     |             |           |        |

Table 3. Summary of Computational Experience

| Set<br>Number | Size<br>m×n | Number of<br>Problems | Average Number of<br>Iterations |       | Average Time (Sec.)<br>(U1108) |       | Range of Times<br>(Sec.) |             |
|---------------|-------------|-----------------------|---------------------------------|-------|--------------------------------|-------|--------------------------|-------------|
|               |             |                       | PRIORITY                        | LIFO  | PRIORITY                       | LIFO  | PRIORITY                 | LIFO        |
| 1             | 2×4         | 5                     | 2.4                             | 2.4   | .0182                          | .0153 | .0072-.0302              | .0064-.0272 |
| 2             | 4×5         | 5                     | 14.8                            | 26.0  | .3273                          | .5264 | .1672-.6642              | .1642-1.504 |
| 3             | 4×5         | 5                     | 9.8                             | 12.4  | .2302                          | .2633 | .0474-.4906              | .0354-.6752 |
| 4             | 4×5         | 5                     | 2.6                             | 2.6   | .0885                          | .0574 | .0326-.1720              | .0246-.0772 |
| 5             | 6×9         | 3                     | 100.7                           | 217.3 | 8.345                          | 17.80 | .0750-19.53              | .0760-46.38 |
| 6             | 6×9         | 3                     | 66.7                            | 71.7  | 6.246                          | 6.822 | 4.215-8.212              | 4.208-8.329 |
| 7             | 6×9         | 3                     | 7.3                             | 7.3   | .8251                          | .6911 | .6948-.8940              | .5698-.7536 |
| 8             | 7×11        | 3                     | 1142.0                          | -     | 214.1                          | -     | 197.3-225.9              | -           |
| 9             | 7×11        | 3                     | 61.0                            | -     | 10.87                          | -     | 9.331-13.23              | -           |
| 10            | 7×11        | 3                     | 20.7                            | -     | 3.501                          | -     | 2.167-5.451              | -           |
| 11            | 5×20        | 5                     | 28.6                            | -     | 8.000                          | -     | 2.881-16.48              | -           |
| 12            | 5×20        | 5                     | 45.8                            | -     | 14.22                          | -     | 2.417-33.26              | -           |
| 13            | 5×20        | 5                     | 5.8                             | -     | 1.461                          | -     | .1888-3.252              | -           |

Table 4. Average Computational Times as a Function of Problem Size and Cost Structure

| Problem Size | Unit Cost Dominating | Costs Equal | Fixed Cost Dominating |
|--------------|----------------------|-------------|-----------------------|
| 4×5          | .0885                | .2302       | .3273                 |
| 6×9          | .8251                | 6.246       | 8.345                 |
| 7×11         | 3.501                | 10.50       | 214.1                 |
| 5×20         | 1.461                | 14.22       | 8.000                 |

## CHAPTER V

## UNIMODULAR STRUCTURE IN THE GROUP PROBLEM

Introduction

The multiparametric integer programming problem for the right-hand side is to minimize  $\underline{c}'\underline{t}$  subject to  $A\underline{t} = \underline{b}(\underline{y})$ ,  $\underline{t} \geq \underline{0}$ ,  $\underline{t} = \underline{0} \pmod{\underline{1}}$ , where  $\underline{b}(\underline{y})$  can be expressed in the form

$$\underline{b}(\underline{y}) = \underline{\bar{b}} + F(\underline{y}),$$

where  $F$  is a matrix of constant coefficients, and  $\underline{y}$  is an integer vector parameter. The group problem associated with any integer programming problem may be viewed as a multiparametric integer programming problem. The purpose of this chapter is to show that the group problem associated with the fixed-charge transportation problem can be viewed as a multiparametric integer programming problem having a totally unimodular constraint matrix.

This chapter will first present some of the properties of unimodular matrices and sets; second, present the group problem for (FCTP); third, demonstrate that the group problem is equivalent to a multiparametric integer programming problem; and last, show that the equivalent problem has a totally unimodular constraint matrix.

Properties of Unimodular Matrices and Sets

The propositions and proofs in this section are special cases of general theorems developed by Heller (21).

*DEFINITION 21*

A *linear transformation*  $T$  on  $R^n$  is a correspondence which maps each  $\underline{x} \in R^n$  into  $T(\underline{x}) \in R^m$  such that for all  $\underline{x}, \underline{y} \in R^n$  and all  $\lambda, \mu \in R$ ,  $T(\lambda\underline{x} + \mu\underline{y}) = \lambda T(\underline{x}) + \mu T(\underline{y})$ .

*DEFINITION 22*

A *basis* of the  $m \times n$  matrix  $A$  with  $\text{rank}(A) = m$  is a set of  $m$  linearly independent columns.

*DEFINITION 23*

The  $m \times n$  matrix  $A$  is said to be *totally unimodular* if every non-vanishing minor of  $A$  has absolute value 1.

*DEFINITION 24*

In  $R^n$  the set  $\{S\}$  with  $\text{rank}(S) = n$  is *unimodular* if, for every basis  $B$  of  $S$ , and every  $\underline{d} \in S$ , the elements of  $\underline{\lambda} = B^{-1}\underline{d}$  are from  $\{-1, 0, 1\}$ .

*DEFINITION 25*

In  $R^n$  the set  $\{S\}$  with  $\text{rank}(S) = n$  is *totally unimodular* if every nonvanishing minor of  $S$  has absolute value 1.

Proposition 12

If  $T: R^m \rightarrow R^m$  is a non-singular linear transformation and  $\{S\}$  is unimodular, then  $\{T(S)\}$  is unimodular.

*Proof.* Let  $\underline{p} \in T(S)$ . Let  $D$  be any basis of  $T(S)$ . Since  $T$  is non-singular, there exist a  $\underline{q} \in \{S\}$  such that  $T(\underline{q}) = \underline{p}$ . Also there exist  $\{B\} \subseteq \{S\}$  such that  $T(B) = D$ . Denote the columns of  $B$  by  $\underline{b}_1, \underline{b}_2, \dots, \underline{b}_m$ . Since  $\{S\}$  is unimodular, there exist  $\underline{\lambda} \in R^m$  such that  $\lambda_i \in \{-1, 0, 1\}$  for all  $i$ , and  $B\underline{\lambda} = \underline{q}$ . Then  $\underline{p} = T(\underline{q}) = T(B\underline{\lambda}) = T(\lambda_1 \underline{b}_1 + \lambda_2 \underline{b}_2 + \dots + \lambda_m \underline{b}_m) = \lambda_1 T(\underline{b}_1) + \lambda_2 T(\underline{b}_2) + \dots + \lambda_m T(\underline{b}_m) = T(B)\underline{\lambda} = D\underline{\lambda}$ . Hence  $\{T(S)\}$  is unimodular. This completes the proof of proposition 12.

### Proposition 13

If  $\{S\}$  is unimodular, then the determinants of all bases of  $S$  have the same absolute value.

*Proof.* Let  $B$  be a basis of  $S$ . Let  $B^*$  be a basis of  $S$  identical to  $B$  except for the  $i$ th column. Let  $\underline{d} \in \{S\}$  be the  $i$ th column of  $B^*$ . Consider the system  $B\underline{\lambda} = \underline{d}$ . By Cramer's Rule

$$\lambda_i = \frac{\det(B^*)}{\det(B)}.$$

Since  $B$  and  $B^*$  are both non-singular and  $\{S\}$  is unimodular,  $|\lambda_i| = 1$ . Hence,  $|\det(B)| = |\det(B^*)|$ . Since every basis of  $S$  can be obtained by beginning with  $B$  and successively replacing one column of  $B$ , the absolute value of the determinants of all bases is 1. This completes the proof of proposition 13.

### Proposition 14

If  $A = \{I_m | D\} \in \{R^m\}$  is unimodular, then every nonvanishing minor in  $D$  has absolute value 1.

*Proof.*  $I_m$  is a basis of  $A$ .  $|\det(I_m)| = 1$ . By proposition 13, the determinants of all bases of  $A$  have absolute value 1. Let  $R$  be any nonvanishing minor of  $D$ .

*Case 1:* Suppose  $R$  is order  $m$ . Then  $R$  is a basis of  $A$ . But all bases have determinants with absolute value 1.

*Case 2:* Suppose  $R$  is of order  $k < m$ . Denote  $\bar{R}$  as the  $m \times k$  matrix formed by completing the columns of  $R$ . Then a basis  $B$  may be formed by selecting columns of  $I_m$  to be appended to  $\bar{R}$ . Then  $B = [J \bar{R}]$ , where  $J$  consists of selected columns from  $I_m$ . After row interchanges this basis will take the form

$$B^* = \begin{bmatrix} \bar{I} & \bar{N} \\ 0 & R \end{bmatrix}.$$

$|\det(B^*)| = |\det(R)|$ . But  $|\det(B^*)| = 1$ . Hence,  $|\det(R)| = 1$ . This completes the proof of proposition 14.

#### Proposition 15

If  $A = [B|N] \in R^{m \times n}$  with  $m \leq n$ , is totally unimodular and  $B$  is a basis of  $A$ , then  $B^{-1}N$  is totally unimodular.

*Proof.* Clearly  $\{A\}$  is unimodular.  $B^{-1}$  exists and is a non-singular linear transformation. By proposition 12,  $\{B^{-1}A\} = \{C\}$  is unimodular.  $\{C\} = \{I_m | B^{-1}N\}$ . By proposition 14,  $B^{-1}N$  is totally unimodular. This completes the proof of proposition 15.

The Group Problem Associated with the  
Fixed-Charge Transportation Problem

A slightly different but equivalent formulation of the fixed-charge transportation problem will be used in this chapter. This formulation does not require that  $\sum_i S_i = \sum_j D_j$  and simplifies the theoretical development to follow. Define the problem as

$$\begin{aligned}
 (\text{FCTP}^*) \quad & \min \sum_{i=1}^m \sum_{j=1}^n \left( c_{ij} x_{ij} + \frac{f_{ij}}{\mu_{ij}} y_{ij} \right) \\
 \text{s.t.} \quad & \sum_{i=1}^m x_{ij} = D_j, \quad (\text{all } j) \\
 & \sum_{j=1}^n x_{ij} + s(S_i) = S_i, \quad (\text{all } i) \\
 & x_{ij} - y_{ij} + s(i,j) = 0, \quad (\text{all } i,j) \\
 & x_{ij}, s(S_i), s(i,j) \geq 0, \quad (\text{all } i,j) \\
 & y_{ij} \in \{0, \mu_{ij}\}, \quad (\text{all } i,j)
 \end{aligned}$$

where all variables are as defined previously and  $s(S_i)$  are slacks which absorb any excess supply.

The linear programming version formed by dropping the integrality restriction is as follows:

$$(\text{FCTP}^*_R) \quad \min \sum_{i=1}^m \sum_{j=1}^n \left( c_{ij} x_{ij} + \frac{f_{ij}}{\mu_{ij}} y_{ij} \right)$$

$$\begin{aligned}
& \text{s.t. } \sum_{i=1}^m x_{ij} = D_j, \quad (\text{all } j) \\
& \sum_{j=1}^n x_{ij} + s(S_i) = S_i, \quad (\text{all } i) \\
& x_{ij} - y_{ij} + s(i,j) = 0, \quad (\text{all } i,j) \\
& x_{ij}, y_{ij}, s(S_i), s(i,j) \geq 0, \quad (\text{all } i,j).
\end{aligned}$$

The transportation problem formed by dropping the  $y_{ij}$ 's is as follows:

$$\begin{aligned}
(\text{TP}) \quad & \min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\
& \text{s.t. } \sum x_{ij} = D_j, \quad (\text{all } j) \\
& \sum x_{ij} + s(S_i) = S_i, \quad (\text{all } i) \\
& x_{ij}, s(S_i) \geq 0, \quad (\text{all } i,j).
\end{aligned}$$

Recall from Chapter III that if one begins with an integer programming problem in the form

$$\begin{aligned}
(\text{IP}) \quad & \min \underline{c}'\underline{x} \\
& \text{s.t. } \underline{A}\underline{x} = \underline{b} \\
& \underline{x} \geq 0 \\
& \underline{x} = \underline{0} \pmod{\underline{1}},
\end{aligned}$$

that the group problem associated with (IP) is

$$(\text{GP}) \quad \min (\underline{c}'_N - \underline{c}'_B B^{-1} N) \underline{x}_N$$

$$\text{s.t. } B^{-1}N\underline{x}_N = B^{-1}\underline{b} \pmod{\underline{1}} \quad (\text{GP.1})$$

$$\underline{x}_N \geq \underline{0} \quad (\text{GP.2})$$

$$\underline{x}_N = \underline{0} \pmod{\underline{1}} \quad (\text{GP.3})$$

where  $B$  is the optimal LP basis for (IP),

$N$  is the matrix of nonbasic columns,

$\underline{x}_N$  is the vector of nonbasic variables,

$\underline{c}_N$  is the vector of costs associated with the nonbasic variables,

and

$\underline{c}_B$  is the vector of costs associated with the basic variables.

It will now be shown that when (IP) is a (FCTP\*), then  $B^{-1}N$  in (GP) will be totally unimodular.

#### Proposition 16

Suppose  $B$  is an optimal basis for (FCTP<sub>R</sub>) with one or more  $y_{ij}$  nonbasic. Then there exists an optimal basis  $B_Y$  with all  $y_{ij}$ 's basic.

*Proof.* If  $y_{ij}$  is nonbasic, then either  $x_{ij}$  or  $s(i,j)$  or both are basic. It is now shown that  $y_{ij}$  can always replace either  $x_{ij}$  or  $s(i,j)$  in the optimal tableau and the new basis will also be optimal. Assume that the problems are to maximize.

*Case 1:* Suppose  $s(i,j)$  is basic, and  $x_{ij}$  is nonbasic. The optimal tableau will be as follows:

| Row No. | Basic Var.  | ... | $x_{ij}$ | ... | $y_{ij}$ | ... | $s(i,j)$ | ... | $\underline{b}$ |
|---------|-------------|-----|----------|-----|----------|-----|----------|-----|-----------------|
| k       | $s(i,j)$    |     | 1        |     | -1       |     | 1        |     | 0 →             |
|         | $z_j - c_j$ |     | $\alpha$ |     | $\beta$  |     | 0        |     |                 |
|         |             |     |          |     | ↑        |     |          |     |                 |

After one iteration of the dual simplex algorithm replacing  $s(i,j)$  with  $y_{ij}$  the optimal tableau is

| Row No. | Basic Var.  | ... | $x_{ij}$         | ... | $y_{ij}$ | ... | $s(i,j)$ | ... | $\underline{b}$ |
|---------|-------------|-----|------------------|-----|----------|-----|----------|-----|-----------------|
| k       | $y_{ij}$    |     | -1               |     | 1        |     | -1       |     | 0               |
|         | $z_j - c_j$ |     | $\alpha + \beta$ |     | 0        |     | $\beta$  |     |                 |

Since  $\alpha, \beta \geq 0$ ,  $\alpha + \beta \geq 0$  and the optimality criterion is still met.

*Case 2:* Suppose  $s(i,j)$  is nonbasic, and  $x_{ij}$  is basic. Then the optimal tableau is

| Row No. | Basic Var.  | ... | $x_{ij}$ | ... | $y_{ij}$ | ... | $s(i,j)$ | ... | $\underline{b}$ |
|---------|-------------|-----|----------|-----|----------|-----|----------|-----|-----------------|
| k       | $x_{ij}$    |     | 1        |     | -1       |     | 1        |     | 0 →             |
|         | $z_j - c_j$ |     | 0        |     | $\alpha$ |     | $\beta$  |     |                 |
|         |             |     |          |     | ↑        |     |          |     |                 |

After one iteration of the dual simplex algorithm replacing  $x_{ij}$  with  $y_{ij}$  the optimal tableau is

| Row No. | Basic Var.  | ... | $x_{ij}$ | ... | $y_{ij}$ | ... | $s(i,j)$         | ... | $\underline{b}$ |
|---------|-------------|-----|----------|-----|----------|-----|------------------|-----|-----------------|
| k       | $y_{ij}$    |     | -1       |     | 1        |     | -1               |     | 0               |
|         | $z_j - c_j$ |     | $\alpha$ |     | 0        |     | $\alpha + \beta$ |     |                 |

Since  $\alpha, \beta \geq 0$ ,  $\alpha + \beta \geq 0$  and the optimality criterion is still met.

Case 3: Suppose both  $s(i,j)$  and  $x_{ij}$  are basic. Since the vectors associated with  $s(i,j)$  and  $y_{ij}$  are linearly dependent, the optimal tableau must take the form

| Row No. | Basic Var.  | ... | $x_{ij}$ | ... | $y_{ij}$ | ... | $s(i,j)$ | ... | $\underline{b}$ |
|---------|-------------|-----|----------|-----|----------|-----|----------|-----|-----------------|
| k       | $x_{ij}$    |     | 1        |     | 0        |     | 0        |     | 0               |
| p       | $s(i,j)$    |     | 0        |     | -1       |     | 1        |     | 0 $\rightarrow$ |
|         | $z_j - c_j$ |     | 0        |     | $\alpha$ |     | 0        |     |                 |

The dual simplex rule for selection of the entering variable is

$\max_j \frac{z_j - c_j}{d_j}$ ,  $d_j < 0$ ; where  $\underline{d}$  is the  $p$ th row of  $B^{-1}N$ . If some variable other than  $y_{ij}$  is selected to enter the basis in place of  $s(i,j)$ , then after this transformation, the tableau is in Case 2. However, if  $y_{ij}$  enters the basis in place of  $s(i,j)$ , then after one transformation the optimal tableau is

| Row No. | Basic Var.  | ... | $x_{ij}$ | ... | $y_{ij}$ | ... | $s(i,j)$ | ... | $\underline{b}$ |
|---------|-------------|-----|----------|-----|----------|-----|----------|-----|-----------------|
| k       | $x_{ij}$    |     | 1        |     | 0        |     | 0        |     | 0               |
| p       | $y_{ij}$    |     | 0        |     | 1        |     | -1       |     | 0               |
|         | $z_j - c_j$ |     | 0        |     | 0        |     | $\alpha$ |     |                 |

Since  $\alpha \geq 0$ , the optimality criterion is still met. By making all non-basic  $y_{ij}$  basic via cases 1, 2 and 3 above, an optimal basis with all  $y_{ij}$  basic can be obtained. This completes the proof of proposition 16.

Let  $B_Y$  be an optimal basis of  $(FCTP_R^*)$ . Then

$$B_Y = \left[ \begin{array}{c|c} \underline{B}_T & 0 \\ \hline \underline{P} & -I \end{array} \right] \begin{array}{l} \text{m+n rows} \\ \text{mn rows} \end{array}$$

where  $B_T$  is a basis for (TP),  $P$  is the last  $mn$  rows of the basic vectors corresponding to the  $x_{ij}$ 's and  $s(S_i)$ 's. By the partitioning procedure,

$$B_Y^{-1} = \left[ \begin{array}{c|c} \underline{B}_T^{-1} & 0 \\ \hline \underline{P}\underline{B}_T^{-1} & -I \end{array} \right]$$

With  $n$  and  $m$  any positive integers, define

$\underline{x}_{i,j}$  to be a  $mn + m + n$  vector with 1's in positions

$j, n + i, m + in + j$ , and 0's elsewhere,

$\underline{s}(S_i)$  to be a  $mn + m + n$  vector with a 1 in  $n + i$

and 0's elsewhere,

$\underline{s}(i,j)$  to be a  $mn + m + n$  vector with a 1 in position

$m + in + j$  and 0's elsewhere, and

$M$  be the matrix with columns  $\underline{x}_{i,j}, \underline{s}(S_i), \underline{s}(i,j)$  for  $i=1,2,\dots,m;$

$j = 1,2,\dots,n.$

Proposition 17

For any  $n, m$ ;  $M$  will be totally unimodular.

*Proof.* The proof is by induction on the size of the sub-matrix of  $M$ . For a  $1 \times 1$  sub-matrix the proposition is obviously true. Assume the proposition is true for  $k-1 \times k-1$  sub-matrix, and let  $A$  be any  $k \times k$  sub-matrix of  $M$ .

*Case 1:* Suppose some column of  $A$  is  $\underline{0}$ . Then clearly  $\det(A) = 0$ .

*Case 2:* Suppose some column of  $M$  has exactly one nonzero entry. Expanding by that column yields  $\det(A) = \pm \det(A^*)$ , where  $A^*$  is the cofactor of the nonzero entry and had determinant  $\pm 1$  or  $0$  by the induction hypothesis.

*Case 3:* Suppose every column of  $A$  has exactly two nonzero entries. Then the columns of  $A$  are made up of certain rows of the vectors  $\underline{x}_{i,j}$ . Suppose one of the rows  $m + n + 1$  through  $m + n + mn$  is included in  $A$ . Then that row has only one non-zero element. Expand by that row.  $\det(A) = \pm \det(A^*)$ , where  $A^*$  is the cofactor of the non-zero entry and has determinant  $\pm 1$  or  $0$  by the induction hypothesis. Suppose the rows of  $A$  are from the first  $m + n$  rows of  $M$ . Since every column will have exactly two non-zero entries, then a row from the set  $R_1 = \{1, 2, \dots, n\}$  can be associated with a row from the set  $R_2 = \{n+1, n+2, \dots, n+m\}$ . Let  $I_1 \subseteq R_1$ ,  $I_2 \subseteq R_2$  denote the rows of  $M$  used in  $A$ . Let  $\underline{a}'_j$ ,  $j \in I_1 \cup I_2$  denote the rows of  $A$ . Then

$$\sum_{j \in I_1} \underline{a}'_j = \sum_{j \in I_2} \underline{a}'_j$$

Hence  $\det(A) = 0$ .

*Case 4:* Suppose there exists one or more column of  $A$  with three non-zero entries with the remaining columns having two non-zero entries. Then a column with three non-zero entries consists of certain elements of one of the  $\underline{x}_{ij}$ 's. Also one of the rows  $m + n + 1$  through  $m + n + mn$  is included in  $A$ . Then that row has only one non-zero element. Expand by that row.  $\det(A) = \pm \det(A^*)$ , where  $A^*$  is the cofactor of the non-zero entry and has determinant  $\pm 1$  or  $0$  by the induction hypothesis. Hence, by induction on  $k = 1$ , the proposition is proved.

Let  $N$  denote the nonbasic columns of  $(FCTP_R^*)$ . Suppose the columns of  $N$  are in the order  $\underline{x}_{ij}$ ,  $\underline{s}(S_i)$  and then  $\underline{s}(i,j)$ . Then

$$N = \begin{bmatrix} N_1 & 0 \\ N_2 & I \end{bmatrix} \begin{array}{l} m + n \text{ rows} \\ mn \text{ rows} \end{array}$$

where  $N_1$  and  $N_2$  consist of the nonbasic  $\underline{x}_{ij}$ 's and  $\underline{s}(S_i)$ 's. Then the group problem constraint matrix is

$$B_Y^{-1}N = \left[ \begin{array}{c|c} B_T^{-1}N_1 & 0 \\ \hline PB_T^{-1}N_1 - N_2 & -I \end{array} \right]$$

The constraint (GP.1) is

$$(5.1) \quad [B_T^{-1}N_1 \quad 0] \underline{x}_N = B_T^{-1} \underline{b}_1 \pmod{\underline{1}}$$

$$(5.2) \quad [PB_T^{-1}N_1 - N_2 \quad -I]x_N = PB_T^{-1}b_{-1} \pmod{\underline{\mu}}$$

where  $\underline{\mu}$  is the vector of  $\mu_{ij}$ 's in the appropriate order. Equation (5.2) is equivalence mod  $\underline{\mu}$  since the basic  $y_{ij}$ 's are restricted to the set  $y_{ij} \in \{0, \mu_{ij}\}$ .

Since  $B_T$  is a basis of (TP) and  $B_T$  is a submatrix of  $M$ , by proposition 6,  $B_T^{-1}b_{-1}$  is an integer vector.

Further, by proposition 13,  $N_1$  is totally unimodular. Therefore constraint (5.1) is satisfied for any integer selection of  $x_N$  and the group constraint set reduces to only constraint (5.2). Note that  $P$  ( $mn \times m+n$ ) can have at most one non-zero entry in any row. Denote the rows of  $P$  by the double subscript  $i,j$  so that the  $(i,j)$ th row will have a non-zero entry if  $x_{ij}$  is basic. This implies that the  $(i-1)n + j$ th row of  $N_2$  will consist of only zeros. Then  $PB_T^{-1}N_1 - N_2$  either deletes some rows of  $B_T^{-1}N_1$  and replaces them with rows with a single non-zero entry or leaves the rows unchanged.

Proposition 18

If  $B_T^{-1}N_1$  is totally unimodular, then  $PB_T^{-1}N_1 - N_2$  is totally unimodular.

*Proof.* Select any square submatrix,  $C$ , from  $PB_T^{-1}N_1 - N_2$ . Suppose  $C$  is also a submatrix of  $B_T^{-1}N_1$ . Then  $|\det(C)| = 1$  or  $0$  since  $B_T^{-1}N_1$  is totally unimodular. Suppose  $C$  is not a submatrix of  $B_T^{-1}N_1$ .

*Case 1:* Suppose some row of  $C$  contains all zeros. Then  $\det(C) = 0$ .

*Case 2:* Suppose one or more rows of  $C$  have a single non-zero

entry which originated in  $N_2$ . Expanding by these rows either reduces  $C$  to a single non-zero entry in which case  $|\det(C)| = 1$ , or results in an adjoint which is a submatrix of  $B_T^{-1}N_1$ . Hence  $|\det(C)| = 0$  or  $1$ . Therefore  $PB_T^{-1}N_1 - N_2$  is totally unimodular. This completes the proof of proposition 18.

The group problem for the fixed-charge transportation problem is

$$\begin{aligned}
 (5.3) \quad & \min \quad (\underline{c}'_B B_Y^{-1} N - \underline{c}'_N) \underline{x}_N \\
 & \text{s.t.} \quad [(PB_T^{-1}N_1 - N_2) \quad -I] \underline{x}_N = PB_T^{-1} \underline{b}_1 \pmod{\underline{\mu}} \\
 & \quad \quad \quad \underline{x}_N = \underline{0} \pmod{\underline{1}} \\
 & \quad \quad \quad \underline{x}_N \geq \underline{0}.
 \end{aligned}$$

From consideration of the original problem it is easily seen that each of the nonbasic variables is bounded. That is,  $x_{ij} \leq \mu_{ij}$ ,  $s(i,j) \leq \mu_{ij}$ , and  $s(S_i) \leq S_i$ . Let  $\underline{q}$  denote the vector of upper bounds for the nonbasic variables.

Since (5.3) is a group problem, there are an infinite number of representations of the right-hand side  $PB_T^{-1} \underline{b}_1$ . Consider only the  $i$ th group equation  $\underline{h}'_i \underline{x}_N = g_i \pmod{\mu_i}$ . Since  $\underline{x}_N$  is bounded,  $g_i$  can assume only a finite number of values all of the form  $g_i + k\mu_i$  with  $k \in Z$ . Let  $\Omega$  be the set of values which the  $i$ th right-hand side may assume. Let  $\Omega = (z_1, z_2, \dots, z_k)$ :  $z_1 \in \Omega_1, z_2 \in \Omega_2, \dots, z_{mn} \in \Omega_k, k = mn$ . Then there exists a smallest element of  $\Omega$  denoted  $\underline{s}$ . Then the group problem can be rewritten as

$$\begin{aligned}
 (5.4) \quad & \min(c'_B B_Y^{-1} N - c'_N) \underline{x}_N \\
 & \text{s.t. } [(PB_T^{-1} N_1 - N_2) \quad -I] \underline{x}_N = \underline{s} \pmod{\underline{\mu}} \\
 & \underline{0} \leq \underline{x}_N \leq \underline{q} \\
 & \underline{x}_N = \underline{0} \pmod{\underline{1}}.
 \end{aligned}$$

The Multiparametric Integer Programming Problem

*DEFINITION 26*

The bounded multiparametric integer programming problem for the right-hand sides is defined as follows:

$$\begin{aligned}
 \min Z(\underline{v}) &= \underline{d}' \underline{t} \\
 \text{s.t. } A \underline{t} &= \underline{f} + F \underline{v} \\
 \underline{0} &\leq \underline{t} \leq \underline{w} \\
 \underline{0} &\leq \underline{v} \leq \underline{r} \\
 \underline{t}, \underline{v} &= \underline{0} \pmod{\underline{1}}.
 \end{aligned}$$

The group problem (5.4) can be placed in the form of a multiparametric integer programming problem by setting

$$\begin{aligned}
 \underline{d}' &= c'_B B_Y^{-1} N - c'_N \\
 \underline{t} &= \underline{x}_N
 \end{aligned}$$

$$\underline{w} = \underline{q}$$

$$A = [PB_T^{-1}N_1 - N_2 \quad -I]$$

$$\underline{f} = \underline{s}$$

$$\underline{F} = K$$

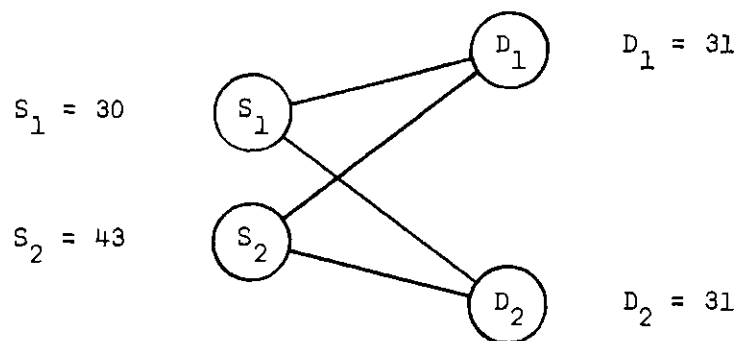
where  $K$  is a diagonal matrix with diagonal elements  $\mu_{ij}$ , and

$\underline{r}$  is determined from  $\Omega$ .

Further, since  $PB_T^{-1}N_1 - N_2$  is totally unimodular  $A$  is totally unimodular. Therefore, the group problem associated with the fixed-charge transportation problem may be viewed as a multiparametric integer programming problem with a totally unimodular constraint matrix.

#### Example 6

Consider the  $2 \times 2$  problem



|               |                              |
|---------------|------------------------------|
| $c_{11} = 5$  | $f_{11} = (400)(30) = 12000$ |
| $c_{12} = 10$ | $f_{12} = (200)(30) = 6000$  |
| $c_{21} = 6$  | $f_{21} = (300)(31) = 9300$  |
| $c_{22} = 4$  | $f_{22} = (100)(31) = 3100$  |

$$B_Y = \left[ \begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{array} \right]$$

$$B_T = \left[ \begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right]$$

$$B_T = \left[ \begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right]$$

$$B_T^{-1} = \left[ \begin{array}{cccc} 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 \end{array} \right]$$

$$P = \left[ \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right]$$

$$B_Y^{-1} = \begin{bmatrix} 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$N = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \bar{N}_1 & 0 \\ N_2 & I \end{bmatrix}$$

$$B_Y^{-1}N = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} B_T^{-1}N_1 & | & 0 \\ \hline PB_T^{-1}N_1 - N_2 & | & -I \end{bmatrix}$$

$$c_B' = [-5 \quad -6 \quad -4 \quad 0 \quad -400 \quad -200 \quad -300 \quad -100]$$

$$\underline{c}'_B B_Y^{-1} \underline{N} - \underline{c}'_N = [7 \quad 99 \quad 400 \quad 200 \quad 300 \quad 100]$$

$$PB_T^{-1} = \begin{bmatrix} 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\underline{b}'_1 = [31 \quad 31 \quad 30 \quad 43]$$

$$PB_T^{-1} \underline{b}'_1 = \begin{bmatrix} 19 \\ 0 \\ 12 \\ 31 \end{bmatrix}$$

Group Problem:

$$\min \quad 7x_{12} + 99s(S_2) + 400s(1,1) + 200s(1,2) + 300s(2,1) + 100s(2,2)$$

s.t.

$$\begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{12} \\ s(S_2) \\ s(1,1) \\ s(1,2) \\ s(2,1) \\ s(2,2) \end{bmatrix} = \begin{bmatrix} 19 \pmod{30} \\ 0 \pmod{30} \\ 12 \pmod{31} \\ 31 \pmod{31} \end{bmatrix}$$

$x_{12}, s(S_2), s(1,1), s(1,2), s(2,1), s(2,2) \geq 0$ , and integer

The bounds for the nonbasic variables are

$$\begin{array}{ll}
 0 \leq x_{12} \leq 30 & 0 \leq s(1,2) \leq 30 \\
 0 \leq s(S_2) \leq 11 & 0 \leq s(2,1) \leq 31 \\
 0 \leq s(1,1) \leq 30 & 0 \leq s(2,2) \leq 31.
 \end{array}$$

Then

$$\Omega_1 = \{-41, -11, 19\}$$

$$\Omega_2 = \{-60, -30, 0\}$$

$$\Omega_3 = \{-50, -19, 12\}$$

$$\Omega_4 = \{0\}.$$

Then  $\underline{s}'$  is  $(-41, -60, -50, 0)$ .

Multiparametric Integer Programming Problem:

$$\min 7t_1 + 99t_2 + 400t_3 + 200t_4 + 300t_5 + 100t_6$$

$$\text{s.t. } \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \underline{t} = \begin{bmatrix} -41 \\ -60 \\ -50 \\ 0 \end{bmatrix} + \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 31 & 0 \\ 0 & 0 & 0 & 31 \end{bmatrix} \underline{y}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \underline{t} \leq \begin{bmatrix} 30 \\ 11 \\ 30 \\ 30 \\ 31 \\ 31 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \underline{v} \leq \begin{bmatrix} 3 \\ 3 \\ 3 \\ 0 \end{bmatrix}$$

$$\underline{t}, \underline{v} = \underline{0} \pmod{\underline{1}}.$$

## CHAPTER VI

## CONCLUSIONS AND RECOMMENDATIONS

The principal objective of this study was to develop a group theoretic algorithm for the fixed-charge transportation problem. This algorithm is presented in Chapter III and computational experience with the method is given in Chapter IV. The largest problems previously solved optimally and reported in the literature had six sources and eight destinations. Problems with various cost structures and  $mn = 100$  have been successfully solved using the algorithm of this research. Further, it is shown that the group problem associated with (FCTP\*) may be viewed as a multiparametric integer programming problem having a totally unimodular constraint matrix.

During the course of this investigation numerous other questions arose and remain unanswered. Six of these which are a logical extension of this work are listed below.

i. *Surrogating Rows of the Group Problem.* ALG-3 uses the group problem to develop penalties by consideration of only a single row at a time. Consequently, any interaction information between two rows of the group problem is lost. It is conjectured that these rows can be surrogated in some manner to yield better information from which to obtain penalties.

ii. *Additional Computational Experience.* Additional computational experience is needed to determine if there is a significant

difference between problems with  $|n-m|$  large and  $|n-m|$  small with the product,  $mn$ , held constant.

iii. *Extension to the Fixed-Cost Flow Problem.* The fixed-cost flow problem for a network  $[N,A]$  may be defined as follows:

$$\begin{aligned}
 \text{(FCFP)} \quad & \min \sum_{(x,y) \in A} a(x,y)f(x,y) + b(x,y)d(x,y) \\
 & \text{st. } f(x,N) - f(N,x) = \begin{cases} s(x), & x \in S \\ 0, & x \in R \\ -t(x), & x \in T \end{cases} \\
 & 0 \leq f(x,y) \leq c(x,y), \quad (x,y) \in A \\
 & \text{and} \\
 & d(x,y) = \begin{cases} 1, & \text{if } f(x,y) > 0 \\ 0, & \text{otherwise.} \end{cases}
 \end{aligned}$$

where  $a(x,y)$  = the unit cost to use  $(x,y)$ ,  
 $f(x,y)$  = the flow in  $(x,y)$ ,  
 $b(x,y)$  = the fixed charge for  $(x,y)$ ,  
 $c(x,y)$  = the capacity of  $(x,y)$ ,  
 $S$  = the set of sources,  
 $T$  = the set of destinations,  
 $R$  =  $N - S - T$ ,  
 $s(x)$  = the supply at source  $x$ , and  
 $t(y)$  = the demand at destination  $y$ .

Zavarei and Frisch (36) have shown that (FCFP) can be converted to a (FCTP) at the expense of additional nodes and arcs. Clearly, (FCFP) can be solved by converting to (FCTP) and applying ALG-3. However, it is conjectured that the theory developed in Chapter III can be extended to solve (FCFP) directly.

iv. *Unimodularity of the Group Problem.* Intuitively, it appears that viewing the group problem as a multiparametric integer programming problem having a totally unimodular constraint matrix would be useful in constructing a solution procedure for the fixed-charge transportation problem. The full strength of this information has not been exploited for ALG-3. There is need for further work investigating means of solving a multiparametric integer programming problem.

v. *Penalty Strength.* There is need for a computational study designed to determine the value of the use of penalties to direct the algorithm. This would involve resolving the 54 test problems using only the LP bound to guide the algorithm and comparison with the results in Table 3.

vi. *Selection Rules.* ALG-3 used the largest penalty to determine the variable to be fixed at the next iteration. Numerous other possibilities for the selection rule exist, and additional information about these rules would be useful.

APPENDIX A  
 FORTRAN CODE FOR ALG-3

```

C PROGRAM TO SOLVE THE FIXED-CHARGE TRANSPORTATION PROBLEM BY ALG-3
C KENNINGTON
C
C VARIABLES
C
C M=NO. OF SOURCES
C N=NO. OF DESTINATIONS
C IS(I)=THE SUPPLY AT SOURCE I
C ID(J)=THE DEMAND AT DESTINATION J
C IC(K)=THE UNIT COSTS
C IF(K)=THE FIXED COSTS
C IU(K)=THE ARC CAPACITIES
C ADJ=AN ADJUSTMENT FACTOR
C ITYPE=A CONTROL VARIABLE ON THE SELECTION PROCEDURE
C (ITYPE=1 IMPLIES PRIORITY, ITYPE=2 IMPLIES LIFO)
C
C
C INPUT
C
C CARD 1 - PROBLEM IDENTIFICATION
C
C CARD 2 - M,N,ADJ,ITYPE (NO. OF SOURCES, NO. OF DESTINATIONS,
C ADJUSTMENT FACTOR TO CHANGE THE FIXED COSTS DIVIDED
C BY THE ARC CAPACITIES INTO AN INTEGER, SELECTION
C RULE)---ALL VARIABLES ARE INTEGER AND ARE READ
C BY (13I6) SPECIFICATION
C
C CARDS 3,4,ETC. - SUPPLIES,DEMANDS, UNIT COSTS WITH THE
C SECOND SUBSCRIPT CHANGING MOST RAPIDLY,
C FIXED COSTS WITH THE SECOND SUBSCRIPT
C CHANGING MOST RAPIDLY. ALL VARIABLES ARE
C INTEGERS AND ARE READ BY (13I6)
C SPECIFICATION
C
C
C RESTRICTION - THE PROGRAM IS RESTRICTED TO PROBLEMS WITH MN
C .LE. 100.
C
C FUNCTION INDEX
C
C INDEX(I,J) = (I-1)*N + J
C
C COMMON M,N,COST(162),LABEL1(62),LABEL2(62),LABEL3(62),PI(62),

```

```

* NL(62,62),MBIG,MN,NPP,NQ,NQQ,NA,NT,          LX(162),
*IX(162),IY(162),
* JEFF,BOUND(5161),INC,LCL,CL(28672),VCPR,CP(100),JLKL(5161),
* SCOST(162),IF(100),IU(100),IC(100),JUDY,TITLE(14),ISOL(100),ITER
* ,JLKF(100),JLKC(100)
INTEGER COST,SCOST,F(162),O9J(200),ADJ,OB,CL,CP,PI,UPEN(100),
* DPEN(100),BOUND,UP,DP,VCPR,Q,PEN,Z
DIMENSION IS(50),ID(50),NX(100,2),NSIJ(100,2),LAM(62),KY(100),
*JY(100)
EQUIVALENCE(F(1),LX(1))
C
NPRT = 100
10 READ 3, TITLE
3  FORMAT(13A6,A2)
PRINT 4,TITLE
4  FORMAT(1H1,13A6,A2 )
C
READ 1,M,N,ADJ,ITYPE
1  FORMAT(13I6)
MN=M*N
IF(MN.LE.100) GO TO 110
C
ERROR MESSAGE
C
IRR=1
100 PRINT 2,IRR
2  FORMAT(6H1ERROR,I5)
GO TO 10
2000 FORMAT(//)
C
110 READ 1,(IS(I),I=1,M) , (ID(I),I=1,N) , (IC(I),I=1,MN),(IF(I),I=1,
1 MN)
C
PRINT PROBLEM
C
PRINT 5,M,N,ADJ,ITYPE
5  FORMAT(15HNO. SOURCES = I3,5X,19HNO. DESTINATIONS = I3, 5X I8,
1 5X I2)
PRINT 9
9  FORMAT(7H0SUPPLY //)
PRINT 7,(IS(I),I=1,M)
7  FORMAT(21I6)
PRINT 8
8  FORMAT(7H0DEMAND //)
PRINT 7,(ID(I),I=1,N)
PRINT 2000
DO 120 I=1,M
DO 120 J=1,N
K=INDEX(I,J)
120 PRINT 6,I,J,IC(K),I,J,IF(K)
6  FORMAT(3HOC( I2,1H, I2, 4H) = I6,5X 2HF( I2,1H,I2,4H) = .I6)
C
CALCULATE ARC CAPACITIES
C
I = ITIME(J,JUDY)
DO 130 I=1,M

```



```

DO 191 I = NQ,NPP
191 LX(I) = IS(I-MN)
DO 192 I = II,NQQ
192 LX(I)=ID(I-NPP)
K = 0
DO 193 I = 1,M
DO 193 J = 1,N
II = M+J
K = K+1
LX(K) = 0
NL(I,II) = K
NL(II,I) = K
IX(K) = I
193 IY(K) = II
DO 194 I = 1,NB
NZ = MN + I
194 NL(NT,I) = NZ
CALL NETWRK
GO TO (209,207), JEFF
207 IRR=4
GO TO 100
209 CONTINUE
CALL SOLCST
C
C     STEPS 3 AND 4
C
INC=0
KOOL=1
DO 211 K = 1,MN
IF(F(K).EQ.0) GO TO 212
IF(F(K).EQ.IU(K)) GO TO 213
KOOL=2
213 INC = INC + IF(K)*IU(K)+IC(K)*F(K)
212 ISOL(K)=F(K)
211 CONTINUE
GO TO (210,220),KOOL
210 CALL OUTPUT
GO TO 10
C
C     STEP 5
C
220 CONTINUE
NXX = 0
NSSIJ = 0
DO 570 I = 1,M
DO 570 J = 1,N
K = INDEX(I,J)
IF(CP(K).EQ.1) GO TO 560
IF(CP(K).EQ.3.AND.F(K).EQ.0) GO TO 550
GO TO 570
560 NSSIJ = NSSIJ+1
NSIJ(NSSIJ,1) = I
NSIJ(NSSIJ,2) = J
IF(F(K).EQ.0) GO TO 550
GO TO 570
550 NXX = NXX + 1

```

```

        NX(NXX,1)=I
        NX(NXX,2) = J
570    CONTINUE
        IF(NXX.EQ.0) GO TO 575
        DO 580 I = 1,NXX
            II = NX(I,1)
            JJ = NX(I,2)
            K = INDEX(II,JJ)
            IF(CP(K).EQ.3) GO TO 571
            OBJ(I) = PI(JJ+M)-PI(II)+SCOST(K)
            GO TO 580
571    OBJ(I) = PI(JJ+M)-PI(II)+IC(K)
580    CONTINUE
575    CONTINUE
        IF(NSSIJ.EQ.0) GO TO 630
        DO 590 I = 1,NSSIJ
            II = NSIJ(I,1)
            JJ = NSIJ(I,2)
            K = INDEX(II,JJ)
            OBJ(I+NXX) = IF(K)
590    CONTINUE
C
630    CONTINUE
        NNN = M + N
        KOOL = NXX + NSSIJ
C
C        STEPS 7,8, AND 9
C
        NODE = MN1+M+N
C
        LUCK=0
        DO 700 I=1,M
            DO 700 J = 1,N
                K=INDEX(I,J)
                IF(CP(K).GT.1) GO TO 700
                IF(F(K).EQ.0.OR.F(K).EQ.IU(K))GO TO 700
                DO 640 II = 1,NNN
540    LAM(II) = 0
                IF(LABEL1(II).EQ.J+M) GO TO 650
                LABEL = -1
                KEY = J + M
                LAM(KEY) = LABEL
                K=LABEL2(KEY)
541    IF(K.EQ.0) GO TO 660
542    IF(K.EQ.0) GO TO 645
543    LAM(K) = LABEL
                KK = K
                K=LABEL2(K)
                GO TO 642
C
545    K=LABEL3(KK)
                IF(K.EQ.0) GO TO 647
                GO TO 643
C
547    K=LABEL1(KK)
                IF(K.EQ.KEY) GO TO 660

```

```

        IF(LABEL3(K).NE.0) GO TO 648
        KK=K
        GO TO 647
C
648    K=LABEL3(K)
        GO TO 643
C
650    KEY = I
        LABEL = 1
        LAM(KEY) = 1
        K = LABEL2(KEY)
        GO TO 641
C
C      EVERYTHING IS NOW LABELED CORRECTLY
C      CALCULATE THE ROW
C
660    CONTINUE
        LUCK=LUCK+1
        UP = 99999999
        DP = 99999999
        IF(NXX.EQ.0) GO TO 685
        DO 670 K=1,NXX
        II=NX(K,1)
        JJ=NX(K,2)
        KEY = LAM(II) - LAM(JJ+M)+2
        OB = OBJ(K)
        GO TO (661,670,663),KEY
661    IF(UP.LE.OB) GO TO 670
        UP = OB
        GO TO 670
663    IF(DP.LE.OB) GO TO 670
        DP=OB
670    CONTINUE
C
685    IF(NSSIJ.EQ.0) GO TO 695
        DO 690 K=1,NSSIJ
        II = NSIJ(K,1)
        JJ = NSIJ(K,2)
        KK = INDEX(II,JJ)
        IF(II.EQ.I.AND.JJ.EQ.J) GO TO 671
        GO TO 690
671    KEY = LAM(JJ+M)-LAM(II)+2
        OB = OBJ(K+NXX)
        GO TO (686,690,687),KEY
686    IF(UP.LE.OB) GO TO 690
        UP = OB
        GO TO 690
687    IF(DP.LE.OB) GO TO 690
        DP=OB
690    CONTINUE
695    CONTINUE
        KY(LUCK) = I
        JY(LUCK)=J
        K=INDEX(I,J)
        UPEN(LUCK)=UP*(IU(K)-F(K))
        DPEN(LUCK)=DP*F(K)

```

```

700  CONTINUE
C
C      STEP 10
C
      IF(LUCK.GT.0) GO TO 750
      IRR = 5
      GO TO 100
750  CONTINUE
      PEN=0
      DO 920 I=1,LUCK
      Z = MIN0(UPEN(I),DPEN(I))
      IF(Z.LE.PEN) GO TO 920
      PEN = Z
920  CONTINUE
C
C      STEP 12
C
      Q = -999999999
      KEY=0
      DO 940 I=1,LUCK
      IF(UPEN(I).LE.Q) GO TO 930
      Q=UPEN(I)
      KEY=I
      JEFF=1
930  IF(DPEN(I).LE.Q)GO TO 940
      Q=DPEN(I)
      KEY=I
      JEFF = 2
940  CONTINUE
      GO TO (950,1000), JEFF
C
C      STEP 13
C
950  CONTINUE
      II = KY(KEY)
      JJ=JY(KEY)
      K=INDEX(II,JJ)
      KOOL = VCPR + Q
      IF(KOOL.GE.INC) GO TO 961
      LCL = LCL + 1
      CP(K)=3
      DO 960 I=1,MN
      II=MN*LCL + I - MN
      JK = (II-1)/18+1
960  FLD(2*(II-(JK-1)*18)-2,2,CL(JK)) = CP(I)
      BOUND(LCL) = VCPR+Q
      NCP = NCP + 1
      JLKL(LCL) = NCP
961  CONTINUE
      KOOL = VCPR + PEN
      IF(KOOL.GE.INC) GO TO 1034
      LCL=LCL+1
      KK=LCL*MN
      IF(KK.LE.516096.AND.LCL.LE.5161) GO TO 970
      IRR=3
      GO TO 100

```

```

C
970  CP(K)=2
      DO 980 I=1,MN
          II=MN*(LCL-1)+I
          JK = (II-1)/18+1
980  FLD(2*(II-(JK-1)*18)-2,2,CL(JK)) = CP(I)
          BOUND(LCL)=VCPR+PEN
          NCP = NCP + 1
          JLKL(LCL) = NCP
          GO TO 1034

C
C      STEP 14
C
1000 CONTINUE
      II = KY(KEY)
      JJ=JY(KEY)
      K=INDEX(II,JJ)
      KOOL = VCPR + Q
      IF(KOOL.GE.INC) GO TO 1011
      LCL = LCL + 1
      CP(K) = 2
      DO 1010 I=1,MN
          II=MN*(LCL-1)+I
          JK = (II-1)/18+1
1010 FLD(2*(II-(JK-1)*18)-2,2,CL(JK)) = CP(I)
          BOUND(LCL)=VCPR+Q
          NCP = NCP + 1
          JLKL(LCL) = NCP

C
1011 KOOL = VCPR + PEN
      IF(KOOL.GE.INC) GO TO 1034
      LCL=LCL+1
      KK=LCL*MN
      IF(KK.LE.516096.AND.LCL.LE.5161) GO TO 1020
      IRR=3
      GO TO 100

C
1020 CP(K)=3
      DO 1030 I=1,MN
          II = MN*(LCL-1) + I
          JK = (II-1)/18+1
1030 FLD(2*(II-(JK-1)*18)-2,2,CL(JK)) = CP(I)
          BOUND(LCL)=VCPR+PEN
          NCP = NCP + 1
          JLKL(LCL) = NCP

C
C      STEP 15
C
1034 CONTINUE
      IF(LCL.GE.1) GO TO 1031
      CALL OUTPUT
      GO TO 10
1031 CONTINUE
      GO TO(1035,1065),ITYPE
1035 CONTINUE
      KEY = 0

```

```

LARG = M8IG
DO 1040 I=1,LCL
IF(BOUND(I).GE.LARG) GO TO 1040
LARG = BOUND(I)
KEY = I
1040 CONTINUE
IF(KEY.GT.0) GO TO 1041
IRR = 8
GO TO 100

C
C PLACE THIS IN CP
C
1041 CONTINUE
DO 1050 I=1,MN
J = MN*(KEY-1)+I
JK = (J-1)/18+1
JL = 2*(J-(JK-1)*18)-2
CP(I) = FLD(JL,2,CL(JK))
K=MN*(LCL-1)+I
JM = (K-1)/18+1
1050 FLD(JL,2,CL(JK)) = FLD(2*(K-(JM-1)*18)-2,2,CL(JM))
BOUND(KEY) = BOUND(LCL)
JLKL(KEY) = JLKL(LCL)
LCL=LCL-1
GO TO 1100

C
C LIFO
C
1065 KEY = 1
LARG = 0
DO 1051 I = 1,LCL
IF(LARG.GT.JLKL(I)) GO TO 1051
LARG = JLKL(I)
KEY = I
1051 CONTINUE
GO TO 1041

C
C STEP 16
C
1100 CONTINUE
DO 1110 I=1,MN
1110 COST(I)=SCOST(I)
C
DO 1130 K = 1,MN
KEY = CP(K)
GO TO(1130,1120,1125),KEY
1120 COST(K) = M8IG
GO TO 1130
1125 COST(K)=IC(K)
1130 CONTINUE
II = NPP + 1
DO 1132 I = 1,MN
1132 LX(I) = 0
DO 1133 I = NQ,NPP
1133 LX(I) = IS(I-MN)
DO 1134 I = II , NQQ

```

```

1134 LX(I) = IO(I-NPP)
      ITER = ITER + 1
C
      CALL NETWRK
C
C
C
      GO TO (1131,1140), JEFF
C
      STEP 17
C
1131 CONTINUE
      DO 1136 K = 1,MN
      IF(CP(K).EQ.2.AND.F(K).GT.0) GO TO 1140
1136 CONTINUE
      CALL SOLCST
C
      IF(VCPR.GE.INC) GO TO 1140
      GO TO 1150
C
      STEP 18
C
1140 IF(LCL.NE.0) GO TO 1031
      CALL OUTPUT
      GO TO 10
C
      STEPS 19 AND 20
C
1150 JEFF=0
      LYNN=1
      DO 1200 K = 1,MN
      IF(F(K).EQ.0) GO TO 1200
      IF(CP(K).EQ.3) GO TO 1180
      IF(F(K).EQ.IU(K))GO TO 1180
      LYNN=2
1180 JEFF=JEFF+IF(K)*IU(K)+IC(K)*F(K)
1200 CONTINUE
      IF(JEFF.GE.INC) GO TO 1055
      INC = JEFF
      DO 1060 I=1,MN
1060 ISOL(I)=F(I)
      CALL UPDATE
1055 GO TO (1034,220),LYNN
      END

```

```

SUBROUTINE SOLCST
C
COMMON M,N,COST(162),LABEL1(62),LABEL2(62),LABEL3(62),PI(62),
* NL(62,62),MBIG,MN,PP,NO,NOR,NB,NT,          LX(162),
* IX(162),IY(162),
* JEFF,BOUND(5161),I,C,LCL,CL(28672),VCPR,CP(100),JLKL(5161),
* SCOST(162),IF(100),IU(100),IC(100),JUDY,TITLE(14),ISOL(100),ITER
* JLKF(100),JLKC(100)
INTEGER COST,SCOST,F(162),          CL,CP,PI,
* BOUND,          VCPR
EQUIVALENCE(F(1),LX(1))
C
VCPR=0
DO 928 K=1,MN
KEY=CP(K)
GO TO (921,928,923),KEY
921 VCPR=VCPR+SCOST(K)*F(K)
GO TO 928
923 VCPR=VCPR+IF(K)*IU(K)+IC(K)*F(K)
928 CONTINUE
RETURN
END

```

```

SUBROUTINE OUTPUT
COMMON M,N,COST(162),LABEL1(62),LABEL2(62),LABEL3(62),PI(62),
* NL(62,62),MRIG,MN,PPP,NQ,NOO,NB,NT,          LX(162),
*IX(162),IY(162),
* JEFF,BOUND(5161),IIC,LCL,C(28672),VCPR,CP(100),JLKL(5161),
* SCOST(162),IF(100),IU(100),IC(100),JUDY,TITLE(14),ISOL(100),ITER
* ,JLKF(100),JLKC(100)
INTEGER COST,SCOST,          CL,CP,PI,
* BOUND,          VCPR
C
I = ITIME(J,JUD)
PRINT 4,TITLE
4  FORMAT(1H1,13A6,A2)
C
PRINT 5
5  FORMAT(7H0SOURCE,5X,11HDESTINATION,5X 4HFLOW // )
DO 10 I=1,M
DO 10 J=1,N
K = (I-1)*N + J
10  PRINT 6,I,J,ISOL(K)
6  FORMAT(15,5X,19,4X,19)
TIME = (JUD-JUDY)/5000.
PRINT 5000,TIME
5000  FORMAT(1H0, 7HTIME = F10.4)
PRINT 6000,ITER
6000  FORMAT(21H0NO. OF ITERATIONS = 19)
KOOL = 0
DO 20 I = 1,MN
IF(ISOL(I).EQ.0) GO TO 20
KOOL = KOOL + ISOL(I)*JLKC(I)+JLKF(I)
20  CONTINUE
PRINT 7000,KOOL
7000  FORMAT(13HTOTAL COST = I10)
RETURN
END

```

```

SUBROUTINE UPDATE
C
COMMON M,N,COST(162),LABEL1(62),LABEL2(62),LABEL3(62),PI(62),
* NL(62,62),MBIG,MN,IPP,NQ,NQR,NB,NT,          LX(162),
* IX(162),IY(162),
* JEFF,BOUND(5161),INC,LCL,CL(28672),VCPR,CP(100),JLKL(5161),
* SCOST(162),TF(100),IU(100),IC(100),JUDY,TITLE(14),ISOL(100),ITER
* ,JLKF(100),JLKC(100)
INTEGER COST,SCOST,          CL,CP,PI,
*          BOUND,          VCPR
C
IF(LCL.EQ.0) RETURN
C
I=1
1041 IF(BOUND(I).LT.INC) GO TO 1050
IF(I.LT.LCL) GO TO 1045
LCL=LCL-1
GO TO 220
1045 DO 1046 K=1,MN
II = (I-1)*MN + K
JJ = (LCL-1)*MN+K
JK = (JJ-1)/18+1
JM = (II-1)/18+1
1046 FLD(2*(II-(JM-1)*18)-2,2,CL(JM)) = FLD(2*(JJ-(JK-1)*18)-2,2,
* CL(JK))
BOUND(I)=BOUND(LCL)
JLKL(I) = JLKL(LCL)
LCL=LCL-1
GO TO 1041
1050 IF(I.EQ.LCL) GO TO 220
I=I+1
GO TO 1041
220 RETURN
END

```

```

SUBROUTINE NETWORK
DIMENSION IAL(40), IAR(40), ITL(40), ITR(40)
COMMON NS,ND,IC(162),ID(62),IU(62),IR(62),IP(62),NL(62,62),
* MBIG,N,NPP,NQ,NQQ,CB,NT, LX(162),IX(162),IY(162),JEFF,
* BOUND(5161),TNC,LCL,CL(28672),VCPR,CP(100),JLKL(5161),
* SCOSI(162),TF(100),
* ZU(100),ZC(100),JUDY,TITLE(14),ISOL(100),ITER
* ,JLKF(100),JLKC(100)
DO 3 I=1,NB
  ID(I)=0
  IU(I)=0
3 IR(I)=0
DO 4 I=1,NS
  K=N+I
4 IP(I)=IC(K)
DO 5 I=1,ND
  K=NPP+I
  J=I+NS
5 IP(J)=-IC(K)
C COMPUTE EVALUATORS
22 J=0
6 J=J+1
  IF(J.GT.N) GO TO 54
  K=IX(J)
  L=IY(J)
  ICB=IC(J)-IP(K)+IP(L)
  IF(ICB.GE.0) GO TO 6
  GO TO 55
54 IF(J.GT.NPP) GO TO 54
  KQ=J-N
  ICB=IC(J)-IP(KQ)
  IF(ICB.GE.0) GO TO 6
  GO TO 60
55 CONTINUE
C ROOT TRACE
C SOURCE SIDE
  I=0
  KX=K
7 I=I+1
  ITL(I)=KX
  KX=ID(KX)
  IF(KX.NE.0) GO TO 7
C DESTINATION SIDE
  KX=L
  M=0
8 M=M+1
  ITR(M)=KX
  KX=ID(KX)
  IF(KX.NE.0) GO TO 8
  IEP=MBIG
  IF(ITL(I).EQ.ITR(M), GO TO 9
C DETERMINE LEAVING VARIABLE
C SOURCE SIDE
  I=0
  KX=K
  IB=-1

```

```

10 KY=KX
   KX=ID(KX)
   IB=-IB
   I=I+1
   IF(KX.EQ.0) GO TO 11
   JM=NL(KY,KX)
   IAL(I)=JM
   IF(IB.LT.0) GO TO 10
   IDT=LX(JM)
   IF(IDT.GT.IEP) GO TO 10
   IEP=IDT
   IFG=1
   KCN=KX
   LV=JM
   GO TO 10
11 IAL(I)=N+KY
   IF(KY.GT.NS) GO TO 12
   IDT=LX(N+KY)
   IF(IDT.GT.IEP) GO TO 12
   IEP=IDT
   KCN=KY
   LV=N+KY
   IFG=2
C   DESTINATION SIDE
12 M=0
   KX=L
   IB=-1
13 KY=KX
   KX=ID(KY)
   IB=-IB
   M=M+1
   IF(KX.EQ.0) GO TO 14
   JM=NL(KX,KY)
   IAR(M)=JM
   IF(IB.LT.0) GO TO 13
   IDT=LX(JM)
   IF(IDT.GT.IEP) GO TO 13
   IEP=IDT
   IFG=3
   KCN=KX
   LV=JM
   GO TO 13
14 IAR(M)=N+KY
   IF(KY.LE.NS) GO TO 16
   IDT=LX(N+KY)
   IF(IDT.GT.IEP) GO TO 16
   IEP=IDT
   KCN=KY
   LV=N+KY
   IFG=4
C   CHANGE VARIABLES STFM TO STFM
16 LX(J)=LX(J)+IEP
   IR=1
   DO 17 IG=1,I
   IB=-IB
   JB=IAL(IG)

```

```

17 LX(JB)=LY(JB)+IEP*Ir
   IB=1
   DO 18 IG=1,M
   IB=-IB
   JB=IAR(IG)
18 LX(JB)=LY(JB)+IEP*Ir
   GO TO (19,19,20,20),IFG
19 CALL GRFT(ITR(1),ITL(1),KCN,KYZ)
   IM=ITL(1)
   IZ=ITR(1)
   IF(IFG.EQ.1) GO TO p1
   IF(I.EQ.1) GO TO 21
   JX=KCN
   JY=ITL(I-1)
   KU=IU(JY)
   ID(JX)=JY
   IU(JY)=JX
   IR(JX)=KU
   GO TO 21
20 CALL GRFT(ITL(1),ITR(1),KCN,KYZ)
   IM=ITR(1)
   IZ=ITL(1)
   IF(IFG.EQ.3) GO TO p1
   IF(M.EQ.1) GO TO 21
   JX=KCN
   JY=ITR(M-1)
   KU=IU(JY)
   ID(JX)=JY
   IU(JY)=JX
   IR(JX)=KU
21 IF(IM.GT.NS) GO TO p0
   IP(IM)=IC(J)+IP(IZ)
   GO TO 81
80 IP(IM)=IP(IZ)-IC(J)
81 CONTINUE
   CALL PIRT(IM)
   GO TO 22
C   SAME TREE-DETERMINE JOINING NODE
   9 I1=I
   M1=M
23 I1=I1-1
   M1=M1-1
   IF(I1.EQ.0.OR.M1.EQ.0) GO TO 40
   IF(ITL(I1).EQ.ITR(M1)) GO TO 23
40 KJN=ITL(I1+1)
   I=0
   KX=K
   IB=-1
   IF(ID(KX).EQ.0) GO TO 25
24 KY=KX
   KX=ID(KY)
   IB=-IB
   IF(KY.EQ.KJN) GO TO 25
   I=I+1
   JM=NL(KY,KX)
   IAL(I)=JM

```

```

IF (IB.LT.0) GO TO 24
IDT=LX(JM)
IF (IDT.GT.IEP) GO TO 24
IEP=IDT
IFG=1
KCN=KX
LV=JM
GO TO 24
25 M=0
KX=L
IB=-1
IF (ID(KX).EQ.0) GO TO 27
26 KY=KX
KX=ID(KY)
IB=-IB
IF (KY.EQ.KJN) GO TO 27
M=M+1
JM=NL(KX,KY)
IAR(M)=JM
IF (IB.LT.0) GO TO 26
IDT=LX(JM)
IF (IDT.GT.IEP) GO TO 26
IEP=IDT
IFG=2
KCN=KX
LV=JM
GO TO 26
C MAKE VARIABLE CHANGE
27 LX(J)=LX(J)+IEP
IB=1
IF (I.EQ.0) GO TO 42
DO 28 IG=1,I
IB=-IB
JB=IAL(IG)
28 LX(JB)=LX(JB)+IEP*IB
42 IB=1
IF (M.EQ.0) GO TO 43
DO 29 IG=1,M
IB=-IB
JB=IAK(IG)
29 LX(JB)=LX(JB)+IEP*IB
43 GO TO (31,32),IFG
31 CALL GRFT(ITR(1),ITL(1),KCN,KY2)
IM=ITL(1)
IZ=ITR(1)
GO TO 33
32 CALL GRFT(ITL(1),ITR(1),KCN,KY2)
IM=ITR(1)
IZ=ITL(1)
33 IF (IM.GT.NS) GO TO 42
IP(IM)=IP(J)+IP(IZ)
GO TO 83
82 IP(IM)=IP(IZ)-IP(J)
83 CONTINUE
CALL PIRT(IM)
GO TO 22

```

```

C ENTERING ROOT
C DETERMINE LEAVING VARIABLE
  60 I=0
    IEP=MBIG
    KX=KQ
    IB=-1
  61 I=I+1
    KY=KX
    ITL(I)=KY
    KX=ID(KX)
    IB=-IB
    IF(KX.EQ.0) GO TO 62
    JM=NL(KY,KX)
    IAL(I)=JM
    IF(IB.LT.0) GO TO 61
    IDT=LX(JM)
    IF(IDT.GT.IEP) GO TO 61
    IEP=IDT
    IFG=1
    KCN=KX
    LV=JM
    GO TO 61
  62 IAL(I)=N+KY
    IF(KY.GT.NS) GO TO 63
    IDT=LX(N+KY)
    IF(IDT.GT.IEP) GO TO 63
    IEP=IDT
    KCN=KY
    LV=N+KY
    IFG=2
  63 LX(J)=LX(J)+IEP
    IB=1
    DO 64 L=1,I
      IB=-IB
      IG=IAL(L)
  64 LX(IG)=LX(IG)+IEP*IB
    KX=KQ
    KD=ID(KX)
    ID(KX)=0
    KR=IR(KX)
    IR(KX)=0
    IF(IU(KD).EQ.KX) GO TO 65
    KB = IU(KD)
  67 IF(IR(KB).EQ.KX) GO TO 66
    KB=IR(KB)
    GO TO 67
  65 IU(KD)=KP
    GO TO 68
  66 IR(KB)=KP
  68 IF(KCN.EQ.KD) GO TO 70
    CALL GRFT(KX,KD,KCN,KYZ)
  70 IM=KX
    IP(IM)=IC(J)
    IF(IFG.EQ.1) GO TO 69
    JX=KCN
    JY=ITL(I-1)

```

```
      KU=IU(JY)
      ID(JX)=JY
      IU(JY)=JX
      IR(JX)=K11
b9  CALL PIRT(IM)
      GO TO 22
34  CONTINUE
      NPPP = NPP + 1
      DO 1000 J = NPPP, NQ3
      IF(LX(1).GT.0) GO TO 1001
1000 CONTINUE
      JEFF = 1
      RETURN
1001 JEFF = 2
      RETURN
      END
```

```

SUBROUTINE GRFT(N,IS,NC,K)
COMMON NS,ND,IC(162),ID(62),IU(62),TR(62),IP(62),NL(62,62),
* MBIG,T,NPP,NQ,NGO,MB,NT, LX(162),IX(162),IY(162),JEFF,
* BOUND(5161),INC,LCL,CL(28672),VCPR,CP(100),JLKL(5161),
* SCOST(162),TF(100),
* ZU(100),7C(100),JUDY,TITLE(14),ISOL(100),ITER
* ,JLKF(100),JLKC(100)
K=1
L=N
M=IS
MR=IR(M)
LU=IU(L)
KD=ID(M)
ID(M)=L
IU(L)=M
IR(M)=LU
IF(M.EQ.NC) GO TO 7
1 L=M
M=KD
IF(M.EQ.0) GO TO 7
IF(IU(M).EQ.L) GO TO 2
MZ=IU(M)
3 IF(IR(MZ).EQ.L) GO TO 4
MZ=IR(MZ)
GO TO 3
2 IU(M)=MR
GO TO 5
4 IR(MZ)=MR
5 IF(M.EQ.NC) GO TO 7
MR=IR(M)
KD=ID(M)
ID(M)=L
IZ=IU(L)
IU(L)=M
IR(M)=IZ
K=K+1
GO TO 1
7 RETURN
END

```

```

SUBROUTINE PIRT(I)
COMMON NS,ND,IC(162),ID(62),IU(62),IR(62),IP(62),NL(62,62),
* M(16,16),NPP,NJ,NQ,NB,NT, LX(162),IX(162),IY(162),JEFF,
*BOUND(5161),INC,LCL,CL(28672),VCPR,CP(100),JLKL(5161),
* SCOST(162),IF(100),
*ZU(100),ZC(100),JUDY,TITLE(14),ISOL(100),ITER
* ,JLKF(100),JLKC(100)
J=I
IF(IU(J).EQ.0) GO TO 1
2 L=IU(J)
10 K=NL(J,L)
IF(L.GT.NS) GO TO 3
IP(L)=IC(K)+IP(J)
GO TO 4
3 IP(L)=IP(J)-IC(K)
4 IF(IU(L).EQ.0) GO TO 7
J=L
GO TO 2
7 IF(IR(L).EQ.0) GO TO 8
L=IR(L)
GO TO 10
8 IF(J.EQ.I)GO TO 1
IF(IR(J).EQ.0)GO TO 9
L=IR(J)
J=ID(J)
GO TO 10
9 J=ID(J)
GO TO 8
1 RETURN
END

```

## BIBLIOGRAPHY

1. Balas, Egon, "Intersection Cuts - A New Type of Cutting Planes for Integer Programming," *Operations Research*, Vol. 19, No. 1, pp. 19-39 (1971).
2. Balinski, M. L., "Fixed Cost Transportation Problems," *NRLQ*, Vol. 8, No. 1 (1961), pp. 41-54.
3. Bellmore, M., and H. D. Ratliff, "Optimal Defense of Multi-Commodity Networks," *Management Science*, 18, B147-B185 (1971).
4. Benders, J. F., "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numerische Mathematik*, Vol. 4 (1962), pp. 238-252.
5. Burdet, Claude-Alain, *Enumerative Cuts I*, Working Paper 94-71-2, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1970).
6. Burdet, Claude-Alain, *On the Diamond Cuts*, Presented at the 41st ORSA Meeting in New Orleans, April, 1972.
7. Burdet, Claude-Alain, *Enumerative Cuts II*, Working Paper 18-71-2, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1972).
8. Cooper, L. and C. Drebes, "An Approximate Solution Method for the Fixed Charge Problem," *NRLQ*, Vol. 14, No. 1 (1967), pp. 101-113.
9. Cooper, L. and A. M. Olson, "Random Perturbations and the MI-MII Heuristics for the Fixed-Charge Problem," Report No. COO-1493-7, Department of Applied Mathematics and Computer Science, Washington University (1968).
10. Denzler, D. R., "An Approximative Algorithm for the Fixed Charge Problem," *NRLQ*, Vol. 16, No. 3 (1969), pp. 411-416.
11. Frank, Ronald S., *On the Fixed Charge Hitchcock Transportation Problem* (unpublished dissertation), The Johns Hopkins University, Baltimore, Maryland (1972).
12. Geoffrion, A. M., and R. E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," in *Perspectives on Optimization: A Collection of Expository Articles*, Edited by Arthur M. Geoffrion, Addison-Wesley.

13. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas Method," *SIAM Review*, Vol. 9 (1967), pp. 178-190.
14. Gomory, R. E., "All-Integer Integer Programming Algorithm," pp. 193-206 in *Industrial Scheduling*, edited by Muth and Thompson, Prentice-Hall (1963).
15. Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs," pp. 269-302 in *Recent Advances in Mathematical Programming*, edited by Graves and Wolfe, McGraw-Hill (1963).
16. Gomory, R. E., "Some Polyhedra Related to Combinatorial Problems," *Linear Algebra and Its Applications*, Vol. 2 (October, 1969), pp. 451-558.
17. Gorry, G. A. and J. F. Shapiro, "An Adaptive Group Theoretic Algorithm for Integer Programming Problems," *Management Science*, Vol. 17, No. 5 (1971), pp. 285-306.
18. Gray, Paul, "Mixed Integer Programming Algorithms for Site Selection and Other Fixed-Charge Problems Having Capacity Constraints," Technical Report No. 101, Department of Operations Research and Statistics, Stanford University (1967).
19. Gray, Paul, "Exact Solution of the Fixed-Charge Transportation Problem," *Operations Research*, Vol. 19, No. 6, pp. 1529-1538 (1971).
20. Hadley, G., *Linear Programming*, Addison-Wesley (1963), 520 pp.
21. Heller, Isidore, "On Unimodular Sets of Vectors," p. 39, *Recent Advances in Mathematical Programming*, Edited by Robert L. Graves and Philip Wolfe, McGraw-Hill, 1963.
22. Hirsch, W. M. and G. B. Dantzig, "The Fixed Charge Problem," *NRLQ*, Vol. 15, No. 3 (1968), pp. 413-424.
23. Hu, T. C., *Integer Programming and Network Flows*, Addison-Wesley (1969), 452 pp.
24. Johnson, Ellis, *Programming in Networks and Graphs*, Operations Research Center Report No. 65-1. University of California, Berkley (1965).
25. Johnson, Ellis, Private Communication (May, 1972).
26. Kuhn, H. W. and W. J. Baumol, "An Approximate Algorithm for the Fixed-Charge Transportation Problem," *NRLQ*, Vol. 9, No. 1 (1962), pp. 1-15.

27. Land, A. H., and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, 28, 497-520 (1960).
28. Langley, Warren, Private Communication (July, 1972).
29. Langley, Warren, and Jeff Kennington, "The Transportation Problem" (To be presented at the 43rd National Meeting of ORSA in Milwaukee, Wisc.).
30. Murty, K. G., "Solving the Fixed Charge Problem by Ranking Extreme Points," *ORSA*, Vol. 16, No. 2 (1968), pp. 268-279.
31. Robers, P. and L. Cooper, "A Study of the Fixed Charge Transportation Problem," Report No. COO-1493-9, Department of Applied Mathematics and Computer Science, Washington University (1969).
32. Steinberg, David I., "The Fixed Charge Problem," *NRLQ*, Vol. 17, No. 2 (1970), pp. 217-236.
33. Thompkins, Curtis J., *Group Theoretic Structures in the Fixed Charge Transportation Problem* (Unpublished dissertation), Georgia Institute of Technology (1971).
34. Tomlin, J. A., "Branch and Bound Methods for Integer and Noninteger Convex Programming," Chapter 21, *Integer and Nonlinear Programming*, J. Abadie Editor, American Elsevier, New York, N. Y.
35. Trotter, Leslie E., *An Implicit Enumeration Algorithm for Integer Programming* (Unpublished masters thesis), Georgia Institute of Technology (1971).
36. Zaverei, M. M., and I. T. Frisch, "On the Fixed Cost Flow Problem," *International Journal of Control*, Vol. 16, No. 5, 897-902 (1972).

## VITA

The author was born in DeKalb, Texas, on August 23, 1945. He graduated from high school in 1963 and received a Bachelor of Science from the University of Arkansas in June of 1968. The following September, he enrolled at Georgia Institute of Technology and received a Master of Science in Industrial Engineering in September of 1970. The author obtained his Ph.D. from the same Institute in 1973.

The author's work experience has consisted of temporary positions with Texas Instruments (Dallas, Texas), Ethyl Corporation (Baton Rouge, Louisiana), General Motors (Freemont, California), NASA (Huntsville, Alabama), University of Arkansas Computing Center (Fayetteville, Arkansas), and Georgia Institute of Technology (Atlanta, Georgia).