

CONTINUOUS AND INTEGER GENERALIZED FLOW PROBLEMS

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

by

Robert Warren Langley

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

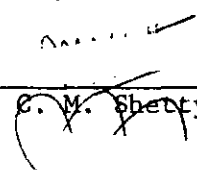
in the School of Industrial and Systems Engineering

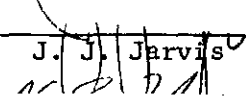
Georgia Institute of Technology

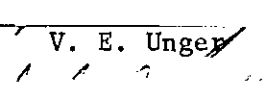
June, 1973

CONTINUOUS AND INTEGER GENERALIZED FLOW PROBLEMS

Approved:

  
\_\_\_\_\_  
C. M. Shetty, Chairman

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

  
\_\_\_\_\_  
V. E. Unger

  
\_\_\_\_\_  
J. J. Goode

Date approved by Chairman: May 16<sup>th</sup> 1973

## ACKNOWLEDGMENTS

I am unable to fully express my appreciation and gratitude to Dr. Mike Shetty. His teaching, guidance, and friendship have sustained me throughout my educational experience at Georgia Tech.

I wish to thank Dr. John Jarvis for introducing me to the work of Ellis Johnson and for his helpful advice and criticism during my dissertation research.

I also thank Dr. Ed. Unger, Dr. Jamie Goode, and Dr. D. E. Fyffe for reading the dissertation and providing helpful criticism.

Acknowledgment is due to Dr. Jeff Kennington for producing the computational results in Table 4.

The guidance and counsel provided by Dr. W. W. Hines and Dr. L. A. Johnson during the early phases of my program of study are greatly appreciated.

Finally, I thank Sherry, Anne, and Kasey for their understanding and consideration.

## TABLE OF CONTENTS

ACKNOWLEDGMENTS. . . . .	ii
LIST OF TABLES . . . . .	v
LIST OF ILLUSTRATIONS. . . . .	vi
SUMMARY. . . . .	vii
Chapter	
I. INTRODUCTION AND LITERATURE SURVEY. . . . .	1
1.1 Introduction	
1.2 Problem Statement	
1.3 Applications of the GFP	
1.4 Solution Procedures for the GFP	
1.5 Network Programming Computational Devices	
1.6 Applications of the IGFP	
1.7 Procedures for Solving the IGFP	
1.8 Concepts of Linear Programming	
1.9 Review of Integer Programming Methods	
1.9.1 Combinatorial Methods	
1.9.2 Enumerative Methods	
1.9.3 Algebraic Methods	
1.10 Summary	
II. BASIS CHARACTERIZATION FOR THE GFP. . . . .	28
2.1 Introduction	
2.2 Basic Graph Terminology	
2.3 Graphical Representation of the GFP	
2.4 GFP Example Problem	
2.5 Basis Characterization	
III. NETWORK PROGRAMMING BY ROW AND COLUMN GENERATION. . . . .	60
3.1 Introduction	
3.2 Simplex Multiplier Calculation	
3.3 Column Generation	
3.4 Row Generation	
3.5 Basis Change and Updating Simplex Multipliers	
3.6 Labeling the Tree	

## TABLE OF CONTENTS (Concluded)

Chapter	Page
IV. CONTINUOUS ALGORITHMS . . . . .	118
4.1 Introduction	
4.2 GFP Algorithm	
4.2.1 Statement of the Algorithm	
4.2.2 Initialization	
4.2.3 Justification of the GFP Algorithm	
4.2.4 Computational Results	
4.3 Ordinary Flow Algorithm	
4.3.1 OFP Algorithm Statement	
4.3.2 Simplifications for the OFP Algorithm	
4.3.3 Computational Results	
4.4 Primal Transportation Algorithm	
4.4.1 Problem Statement	
4.4.2 Simplifications for the Transportation Problem	
4.4.3 Computational Results	
4.5 Resolution of Degeneracy	
4.6 Summary	
V. INTEGER GENERALIZED FLOW PROBLEM CHARACTERISTICS. . . . .	158
5.1 Introduction	
5.2 IGFP Group Formulation	
5.3 Determinant Calculation	
5.4 Structure of the Nonbasic Columns and Basic Rows	
5.5 Penalties	
5.6 Summary	
VI. ALGORITHM FOR THE IGFP. . . . .	195
6.1 Introduction	
6.2 Selection of Method of Solution	
6.3 Branch and Bound Procedure	
6.4 Computational Results	
6.5 Obtaining a Feasible Integer Solution	
VII. CONCLUSIONS AND RECOMMENDATIONS . . . . .	208
7.1 Conclusions	
7.2 Recommendations	
BIBLIOGRAPHY . . . . .	212
VITA . . . . .	220

## LIST OF TABLES

Table	Page
1. Simplex Iterations for GFP Example. . . . .	35
2. Generalized Flow Computational Results. . . . .	125
3. OFP Computational Results . . . . .	139
4. Transportation Computational Results. . . . .	149
5. IGFP Computational Results. . . . .	202

## LIST OF ILLUSTRATIONS

Figure		Page
1.	Examples of Graphical Structures . . . . .	29
2.	Generalized Flow Example. . . . .	33
3.	Graphical Representation of Simplex Iterations. . . . .	40
4.	Block Diagonal Matrix . . . . .	44
5.	Basis Components. . . . .	45
6.	Entering Arc Configurations . . . . .	69
7.	Basis Exchange Configurations . . . . .	92
8.	Comparison of OFP and SHARE OTK Algorithms (Nodes) . . . . .	143
9.	Comparison of OFP and SHARE OTK Algorithms (Arcs). . . . .	144
10.	Branch and Bound Procedure. . . . .	199

## SUMMARY

The characterization of the basis for ordinary and generalized network problems given by Dantzig, Johnson, and others is shown to provide a powerful tool for exploiting the sparseness of the constraint set for these problems. This graphical representation is used to directly compute the nonzero entries in specified columns and/or rows of the basis inverse. Based on this, algorithms for the generalized flow (GFP), ordinary flow (OFF), and transportation (TP) problems are developed and computational results for the resultant computer codes are given. The power of the approach is illustrated by the fact that the solution times for the OFF code are five times faster than the Share out-of-kilter code on the same problems. The details of the computer implementation are also given.

The integer generalized flow problem (IGFP) is considered and the previously mentioned basis representation for the GFP is used to identify the interaction between variables in the group theoretic formulation of the problem. Several properties of the IGFP are noted. A branch and bound algorithm is developed and computational results for the IGFP are given.

## CHAPTER I

### INTRODUCTION AND LITERATURE SURVEY

#### 1.1 Introduction

A great deal of recent research in mathematical programming has been concentrated in the specialized field of integer programming (IP). This research is stimulated by the wide application of such models and the need to be able to find solutions efficiently. To date no effective algorithm, say one comparable to Dantzig's simplex method for the linear programming problem, for the general IP problem has been found. Very good results have been obtained for some IP problems by taking advantage of the special nature of a particular type of problem (e.g., matching, set covering). This successful exploitation of problem structure provided the motivation for the research of this study. The problem chosen is the integer generalized flow problem (IGFP) which is defined in the next section. The problem is important in its own right to warrant study. Besides, it offers a reasonable chance of yielding a successful solution technique and of providing insight into the process of exploiting the problem structure in integer programming. The objectives of this study are:

1. To develop a detailed characterization of the structure of the generalized flow problem (without integer constraints).
2. To develop or extend existing solution procedures for the GFP which will be compatible with the integer programming method chosen to

solve the IGFP.

3. To use the characterization in (1) to develop a characterization of the IGFP.

4. To develop an effective solution technique for the IGFP.

In the chapters that follow, several topics are investigated and brought together in accomplishing these primary objectives. The associated continuous generalized flow problem (GFP) is studied and an efficient solution procedure derived from a characterization of the structure of its basis. Algorithms which take advantage of this structure are presented for two special cases of the GFP, the ordinary minimum cost flow and transportation problems. The IGFP is characterized from a group theoretic point of view, and efficient means for utilizing information from this formulation in a branch and bound solution procedure are developed. The results from the study of the structure of the GFP and IGFP are brought together in a branch and bound integer programming procedure. A summary of the main results of this study and areas for additional research are given in Chapter VII.

Throughout the dissertation, if material of a broad nature is being discussed (e.g., linear programming), one or two general references are given which are not meant to be considered the only work in that field. When specific methods are addressed, the associated reference(s) are given. In some cases references are limited to the most recent work in a field or the ones specifically related to a topic in the dissertation.

This introductory chapter contains a mathematical statement of the

IGFP and the associated GFP. Applications of the GFP are cited from the literature, and solution procedures for the GFP are surveyed. An example from Dantzig's ubiquitous book [15] motivates the restriction of the variables for the GFP to integer values. The work of Estabrook [20] on a special case of the IGFP is then reviewed. Necessary notation and terminology from linear and integer programming used later in the dissertation are summarized.

## 1.2 Problem Statement

The integer generalized flow problem is stated mathematically as:

(IGFP)

$$\text{Minimize} \quad \sum_{j=1}^n c_j x_j \quad (1)$$

Subject to:

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i=1, \dots, m \quad (2)$$

$$0 \leq x_j \leq M_j \quad j=1, \dots, n \quad (3)$$

$x_j$  integer for all  $j$ .

$a_{ij} \neq 0$  for at most two  $i \in \{1, \dots, m\}$

where  $c_j$ ,  $a_{ij}$ ,  $M_j$  integer for all  $i$  and  $j$ .

For all integer programming problems it will be assumed that all coefficients and bounds are integer.

The continuous linear program associated with the IGFP when the integer requirement is relaxed is called the generalized flow problem (GFP) and is:

(GFP)

$$\text{Minimize} \quad \sum_{j=1}^n c_j x_j \quad (4)$$

Subject to:

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i=1, \dots, m \quad (5)$$

$$0 \leq x_j \leq M_j \quad j=1, \dots, n \quad (6)$$

$$a_{ij} \neq 0 \text{ for at most two } i \in \{1, \dots, m\}$$

The flow with gains problem considered by Jewell [56] and others is the same as the generalized flow problem studied in this dissertation.

An important special case of the GFP has been called by Dantzig [15] the weighted distribution problem and by Balas and Ivanescu [3] and others the generalized transportation problem (GTP). The latter terminology will be adopted here. The generalized transportation problem is thus:

(GTP)

$$\text{Minimize} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (7)$$

Subject to:

$$\sum_{j=1}^n a_{ij} x_{ij} \leq S_i \quad i=1, \dots, m \quad (8)$$

$$\sum_{i=1}^m x_{ij} = T_j \quad j=1, \dots, n \quad (9)$$

$$0 \leq x_{ij} \leq M_{ij} \quad \begin{matrix} i=1, \dots, m \\ j=1, \dots, n \end{matrix} \quad (10)$$

To see that the GTP is a special case of the GFP, note that each variable appears in only two equations, one from the set of Eqs. (8) and one from Eqs. (9). If slack variables were added to Eqs. (8) they would appear in only one equation. This is one form of the GTP. Other forms may consist of different inequality or equality relations on the two sets of equations (8) and (9) or the coefficients in Eqs. (9) may be something other than one. This does not alter the fact that this is still a special case of the GFP.

The GFP and the GTP are generalizations of the transshipment and transportation problems, respectively. In these ordinary flow problems the nonzero  $a_{ij}$  in Eq. (2) are either plus one or minus one. If there are two nonzero coefficients, one is a plus one and the other is a minus one. The ordinary flow problem has been studied extensively and detailed references may be found in the books of Ford and Fulkerson [21] and Hu [54].

### 1.3 Applications of the GFP

Jewell [56] and Dantzig [15] cite several examples in which mathematical models for resource allocation problems have the form of the GFP. Arms [1] uses the generalized transportation model to solve optimum weapons-allocation problems and Eisenman [19] puts the machine loading problem in the GTP form. Jensen [55] uses a GFP formulation to study water resource allocation. Jarvis and Jezior [57] use a generalized flow formulation to model a health care system. Demmy [16] requires the solution of generalized flow problems as subproblems in his master procedure for a more general linear programming model. In a different field,

Fujisawa [23] and Onaga [76,77] discuss electrical and communication network problems which can be formulated as generalized flow problems. According to Eisenman [19] by taking advantage of the special structure of the generalized flow problem, one is able to solve large scale problems whose solutions would be impractical by direct application of the simplex method.

The range of application of the generalized flow problem justifies the investigation of the specialized solution procedures detailed in the next section.

#### 1.4 Solution Procedures for the GFP

Development of solution procedures for the GFP has paralleled that of ordinary flow problems; that is, applying the various simplex-based techniques and making use of the resulting simplifications. The solution procedures are categorized as primal-dual, primal, and dual following the linear programming terminology. In addition, some studies focus on the graphical representation of the problem to develop solution procedures. These procedures, however, can just as well be interpreted in linear programming terms.

Jewell originally considered the GFP under the name of flow with gains [56]. He extended the work of Ford and Fulkerson [21] for ordinary networks to the GFP in developing a primal-dual algorithm. He also considered the graphical nature of the problem producing a max-flow min-cut theorem analogous to that of Ford and Fulkerson. Recently Minieka [74] has presented a method to initialize Jewell's algorithm, modified it to ensure finite convergence, and interpreted it in terms of the out-of-kilter

algorithm for ordinary flow problems. Jarvis and Jezior [57], Smith [86], and Grinold [49] all apply Jewell's ideas to the maximization of flow into a specified sink node in the generalized flow setting. They take advantage of the structure of the GFP as well as additional structure to produce efficient specialized algorithms.

Dantzig [15] applied the primal linear programming method to ordinary flow problems and extended the results to a special case of the GFP, namely the GTP discussed in section 1.2. Johnson [58] brought together the algebraic approach of Dantzig with the graphical methods of Ford and Fulkerson for the ordinary flow problem and of Jewell for the GFP. He also suggested a method of keeping track of the basis for the ordinary flow problem. Simonnard [85] also interpreted distribution problems in terms of a graph. Recently Maurras [71] has reported computational results of applying a specialization of the primal linear programming method to large generalized flow problems.

In a set of three related papers in 1964 the generalized transportation problem was considered. Balas and Ivanescu [3] developed the theory for extending primal solution techniques for the ordinary transportation problem to that of the GFP. They focused on the simplifications in calculating the new set of basic variables for a specified basis change. Eisenman [19] presented an algorithm for carrying out the method of Balas and Ivanescu and extended it to the capacitated problem. Lourie [67] indicated efficient means for implementing Eisenman's algorithm on a computer. More recently, Arms [1] considered a solution procedure for the GTP and presented limited computational results.

The approach taken in this dissertation is also based on the primal

simplex algorithm. A graphical representation of the basis of the problem is used which allows the identification, computation, and updating of only that information directly required to carry out the simplex operations. Additionally, this representation can be efficiently maintained and updated. This is in contrast to the graphical solution methods such as the out-of-kilter algorithm of Ford and Fulkerson for ordinary flow problems and the algorithm of Jewell for the GFP. These methods are based on such concepts as flow augmenting paths, cut sets, etc. and information (e.g., labels) from one iteration is not used in the next iteration.

Dual methods were applied to the GFP by Balas [4] and Takahashi [88] independently in 1966. Balas specifically considered the GTP but his ideas are readily extendable to the GFP. In the same year Charnes and Raike [13] applied dual methods to some special cases of the GFP. Specifically, they considered the "shortest path problem" whose solution for ordinary networks is given by Dijkstra and others. Glover and Klingman [32] extended the work of Charnes and Raike to provide an initial dual feasible solution for the capacitated GFP.

As mentioned earlier, there are studies which take advantage of the graphical representation of the problem in the spirit of the Ford and Fulkerson algorithm for maximum flow for the ordinary flow problem. Some of these studies arose from the field of electrical engineering for the problem of maximizing flow into a sink node with the GFP constraint set. Fujisawa [23] developed a procedure for maximizing flow into the sink by successively finding paths between the source and sink along which flow could be increased. Mayeda and Van Valkenberg [73] investigated the

properties of the GFP analagous to those for the ordinary flow problem studied by Ford and Fulkerson [21] and others. Onaga [76,77] considered the graph-theoretic properties of the GFP and adopted a dynamic programming algorithm for minimizing the flow out of a specified source node while maximizing flow into the sink node.

Rutenberg [79] has discussed the GFP with a nonlinear objective function and has extended the convex-simplex method to this problem. Glover and Klingman [31] have shown a method for transforming some generalized networks into equivalent pure networks. They have also discussed a computational simplification for the GFP [35]. Malek-Zavarei and Aggarwal [69] discuss equivalences and properties of the GFP.

### 1.5 Network Programming Computational Devices

The close relationship between the ordinary flow problem and the generalized flow problem is indicated by the articles detailed in the previous section. Recently, several computational devices for ordinary flow problems have been reported. Since methods for the ordinary flow problem may be extended to the GFP, these recent results will be outlined.

The primal computational methods for the ordinary flow problem center around the manner of identifying the current basis and of changing from one basis to another. Glover and Klingman [29] and Glover, Klingman, and Kearny [34] developed a means of doing this which is essentially that of Johnson [59]. They include the details of changing from one basis to another. Srinivasan and Thompson [87] consider similar methods and develop devices to reduce the effort required for determining leaving variables. Both of these efforts are specifically addressed to

the transportation problem. Tomizawa [92] also indicates improvements in methods for solving transportation problems.

The computational results reported by Glover, Klingman, Napier, and Kearny [30,36] and Thompson and Srinivasan [91] for the transportation problem and for the improved out-of-kilter algorithm of Glover, Klingman, and Barr [38] for the ordinary flow problem show the improvements that can be obtained by effectively exploiting the specialized basis structure of the ordinary flow problem.

### 1.6 Applications of the IGFP

To illustrate the need to consider the requirement that the variables in a GFP be integers, the following example is taken from Dantzig [15]:

Consider the GFP used to model the following situation. An airline has a fleet composed of several types of aircraft. These aircraft must be allocated to various routes to satisfy a specified passenger demand. Since the cost of operating a particular aircraft on a specific route varies from route to route, the allocation should be made to minimize the total cost, while satisfying demand and utilizing only the aircraft available. The appropriate definitions and mathematical problem statement are:

- $x_{ij}$  - number of aircraft type  $i$  to be allocated to route  $j$
- $c_{ij}$  - cost of operating one type  $i$  aircraft on route  $j$
- $p_{ij}$  - number of passengers accommodated by aircraft type  $i$  on route  $j$
- $a_i$  - number of type  $i$  aircraft available

$b_j$  - passenger demand on route  $j$

The problem is then:

$$\text{Minimize} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (11)$$

Subject to:

$$\sum_{j=1}^n x_{ij} \leq a_i \quad i=1, \dots, m \quad (12)$$

$$\sum_{i=1}^m p_{ij} x_{ij} \geq b_j \quad j=1, \dots, n \quad (13)$$

$$x_{ij} \geq 0 \quad (14)$$

The following example is for two aircraft types to be allocated to two routes.

Minimize

$$20x_{11} + 110x_{12} + 50x_{21} + 300x_{22} \quad (15)$$

Subject to:

$$x_{11} + x_{12} \leq 4 \quad (16)$$

$$x_{21} + x_{22} \leq 3$$

$$50x_{11} + 100x_{21} \geq 150$$

$$40x_{12} + 100x_{22} \geq 100$$

$$x_{ij} \geq 0 \quad \text{for all } i \text{ and } j.$$

The optimum linear programming solution is

$$\begin{aligned}
 x_{11} &= 1.5 \\
 x_{12} &= 2.5 \\
 x_{21} &= .75 \\
 x_{22} &= 0
 \end{aligned}$$

Total cost - 342.5

Of course it is not possible to allocate a fraction of an aircraft. The solution,  $x_{11} = 2$ ,  $x_{12} = 3$ ,  $x_{21} = 1$ ,  $x_{22} = 0$ , obtained by rounding up the nonintegral values is not feasible. But consider the solution  $x_{11} = 1$ ,  $x_{12} = 3$ ,  $x_{21} = 1$ ,  $x_{22} = 0$ . This solution is feasible and has a cost of 400. Another feasible integer solution is  $x_{11} = 1$ ,  $x_{12} = 0$ ,  $x_{21} = 1$ ,  $x_{22} = 1$  with a cost of 370. The optimal integer solution is  $x_{11} = 3$ ,  $x_{12} = 0$ ,  $x_{21} = 0$ , and  $x_{22} = 1$  with a cost of 360. This simple example points out the well known fact that it is not simple to obtain an optimum integer solution to a system of linear constraints from the solution of the associated linear program. From this example it is clear that the addition of the integer requirement to the GFP is worth investigation.

### 1.7 Procedures for Solving the IGFP

There are no published algorithms for specifically solving the integer generalized flow problem. Estabrook's algorithm [20] solves a specialized case of the IGFP called the integer generalized transportation problem (IGTP). This problem is stated:

$$\begin{aligned}
 &\text{Minimize} && \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} && (17)
 \end{aligned}$$

Subject to:

$$\sum_{j=1}^n a_{ij}x_{ij} \leq S_i \quad i=1, \dots, m \quad (18)$$

$$\sum_{i=1}^m x_{ij} = T_j \quad j=1, \dots, n \quad (19)$$

$$x_{ij} \geq 0 \text{ and integer for all } i \text{ and } j. \quad (20)$$

$$c_{ij}, S_i, T_j, a_{ij} \geq 0 \text{ and integer for all } i \text{ and } j.$$

As mentioned previously, several persons have developed solution procedures for the GTP, which is the above problem, without requiring  $x_{ij}$  to be integers. Estabrook uses the unique structure of the basis for the GTP and the particular form of the equations for the GTP to simplify the rounding algorithm of Gomory [40] when applied to the IGTP.

Two drawbacks of the rounding algorithm are that it does not guarantee that a feasible solution will be found and the amount of storage required is dependent on the size of the determinant of the optimal LP basis. These are true of the algorithm of Estabrook also. However, his computational results indicate that the use of the special structure of the GTP basis and the group formulation of the IGTP are powerful ways of considering the problem. The extension of some of Estabrook's simplifications of the group structure of the IGTP will be given in Chapter V.

### 1.8 Concepts of Linear Programming

Certain terminology, definitions, and concepts of linear programming will be used extensively in the development in later chapters. To provide a common basis for discussion, a summary of relevant material from linear programming will be given in this section. The material follows

the presentation in Lasdon [66] which was condensed from the definitive work of Dantzig [15]. Similar material is covered in many books including Simonnard [85] and Hadley [51] among others.

In matrix form an upper bounded linear programming problem (P) can be stated as:

(P)

$$\text{Minimize} \quad cx \quad (21)$$

$$\text{Subject to:} \quad Ax = b \quad (22)$$

$$0 \leq x_i \leq M_i \quad i=1, \dots, n \quad (23)$$

A is an  $m \times n$  matrix,  $x$  is an  $n \times 1$  column vector, and the remaining vectors are conformable.

It will be assumed that the rank of A is  $m$ . A feasible solution to P is a vector  $x = (x_1, \dots, x_n)$  which satisfies equations (22) and (23). A basis is an  $m \times n$  nonsingular matrix, B, formed by some  $m$  columns of the constraint matrix A. (Since the rank of A is  $m$ , it contains at least one basis.) The variables  $x_1, \dots, x_m$  associated with the columns of the basis B are called basic variables. The remaining variables are called nonbasic. Thus, if the nonbasic variables are assigned values at their upper or lower bounds, the remaining basic variables are determined uniquely by solving the resulting set of equations. Such a solution is called a basic solution.

A basic feasible solution is a basic solution which satisfies Eq. (23). It follows that a basic feasible solution may have at most  $m$  components strictly between zero and the upper bounds ( $M_i$ ). A nondegenerate basic feasible solution is one with exactly  $m$  components strictly between

the upper and lower bounds. The cannonical form of a linear programming problem is the system of equations obtained when equation (22) has been rearranged or transformed so that the first  $m$  columns of  $A$  form an identity matrix.

To introduce certain terminology consider problem  $P$  without the upper bound  $x_i \leq M_i$ . If the objective function Eq. (21) is expressed in terms of the current nonbasic variables, the coefficient for each nonbasic variable is called its relative cost factor or current cost. For a given basis  $B$  the row vector  $\pi$  is defined by the equation:

$$\pi B = c_B \quad (24)$$

The vector  $c_B$  is the vector of cost coefficients associated with the basic variables. The components of the vector  $\pi$  are called the simplex multipliers associated with the basis  $B$ . It follows directly that the relative cost factor ( $\bar{c}_j$ ) for a nonbasic variable  $x_j$  may be calculated by:

$$\bar{c}_j = c_j - \pi A_j \quad (25)$$

The vector  $A_j$  is the corresponding column from the original set of equations (22), and we must have  $\bar{c}_j \geq 0$  for optimality.

The optimality conditions for the bounded variables problem  $P$  are readily related to the simplex multipliers and relative cost factors described above.

Corresponding to the problem  $P$  (called the primal problem) is another linear programming problem called the dual problem (D). The form of the dual problem is:

$$(D) \quad \text{Maximize} \quad ub - \sum_{i=1}^M v_i M_i \quad (26)$$

Subject to:

$$uA - v \leq c \quad (27)$$

$$v \geq 0, u \text{ unrestricted} . \quad (28)$$

The  $u$  variables are associated with the primal constraints Eq. (22) and the  $v$  variables are associated with upper bound constraints contained in Eq. (23).

The well known complementary slackness conditions of linear programming for this primal-dual pair of programs are:

$$x_j (c_j - uA_j + v_j) = 0 \quad (29)$$

$$v_j (x_j - M_j) = 0 \quad (30)$$

If  $0 < x_j < M_j$ , then by Eq. (30)  $v_j = 0$  and:

$$c_j - uA_j = 0 . \quad (31)$$

Now let  $B$  be the columns of  $A$  corresponding to the basis, i.e., for the variables  $0 < x_j < M_j$  if the problem is not degenerate. Let  $v_B$  and  $c_B$  be the components of  $v$  and  $c$  corresponding to the basic variables. Then, from Eq. (29) and (30),

$$\begin{aligned} v_B &= 0 \\ c_B - uB &= 0 \quad \text{or} \quad uB = c_B \end{aligned} \quad (32)$$

Equation (32) is the same as Eq. (24) with  $\pi = u$ . When  $x_j = 0$ , then again from Eq. (30)  $v_j = 0$  and the relative cost factor Eq. (25) becomes:

$$(c_j - uA_j + v_j) = (c_j - uA_j) = \bar{c}_j \geq 0. \quad (33)$$

When  $x_j = M_j$ , from Eq. (29) we get  $c_j - uA_j + v_j = 0$  which can be satisfied by letting  $v_j = -(c_j - uA_j)$ . The nonnegativity restriction on  $v_j$  then yields the optimality condition as:

$$\bar{c}_j = (c_j - uA_j) \leq 0 \quad \text{when } x_j = M_j. \quad (34)$$

This discussion relating the primal and dual problems is relevant to the presentation in later chapters. Detailed results in this area may be found in the previously cited references.

### 1.9 Review of Integer Programming Methods

This section will briefly describe the various strategies for solving general integer programming problems which are discussed in relation to the IGFP in later chapters. The terminology necessary to subsequent sections will be defined here. There are several excellent survey articles on integer programming notably those of Balinski and Spielberg [7] and Geoffrion and Marsten [26]. This section is not intended to supplant those articles but to present the ideas necessary to the understanding of later topics. Following the manner in [7], the methods of integer programming are separated into (i) combinatorial, (ii) algebraic, and (iii) enumerative. A fourth category may be added, namely (iv) approximate or heuristic methods which attempt to obtain a so-called "good"

integer solution, but do not validate its optimality.

#### 1.9.1 Combinatorial Methods

Combinatorial methods encompass algorithms which consider a specific combination of values and directly determine optimality or lead to a new combination. In particular, these methods are characterized by an algebraic bound on the number of iterations as a function of a problem parameter rather than an exponential bound as in the general case. Solution methods for the assignment and ordinary transportation problems have been developed which come into this category. However, these are not considered as integer programming problems since the integer requirement is redundant because of the special nature of the constraint set. Few actual integer programming algorithms are in the combinatorial class. The notable exception is the matching problem and its variations studied principally by Edmonds and Johnson [18] and Balinski [8]. For those problems a particularly effective way of enforcing the integrality requirements has been developed which leads to efficient algorithms. However, Padberg [78] has recently shown that the problem of edge covering by nodes, seemingly next in degree of difficulty after the matching problem, is much more difficult to attack with a combinatorial approach. That leads one to expect that combinatorial methods are extremely limited in the scope of their applicability.

#### 1.9.2 Enumerative Methods

The methods which have proven computationally most successful to date are in the class called enumerative. The fundamental strategy of the enumerative approach is to develop a tree hierarchy of nodes representing candidate problems whose solution may provide a solution to the

integer programming problem. The search adds nodes to the tree until a solution is found which is shown to be the best over all others which have been found or which may be found as a result of searching the remaining candidate problems. The characteristics of an effective enumerative method are (i) scheme for recording the state of nodes considered and keeping track of those to be considered, (ii) means for identifying a solution when it has been found, (iii) rule to choose the next candidate problem (node) for consideration, and (iv) strategy for directing the search of the tree.

The enumeration methods are categorized as implicit enumeration or branch and bound according to the method used to direct the tree search.

The branch and bound method evolved from the initial work of Land and Doig [65] as improved by Dakin [14]. These are, in fact, the basis for most commercially available general IP codes such as UMPIRE, OPHELIE, and MPS-MIP. The strategy at a node is to solve the associated linear programming problem, i.e., the IP problem with integrality relaxed. If the solution is all integer, then it becomes a candidate solution. If one or more variables are not integer, one is chosen ( $x_i$ ) and two new nodes are created in the tree with the restriction that:

$$x_i \geq [\bar{x}_i] + 1 \quad \text{or} \quad x_i \leq [\bar{x}_i]. \quad (35)$$

$\bar{x}_i$  is the value of the fractional variable in the LP solution and  $[y]$  is the greatest integer less than or equal to  $y$ .

If a candidate solution is found, it is compared to the best solution found so far (called the incumbent solution). If it is better, it replaces the incumbent. That branch of the tree is terminated since any

further restriction of this problem can result in only worse solutions.

To generate two descendant nodes from a given node requires the choice of a branching variable. According to Johnson [61] this choice is crucial in determining the effectiveness of the search. Suppose the value of the objective function at the current node is  $Z_{LP_i}$ . It is easily shown that  $Z_{LP_i}$  forms a lower bound on the objective function value for all nodes below the current node. A great deal of work has been done by Driebeck [17], Dakin [14], Tomlin [93], Johnson and Spielberg [60], and others to develop a penalty ( $P_i$ ) such that  $Z_{LP_i} + P_i$  can be used as a lower bound instead of  $Z_{LP_i}$ . Once an incumbent solution is obtained, the effective use of penalties allows rapid curtailment of the search. That is, if a penalty is large enough so that  $Z_{LP_i} + P_i > Z_{inc}$ , the cost of the incumbent, then that branch may be terminated or fathomed; since no better solution may be found. Likewise a branch is fathomed if the associated LP is infeasible, since obviously there cannot be any feasible integer solutions along that branch.

After either finding a candidate solution or creating two nodes to be added to the list of candidate problems, the node to be considered next must be selected. Various heuristic rules have been used to make this choice. A trade-off must be made between flexibility of choice and the difficulty in solving the linear program at each new node. The use of the solution of the LP at one node in solving the LP at a nearby node is highly desirable. This also points out the need for an efficient LP solution procedure; particularly one which takes advantage of problem structure, if that is possible. The node choice is also related to the requirement to store information for each node to characterize its candi-

date problem and perhaps to facilitate its solution.

In summary, the devices important to the development of an effective branch and bound scheme include effective choice of branching variables, good strategy for choice of node to be solved, efficient LP solution procedures, means to store the candidate problem for each node, and calculation of penalties which reduce the number of nodes which must be explicitly considered. An additional device which might be considered is a means to determine a "good" integer solution early in the search so that the depth of the tree (in terms of LP objective value) may be limited.

The implicit enumeration strategy (e.g., Balas [5], Glover [27], Geoffrion [25], and Trotter [95]) sacrifices the flexibility of the tree search of branch and bound for efficient means of storing the tree and identifying that portion which has yet to be searched. Associated with each node of the implicit enumeration tree will be a set of fixed variables and a set of free variables. The fixed variables have been fixed at specific values at nodes higher in the tree. The free variables are examined in hopes of finding a new incumbent solution or determining that searching along that branch is no longer fruitful. This is done by considering the logical implications of the constraints of the associated LP one at a time. The information contained in the solution of the LP has not been used extensively in implicit enumeration schemes with the notable exception of the surrogate constraint concept of Glover [28] and Geoffrion [25]. The choice of the branching variable (i.e., the free variable to be fixed) is done primarily on logical considerations and penalties

have not been used. (Geoffrion and Marsten [26] mention the use of penalties with Geoffrion's 0-1 code RIP30C but no report of their results has been published.) The rigidity of the search is compensated by the simplicity in maintaining the tree. The clever devices used for this are reported in Geoffrion [25] for the 0-1 problem and by Zionts [96] and Trotter [95] for the general integer problem. The main consideration for implementing an implicit enumeration algorithm is an effective means for utilizing the logical information in the linear programming constraints. Information derived from solving the associated LP may also be used (e.g., via the surrogate constraints).

### 1.9.3 Algebraic Methods

The so-called algebraic approach has emanated from the extensive theoretical work of Ralph Gomory [41,42,43,44]. They are divided into cutting plane methods and group theoretic methods, although the two are closely related. The intuitive idea of the cutting plane method is to use the original LP constraints and the integer requirements to develop additional linear constraints or cutting planes which reduce the linear programming solution space. If the added constraints sufficiently reduce the LP solution space, the optimal integer solution may be found by ordinary linear programming techniques. Unfortunately, the information used for constructing these additional constraints has not led to a rapidly converging algorithm with the notable exception of Martin's work [70] with the set covering problem. Recently, work by several authors, principally Balas [6], Glover [39], and Burdet [10], has focused on geometrical ideas to derive cutting planes. Burdet [11] does much to relate these

geometrically motivated ideas to the algebraic notions of Gomory. Unfortunately, except for the unique case of the matching problem mentioned previously, the derived cuts are unrelated to the structure of the original linear programming problem. Thus the power of special purpose LP algorithms is lost when one uses a cutting plane method. In fact, the concept of exploiting special structures to solve integer programming problems does not seem to be directly compatible with the usual methods of adding additional linear constraints. This severely limits the use of cutting planes when developing an algorithm particularized to take advantage of the special structure of the associated linear program.

The group theoretic method is an extension of the methods of Gomory for constructing cutting planes. The main theoretical ideas in Gomory's papers have been extended by Shapiro [81,82,83]. The description below follows that presented in [7]. More detailed descriptions are found in the texts of Hu [54] and Garfinkel and Nemhauser [24] among others.

Consider the following linear programming problem in canonical form associated with an integer problem.

(P1)

$$\text{Minimize} \quad cx \quad (36)$$

$$\text{Subject to:} \quad Ax + Is = b \quad (37)$$

$$x, s \geq 0 \quad (38)$$

Suppose that the optimal basis is  $B$  and the variables  $x$  and  $s$  are divided into the LP optimum basic variables  $x_B$  and corresponding nonbasic variables  $x_N$ . The integer problem can be rewritten as:

(P2)

Minimize

$$(c_N - c_B B^{-1} A_N) x_N + c_B B^{-1} b \quad (39)$$

Subject to:

$$x_B + B^{-1} A_N x_N = B^{-1} b \quad (40)$$

$$x_B \geq 0 \quad x_N \geq 0 \quad x_B, x_N \text{ integer}$$

$c_N$  and  $A_N$  are the original costs and constraint columns associated with the nonbasic variables.

The integer programming problem can be restated as: find an integer vector  $x_N \geq 0$  such that  $x_B \geq 0$  and integer where:

$$x_B = B^{-1} b - B^{-1} A_N x_N \quad (41)$$

and Eq. (39) is minimized. Shapiro [84] calls the vector  $-B^{-1} A_N x_N$  the correction to the LP solution  $x_B = B^{-1} b$  which makes it integer. Using group theory of abstract algebra, Gomory, Shapiro, and others have formalized these ideas into a unified theory and developed various properties of this formulation in the references cited earlier. Much of recent research in integer programming has been devoted to using the theory of Gomory to develop algorithms for solving the general integer problem. To date there has been only limited success with general IP algorithms derived from group theory. The best reported results are those of Gomory and Shapiro [48]. The amount of effort required with the group algorithms reported so far is proportional to the determinant of the optimal LP basis  $B$ . This, in general, may be very large.

Some applications of group theory, to specially structured problems,

have produced seemingly efficient algorithms. Thierez [90] developed an algorithm based on group theory which successfully solved large set covering problems. This is not surprising considering the relationship between group methods and cutting planes and the fact that Martin [70] has solved set covering problems with cutting plane methods since 1965.

Tompkins characterized the structure of the group problem for the fixed charge transportation problem in his doctoral dissertation. He developed an algorithm from this characterization, but reported no computational results.

Estabrook applied Gomory's group theoretic rounding algorithm [40] to the generalized transportation problem in his doctoral dissertation [20] with good computational success. His development will be discussed in Chapter V.

Recently, results from group theory have been used to enhance other IP methods, principally branch and bound procedures. Gomory and Johnson [46,47] have shown how useful cutting planes may be developed by applying group theoretical ideas to a single row of an optimal LP tableau. Johnson [62] has developed an algorithm to construct these constraints. These constraints may be used to obtain a penalty associated with integerizing a particular fractional basic variable. These penalties can be used to aid selection of a branching variable and to provide improved bounds as described in the description of branch and bound methods. Johnson and Spielberg [60] have reported promising experimental results when these penalties were used in solving knapsack problems. Kennington [63] has used the group theoretic characterization of Tompkins

and the penalty ideas of Johnson to derive a powerful branch and bound algorithm for the fixed charge transportation problem.

The group characterization of the integer programming problem focuses on the underlying mathematical structure of the problem. Applying the results directly to a problem has not produced computationally successful algorithms. It does seem that by using the group theory to identify the structure of the problem one is able to construct algorithms to best exploit the special structure of a particular problem. This is the main thrust of Chapter V where the IGFP will be formulated as a group problem.

#### 1.10 Summary

In this chapter we have given the general background for the study of the IGFP along with a survey of solution procedures available for solving the GFP. A brief survey of linear programming and integer programming has been given to provide a common terminology for discussion in later chapters.

Chapter II contains the characterization of the GFP. The methods emanating from this characterization for network programming are presented in Chapter III. In Chapter IV these methods are organized into algorithms for the GFP and two important special cases; the implementation of these algorithms on a computer is also discussed and computational results reported. In Chapter V the characterization of the IGFP is given in terms of its group representation and some results of Estabrook are extended to the IGFP. In Chapter VI the motivation for selecting a branch and bound procedure is given and the algorithm and computa-

tional results presented. Additional means of enhancing the performance of the algorithm are discussed. In Chapter VII the results are summarized and recommendations for future research are made.

## CHAPTER II

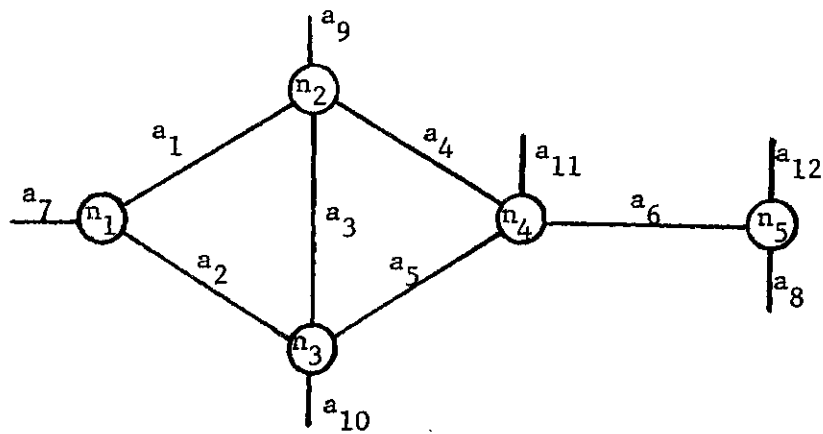
### BASIS CHARACTERIZATION FOR THE GFP

#### 2.1 Introduction

In this chapter the definitions and correspondences necessary for describing the GFP in terms of a graph are given. The simplex procedure is applied to an example problem and the operations performed in the standard simplex tableau are contrasted with the ones which result when the information structure depicted by the graph is used. The special structure of the basis for the GFP which results in the simplifications indicated in the example is formally characterized. The definitions and basis characterization given here are similar to those of Dantzig [15] and Johnson [58].

#### 2.2 Basic Graph Terminology

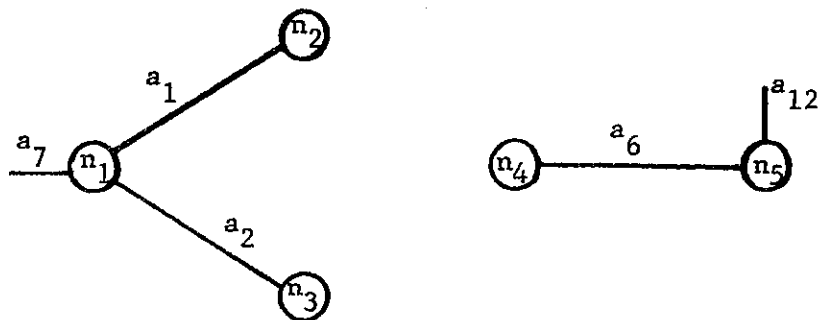
A graph  $G(N,A)$  is a finite set of nodes  $N$ (vertices) and a finite set of arcs  $A$ (edges) connecting the nodes. The set of arcs may be partitioned into two disjoint sets,  $A_r$  and  $A_s$ , defined such that each arc  $a_r$  in the set  $A_r$  is associated with a particular node pair  $(n_i, n_j)$  and each arc  $a_s$  in the set  $A_s$  is associated with a node singleton  $(n_i)$ . Arc  $a_r \in A_r$  is called a regular arc incident on nodes  $n_i$  and  $n_j$  and arc  $a_s \in A_s$  is called a slack arc incident on node  $n_i$ . Thus  $A_r$  is called the set of standard or regular arcs, and set  $A_s$  is called the set of slack arcs. A graph is shown in Figure 1(a).



$$A_r = \{a_i : i=1, \dots, 6\} \quad A_s = \{a_i : i=7, \dots, 12\}$$

$$N = \{n_i : i=1, \dots, 5\}$$

(a) A Graph



(b) A Spanning Subgraph

Figure 1. Examples of Graphical Structures

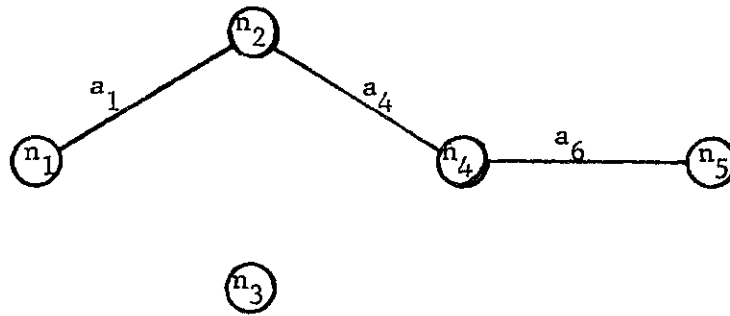
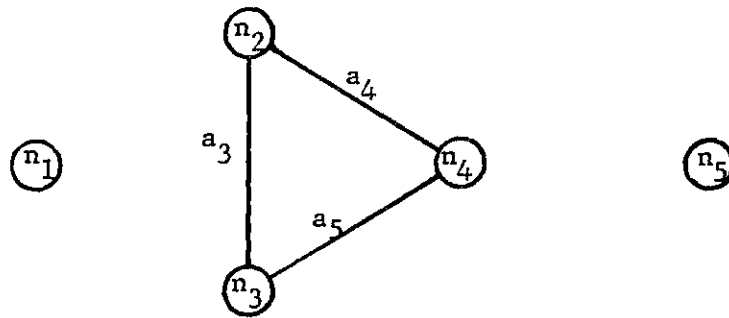
(c) A Path  $(a_1, a_4, a_6)$ (d) A Cycle  $(a_3, a_4, a_5)$ 

Figure 1. (Concluded)

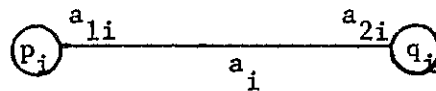
A subgraph  $G'$  of  $G$  is a graph whose node set  $N'$  is contained in the node set  $N$  (i.e.,  $N' \subset N$ ) and whose arc set  $A'$  is contained in the arc set  $A$  of  $G$  (i.e.,  $A' \subset A$ , see Fig. 1(b)). A spanning subgraph is a subgraph whose node set is the same as that of the graph (i.e.,  $N' = N$ ).

Consider a sequence of nodes and connecting arcs such as  $(n_1, a_1, n_2, a_2, \dots, a_{m-1}, n_m)$ . This sequence will be called a path between node  $n_1$  and node  $n_m$ . This sequence is called a simple path if no node is repeated. A path may be denoted by the sequence of arcs only since by definition an arc goes between two unique nodes. In a path if a node is

repeated (i.e.,  $n_i = n_j$  for some  $i$  and  $j$ ), then the portion of the path starting with  $n_i$  and ending with  $n_j$  is called a cycle. If no other node between  $n_i$  and  $n_j$  is repeated, then it is called a simple cycle.

### 2.3 Graphical Representation of the GFP

The structure of the GFP lends itself to a graphical representation. This representation focuses attention on the relationship among the variables and equations and provides insight into how the operation of various solution procedures can be efficiently carried out. Suppose the equations of a GFP are numbered  $(1, \dots, m)$ , the variables which have nonzero coefficients in two equations are  $x_1, \dots, x_n$ , and the variables which have a nonzero coefficient in one equation are  $x_{n+1}, \dots, x_{n+s}$ . To display this problem as a graph, define a set of nodes  $N = (n_1, \dots, n_m)$ . For each  $x_i \in X_s = \{x_i : i = n+1, \dots, n+s\}$ , if  $x_i$  has its nonzero coefficient in equation  $p_i$ , draw a line attached to node  $p_i$  and call it slack arc  $a_i$ . For a particular  $x_i \in X_r = \{x_i : i = 1, \dots, n\}$ , suppose a nonzero coefficient  $a_{1i}$  is in equation  $p_i$  and a nonzero coefficient  $a_{2i}$  appears in equation  $q_i$ . Connect nodes  $n_{p_i}$  and  $n_{q_i}$  and call the connecting line arc  $a_i$ . Label the arc  $a_i$  with  $a_{1i}$  at the node  $n_{p_i}$  and  $a_{2i}$  at the node  $n_{q_i}$  as follows:



Further, on the graph by placing an arrowhead on the end of arc  $a_i$  incident on node  $n_{q_i}$  we mean the coefficient near node  $n_{q_i}$  is the second nonzero coefficient of the variable  $x_i$ . In tracing a path, to traverse a forward arc  $a_i$  we mean that the node(equation) associated with  $a_{1i}$  is

encountered before the node(equation) associated with  $a_{2i}$ . To traverse a reverse arc we mean that the node(equation) associated with  $a_{2i}$  is encountered before the one associated with  $a_{1i}$ . Graphically, in a path a forward arc is traversed tail to head and a reverse arc is traversed head to tail.

For the methods used to solve the generalized flow problem in this dissertation these arbitrary definitions do not mean as much as for other methods. If the signs of  $a_{1i}$  and  $a_{2i}$  are opposite for all  $i$ , then the usual interpretation on the graph is of flow from the node with positive coefficient to the node with negative coefficient. These conventions are consistent with those for ordinary flow problems used by Ford and Fulkerson [21] with  $a_{1i} = +1$  and  $a_{2i} = -1$  for all regular arcs.

A convention similar to that above can be adopted for slack arcs. Usually slack arcs will correspond to slack or artificial variables which appear with a plus or minus one in a single equation. For slack arcs in general, the convention will be if the variable  $x_i$  appears with a positive coefficient in equation  $p_i$ , then the corresponding arc  $a_i$  will have an arrowhead on the end not attached to node  $p_i$ . On the other hand, if  $x_i$  appears with a minus sign in equation  $p_i$ , the arrowhead is on the end of arc  $a_i$  incident on node  $p_i$ . Figure 2 is an example of a generalized flow problem and the corresponding graph. The slack variables  $x_7$  and  $x_8$  and the artificials  $x_9$  to  $x_{12}$  have been added to the problem and are shown as slack arcs on the graph. The coefficients are shown at the respective ends of each arc and the costs and bounds are shown in parentheses on each arc as  $(c_i, M_i)$ . No distinction is made between slack and artificial variables on the graph as both are represented as slack arcs.

Minimize  $3x_1 + 2x_2 + 2x_3 + 3x_4 + x_5 + 2x_6$

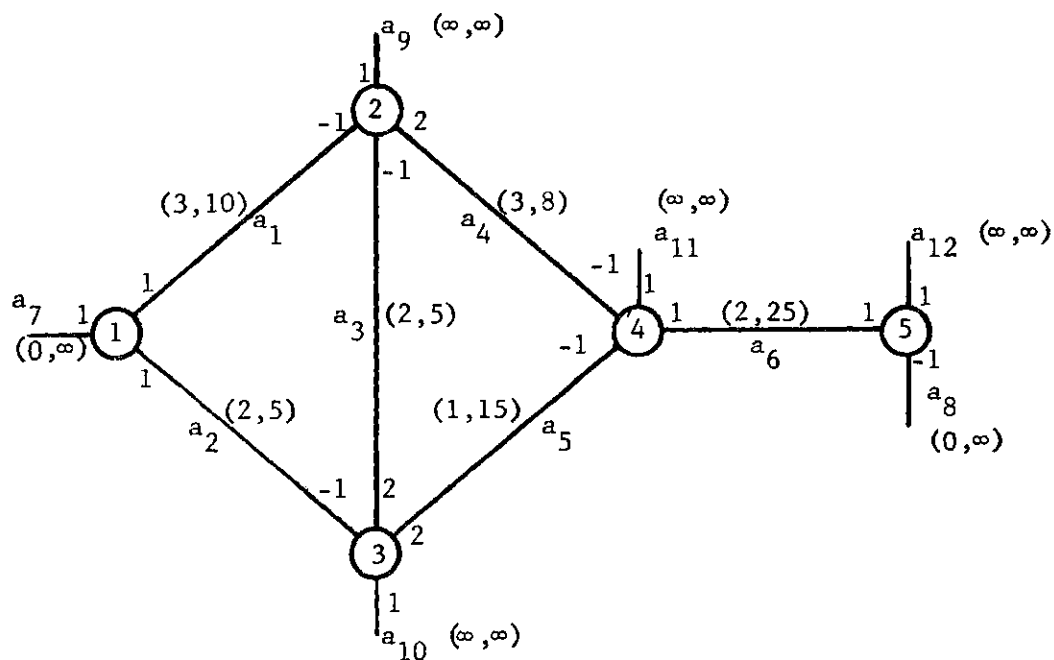
Subject to:

$$\begin{array}{rcl}
 x_1 + x_2 & + x_7 & = 20 \\
 -x_1 - x_3 + 2x_4 & + x_9 & = 0 \\
 -x_2 + 2x_3 + 2x_5 & + x_{10} & = 0 \\
 -x_4 - x_5 + x_6 & + x_{11} & = 0 \\
 +x_6 - x_8 & + x_{12} & = 5
 \end{array}$$

$$0 \leq x_1 \leq 10 \quad 0 \leq x_2 \leq 5 \quad 0 \leq x_3 \leq 5$$

$$0 \leq x_4 \leq 8 \quad 0 \leq x_5 \leq 15 \quad 0 \leq x_6 \leq 25$$

(a) A Generalized Flow Problem



(b) Graphical Representation of a GFP

Figure 2. Generalized Flow Example

There is a one to one correspondence between equations and nodes, and variables and arcs. Normally when referring to the graph, the terms nodes and arcs will be used, but on occasion to facilitate the presentation, some liberty will be taken in using these terms interchangeably.

The approach taken throughout this research is to utilize the information structure of the GFP as displayed in its corresponding graph to develop solution techniques for the GFP and related special cases of the GFP as linear programming problems. This is in sharp contrast to the approach of Jewell [56] and Ford and Fulkerson [21] where properties of the associated graph are used as the basis of the solution technique.

#### 2.4 GFP Example Problem

An example of a GFP problem is given in Figure 2 along with its corresponding graphical representation. Slack and artificial variables have been added in Figure 2(b). The application of the upper bounded simplex method to the problem generates the sequence of tableaux in Table 1. The big M method was used, and the first nonbasic variable encountered whose current cost indicated improvement of the objective function was chosen as the entering variable. The sequence of subgraphs in Figure 3 shows the basic variables at each iteration of the simplex and the entering nonbasic variable is shown in dashed lines.

To highlight the use of the graph for characterizing the GFP and its use in providing the necessary information to solve the problem with a simplex procedure, several observations will be made on the solution of the example problem. This is done to highlight areas where the graphical representation provides information for solving the problem more effec-

Table 1. Simplex Iterations for GFP Example

		$c_j$	3	2	2	3	1	2	0	0	100	100	100	100	
		$M_j$	10	5	5	8	15	25	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	
Basic	Var. $\bar{b}$	$\bar{b}$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$\pi$
A	$x_7$	$\infty$	20	1	1				1						0
	$x_9$	$\infty$	0	-1		-1	2				1				100
	$x_{10}$	$\infty$	0		-1	2		2				1			100
	$x_{11}$	$\infty$	0				-1	-1	1				1		100
	$x_{12}$	$\infty$	5						1		-1			1	100
			103	102	-98	-97	-99	2			100				

$x_{10}$  leaves,  $x_5$  enters. Rows 3 and 4,  $\pi_3$ ,  $\bar{c}_2$ ,  $\bar{c}_3$ , and  $\bar{c}_5$  change.

Table 1. (Continued)

	Basic Var.	$\bar{b}'$	$\bar{b}$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$\pi$
B	$x_7$	$\infty$	20	1	1					1						0
	$x_9$	$\infty$	0	-1		-1	2					1				100
	$x_5$	15	0		$-\frac{1}{2}$	1		1					$\frac{1}{2}$			$\frac{101}{2}$
	$x_{11}$	$\infty$	0		$-\frac{1}{2}$	1	-1		1				$\frac{1}{2}$	1		100
	$x_{12}$	$\infty$	5						1		-1				1	100
				103	$\frac{103}{2}$	1	-97		2		100					

$x_9$  leaves,  $x_4$  enters. Rows 2 and 4,  $\pi_2$ ,  $\bar{c}_1$ ,  $\bar{c}_3$ , and  $\bar{c}_4$  change.

C	$x_7$	$\infty$	20	1	1					1						0
	$x_4$	8	0	$-\frac{1}{2}$		$-\frac{1}{2}$	1					$\frac{1}{2}$				$\frac{103}{2}$
	$x_5$	15	0		$-\frac{1}{2}$	1		1					$\frac{1}{2}$			$\frac{101}{2}$
	$x_{11}$	$\infty$	0	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$			1			$\frac{1}{2}$	$\frac{1}{2}$	1		100
	$x_{12}$	$\infty$	5						1		-1				1	100
				$\frac{109}{2}$	$\frac{103}{2}$	$-\frac{89}{2}$			2		100					

$x_{11}$  leaves,  $x_3$  enters. Rows 2, 3, and 4,  $\pi_2$ ,  $\pi_3$ , and  $\pi_4$ ,  $\bar{c}_3$ ,  $\bar{c}_1$ ,  $\bar{c}_2$ , and  $\bar{c}_6$  change.

Table 1. (Continued)

	Basic Var. $\bar{b}$	$\bar{b}$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$\pi$
D	$x_7$	$\infty$	20	1	1				1						0
	$x_4$	8	0	-1	$-\frac{1}{2}$	1		1			1	$\frac{1}{2}$	1		4
	$x_5$	15	0	1	$\frac{1}{2}$		1	-2			-1	$-\frac{1}{2}$	-2		3
	$x_3$	5	0	-1	-1	1		2			1	1	2		5
	$x_{12}$	$\infty$	5					1		-1				1	100
				7	5			-103		100					
$x_4$ leaves, $x_6$ enters. Rows 3, 4, and 5, $\pi_2$ , $\pi_3$ , and $\pi_4$ , $\bar{c}_6$ , $\bar{c}_4$ , $\bar{c}_1$ , and $\bar{c}_2$ change.															
E	$x_7$	$\infty$	20	1	1				1						0
	$x_6$	25	0	-1	$-\frac{1}{2}$	1		1			1	$\frac{1}{2}$	1		-99
	$x_5$	15	0	-1	$-\frac{1}{2}$	2	1				1	$\frac{1}{2}$			$-\frac{97}{2}$
	$x_3$	5	0	1		1	-2				-1				-98
	$x_{12}$	$\infty$	5	1	$\frac{1}{2}$		-1			-1				1	100
				-96	$-\frac{93}{2}$		103			100					
$x_3$ leaves, $x_1$ enters. Rows 1, 2, 3, and 4, $\pi_2$ , $\bar{c}_1$ and $\bar{c}_3$ change.															

Table 1. (Continued)

	Basic Var.	$\bar{b}'$	$\bar{b}$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$\pi$
F	$x_7$	$\infty$	20		1	-1	2			1						0
	$x_6$	25	0		$-\frac{1}{2}$	1	-1		1			1				-3
	$x_5$	15	0		$-\frac{1}{2}$	1		1					$\frac{1}{2}$	1		$-\frac{97}{2}$
	$x_1$	10	0	1		1	-2						$\frac{1}{2}$			-98
	$x_{12}$	$\infty$	5		$\frac{1}{2}$	-1	1				-1	-1			1	100
					$-\frac{93}{2}$	96	-89				100					
	$x_{12}$ leaves, $x_4$ enters. Rows 1,2,4, and 5, $\pi_3$ , $\pi_4$ , and $\pi_5$ , $\bar{c}_2, \bar{c}_3, \bar{c}_4$ , and $\bar{c}_8$ change.															
G	$x_7$	$\infty$	10			1				1	2	2			-2	0
	$x_6$	20	5						1		-1				1	-3
	$x_5$	15	0		$-\frac{1}{2}$	1		1					$\frac{1}{2}$	1		-4
	$x_1$	0	10	1	1	-1					-2	-2	$\frac{1}{2}$		2	-9
	$x_4$	8	5		$\frac{1}{2}$	-1	1				-1	-1			1	11
					-2	7					9					

$x_2$  enters at upper bound. No changes except in right hand side.

Table 1. (Continued)

	Basic Var.	$\bar{b}'$	$\bar{b}$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$\pi$
H	$x_7$	$\infty$	10			1				1	2	2			-2	0
	$x_6$	20	5						1		-1			1	1	-3
	$x_5$	$12\frac{1}{2}$	$2\frac{1}{2}$		$-\frac{1}{2}$	1		1					$\frac{1}{2}$			-4
	$x_1$	5	5	1	1	-1					-2	-2	$\frac{1}{2}$		2	-9
	$x_4$	$5\frac{1}{2}$	$2\frac{1}{2}$		$\frac{1}{2}$	-1	1				-1	-1			1	11
					-2	7					11					
					U											

The optimal solution is  $x_1 = 5$ ,  $x_2 = 5$ ,  $x_3 = 0$ ,  $x_4 = 2\frac{1}{2}$ ,  $x_5 = 2\frac{1}{2}$ ,  $x_6 = 5$ ,

$x_7 = 10$ , and  $x_8 = 0$ .

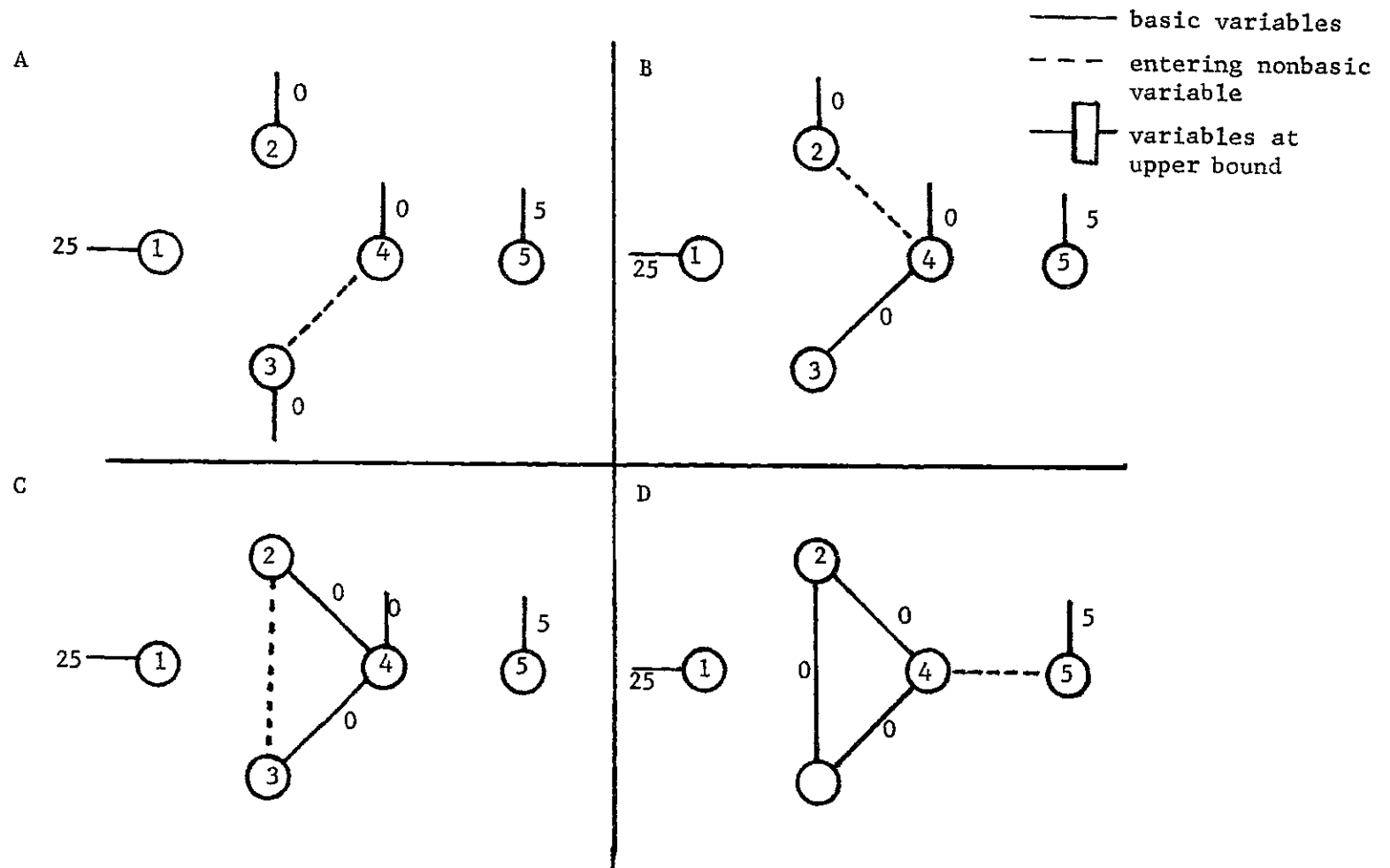
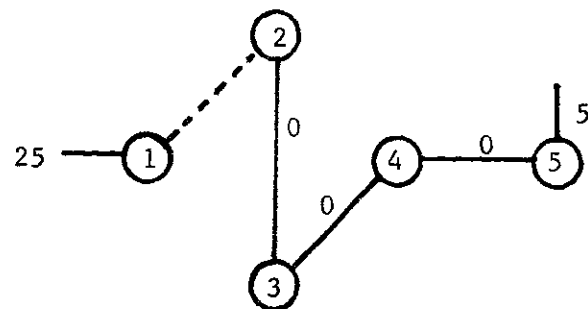
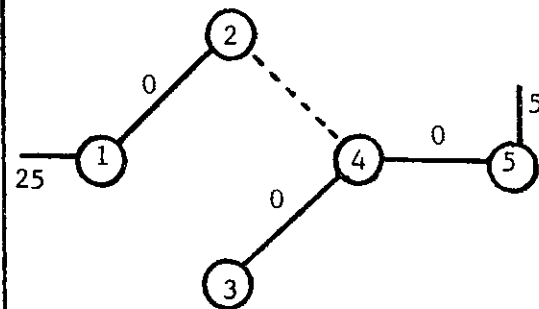


Figure 3. Graphical Representation of Simplex Iterations

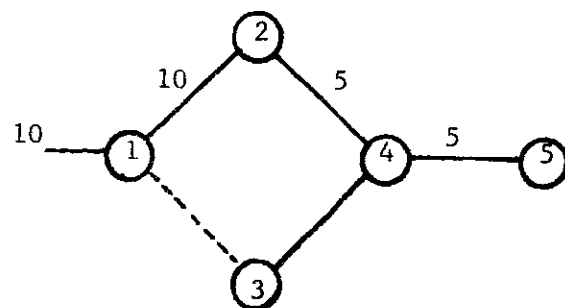
E



F



G



H

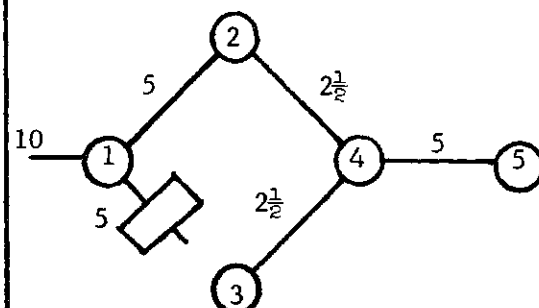


Figure 3. (Concluded)

tively than the straightforward application of the simplex method. Certain assertions will be made without proof at this stage. However, these will be proved later in Chapters II and III when the problem structure and solution procedure are formally characterized.

From either end of a nonbasic variable a simple path containing only basic arcs may be traced to either (i) a basic slack arc or (ii) a simple cycle of only basic arcs. In diagram F of Figure 3 nonbasic arc  $a_4$  goes between nodes 2 and 4. From node 2 the path  $(2, a_1, 1)$  leads to basic slack arc  $a_7$ . From node 4 the path  $(4, a_6, 5)$  leads to basic slack arc  $a_9$ . In diagram D nonbasic arc  $a_6$  goes between node 4 and node 5. Node 4 is contained in the simple cycle  $(4, a_5, 3, a_3, 2, a_4, 4)$  and the path (5) containing a single node leads to the basic slack arc  $a_{12}$ . This relationship between nonbasic and basic variables leads to the development of efficient solution procedures for network type problems (ordinary flow, transportation, etc.) and is a direct result of the basis structure of these problems. This nonbasic-basic variable relationship leads to an efficient procedure for determining the current column representation of a nonbasic variable. In the simplex method this is required to determine the variable leaving the basis.

In the primal simplex method the simplex multipliers (dual variables) correspond to the nodes and are used to determine when a nonbasic variable is to enter the basis. Note that, in the example problem, usually not all of the simplex multipliers change from one tableau to the next. The fact that the simplex multipliers which change can be identified directly leads to an efficient procedure for the pricing operation

in the simplex calculations. The structure of the GFP can be exploited to perform the two primary operations of the simplex: determination of an entering nonbasic variable and determination of the leaving basic variable.

## 2.5 Basis Characterization

The concept of a basic solution is fundamental to the simplex procedure. A basis, basic solution, and related ideas were summarized in Section 1.8. Another characterization of a basis is given by the statement that  $B$  is a basis of  $A$ , if and only if each column of  $A$  can be represented as a unique linear combination of the columns of  $B$ . The proof of this may be found in any book on linear algebra (e.g., Hildebrand [52]).

In the GFP the basis has a special structure. To discuss this we need the following terminology. A matrix arranged as in Figure 4 with identifiable square blocks which have no nonzero entries in common rows or columns is called a block diagonal matrix. A slack arc whose associated variable is basic will be called a root and the node on which it is incident will be called the root node. If all the variables corresponding to the arcs contained in a simple cycle are basic, the cycle will be called a pseudoroot. Assume that the matrix in Figure 4 is the basis for a GFP problem. Block  $B_1$  contains the root  $a_1$  shown in the subgraph corresponding to  $B_1$  in Figure 5(a). Block  $B_2$  contains the pseudoroot  $(a_6, a_7, a_8, a_9)$  shown graphically in Figure 5(b).

We will now state several lemmas and a theorem related to the structure of the basis. Theorem 2.1 below is essentially Johnson's result in [58]. However, we have adopted a constructive proof which forms the nu-

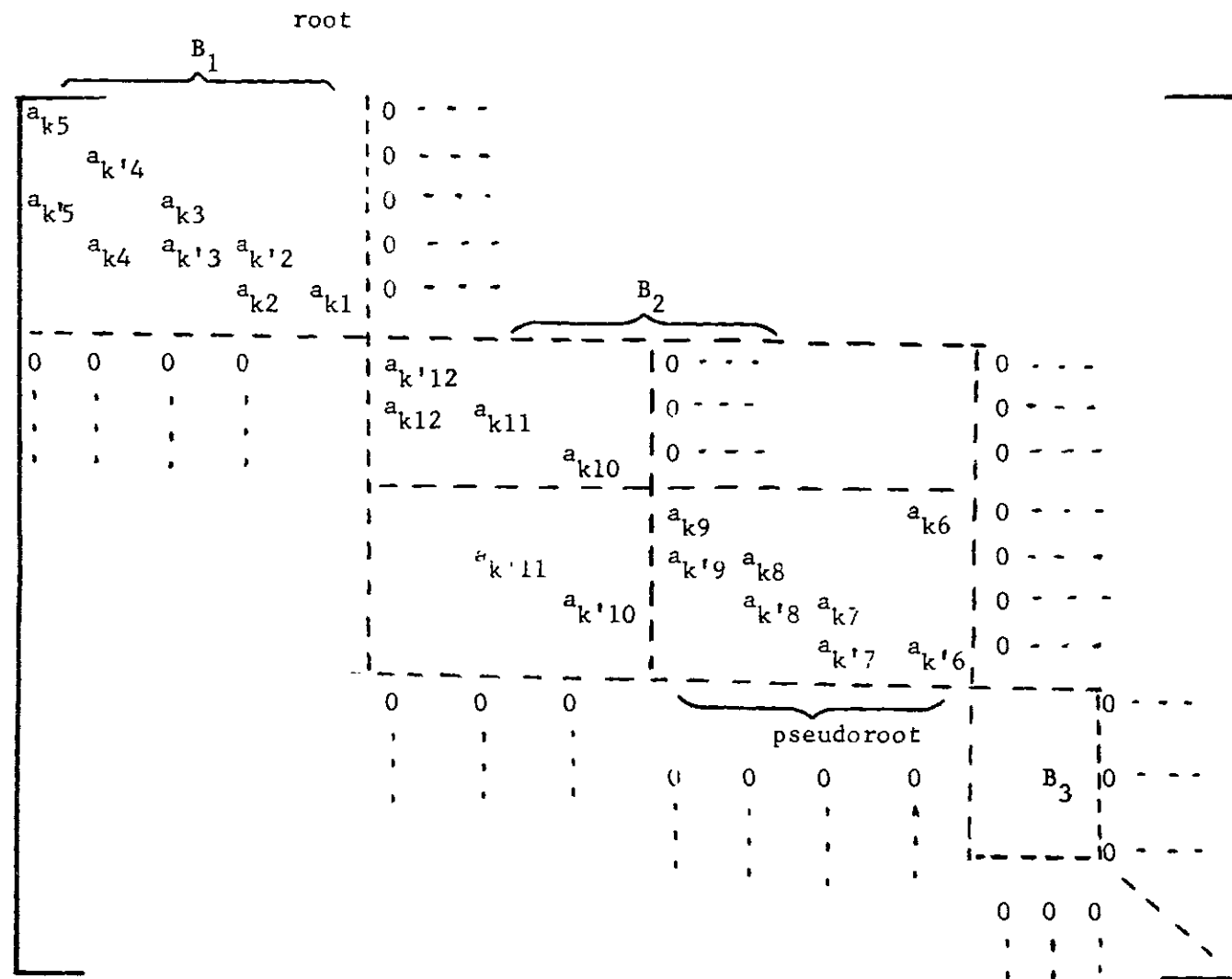
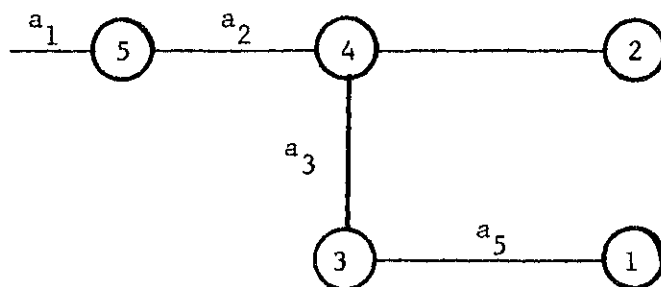
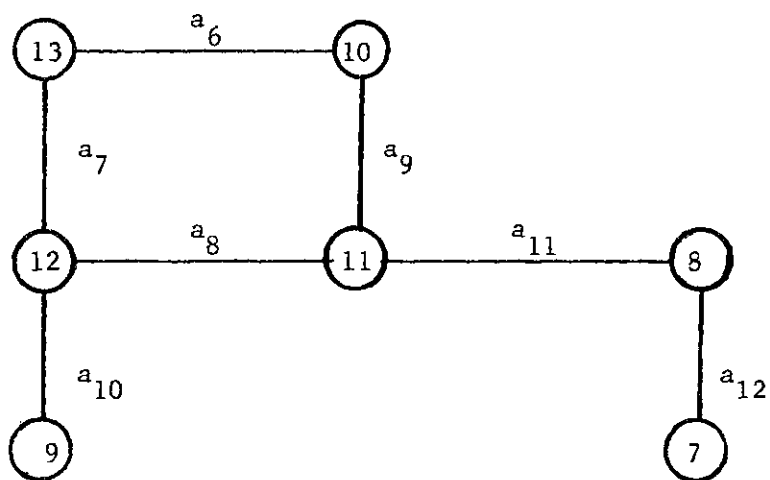


Figure 4. Block Diagonal Matrix



(a) Rooted Component

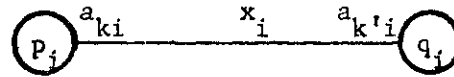


(b) Pseudorooted Component

Figure 5. Basis Components

cleus of the solution procedure discussed in Chapters III and IV. Before the statement of the lemmas and the theorem, we will define some notation and terminology which are used throughout the remainder of the presentation.

When going from node  $p_i$  to node  $q_i$  along the arc for the variable  $x_i$ , the first coefficient encountered, which is in equation  $p_i$ , will be denoted  $a_{ki}$  and the second coefficient will be denoted  $a_{k'i}$  and is in equation  $q_i$ . An example is:



Consider an arc  $a_1$  and the sequence of arcs  $(a_2, \dots, a_{r+1})$  following  $a_1$  in a simple path. The ordered set of coefficient pairs for the sequence of arcs  $(a_1, a_2, \dots, a_{r+1})$  is  $(a_{k1}, a_{k'1}), (a_{k2}, a_{k'2}), \dots, (a_{kr+1}, a_{k'r+1})$  in the notation defined above. Associate with each arc  $a_{i+1}$  after the designated arc  $a_1$  the weight  $d_i$  defined in the following manner:

$$d_1 = \frac{a_{k'1}}{a_{k2}}, \quad d_2 = -\frac{d_1 a_{k'2}}{a_{k3}}, \dots, \quad d_r = -\frac{d_{r-1} a_{k'r}}{a_{kr+1}}$$

or

$$d_k = (-1)^{k-1} \prod_{i=1}^k \left( \frac{a_{k'i}}{a_{ki+1}} \right) \quad (1)$$

We will now proceed to the lemmas and theorem.

Let  $B_i$  be a set of columns of a basis  $B$  for the GFP with the following property. The original rows of  $B$  corresponding to  $B_i$  contain

no nonzero entries other than in the columns of  $B_i$ . Also, any nonzero entry in  $B_i$  can be reached from any other nonzero entry in  $B_i$  by a series of alternating row and column moves through other nonzero entries.

Lemma 2.1

If  $B_i$  is defined as above, then it cannot contain two columns which correspond to slack arcs (roots).

Proof:

Assume to the contrary. That is, let  $B_i$  contain columns  $b_1$  and  $b_{r+1}$  which contain the single nonzero entries  $a_{k'1}$  and  $a_{kr+1}$ , respectively. The nonzero entry  $a_{kr+1}$  can be reached from  $a_{k'1}$  by a series of alternating row and column moves. Let the columns encountered in such a trace be used to define a path. Remove the appropriate arcs (and columns) from the path to make it a simple path. Let the path now be  $(a_1, a_2, \dots, a_{r+1})$  with  $a_1$  and  $a_{r+1}$  as the slack arcs. Consider the linear combination of the associated columns  $b_2, \dots, b_{r+1}$  obtained using the arc weights previously defined:

$$g = \sum_{i=1}^r d_i b_{i+1}$$

Considering only the rows of the  $B_i$  with nonzero elements in the path:

$$g' = \begin{bmatrix} d_1 a_{k2} \\ d_1 a_{k'2} + d_2 a_{k3} \\ \vdots \\ \vdots \\ d_{r-1} a_{k'r} + d_r a_{kr+1} \end{bmatrix} \quad (2)$$

The vector  $g'$  is denoted as  $g' = (g'_1, \dots, g'_r)^\top$

The first element is:

$$g'_1 = a_{k2} \left( \frac{a_{k'1}}{a_{k2}} \right) = a_{k'1} \quad (3)$$

For any other entry  $g'_i$ ,  $i = 2, \dots, r$

$$g'_i = d_{i-1} a_{k'i} + d_i a_{ki+1}$$

But:

$$d_i = - \frac{d_{i-1} a_{k'i-1}}{a_{ki+1}}$$

Thus:

$$g'_i = d_{i-1} a_{k'i} - \left( \frac{d_{i-1} a_{k'i-1}}{a_{ki+1}} \right) a_{ki+1} = 0 \quad (4)$$

The vector  $g$  is thus:

$$g = \begin{bmatrix} 0 \\ \vdots \\ a_{k'1} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} = b_1 \quad (5)$$

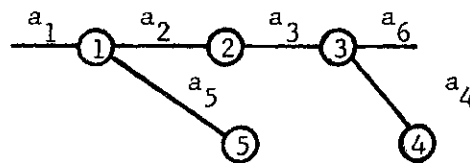
The column  $b_1$  has been represented as a linear combination of other columns of the basis. This contradicts the definition of a basis and the lemma is proved.

An example illustrating the lemma is given below.

Let  $B$  be:

$$B = \begin{bmatrix} a_{k'1} & a_{k2} & & a_{k5} & & \\ & a_{k'2} & a_{k3} & & & \\ & & a_{k'3} & a_{k6} & & \\ & & & & a_{k'5} & \\ & & & & & a_{k'4} \end{bmatrix} \quad (6)$$

The subgraph corresponding to B is:



$B_i$  is:

$$B_i = \begin{bmatrix} a_{k'1} & a_{k2} & & \\ & a_{k'2} & a_{k3} & \\ & & a_{k'3} & a_{k6} \end{bmatrix} \quad (7)$$

The two slack arcs are  $a_1$  and  $a_6$ . The set of arcs containing these two arcs and the path between them is  $(a_1, a_2, a_3, a_6)$ . The weights  $d_i$  are:

$$d_1 = \frac{a_{k'1}}{a_{k2}} \quad d_2 = - \frac{a_{k'1}a_{k'2}}{a_{k2}a_{k3}} \quad d_3 = + \frac{a_{k'1}a_{k'2}a_{k'3}}{a_{k2}a_{k3}a_{k6}} \quad (8)$$

The linear combination of the columns for  $a_2$ ,  $a_3$ , and  $a_6$  is:

$$g = \begin{bmatrix} \frac{a_{k'1}}{a_{k2}} a_{k2} \\ \frac{a_{k'1}}{a_{k2}} a_{k'2} - \frac{a_{k'1} a_{k'2}}{a_{k2} a_{k3}} a_{k3} \\ - \frac{a_{k'1} a_{k'2} a_{k'3}}{a_{k2} a_{k3}} + \frac{a_{k'1} a_{k'2} a_{k'3}}{a_{k2} a_{k3} a_{k6}} a_{k6} \end{bmatrix} = \begin{bmatrix} a_{k'1} \\ 0 \\ 0 \end{bmatrix} \quad (9)$$

which, with additional zero entries, is the column for  $a_1$ .

Lemma 2.2

Let  $B_i$  be defined as before, then  $B_i$  cannot have a set of columns corresponding to a simple cycle (pseudoroot) and a column corresponding to a slack arc (root).

Proof:

Assume to the contrary, that is, assume  $B_i$  has a simple cycle and a slack arc. As in Lemma 2.1 a series of row and column moves can be made to define a simple path between the slack arc and an arc in the cycle. Choose a column from the cycle.

$$\underline{b} = \begin{bmatrix} 0 \\ \vdots \\ a_{kp} \\ 0 \\ a_{k'p} \end{bmatrix} \quad (10)$$

This column can be represented as the sum of two columns each with one nonzero entry:

$$\underline{b} = \underline{b}_t + \underline{b}_s = \begin{bmatrix} 0 \\ \vdots \\ a_{kp} \\ 0 \\ \vdots \\ \vdots \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ a_{k'p} \\ 0 \\ \vdots \end{bmatrix} \quad (11)$$

Taking the root and  $b_t$  as the two slack arcs in Lemma 2.1, there exist multipliers  $d'_i$  associated with the arcs in the path from the root to  $b_t$  such that

$$b_t = \sum_{i=1}^{r_1} d'_i b'_i \quad (12)$$

Likewise, there exist multipliers  $d''_i$  associated with arcs in the path from the root to  $b_s$  such that:

$$b_s = \sum_{i=1}^{r_2} d''_i b''_i$$

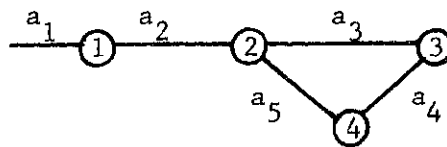
Adding:

$$b = b_t + b_s = \sum_{i=1}^r d_i b_i \quad (13)$$

The components  $b_i$ ,  $i=1, \dots, r$  correspond to the arcs in the two paths considered, and  $d_i = d'_i + d''_i$  for arcs contained in both paths and  $d_i = d'_i$  or  $d''_i$  if they are in only one of the paths. An example of the situation covered in this lemma is:

$$B_i = \begin{bmatrix} a_{k1} & a_{k'2} & & & \\ & a_{k2} & a_{k'3} & & a_{k'5} \\ & & a_{k3} & a_{k'4} & \\ & & & a_{k4} & a_{k5} \end{bmatrix} \quad (14)$$

The associated subgraph is:



Choose the column for  $a_4$  as the column from the cycle to separate.

$$b = \begin{bmatrix} 0 \\ 0 \\ a_{k'4} \\ a_{k4} \end{bmatrix} = b_t + b_s = \begin{bmatrix} 0 \\ 0 \\ a_{k'4} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ a_{k4} \end{bmatrix} \quad (15)$$

The path for  $b_t$  is  $(a_3, a_2, a_1)$  and for  $b_s$   $(a_5, a_2, a_1)$ . The multipliers are:

$$\begin{aligned} d'_1 &= \frac{a_{k'4}}{a_{k3}} & d'_2 &= -\frac{a_{k'4}a_{k'3}}{a_{k3}a_{k2}} & d'_3 &= \frac{a_{k'4}a_{k'3}a_{k'2}}{a_{k3}a_{k2}a_{k1}} \\ d''_1 &= \frac{a_{k4}}{a_{k5}} & d''_2 &= -\frac{a_{k4}a_{k'5}}{a_{k5}a_{k2}} & d''_5 &= \frac{a_{k4}a_{k'5}a_{k'2}}{a_{k5}a_{k2}a_{k1}} \end{aligned} \quad (16)$$

The linear combination is:

$$b = (d'_1 + d''_1)b_1 + (d'_2 + d''_2)b_2 + d'_3b_3 + d''_5b_5 \quad (17)$$

Hence  $b$  has been expressed as a linear combination of other basic vectors which is impossible.

Lemma 2.3

With  $B_i$  defined as before,  $B_i$  cannot have a set of columns corresponding to two simple cycles (pseudoroots).

Proof:

Again by contradiction assume that  $B_i$  contains columns corresponding to the two cycles. Define the path between the two cycles as before. Choose a column corresponding to an arc in one of the cycles and separate it into two columns containing only one nonzero entry.

But in the proof of Lemma 2.2 we have shown that each of these can be expressed as a linear combination of basic arcs. Adding, the arc we separated can then be represented as a linear combination of basic arcs. This contradicts the assumption that  $B$  is a basis and the lemma is proved.

Lemma 2.4

For a set of  $k$  columns,  $B_i$ , as considered in the previous lemmas, there are exactly  $k$  rows with nonzero entries.

Proof:

Assume that there are nonzero entries in less than  $k$  rows. Of the  $k$  columns of  $B_i$  either two columns have a single nonzero entry, or one column has one nonzero entry and a subset of the columns forms a cycle, or two distinct subsets of the columns form cycles. But by Lemmas 2.1, 2.2, and 2.3 none of these conditions are possible and hence at least  $k$  rows of  $B_i$  must contain nonzero entries.

Now assume that more than  $k$  rows of  $B_i$  contain nonzero entries. By definition of  $B_i$  this can only happen if each of the columns of  $B_i$  contains two nonzero entries and no subset of the columns forms a cycle (the set of columns contains no root or pseudoroot). Thus there can be at most  $k+1$  rows of  $B_i$  with nonzero entries. Let the remaining columns of the basis  $B$  be arranged into sets in the same manner as  $B_i$ . Since no set of columns has nonzero entries in rows where another set has nonzero entries, and since the total number of rows equals the total number of columns, then at least one of the sets of columns must have nonzero entries in one fewer rows than it has columns. But we have shown this to be impossible in the first part of the proof. Hence  $B_i$  must have nonzero entries in  $k$  rows, where  $k$  is the number of columns in  $B_i$ .

A characterization of the basis for the GFP can now be given using the lemmas. The following theorem is fundamental to the remainder of the presentation.

Theorem 2.1

A basis for the generalized flow problem is block diagonal with each block containing either one column corresponding to a slack arc in the associated subgraph or a set of columns corresponding to a single cycle in the subgraph but not both. That is, no block contains columns for two roots, two pseudoroots, or a root and a pseudoroot.

Proof:

We will first establish constructively that the basis is block diagonal. A basis consists of  $m$  linearly independent columns of  $A$ .

Choose one of these columns. Either it has one or two entries:

$$(i) \begin{bmatrix} 0 \\ \vdots \\ a_{ik} \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad \text{or} \quad (ii) \begin{bmatrix} 0 \\ \vdots \\ a_{ik} \\ 0 \\ \vdots \\ \vdots \\ a_{ik'} \\ 0 \\ \vdots \\ \vdots \end{bmatrix} \quad (19)$$

Select all other columns with entries in the row containing  $a_{ik}$  in case (i) or in the rows containing  $a_{ik}$  and  $a_{ik'}$  in case (ii). For each column so selected repeat the process. Continue until either all the  $m$  columns have been selected or until no columns have entries in the previously selected columns. If  $m$  columns have been selected, then there is one set of columns. Otherwise repeat the procedure to define sets  $2, \dots, R$ .

The maximum number of sets is obviously  $m$  with a column for a slack arc in each set.

By the constructive procedure used to define the sets, each set of columns is of the form of  $B_i$  in Lemmas 2.1 through 2.4. Let the set of rows with nonzero entries in each set define the block for each set of columns. By Lemmas 2.1 through 2.3, no block contains two roots, a root and a pseudoroot, or two pseudoroots. Also by the proof of Lemma 2.4 each block must contain a root or a pseudoroot. The theorem is thus proved.

Several definitions can now be made which are useful in describing the algebraic operations of an algorithm in terms of the corresponding graph. The connected subgraph corresponding to a block  $B_i$  of the basis is called a tree. If the block  $B_i$  contains a slack column, the tree is a rooted tree. If the block contains a set of columns corresponding to a simple cycle, then the tree is a pseudorooted tree. The set of connected subgraphs, one for each block of the basis, is called the basis forest. Each tree (rooted or pseudorooted) will be called a component of the basis forest and its corresponding block will be called a component of the basis matrix  $B$ .

To facilitate the presentation these definitions differ slightly from the usual definitions of a tree and forest (e.g., Johnson [58]) in that the slack column or root and the cycle or pseudoroot are incorporated into the definitions instead of being considered separately.

Two useful corollaries follow directly from Theorem 2.1.

Corollary: There is a unique path of basic arcs from each node in a tree to the root or pseudoroot.

Proof:

Assume to the contrary. That is, suppose that there are two paths from node  $n_k$  to the root or the pseudoroot. Denote these paths as

$$P_1 = (a_{i_1}, \dots, a_{i_p}) \quad (20)$$

$$P_2 = (a_{j_1}, \dots, a_{j_q})$$

Consider two cases. First assume the tree is rooted. Then arcs  $a_{i_p}$  and  $a_{j_q}$  are incident on the same node to which the root is attached. Also,  $a_{i_1}$  and  $a_{j_1}$  are both incident on node  $n_k$ . Then  $(a_{i_1}, \dots, a_{i_p}, a_{j_q}, \dots, a_{j_1})$  defines a cycle of basic arcs in the tree. But by Theorem 2.1 a tree cannot contain a cycle and a root, thus for a rooted tree the corollary is proved.

For the case of a pseudorooted tree, if  $P_1$  and  $P_2$  are incident on the same node of the pseudoroot, the proof is the same as if the tree were rooted. Otherwise, let  $n_p$  and  $n_q$  be the nodes of the pseudoroot at which  $P_1$  and  $P_2$  end. Let  $a_{k_1}, \dots, a_{k_r}$  be the arcs in the path between  $n_p$  and  $n_q$  in an arbitrary direction around the pseudoroot. Since  $a_{i_1}$  and  $a_{j_1}$  are both incident on node  $n_k$  and  $P_1 \neq P_2$ , then  $(a_{i_1}, \dots, a_{i_p}, a_{k_1}, \dots, a_{k_r}, a_{j_q}, \dots, a_{j_1})$  defines a cycle of basic arcs, different from the pseudoroot. Thus the basis tree contains two cycles which contradicts Theorem 2.1 and the corollary is proved.

Corollary: Let  $b_i$  be a column of a component  $B_k$  of the basis. Then, deleting  $b_i$  will partition  $B_k$  into two subcomponents  $B'_k$  and  $B''_k$  (one of them possibly empty) such that the corresponding two subblocks of the matrix have no nonzero entries in the intersection of their rows and

columns. If  $b_i$  is not the root or part of the pseudoroot, then one of the subcomponents contains the root or the pseudoroot of  $B_k'$ .

Proof:

Assume  $b_i$  has nonzero entries in two rows. Select the row containing one of these nonzero entries and proceed as in the proof of Theorem 2.1 for constructing components by selecting all columns with nonzero entries in the row, then considering the rows where these have other nonzero entries, etc. In this manner  $B_k''$  is constructed and the remaining columns of  $B_k$  constitute  $B_k'$  (possibly  $B_k'$  is empty). For the same reason that no two components in Theorem 2.1 have nonzero entries in common, neither will  $B_k'$  and  $B_k''$ . The corollary is proved.

It may be noted that, if  $b_i$  corresponds to the root or an arc in the cycle denoting the pseudoroot, then deletion of  $b_i$  will not yield two separate blocks, i.e., we will have  $B_k' = \emptyset$  and  $B_k'' = B_k$  without column  $b_i$ .

Thus, in all cases removal of column  $b_i$  yields a subcomponent  $B_k''$  which does not contain the root or the complete pseudoroot of  $B_k$ . We will have frequent occasions to refer to this block in later chapters. We will call  $B_k''$  the portion of  $B_k$  above  $b_i$  or above  $x_i$ , the variable associated with  $b_i$ . The terminology is more meaningful in terms of the graph of  $B_k$ .  $B_k''$  will correspond to that part of the tree above the arc for  $x_i$  (i.e., away from the root) and the unique path from any node in  $B_k''$  to the root will contain the arc for  $x_i$ .

This corollary is illustrated by the following example:

$$B_k = \begin{bmatrix} a_{11} & & & & & a_{16} \\ a_{21} & & & a_{14} & & \\ & a_{12} & & & & \\ & & a_{13} & a_{24} & a_{15} & \\ & a_{22} & a_{23} & & & \\ & & & & a_{25} & \end{bmatrix} \quad (21)$$

Choose  $b_1 = b_4$  which has nonzero entries in rows two and four. Selecting the entry in row two places  $b_1$  in  $B'_k$  which, in turn, places  $b_6$  in  $B'_k$ .

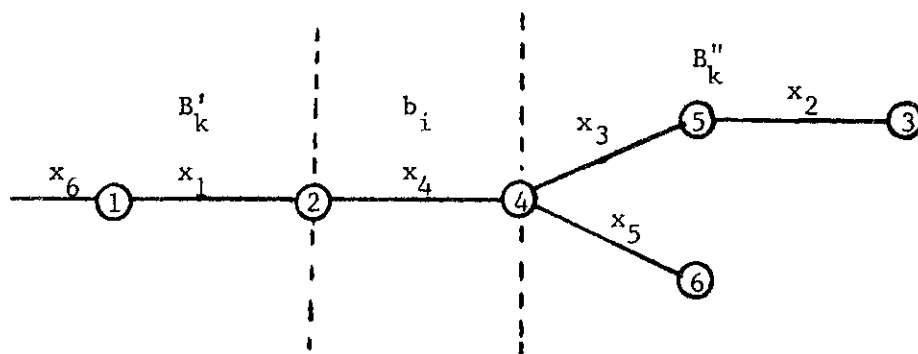
$$B'_k = \begin{bmatrix} a_{11} & a_{16} \\ a_{21} & 0 \end{bmatrix} \quad (22)$$

$$B''_k = \begin{bmatrix} 0 & 0 & a_{12} \\ a_{13} & a_{15} & 0 \\ a_{23} & 0 & a_{22} \\ 0 & a_{25} & 0 \end{bmatrix}$$

By rearrangement of rows and columns  $B_k$  can be written:

$$B_k = \begin{bmatrix} a_{12} & & & & & \\ & a_{25} & & & & \\ a_{22} & & a_{23} & & & 0 \\ & a_{15} & a_{13} & a_{24} & & \\ & & & a_{14} & a_{21} & \\ & 0 & & & a_{11} & a_{16} \end{bmatrix} = \begin{bmatrix} & & & & & \\ & & & & & \\ B''_k & & & & 0 & \\ & & & b_1 & & \\ 0 & & & & B'_k & \end{bmatrix}$$

The graphical representation with nodes corresponding to the original row order:



Node four is directly above  $x_4$  and nodes (4, 5, 6, 3) and arcs ( $a_5$ ,  $a_2$ ,  $a_3$ ) comprise the portion of the graph above  $x_4$ .

## CHAPTER III

### NETWORK PROGRAMMING BY ROW AND COLUMN GENERATION

#### 3.1 Introduction

In Chapter II a constructive method was used to exhibit the structure of a basis of the generalized flow problem. This constructive characterization can be used to develop efficient algorithms for the GFP. All of the algorithms presented in this paper are derived by applying the simplex method to the GFP or a special case of the GFP. The simplifications which result from the special nature of the GFP basis structure permit identification and utilization of only that information necessary to make decisions at each step of a simplex algorithm. When organized for use on a computer, these simplifications greatly reduce the amount of storage required over that of the standard simplex techniques or one of its efficient implementations. The extent of this improvement is indicated by the results of Murras [17].

In this chapter it is shown how the operations required in a simplex procedure (primal, dual, etc.) can be simplified by using the information structure of the GFP as depicted in graphical form. The main thesis is that the inverse of the current basis does not have to be known explicitly but may be effectively computed as needed. This leads to efficient procedures for generating a row or column of the current tableau as required. A method for recomputing the simplex multipliers is given in Section 3.5 which significantly reduces the effort required to update

the multipliers at each basis change. A method for identifying the current basis is derived by extending the triple labeling method of Johnson [59] for the ordinary flow problem. The way this scheme is used in carrying out the simplex operation is exhibited and procedures to change the labels to reflect a change of basis are given in Section 3.6.

### 3.2 Simplex Multiplier Calculation

The two decisions to be made in the primal simplex method are, in order, (i) choose a nonbasic variable to enter the basis and (ii) choose a basic variable to leave the basis. Usually, a nonbasic variable is chosen to enter from the set whose current cost ( $\bar{c}_j$ ) indicates a reduction (for minimization) in the objective function if the variable is brought into the basis. That is, if the nonbasic variable  $x_j$  is at its upper bound and  $\bar{c}_j > 0$ , reducing  $x_j$  will lower the objective value, and likewise, if  $x_j$  is 0, then if  $\bar{c}_j < 0$  increasing  $x_j$  will reduce the objective value. Consider the computation of the current costs for a generalized flow problem. Let the column associated with  $x_i$  have nonzero coefficients  $a_{ki}$  and  $a_{k'i}$  in equations  $p_i$  and  $q_i$ , respectively. Let the simplex multipliers associated with equations  $p_i$  and  $q_i$  be  $\pi_{p_i}$  and  $\pi_{q_i}$ . The optimality conditions are:

$$x_i = 0 \quad \text{implies} \quad \bar{c}_i = c_i - \pi_{p_i} a_{ki} - \pi_{q_i} a_{k'i} \geq 0 \quad (1)$$

$$x_i = M_i \quad \text{implies} \quad \bar{c}_i = c_i - \pi_{p_i} a_{ki} - \pi_{q_i} a_{k'i} \leq 0 \quad (2)$$

Since  $\bar{c}_i = 0$  for all basic variables, the relative costs are computed for nonbasic variables and if the conditions of Eqs. (1) and (2) do not hold,

a variable is chosen to enter the basis. The usual rule is to choose  $i^*$  such that:

$$\bar{c}_{i^*} = \min (\bar{c}_i') \quad (3)$$

where  $\bar{c}_i' = \bar{c}_i$  if  $x_i = 0$  and  $\bar{c}_i' = -\bar{c}_i$  if  $x_i = M_i$ . For network problems there is some reason to believe computational advantage can be gained by choosing some other candidate for entry which would not necessitate the calculation of all  $\bar{c}_i'$  as required when the minimum rule is used. The simplex multipliers ( $\pi$ ) are found by solving the equation:

$$\pi B = c_B \quad (4)$$

where  $c_B$  is the vector of cost coefficients of the basic variables corresponding to the columns of  $B$ . Since  $B$  is block diagonal, this separates into the solution of  $R$  sets of equation:

$$\pi_1 B_1 = c_{B_1}, \quad \pi_2 B_2 = c_{B_2}, \quad \dots, \quad \pi_R B_R = c_{B_R}. \quad (5)$$

Consider the solution for one of the blocks:

$$\pi_k B_k = c_{B_k}. \quad (6)$$

Assume that the  $n_k \times n_k$  matrix  $B_k$  contains a slack column and has been arranged in lower triangular form with the equations and variables numbered as indicated:



$$\pi_1 a_{11} = c_1, \quad \pi_1 = c_1/a_{11} \quad (9)$$

$$a_{12}\pi_1 + a_{22}\pi_2 = c_2, \quad \pi_2 = \frac{c_2 - \pi_1 a_{12}}{a_{22}}$$

$$a_{13}\pi_1 + a_{23}\pi_3 = c_3, \quad \pi_3 = \frac{c_3 - \pi_1 a_{13}}{a_{23}}$$

In general for nodes  $p_i$  and  $q_i$  associated with variable  $x_i$ :

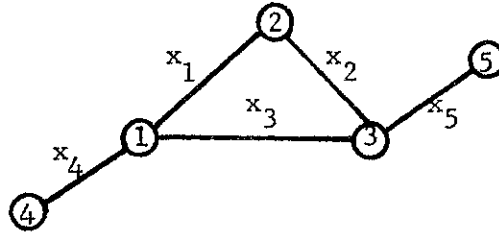
$$\pi_{p_i} = \frac{c_i - a_{2i}\pi_{q_i}}{a_{1i}} \quad \text{or} \quad \pi_{q_i} = \frac{c_i - a_{1i}\pi_{p_i}}{a_{2i}} \quad (10)$$

The equation used depends on whether node  $p_i$  or  $q_i$  is encountered first. To compute these values a device is needed to identify root nodes and to trace a tree outwardly from the root. In this manner the simplex multipliers are computed so that the current costs for each nonbasic arc can be found to determine optimality.

Because of the nature of the basis and basis exchanges all simplex multipliers are calculated initially and a subset is recalculated at each basis change. The method for doing this is presented later in this chapter.

If  $B_i$  contains a cycle the simplex multipliers of the cycle are determined and then each node in the cycle is treated as a root node and the simplex multipliers above it are determined as before. For example, suppose

$$B_k = \left[ \begin{array}{ccc|cc} a_{15} & & & & \\ & a_{24} & & & \\ \hline & & a_{23} & a_{12} & \\ a_{25} & & & a_{22} & a_{21} \\ & & & & \\ & a_{14} & a_{13} & & a_{11} \end{array} \right] \quad (11)$$



The set of equations to be solved is:

$$a_{11}\pi_1 + a_{21}\pi_2 = c_1 \quad (12)$$

$$a_{22}\pi_2 + a_{12}\pi_3 = c_2$$

$$a_{23}\pi_3 + a_{13}\pi_1 = c_3$$

$$a_{14}\pi_1 + a_{24}\pi_4 = c_4$$

$$a_{25}\pi_3 + a_{15}\pi_5 = c_5$$

Solving for  $\pi_1$ :

$$\pi_1 = \frac{\frac{c_3}{a_{13}} - \frac{a_{23}c_2}{a_{13}a_{12}} - \frac{a_{23}a_{22}}{a_{13}a_{12}a_{21}} c_1}{1 - \left(-\frac{a_{23}}{a_{13}}\right)\left(-\frac{a_{22}}{a_{12}}\right)\left(-\frac{a_{11}}{a_{21}}\right)} \quad (13)$$

In turn:

$$\pi_2 = \frac{c_1 - a_{11}\pi_1}{a_{21}}, \quad \pi_3 = \frac{c_2 - a_{22}\pi_2}{a_{12}}, \quad \pi_4 = \frac{c_4 - a_{14}\pi_1}{a_{24}},$$

$$\pi_5 = \frac{c_5 - a_{25}\pi_3}{a_{15}}$$

In general, to determine the dual variables corresponding to a cycle, assume that the nodes in the cycle are numbered  $1, 2, \dots, r$ . Thus the

numbering assigns a direction around the cycle. Assume that the corresponding coefficients on the basic variables around the cycle are  $(a_{ki}, a_{k'i})$  as encountered and the costs are  $c_i$ ,  $i = 1, \dots, r$ . Define the set of constants

$$p_i = \frac{a_{ki}}{a_{k'i-1}} \quad (14)$$

Then the formula for  $\pi_1$  is:

$$\pi_1 = \frac{1}{a_{k'r}} \frac{[c_r - p_r c_{r-1} - p_r p_{r-1} c_{r-2}, \dots, \prod_{i=2}^r p_i c_1]}{(1 - (-1)^r \prod_{i=1}^r p_i)} \quad (15)$$

The quantity  $(1 - (-1)^r \prod_{i=1}^r p_i) \neq 0$  because otherwise  $B_k$  could not be a part of the basis. This is shown below.

The cycle corresponds to the square submatrix  $B'_k$ , where:

$$B_k = \left[ \begin{array}{c|c} B_A & 0 \\ \hline B_u & B'_k \end{array} \right] \quad (16)$$

But  $\text{Det}(B_k) = \text{Det}(B_A) \text{Det}(B'_k)$ . Thus  $\text{Det}(B'_k) \neq 0$  since  $B_k$  is part of the basis. Then:

$$\text{Det}(B'_k) = \prod_{i=1}^r a_{ki} - (-1)^r \prod_{i=1}^r a_{k'i} \quad (17)$$

Suppose:

$$1 - (-1)^r \prod_{i=1}^r p_i = 0 \quad (18)$$

Then:

$$1 - (-1)^r \prod_{i=1}^r \frac{a_{ki}}{a_{k'i}} = 0 \quad (19)$$

$$\prod_{i=1}^r a_{k'i} - (-1)^r \prod_{i=1}^r a_{ki} = 0 \quad (20)$$

But

$$(-1)^{-r} = (-1)^r \quad \text{for } r \text{ an integer} \quad (21)$$

Thus

$$\prod_{i=1}^r a_{ki} - (-1)^r \prod_{i=1}^r a_{k'i} = 0 = \text{Det}(B'_k) \quad (22)$$

which contradicts  $\text{Det}(B'_k) \neq 0$ .

To calculate the simplex multipliers for a cycle requires tracing around the cycle from a node in the cycle. Then by tracing out from each node in the cycle, the simplex multipliers corresponding to the remaining nodes can be determined. Since all components of the basis are either rooted or pseudorooted, the complete set of simplex multipliers can be calculated by the methods shown. The current costs  $\bar{c}_i$  can then be calculated and an entering variable chosen by an appropriate rule.

### 3.3 Column Generation

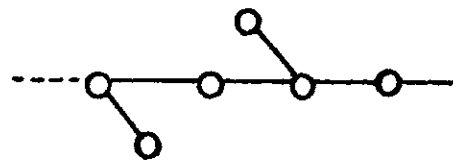
Once a nonbasic variable has been selected to enter the basis, its representation in terms of the current basic variables must be found so that the variable to leave the basis may be determined. For the primal simplex method, the leaving or blocking variable is the one which allows the maximum change in the entering variable consistent with maintaining primal feasibility. The generation of the required column, the determina-

tion of the blocking variable, and the change in the values of the basic variables can be done efficiently using a procedure derived from the constructive proof of Lemmas 2.1 - 2.3 in Chapter II.

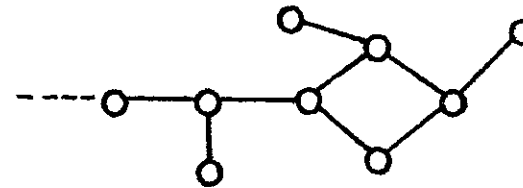
First consider the various relationships an entering nonbasic variable can have to the basic variables as shown in Figure 6. First an entering variable can have a nonzero coefficient in one (Figure 6(a,b) or two equations (Figure 6(c,d,e,f,g,h)). If it has a nonzero entry in one equation, the tree containing the corresponding node can be rooted or pseudorooted (see Theorem 2.1 of Chapter II).

If the entering variable has nonzero coefficients in two equations, the corresponding nodes can be in the same tree (Figure 6(g,h)) or different trees (Figure 6(c,d,e,f)). If they are in the same tree, it can be a rooted tree (Figure 6(g)) or pseudorooted tree (Figure 6(g,h)). If the two ends of the arc corresponding to the entering variables are in different trees, either both trees are rooted (Figure 6(c)), both trees are pseudorooted (Figure 6(e)), or one tree is rooted and the other is pseudorooted (Figure 6(e,f)). In the case that one is rooted and the other pseudorooted, without loss of generality, denote the tree containing node  $p_i$  as the left tree and  $q_i$  the right tree. Then either the left tree is rooted and the right tree pseudorooted or vice versa. This exhausts the possible combinations of an entering variable and the associated basis component.

For all of the situations depicted in Figure 6, the same operation must be accomplished; that is, the nonbasic column corresponding to the entering variable must be constructed and the blocking variable determined. The breakdown into the various cases is useful since the column construc-

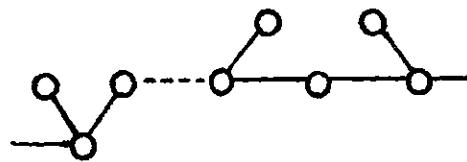


(a) Slack Arc Incident on a Rooted Tree

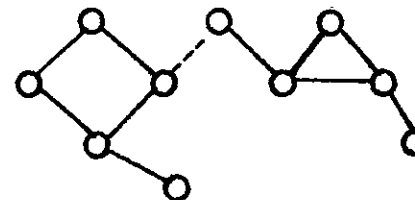


(b) Slack Arc Incident on a Pseudorooted Tree

--- Nonbasic Arc  
 — Basic Arc

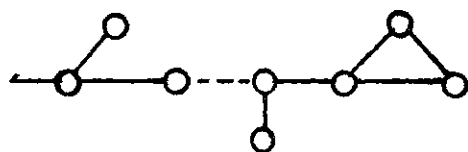


(c) Regular Arc Between Two Rooted Trees

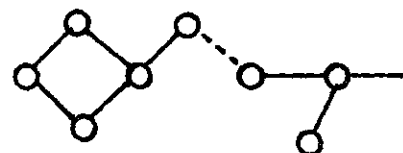


(d) Regular Arc Between Two Pseudorooted Trees

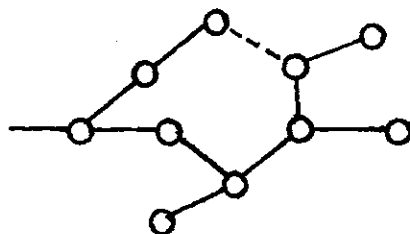
Figure 6. Entering Arc Configurations



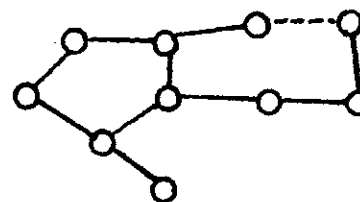
(e) Regular Arc with Left Tree Rooted and Right Tree Pseudorooted



(f) Regular Arc with Left Tree Pseudorooted and Right Tree Rooted



(g) Regular Arc with Both Ends in the Same Rooted Tree

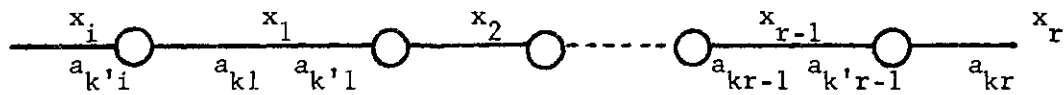


(h) Regular Arc with Both Ends in the Same Pseudorooted Tree

Figure 6. (Concluded)

tion procedure will be slightly different for each case. The column generation procedures for cases (a) through (h) in Figure 6 are presented next.

First consider case (a). Let  $x_i$  be the entering nonbasic variable with its nonzero coefficient in equation  $p_i$ . The basis  $B$  can be partitioned into  $B_p$  and  $B'_p$  where  $B_p$  is associated with the columns in the unique path from node  $p_i$  to the root (including the slack column for the root) and  $B'_p$  is associated with the remaining columns (see corollary to Theorem 2.1). The path is:



The matrix  $B_p$  has the form:

$$B_p = \begin{bmatrix} a_{k1} & & & & & \\ a_{k'1} & a_{k2} & & & & \\ & a_{k'2} & & & & \\ & & & & & \\ & & & & a_{kr-1} & \\ & & & & a_{k'r-1} & a_{kr} \end{bmatrix} \quad (23)$$

Suppose the column for  $x_i$  is:

$$b_i = \begin{bmatrix} a_{k'i} \\ 0 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad (24)$$

Let the representation of  $b_i$  in terms of the current basis be the vector

$d$  with components  $d_i$ . Then:

$$\left[ \begin{array}{c|c} B_p & B'_p \\ \hline 0 & \end{array} \right] \left[ \begin{array}{c} d_p \\ \hline d'_p \end{array} \right] = \left[ \begin{array}{c} a_{k'i} \\ 0 \\ \vdots \\ 0 \end{array} \right] \quad (25)$$

The vector  $d$  has been partitioned into  $(d_p/d'_p)^t$ . From the proof of Lemma 2.1, there exists a solution  $d_p^*$  to  $B_p d_p = (a_{k'i}, 0, 0, \dots, 0)^t$  so that  $(d_p^*, 0)$  solves the above system and is the unique representation of the column of  $x_i$  in terms of the current basis vectors. Let  $(x_1, \dots, x_r)$  be the basic variables in the path from node  $p_i$  to the root, with  $x_r$  as the slack variable at the root. From Lemma 2.1 the components of  $d_p^*$  are:

$$d_1^* = \frac{a_{k'i}}{a_{k1}}, \dots, d_r^* = \frac{-a_{k'r-1}}{a_{kr}} d_{r-1} \quad r = 2, \dots \quad (26)$$

These are precisely the weights defined in Eq. (1) of Chapter II.

Now consider the inverse of the basis  $B$ . Let  $\bar{d}$  denote the  $j^{\text{th}}$  column corresponding to equation  $p_j$ . If  $e_{p_j}$  is a column vector with a one in row  $p_j$ , then:

$$\left[ \begin{array}{c|c} B_p & B'_p \\ \hline 0 & \end{array} \right] \left[ \begin{array}{c} \bar{d}_p \\ \hline \bar{d}'_p \end{array} \right] = \left[ \begin{array}{c} 0 \\ 1 \\ 0 \end{array} \right] \leftarrow \text{row } p_j \quad (27)$$

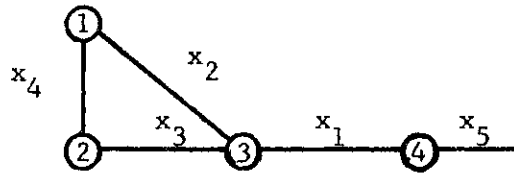
Comparing this with Eq. (25), we readily have the solution:

$$\bar{d} = \frac{d}{a_{k'i}} \quad (28)$$

with nonzero entries  $\bar{d}_i$  only for  $i = 1, \dots, r$ .

The graphical representation of the basis identifies the nonzero elements in the basis inverse and provides a means for calculating them.

If the basic path from node  $p_j$  ends in a pseudoroot as in case 6(b), column construction is similar with additional consideration for the basic variables in the pseudoroot. The following example illustrates the differences.



$$B_p = \begin{bmatrix} a_{k4} & & a_{k'2} \\ a_{k'4} & a_{k3} & \\ & a_{k'3} & a_{k2} & a_{k1} \\ & & & a_{k'1} \end{bmatrix} \quad b_s = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a_{k5} \end{bmatrix} \quad (30)$$

The equation to be solved is:  $b_s = B_p d$ . For the path to the pseudo-

root:  $d_1 = \frac{a_{k5}}{a_{k'1}}$ . For the cycle define the constant  $q$ :

$$q = \left( - \frac{a_{k'4}}{a_{k4}} \right) \left( - \frac{a_{k'3}}{a_{k3}} \right) \left( \frac{a_{k'2}}{a_{k2}} \right) \quad (31)$$

The remaining coefficients are:

$$d_2 = \left( \frac{a_{k1}}{a_{k'2}} - a_{k1} q \right) d_1$$

$$d_3 = \left( - \frac{a_{k'2}}{a_{k4}} \right) d_2$$

$$d_4 = \left( - \frac{a_{k'4}}{a_{k3}} \right) d_3$$

The current representation of the column for  $x_5$  contains the nonzero entries  $d_1, \dots, d_4$  corresponding to the basic variable  $x_1$  in the path to the pseudoroot and the basic variables  $x_2, x_3$ , and  $x_4$  in the pseudoroot. For this example, the nonzero entries in the fourth column of  $B^{-1}$  are:

$$d'_1 = \frac{d_1}{a_{k5}} = \frac{1}{a_{k'1}} \quad (32)$$

$$d'_2 = \frac{d_2}{a_{k5}} = \frac{a_{k1}}{a_{k'1}} \left( \frac{1}{a_{k'2}} - q \right)$$

$$d'_3 = \frac{d_3}{a_{k5}} = - \frac{a_{k'2} a_{k1}}{a_{k4} a_{k2}} \left( \frac{1}{a_{k'2}} - q \right)$$

$$d'_4 = \frac{d_4}{a_{k6}} = - \frac{a_{k'2} a_{k1} a_{k'4}}{a_{k3} a_{k4} a_{k2}} \left( \frac{1}{a_{k'2}} - q \right)$$

Suppose the column corresponding to the entering variable  $x_i$  has nonzero entries in two rows as in cases (c) through (h).

$$b_i = \begin{bmatrix} 0 \\ \vdots \\ a_{ki} \\ 0 \\ \vdots \\ a_{k'i} \\ 0 \end{bmatrix} \quad (33)$$

Then, as in the proof of Lemma 2.2,  $b_i$  can be expressed as the sum of two slack columns:

$$b_i = b_i' + b_i'' = \begin{bmatrix} 0 \\ a_{ki} \\ 0 \\ \vdots \\ \vdots \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ a_{k'i} \\ 0 \\ \vdots \end{bmatrix} \quad (34)$$

The method for determining the current representation of  $\underline{b}_i'$  and  $\underline{b}_i''$  has already been shown. Suppose nodes  $p_i$  and  $q_i$  corresponding to the nonzero entries  $a_{ki}$  and  $a_{k'i}$  are in different trees as in cases (c,d,e, or f). Then the representation is precisely as before with the two columns of the inverse matrices involved, being computed for two different blocks  $B_j$  and  $B_k$ .

If both nodes  $p_i$  and  $q_i$  are in the same tree, then the order of determining the representation is slightly different. For a rooted tree (Figure 6(g)), from both node  $p_i$  and  $q_i$ , there is a unique simple path of basic arcs to the root. Since the paths both end at the root node, they must join at some node,  $s$ , which must be the root node or some node above it. Denote the path starting at  $p_i$  the left path ( $P_L$ ) and at  $q_i$  the right path ( $P_R$ ).

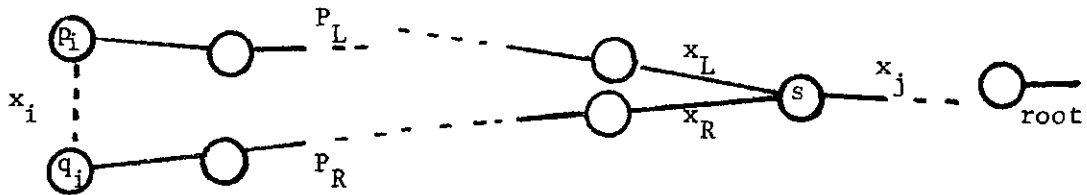
Let the variable corresponding to the arc in  $P_L$  incident on node  $s$  be  $x_L$  with associated column  $b_L$  and the variable  $x_R$  with column  $b_R$  correspond to the arc in  $P_R$  incident on  $s$ . As in the corollary to Theorem 2.1, partition  $B_k$  into  $B_L$ ,  $B_R$ , and  $B_k'$  where  $B_L$  corresponds to the path above  $x_L$  and  $B_R$  to the path above  $x_R$ . The root is contained in  $B_k'$ . The partitioned matrix is:

$$\begin{bmatrix}
 P_L & x_L & P_R & x_R \\
 B_L & a_{k'L} & B_R & a_{k'R} \\
 a_{kL} & 0 & a_{kR} & 0 \\
 0 & 0 & 0 & 0
 \end{bmatrix}
 \quad (35)$$

The column for  $x_i$  is:

$$b_i = \begin{bmatrix} a_{ki} \\ 0 \\ \vdots \\ a_{k'i} \\ 0 \\ \vdots \end{bmatrix} \quad \begin{matrix} \leftarrow \text{a row in } B_L \\ \leftarrow \text{a row in } B_R \end{matrix} \quad (36)$$

The coefficient  $a_{ki}$  is in a row corresponding to  $B_L$  and  $a_{k'i}$  is in a row corresponding to  $B_R$ . The graphical representation is:



For the basic variables in  $P_L$  and in  $P_R$ , the weights (nonzero entries) are calculated as before. If the cycle formed by the two paths to  $s$  and the entering arc correspond to a set of linearly dependent variables, the representation is complete and the variables in the path to the root and the slack variable at the root do not have to be considered. One would like to detect this linear dependence before calculating the weights and entries for the variables below node  $s$ . The following lemma gives a means

for doing this.

Let the weight for the variable  $x_L$  be  $d_L$  and the weight for  $x_R$  be  $d_R$ .

Lemma 3.1

If both ends of an arc corresponding to a nonbasic variable  $x_i$  are in the same rooted tree, then the set of columns corresponding to the cycle formed by the entering arc and the portions of the paths to the root from both ends of the nonbasic arc until they join is linearly dependent if  $d_L a_{k'L} + d_R a_{k'R} = 0$ .

Proof:

By the constructive proof in Lemma 2.2, the weight for the variable  $x_j$  corresponding to the arc below  $s$  in the path to the root is:

$$d_j = \frac{-d_L a_{k'L}}{a_{kj}} + \frac{-d_R a_{k'R}}{a_{kj}} = \frac{-1}{a_{kj}} (d_L a_{k'L} + d_R a_{k'R}) \quad (37)$$

But the quantity in parenthesis on the right hand side of Eq. (37) is precisely the quantity in the lemma. Thus, if  $d_L a_{k'L} + d_R a_{k'R} = 0$ , then  $d_j = 0$ . But if  $d_j$  is equal to zero, then all of the weights in the remainder of the path including the root are zero. The column for the nonbasic variable  $x_i$  can thus be expressed as a linear combination of the columns corresponding to  $P_L$  and  $P_R$ , hence the set of columns is linearly dependent.

The lemma suggests the following method for determining the current representation for the column for  $x_i$ .

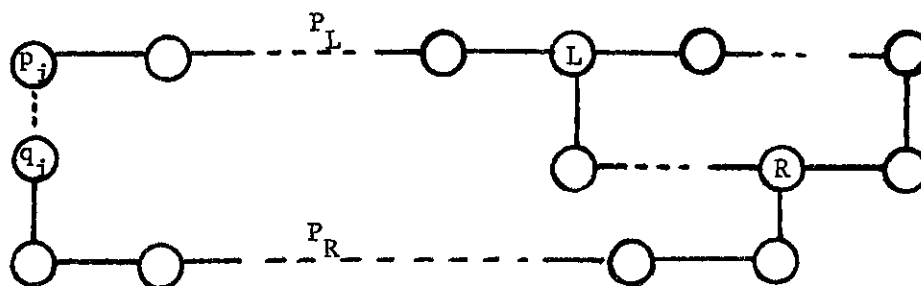
1. Compute the weights for the variables in  $P_L$  to  $s$ .

2. Compute the weights for the variables in  $P_R$  to  $s$ .
3. Check the linear dependence condition in the lemma, i.e.,  
 $d_L a_{k'L} + d_R a_{k'R} = 0$ ? If so, then the representation is complete.
4. If not, compute the weight for  $x_j$ :

$$d_j = \frac{-1}{a_{kj}} (d_L a_{k'L} + d_R a_{k'R})$$

5. Continue to the root computing the weights iteratively as before.

The last case to be considered is case (h) when both ends of a nonbasic arc are in the same pseudorooted tree. If the left path and right path join at a node above the pseudoroot or at the same node in the pseudoroot, the difference between case (g) and case (h) is the same as the difference between cases (a) and (b). If the left and right paths meet the pseudoroot at different nodes, a slightly different method is used. Consider the example shown below.



As before, the entries are computed down the left path to  $L$  and down the right path to  $R$ . An arbitrary direction is chosen around the cycle and the variables numbered as encountered, starting with the first one after node  $L$ , as  $x'_1, \dots, x'_u, x'_{u+1}, \dots, x'_r$  with nonzero entries  $(a'_{k1}, a'_{k,1}), \dots, (a'_{kr}, a'_{k,r})$ .  $x'_u$  is the basic variable before node  $R$  and  $x'_{u+1}$  is the

variable after node R in the cycle.

Define:

$$g_1 = \prod_{j=1}^{u-1} - \frac{a_{kj+1}}{a_{k',j}} \quad (38)$$

$$g_2 = \prod_{j=u+1}^{v-1} - \frac{a_{kj+1}}{a_{k',j}} \quad (39)$$

Let  $d'_i$  correspond to the weight for  $x'_i$  in the cycle. Then:

$$d'_1 = - \frac{\left( a'_{k'R} d_R - \frac{a'_{k'L} d_L a'_{ku+1}}{a'_{k',v} g_2} \right)}{\left( \frac{a'_{k'u}}{g_1} - \frac{a'_{k'l} a'_{ku+1}}{a'_{k',v} g_2} \right)} \quad (40)$$

$$d'_j = - d'_{j-1} \left( \frac{a'_{kj-1}}{a'_{k',j}} \right) \quad j = 2, \dots, u \quad (41)$$

$$d'_{u+1} = \frac{\left( a'_{k'L} d_L - \frac{a'_{k'R} d_R a_{k1}}{a'_{k',u} g_1} \right)}{\left( \frac{a'_{k',v}}{g_2} - \frac{a_{ku+1} a_{k1}}{a'_{k',u} g_2} \right)} \quad (42)$$

$$d'_j = - d'_{j-1} \frac{a'_{kj-1}}{a'_{k,j}} \quad j = u+2, \dots, v \quad (43)$$

This method of obtaining the entries for the basic variables in the cycle is more complicated than considering the entering variable as the sum of two slack variables, getting the representations separately, and adding the columns together. However, this method requires tracing around the cycle twice, while the other method would require tracing

around the cycle four times. The denominators in Eqs. (40) and (41) can be shown to be nonzero in the case that the variables corresponding to the cycles formed by  $x_1$ ,  $P_L$ ,  $P_R$ , and the path from node R to node L are linearly independent. Calculating these quantities provides a test of where the blocking variable might be.

We have discussed above, for all the different cases, the methods for generating the current representation for a nonbasic variable in terms of the current basic variables.

### 3.4 Row Generation

The discussion in the previous two sections related to the operations involved in a primal simplex network algorithm for the generalized flow problem. For a dual algorithm another similar concept is needed. In general, a dual simplex method chooses as a departing variable a basic variable violating the primal constraints. Once this variable is chosen, its corresponding row of nonbasic entries is needed, and the entering variable is chosen as the one allowing the maximum change consistent with maintaining dual feasibility. A dual algorithm requires the generation of the row associated with a specified basic variable. The method for doing this is essentially the same as that for determining the simplex multipliers and updated costs for nonbasic arcs in the primal method. This close connection is evident since in the primal case the current costs are in fact the row associated with the objective function.

Suppose the current row associated with a basic variable  $x_j$  whose associated column has its nonzero entries in block  $k$  is to be generated. For basic variable  $x_j$  this row ( $\bar{a}_j$ ) is obtained by multiplying the corres-

ponding row of the basis inverse ( $\underline{b}_j'$ ) and the original columns for the nonbasic columns ( $A_N$ ).

$$\underline{\bar{a}}_j = \underline{b}_j' A_N \quad (44)$$

The calculation of the required row of the inverse can be accomplished in the following manner. First, note that the inverse of a block diagonal matrix is a block diagonal matrix of the inverses of the original blocks. Thus the row will contain zero entries for all columns not associated with  $B_k$ . Next partition  $B_k$  into  $B_k'$  and  $B_k''$  using column  $b_j$  for the basis variable  $x_j$  as the partitioning column (see the partitioning corollary after Theorem 2.1).

$$B_k = \left[ \begin{array}{c|c|c} & 0 & 0 \\ & \vdots & \vdots \\ B_k' & a_{k',j} & \vdots \\ \hline 0 & a_{kj} & \\ \vdots & 0 & B_k'' \\ \vdots & \vdots & \end{array} \right] \quad (45)$$

The block  $B_k''$  contains the root or pseudoroot. The required row of  $B_k^{-1}$  is the solution to:

$$\underline{b}_j^* B_k = e^j = (0, \dots, \underset{\substack{\uparrow \\ j^{\text{th}} \text{ position}}}{1}, 0, \dots, 0) \quad (46)$$

For convenience, the subscript  $j$  will be dropped when referring to  $\underline{b}_j^*$  and  $b_1^*, \dots, b_r^*$  will be the components of  $\underline{b}_j^*$ . Let  $b_1^*, \dots, b_v^*$  be the elements of  $\underline{b}_j^*$  which multiply  $B_k'$  and  $b_{v+1}^*, \dots, b_r^*$  multiply  $B_k''$ . Also let  $b_v^*$  multiply  $a_{k',j}$  and  $b_{v+1}^*$  multiply  $a_{kj}$ . Then Eq. (46) can be rewritten:

$$[b_1^*, \dots, b_v^*] \begin{bmatrix} & 0 \\ B'_k & \\ & a_{k'j} \end{bmatrix} = [0 \dots 1] \quad (47)$$

$$[b_{v+1}^*, \dots, b_r^*] \begin{bmatrix} a_{kj} & \\ 0 & B''_k \end{bmatrix} = [0 \dots 0] \quad (48)$$

The solution to Eq. (48) is obviously:

$$b_i^* = 0 \quad i = v+1, \dots, r \quad (49)$$

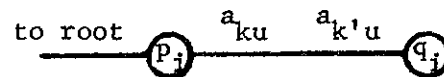
Equation (47) has the same form as Eq. (32) in Chapter I for determining the simplex multipliers, i.e.,

$$uB = c_B \quad (50)$$

If  $b_1^*, \dots, b_r^*$  are identified with elements of  $u$  and if the vector  $(0, \dots, 1)$  is identified with the vector  $c_B$ , Eqs. (50) and (47) are identical. But in Section 3.2 an iterative method was presented for solving the system of equations (50). Thus Eq. (47) can be solved in the same iterative manner, tracing out from the node above the arc for  $x_j$ . The entry for this node is calculated by noting that:

$$a_{k'j} b_v^* = 1 \quad \text{thus} \quad b_v^* = \frac{1}{a_{k'j}} \quad (51)$$

Let two nodes  $p_i$  and  $q_i$  correspond to any two rows of  $[B'_k \ b_j]$  and let  $p_i$  and  $q_i$  be connected by the arc for the basic variable  $x_u$ . Let node  $p_i$  be nearer the root.



Let  $b_s^*$  be associated with node  $p_i$  and  $b_t^*$  be associated with node  $q_i$  and assume  $b_s^*$  has been calculated. Then  $b_t^*$  is calculated using the relationship:

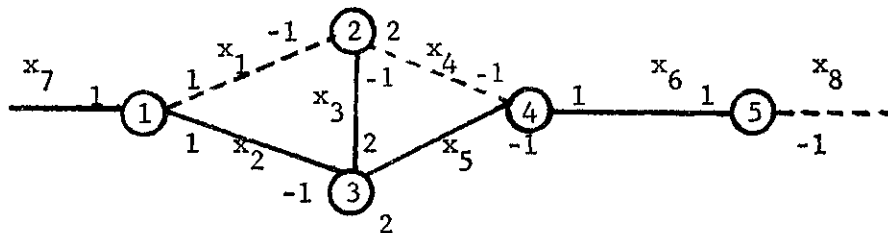
$$a_{ku} b_s^* + a_{k'u} b_t^* = 0, \quad (52)$$

i.e.,

$$b_t^* = \frac{-a_{ku}}{a_{k'u}} b_s^*$$

This is the relationship used to iteratively calculate the nonzero elements  $b_1^*, \dots, b_{v-1}^*$  in  $\underline{b}_j^*$ . The above development has shown that the only nonzero elements of the row of the inverse corresponding to  $x_j$  are the ones associated with the rows of  $B'_k$  and hence the nodes above the arc for  $x_j$  in the associated graph. Moreover, each of these elements will in fact be nonzero. Once these nonzero elements have been calculated, the current row for  $x_j$  can be obtained using Eq. (44).

An example of the construction of a row of the inverse and the current row for a basic variable is now given using the sample problem from Chapter II.



The arcs corresponding to basic variables are in heavy lines and for non-

basic variables in broken lines. To generate the row associated with  $x_5$ , the fourth row of  $B_p^{-1}$  must be calculated since the column for  $x_5$  is the fourth column.

$$B_k = \begin{array}{c} \begin{array}{ccccc} x_7 & x_2 & x_3 & x_5 & x_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 2 & 2 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad (53)$$

To generate the row for  $x_5$ , partition  $B_k$  about the column for  $x_5$ .

$$B_k = \left[ \begin{array}{c|c|c} B'_k & a_{k,5} & 0 \\ \hline 0 & a_{k5} & B''_k \end{array} \right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 2 & 2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (54)$$

The row of the inverse to be computed is:

$$b^* = (b_1^*, \dots, b_5^*) \quad (55)$$

$$\text{But by Eq. (49):} \quad b_5^* = b_4^* = b_3^* = 0$$

$$\text{And by Eq. (51):} \quad b_2^* = \frac{1}{-1} = -1$$

$$\text{And by Eq. (52):} \quad b_1^* = \frac{-(1)b_2^*}{(1)} = 1$$

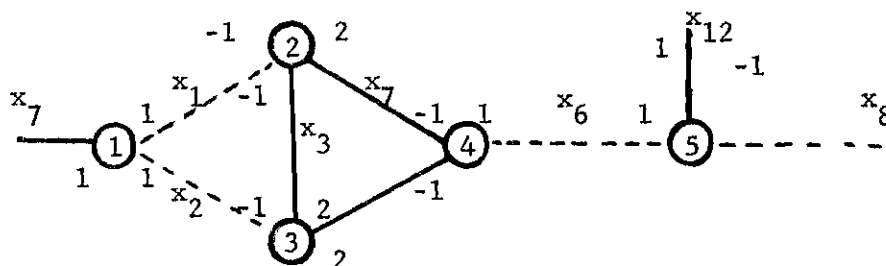
The fourth row of  $B_p^{-1}$  is:

$$\underline{b}_4' = [0 \ 0 \ 0 \ -1 \ 1] \quad (56)$$

To determine the entries for the nonbasic variables  $x_1$ ,  $x_4$ , and  $x_8$ , the appropriate original columns are multiplied by the constructed row.

$$\bar{a}_i = \bar{b}_i' A_N = [0 \ 0 \ 0 \ -1 \ 1] \begin{bmatrix} 1 & 0 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = [0 \ 1 \ -1] \quad (57)$$

For the case of a pseudorooted component consider:



To generate the row associated with  $x_5$ , the third row of  $B_k^{-1}$  is calculated:

$$B_k = \begin{matrix} & x_3 & x_4 & x_5 \\ \begin{bmatrix} -1 & 2 & 0 \\ 2 & 0 & 2 \\ 0 & -1 & -1 \end{bmatrix} & & & \end{matrix} \quad (58)$$

$$\begin{aligned} -1b_1' + 2b_2' &= 0 \\ 2b_1' - b_3' &= 0 \\ 2b_2' - b_3' &= 1 \end{aligned} \quad (59)$$

Solving:

$$\begin{aligned} b_1' &= -1 \\ b_2' &= -\frac{1}{2} \\ b_3' &= -2 \end{aligned} \quad (60)$$

The entries in the generated row corresponding to  $x_1$ ,  $x_2$ , and  $x_6$  are calculated using Eq. (44).

$$\bar{a}_1 = (-1)(-1) = 1 \quad (61)$$

$$\bar{a}_2 = -1(-\frac{1}{2}) = \frac{1}{2}$$

$$\bar{a}_6 = 1(2) = -2$$

Using these methods any row of the inverse matrix corresponding to a specific basic variable and the corresponding entries for the nonbasic variables in the updated representation of that row may be generated directly from the tree structure of the basis and the original coefficient matrix.

An interpretation of the nonzero entries in an updated row may also be given in terms of the current basis graph. Consider the nodes in the tree containing the basic arc (variable) whose row is being generated. Then only nonbasic arcs incident on these nodes can have nonzero entries. Further, only nonbasic arcs which are incident on the tree above the generating arc can have nonzero entries. This is evident from the construction of the corresponding row of the basis inverse already given. Nonbasic arcs which have both ends in the tree above the generating arc will have a nonzero entry only if the simple cycle of basic arcs and the particular nonbasic arc form a set of linearly independent vectors.

These characteristics may be stated as a theorem.

#### Theorem 3.1

The row in the current tableau for basic variable  $x_i$  will contain

nonzero entries for only the nonbasic variables corresponding to:

1) Arcs which are incident on exactly one node in the basic tree above the arc for  $x_i$ .

2) Arcs which are incident on two nodes in the basis tree above  $x_i$  such that the cycle formed by the arc for the nonbasic variable and arcs for basic variables above  $x_i$  constitute to a set of linearly independent vectors.

Proof:

If an arc corresponding to the nonbasic variable  $x_j$  is not incident on the tree above the variable  $x_i$ , then  $x_j$  will have a zero entry in the current row for  $x_i$ . This is seen since the entry for  $x_j$  in the row for  $x_i$  is calculated:

$$\bar{a}_{ij} = b'_i b_j \quad (62)$$

where  $b'_i$  is the complete row of the inverse for  $x_i$  and  $b_j$  is the original column for  $x_j$ . But it has just been shown that all of the elements of  $b'_i$  not corresponding to nodes above  $x_i$  are zero. Thus, since the nonzero element(s) of  $b_j$  occur in rows where the elements of  $b'_i$  are zero  $\bar{a}_{ij}$  must equal zero.

If the arc corresponding to  $x_j$  has both ends in the tree above  $x_i$  then the paths from both ends of  $x_j$  to the root or pseudoroot must include  $x_i$ . Hence  $x_j$  and some set of basic variables associated with arcs above  $x_i$  must form a cycle. Suppose that the columns corresponding to the cycle are linearly dependent, but that  $x_j$  has a nonzero entry ( $\bar{a}_{ij}$ ) in the current row for the basic variable  $x_i$ . Then, disregarding primal feasibility, a simplex pivot could be made on  $\bar{a}_{ij}$  making  $x_j$  basic and  $x_i$

nonbasic. In terms of the graph this would mean that the tree which contained  $x_i$  would break into two components, one tree consisting of the nodes and arcs not above  $x_i$ , which would retain the root or pseudoroot, and another tree consisting of the nodes and arcs above  $x_i$ . The latter tree would have a pseudoroot formed by the cycle containing  $x_i$  described previously. But it was assumed that the columns of the cycle were linearly dependent and hence the definition of a basis is violated. This contradiction proves the last part of the theorem.

This interpretation has a special significance for ordinary flow problems where in each column for a regular variable one nonzero entry is a plus one and one is a minus one. The single nonzero entry for the columns of the slack or artificial variables is a plus or minus one.

It is well known that a basis for these ordinary flow problems can contain no pseudoroots (cycles) (Dantzig [15], Johnson [58]). This is easily seen by considering the  $p \times p$  matrix of such a cycle:

$$B_c = \begin{bmatrix} a_{k1} & & & & a_{k'p} \\ a_{k'1} & a_{k2} & & & \\ & a_{k'2} & & & \\ & & \ddots & & \\ & & & a_{k'p-1} & a_{kp} \end{bmatrix} \quad (63)$$

The determinant of such a matrix is:

$$\text{Det}(B_c) = \prod_{i=1}^p a_{ki} - (-1)^p \prod_{i=1}^p a_{k'i} \quad (64)$$

The coefficients for each variable  $x_i$  are given by:

$$a_{ki} = \pm 1 \quad \text{and} \quad a_{k'i} = -a_{ki}$$

Since  $B_c$  is a cycle, then there are precisely  $p$  plus one and  $p$  minus one entries in  $B_c$ . Suppose  $r$  of the  $a_{ki}$  are equal to plus one. Then  $(p-r)$  of the  $a_{k'i}$  are also equal to plus one. Similarly,  $(p-r)$  of the  $a_{ki}$  and  $r$  of the  $a_{k'i}$  are equal to minus one. The notation  $a_{ki}$  and  $a_{k'i}$  implies a direction around the cycle. In this direction the above discussion implies we will have  $r$  forward arcs and  $(p-r)$  reverse arcs. The calculation of the determinant of  $B_c$  is:

$$\prod_{i=1}^p a_{ki} = (-1)^{p-r} (1)^r = (-1)^{p-r} \quad (65)$$

$$\prod_{i=1}^p a_{k'i} = (-1)^r (1)^{p-r} = (-1)^r \quad (66)$$

$$\text{Det}(B_c) = (-1)^{p-r} - (-1)^p (-1)^r \quad (67)$$

But

$$(-1)^r = (-1)^{-r} \quad (68)$$

Thus

$$\text{Det}(B_c) = (-1)^{p-r} - (-1)^p (-1)^{-r} = 0 \quad (69)$$

Since  $r$  was chosen arbitrarily, no cycle may be in the basis for the ordinary flow problem. This leads to a corollary to Theorem 3.1.

Corollary: For the ordinary flow problem, the only nonzero entries in the current row for basic variable  $x_i$  are for nonbasic variables  $x_j$  whose corresponding arcs are incident on one node of the tree above the arc for variable  $x_i$ .

Proof:

If the arc for nonbasic variable  $x_j$  is incident on two nodes above  $x_i$ , then it is contained in a cycle with a set of basic arcs above  $x_i$  and the cycle corresponds to a set of linearly dependent columns. Hence, by Theorem 3.1 the corollary is true.

It can also be noted that, for the matching problem, where  $a_{ki} = a_{k'i} = 1$  for all  $i$ , no even cycles can be in the basis. If they were, then the determinant of the submatrix of the cycle would be given by Eq. (64) as:

$$\text{Det}(B_c) = \prod_{i=1}^p a_{ki} - (-1)^p \prod_{i=1}^p a_{k'i} \quad (70)$$

but

$$a_{ki} = a_{k'i} = 1 \quad \text{for all } i \text{ and } p \text{ is even.}$$

$$\text{Det}(B_c) = (1) - (-1)^p 1 = 1 - 1 = 0 \quad (71)$$

which is not possible. This result is well known and has been proved by Johnson [58] and others.

### 3.5 Basis Change and Updating Simplex Multipliers

Row generation has a close relationship to the updating of simplex multipliers and nonbasic evaluators after a basis change. Consider the primal simplex operations performed in updating the objective row in a tableau. The pivot element is in the column of the entering nonbasic variable  $x_j$  and in the row of the departing basic variable  $x_i$ . To update the objective function row the ratio  $\bar{c}_j / \bar{a}_{ij}$  is multiplied by the row cor-

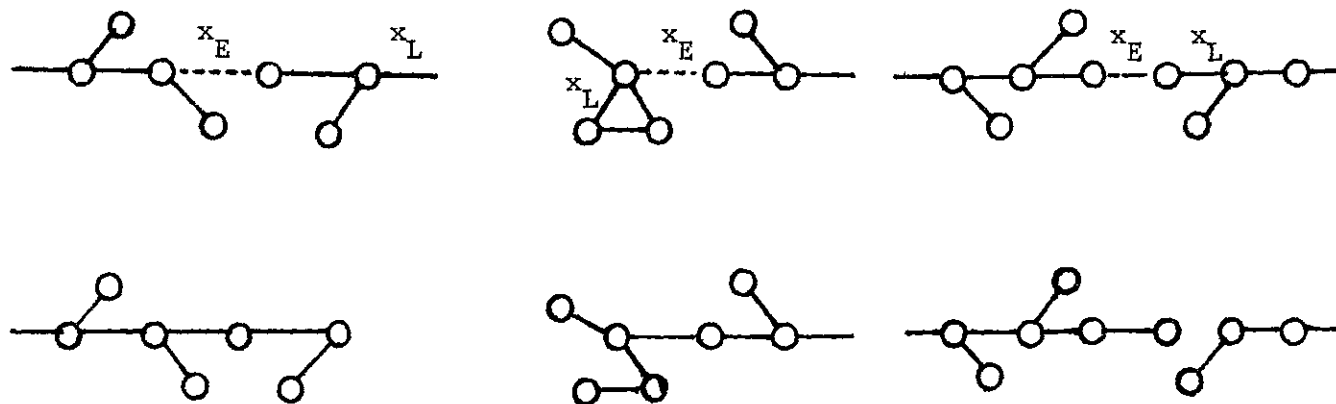
responding to  $x_i$  and subtracted from the current updated cost row. Hence only the simplex multipliers and current costs with nonzero entries in the row corresponding to the basic variable are changed. This property is the basis for the theorem of Glover, Klingman, and Kearny [34] for updating the objective function row for the ordinary transportation problem.

By observing the structure of the basis before and after a basis change is made, the simplex multipliers which must be recalculated can be specified exactly. The nature of the change in the structure of the basis is detailed in the two lemmas and a corollary which follow. Several examples are given in Figure 7.

First denote two distinct components of the basis as  $B_1$  and  $B_2$ . Let  $x_i$  be the entering nonbasic variable with associated column  $b_i$  and  $x_j$  be the leaving basic variable with associated column  $b_j$  which is a column of  $B_2$ . By the second corollary to Theorem 2.1, let  $B_2$  be partitioned into two blocks by  $b_j$  so that  $B_2^I$  contains the root or pseudoroot of  $B_2$  and  $B_2^{II}$  is above  $b_j$ . If  $b_j$  is a slack column or in a cycle then  $B_2^{II} = B_2$  and  $B_2^I$  is empty by definition.

Lemma 3.2

Let  $b_i$  replace  $b_j$  in the basis. If  $b_i$  has its nonzero entries in rows corresponding to the two distinct blocks  $B_1$  and  $B_2$ , then after the basis change  $B_2^I$  is a distinct block and  $B_1$  together with  $b_i$  and  $B_2^{II}$  is a distinct block. A trace from a nonzero element of  $B_1$  to a nonzero element of  $B_2^{II}$  in a series of alternating row and column move on nonzero element must contain the nonzero elements of  $b_i$ .



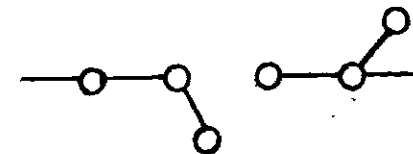
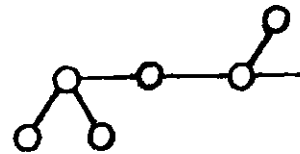
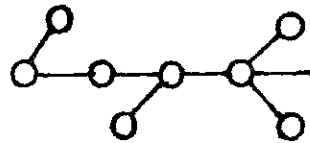
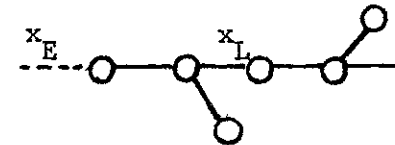
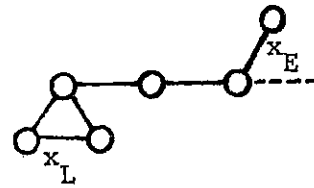
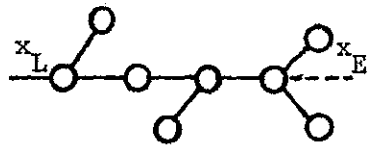
(a) Arc enters,  
Root leaves.

(b) Arc enters,  
Pseudoroot leaves.

(c) Arc enters,  
Arc leaves.

$\text{---} \overset{x_E}{\text{---}}$  entering arc  
 $\text{---} \overset{x_L}{\text{---}}$  leaving arc

Figure 7. Basis Exchange Configurations



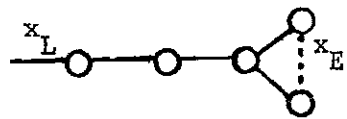
(d) Root enters,  
Root leaves.

(e) Root enters,  
Pseudoroot leaves.

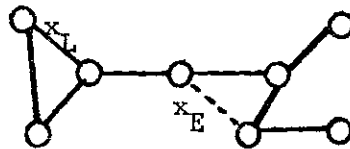
(f) Root enters,  
Arc leaves.

$\text{---} \frac{x_E}{\text{---}}$  entering arc  
 $\text{---} \frac{x_L}{\text{---}}$  leaving arc

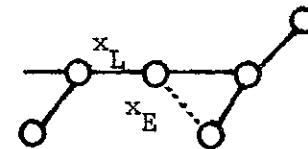
Figure 7. (Continued)



(g) Pseudoroot enters,  
Root leaves.



(h) Pseudoroot enters,  
Pseudoroot leaves.



(i) Pseudoroot enters,  
Arc leaves.

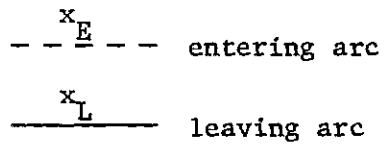


Figure 7. (Concluded)

Proof:

The portion of the basis corresponding to  $B_1$  and  $B_2$  is given in partitioned form as:

$$B' = \left[ \begin{array}{c|c|c|c} \overline{B_1} & 0 & & \\ \hline 0 & B_2'' & b_j & 0 \\ \hline & & & \overline{B_2'} \end{array} \right] \quad (72)$$

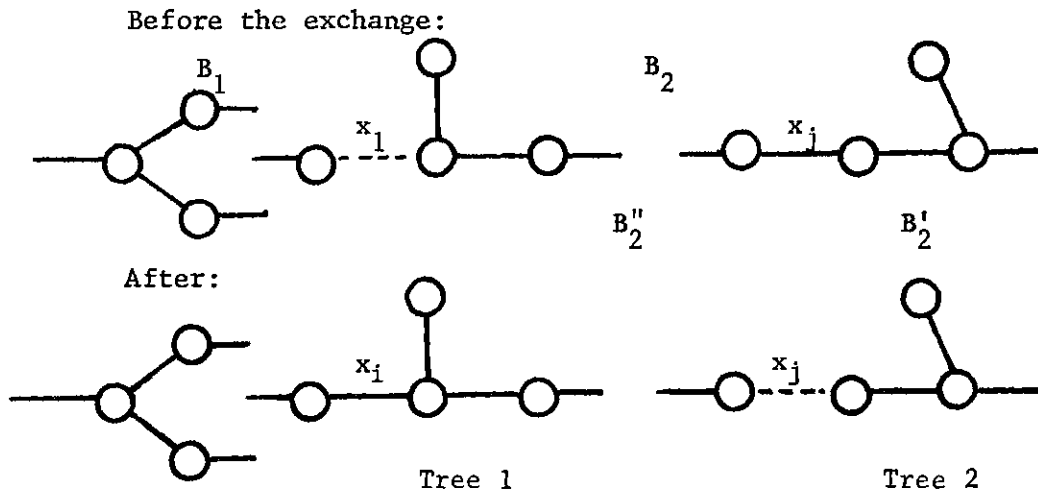
After the basis change it becomes:

$$B' = \left[ \begin{array}{c|c|c|c} \overline{B_1} & & 0 & 0 \\ \hline 0 & b_i & B_2'' & \\ \hline & & & \overline{B_2'} \end{array} \right] \quad (73)$$

That  $B_1$  remains unchanged is clear, since its columns are not affected by the exchange. Since  $B_2'$  and  $B_2''$  have no nonzero entries in the intersection of their rows and columns, the deletion of  $b_j$  leaves  $B_2'$  as a distinct block.

The entering column  $b_i$  has a nonzero entry in a row corresponding to  $B_2''$  by definition of entering and leaving variable. Thus, with the addition of  $b_i$  to the basis all of  $B_2''$  would become part of the same component as  $B_1$  using the constructive process for defining a block (see Theorem 2.1). Since they were distinct blocks,  $B_2''$  originally had no nonzero entries in common with  $B_1$ . Thus,  $b_i$  is the only column with a nonzero entry in a row of  $B_1$  and a nonzero entry in a row of  $B_2''$ , proving the last part of the lemma.

The lemma can be stated in terms of the associated graph. Before the basis change  $B_1$  and  $B_2$  are both rooted (pseudorooted) trees.  $B_2'$  contains the root or pseudoroot for  $B_2$  (if it is not empty).  $B_2''$  is connected to  $B_2'$  by the arc for  $x_j$  and is in fact above this arc. After the basis change  $B_2'$  corresponds to a rooted tree.  $B_1$ ,  $b_i$ , and  $B_2''$  form a rooted tree and  $B_2''$  is above the arc for  $x_i$ . An example of this is:



Other examples illustrating the lemma are given in Figure 7(a,b,c).

If the leaving variable is a root or part of a pseudoroot, then then  $B_2'' = B_2$  and the blocks  $B_1$  and  $B_2$  will become one block after the exchange; otherwise, they will remain as two distinct blocks after the exchange. In the case that the entering column has only one nonzero entry the following corollary to the lemma holds.

Let the column  $b_i$  associated with the nonbasic entering variable  $x_i$  have only one nonzero entry in a row corresponding to the block  $B_2$ . Let  $B_2$  be partitioned into  $B_2'$  and  $B_2''$  by the column  $b_j$  of the leaving variable  $x_j$ .

Corollary: After the basis exchange  $B_2'$  is a distinct block and  $B_2''$  is a distinct block with  $b_i$  as its root.

Proof:

The proof is the same as for the lemma with  $B_1$  empty.

Examples of the corollary are in Figure 7(d,e,f).

On the other hand, consider the case where the column  $b_i$  for the nonbasic entering variable  $x_i$  has both nonzero entries in two rows of the same component  $B_1$ . Let  $x_j$  and  $b_j$  be defined as before. Let  $s$  be the node where the paths from the rows of the nonzero entries of  $b_i$  join before reaching the root or pseudoroot. Let the nodes where the paths meet the pseudoroot be  $n_L$  and  $n_R$ , respectively. Let  $b_j$  partition  $B_1$  into  $B_1'$  and  $B_1''$  as before.

Lemma 3.3

If  $b_i$  has its nonzero elements in two rows of the same block  $B_1$ , then after the basis exchange either 1) a nonzero entry of  $B_1''$  can be reached from a nonzero entry of  $B_1'$  through a series of alternating row and column moves on nonzero elements only by going through the nonzero elements of  $b_i$  or 2)  $B_1'$  is a distinct block and  $B_1''$  is a distinct block with a pseudoroot comprised of the column  $b_i$  for the entering variable  $x_i$  and the columns of the variables in the paths from both ends of  $x_i$  to the joining node(s)  $s$  ( $n_L$  and  $n_R$ ).

Proof:

For (1) the proof is essentially the same as Lemma 3.1. For (2)  $B_1'$  contains a root or pseudoroot and has no common nonzero entries with  $B_1''$  after the deletion of  $b_j$ ; thus it is a distinct component. If  $B_1$

contained a root or if  $s$  was above the root or pseudoroot, then  $B_1''$  has as its pseudoroot the entering variable  $x_i$  and all variables in the paths to  $s$  from both ends of  $x_i$ . If  $n_L$  and  $n_R$  are distinct, then the pseudoroot of  $B_1''$  is comprised of  $x_i$ , the variables in the paths to  $n_L$  and  $n_R$ , and the part of the cycle between  $n_L$  and  $n_R$  not containing the leaving variable  $x_j$ . If the leaving variable is a root or in a pseudoroot, then  $B_1'$  is empty.

Some cases covered by the lemma are shown in Figure 7(g,h,i).

The following theorem uses the structure of a basis change established in the preceding lemmas and corollary to specify the recalculation of simplex multipliers.

#### Theorem 3.2

The only simplex multipliers which need to be recalculated after a basis change are those associated with nodes above the entering arc.

Proof:

The definitions and terminology used in the lemmas and corollary are used here. First consider the case where the column  $b_i$  of the entering variable  $x_i$  has its two nonzero entries in the rows of two distinct components  $B_1$  and  $B_2$ . Let the column  $b_j$  of the leaving variable  $x_j$  partition  $B_2$  into  $B_2'$  and  $B_2''$  as before. Since  $B_1$  and  $B_2'$  retain the same form, the corresponding simplex multipliers remain the same. Let  $\pi_r$  be the simplex multiplier associated with the row of  $B_1$  containing the nonzero entry  $a_{ki}$  of  $b_i$ . Then for the equation of  $B_2''$  containing the other nonzero entry, the simplex multiplier ( $\pi_s$ ) is determined by the equation:

$$a_{ki}\pi_r + a_{k'i}\pi_s = c_i \quad (74)$$

But  $\pi_r$  does not change. Thus:

$$\pi_s = \frac{c_i - \pi_r a_{ki}}{a_{k'i}} \quad (75)$$

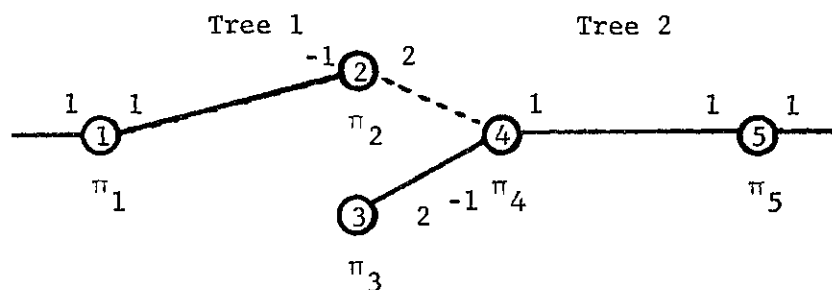
Since  $B_2''$  contains no other nonzero entries in common with  $B_1$ , the remaining simplex multipliers are determined iteratively as demonstrated in Section 3.2.

If the entering variable is a root as in the corollary, then the simplex multipliers have to be determined only for  $B_2''$  with the one at the root ( $\pi_r$ ) determined by:

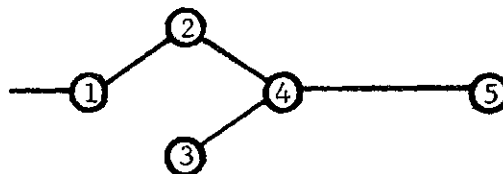
$$\begin{aligned} a_{ki} \pi_r &= c_r \\ \pi_r &= c_r / a_{ki} \end{aligned} \quad (76)$$

The case where both nonzero entries of  $b_i$  are in the same tree as in Lemma 3.3 follows the same pattern. If the dropping variable is above the joining node  $s$ , then the recalculation of the simplex multipliers is the same as the first instance. If a distinct component  $B_2''$  is formed with a pseudoroot, then the simplex multipliers of the cycle are calculated and the remainder calculated tracing out from each node of the cycle as before.

The following examples from Figure 3(f,g) of the example of Section 2.4 demonstrate the use of the theorem. The graphical representation of the problem before the basis change, with  $x_4$  entering and  $x_2$  leaving is:



After the basis change:



The equation for the simplex multipliers before the change was:

$$\pi B_1 = c_{B_1} \quad (77)$$

$$\pi' B_2 = c_{B_2}$$

with:

$$B_1 = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} \quad B_2 = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad \pi = [\pi_1 \quad \pi_2] \quad c_{B_1} = [0 \quad 3]$$

$$\pi' = [\pi_3 \quad \pi_4 \quad \pi_6] \quad c_{B_2} = [1 \quad 2 \quad 1000]$$

Using the iterative scheme tracing out from the root of a tree, the simplex multipliers can be calculated.

For Tree 1:

$$\begin{aligned} (1)\pi_1 &= 0 & \pi_1 &= 0 \\ \pi_1 - \pi_2 &= 3 & \pi_2 &= -3 \end{aligned} \quad (78)$$

For Tree 2:

$$(1)\pi_5 = 1000 \quad \pi_5 = 1000 \quad (79)$$

$$\pi_5 + \pi_4 = 2 \quad \pi_4 = -998$$

$$2\pi_3 - \pi_4 = 1 \quad \pi_3 = -997/2$$

After the basis change there is only one tree, since a root left the basis.

$$[\pi \quad \pi'] \begin{bmatrix} B_1 & & 0 \\ \hline & b_1 & \\ 0 & & B_2'' \end{bmatrix} = \begin{bmatrix} c_{B_1} \\ c_{B_2} \end{bmatrix} \quad (80)$$

$$[\pi_1 \quad \pi_2 \quad \pi_3 \quad \pi_4 \quad \pi_5] \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 3 \\ 1 \\ 2 \end{bmatrix} \quad (81)$$

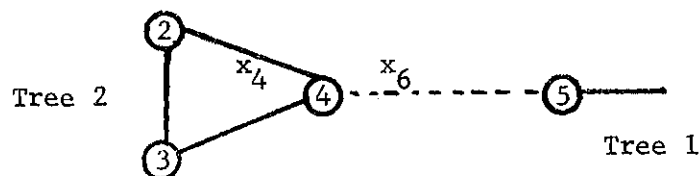
$\pi_1$  and  $\pi_2$  remain the same. The calculations for the simplex multipliers above  $x_4$  are:

$$2\pi_2 - \pi_4 = 3 \quad \pi_4 = -9 \quad (82)$$

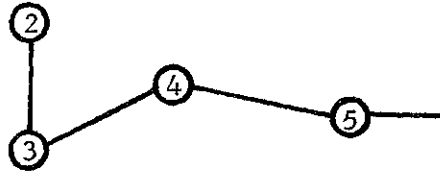
$$2\pi_3 - \pi_4 = 1 \quad \pi_3 = -4$$

$$\pi_4 + \pi_5 = 2 \quad \pi_5 = 11$$

Another example with a pseudoroot is taken from the iterations shown in Figures 3(d,e). Before the change, with  $x_6$  entering and  $x_4$  leaving:



After the change:



$$\pi'_{B_2} = c_{B_2} \quad \pi_{B_1} = c_{B_1} \quad (83)$$

with

$$B_1 = [1] \quad B_2 = \begin{bmatrix} -1 & 2 & 0 \\ 2 & 0 & 2 \\ 0 & -1 & -1 \end{bmatrix} \quad \pi = [\pi_5] \quad c_{B_1} = [1000] \\ \pi' = [\pi_3 \quad \pi_2 \quad \pi_4] \quad c_{B_2} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

The calculations for the multipliers before the change:

For Tree 1

$$-\pi_2 + 2\pi_3 = 2 \quad \pi_2 = 4 \quad (84)$$

$$2\pi_2 - \pi_4 = 3 \quad \pi_3 = 3$$

$$2\pi_3 - \pi_4 = 1 \quad \pi_4 = 5$$

For Tree 2

$$\pi_5 = \frac{(1000)}{1} = 1000 \quad (85)$$

After the basis change, there is one component

$$[\pi \quad \pi'] \begin{bmatrix} B_1 & & 0 \\ & b_i & \\ 0 & & B_2'' \end{bmatrix} = \begin{bmatrix} c_{B_1} \\ c_{B_2} \end{bmatrix} \quad (86)$$

$$[\pi \ \pi'] = [\pi_5 \ \pi_4 \ \pi_3 \ \pi_2] \quad (87)$$

$$B_P = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad c_B = \begin{bmatrix} 1000 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

$\pi_1$  remains the same. Nodes 2, 3, and 4 are above the arc for  $x_6$ .

$$\pi_4 = c_6 - \pi_5 = 2 - 1000 = -998 \quad (88)$$

$$\pi_3 = \frac{c_5 + \pi_5}{2} = \frac{1 - 998}{2} = \frac{-997}{2}$$

$$\pi_2 = -c_3 + 2\pi_3 = -2 - 997 = -999$$

### 3.6 Labeling the Tree

In the earlier sections, methods for carrying out the simplex operations for the GFP have been developed which utilize a graphical representation of the problem. The procedures need an efficient means for identifying the tree structure and for tracing up and down a rooted or pseudorooted tree. The method of doing this will be discussed now. Glover, Klingman, and Kearny [34] have presented a scheme called the augmented predecessor index method for maintaining the tree for ordinary transportation problems. They indicate that it is essentially the triple labeling method of Johnson [59]. Srinivasan and Thompson [87] have also presented a similar scheme, and Maurras [71] indicates a labeling procedure for the GFP but does not give the details. The methods given here are extensions of Johnson's triple labeling scheme for the ordinary flow

problem.

The requirements for a labeling scheme are:

- 1) Trace from a node in the tree to the root.
- 2) Trace all nodes above a specified node.
- 3) Detect when a root or pseudoroot has been reached in a tracing down a tree.
- 4) Trace the nodes of a pseudoroot.
- 5) Efficient updating of labels when a basis change is made.

The essence of the row and column generation procedures is to use the labels to calculate the current representation of a row or column directly, without the node-labeling search of an out-of-kilter type algorithm. Two schemes will be presented: one with the restriction that if there is an arc between two nodes it is unique and the other without this restriction.

For the first method the labels will be node numbers of other nodes in the same tree. Consider node  $i$ . Then, if node  $j$  is connected to node  $i$  by a basic arc, a device is needed to determine the arc between  $i$  and  $j$ ; that is, which variable ( $x_k$ ) occurs in equation  $i$  and in equation  $j$ . From this  $a_{1k}$  and  $a_{2k}$  may be found. Let there be  $n$  regular variables (two ended arcs) and  $s$  slack arcs (slack and artificial variables). Define the function  $N_{ij}$  such that  $N_{ij} = k$  if the nonzero coefficient  $a_{1k}$  of  $x_k$  occurs in equation  $i$  and  $a_{2k}$  occurs in equation  $j$  and  $N_{ij} = -k$  if  $a_{1k}$  is in equation  $j$  and  $a_{2k}$  in equation  $i$ .

$$N_{ij} = k \quad \text{if } b_k = \begin{bmatrix} 0 \\ \vdots \\ a_{1k} \\ 0 \\ \vdots \\ a_{2k} \\ 0 \\ \vdots \end{bmatrix} \quad \begin{array}{l} \text{row } i \\ \text{row } j \end{array} \quad (89)$$

Further

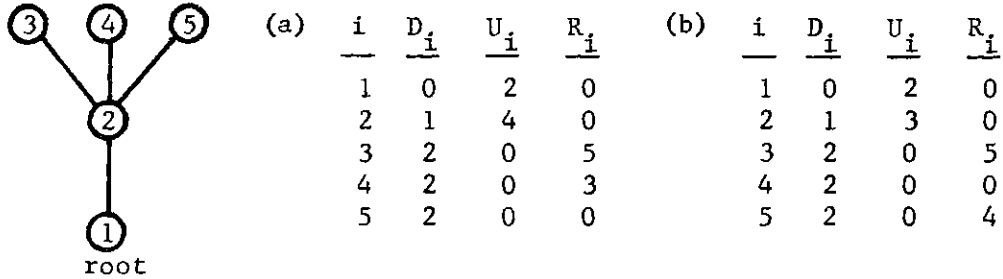
$$N_{ij} = -k \quad \text{if } b_k = \begin{bmatrix} 0 \\ \vdots \\ a_{2k} \\ 0 \\ \vdots \\ a_{1k} \\ 0 \\ \vdots \end{bmatrix} \quad \begin{array}{l} \text{row } i \\ \text{row } j \end{array} \quad (90)$$

In terms of direction of an arc as defined previously  $N_{ij} = k$ , if arc  $k$  is directed away from node  $j$  toward node  $i$  and  $N_{ij} = -k$  if arc  $k$  is from  $j$  to  $i$ . Let there be  $m$  equations (nodes). If there is a slack arc at equation  $i$  address it by defining  $N_{m+1 i} = k$ .

Associated with each node  $i$  will be three labels: the down label ( $D_i$ ), the up label ( $U_i$ ), and the right label ( $R_i$ ). The down label will indicate the next node closer to the root (or pseudoroot) connected to the current node by a basic arc. The node on which a basic slack arc is incident is called the root node and has a down label of zero. If a node is part of a pseudoroot, its down label will be negative.

Thus,  $D_i = j$  indicates that node  $j$  is the next node down the tree connected to node  $i$  by arc  $k = |N_{ij}|$ . If  $D_i = 0$ , node  $i$  is the root node and the associated basic slack arc is  $k = N_{m+1 i}$ . If  $D_i < 0$  then node  $i$  is part of a cycle of basic arcs.

The up label  $U_i$  indicates a node connected to node  $i$  by a basic arc, one arc farther away from the root or pseudoroot. If  $U_i = j$ , then arc  $k = |N_{ij}|$  is the basic arc connecting nodes  $i$  and  $j$  and arc  $k$  is in the path from node  $j$  to the root or pseudoroot.  $U_i = 0$  indicates that there are no nodes above node  $i$ ; that is, node  $i$  is at the end of a tree. Since more than one node may be directly above a particular node, the right label  $R_i$  indicates a node (other than  $i$ ) directly above the node  $D_i$ . That is, nodes  $i$  and  $R_i$  are both above the same node ( $D_i$ ).  $R_i = 0$  indicates that there are no other nodes above node  $D_i$  which have not been specified by an up or right label. For example:



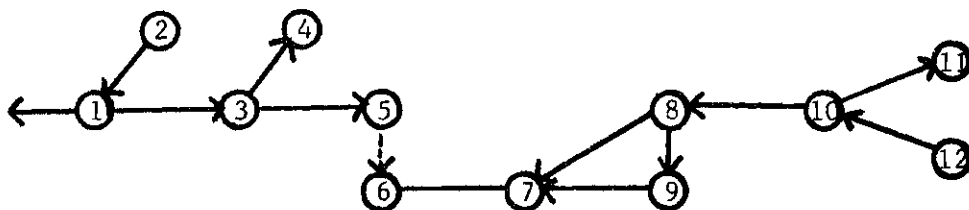
The labels in (a) or (b) may be used to represent the tree structure in the graph showing that a particular set of labels is not necessarily the unique representation. If  $D_i < 0$ , then node  $i$  is in a cycle, and  $R_i$  indicates an adjacent node in the cycle.

This labeling scheme allows tracing up or down a tree as required to carry out the simplex operations for the GFP which have been described. To trace down a tree, the path to the root or pseudoroot is found by:

$$n_{i+1} = D_i, \quad a_i = |N_{n_i, n_{i+1}}|$$

This proceeds until  $D_i \leq 0$ . If  $D_i = 0$ , then the slack arc is  $a_s = N_{m+1,i}$ . If  $D_i < 0$ , then let  $p = i$ , and let  $n_{i+1} = R_i$ ,  $a_i = |N_{n_i} n_{k+1}|$  until  $n_i = p$ , in which case the cycle has been traced.

The example below shows the down trace required to obtain the representation of a nonbasic arc between a rooted and a pseudorooted tree.



$n = 12$

$N_{13} 1$		=	1
$N_{21}$	= $-N_{12}$	=	2
$N_{13}$	= $-N_{31}$	=	3
$N_{34}$	= $-N_{43}$	=	4
$N_{53}$	= $-N_{35}$	=	5
$N_{56}$	= $-N_{65}$	=	6
$N_{67}$	= $-N_{76}$	=	7
$N_{87}$	= $-N_{78}$	=	8
$N_{79}$	= $-N_{97}$	=	9
$N_{89}$	= $-N_{98}$	=	10
$N_{10} 8$	= $-N_{8} 10$	=	11
$N_{10} 11$	= $-N_{11} 10$	=	12
$N_{12} 10$	= $-N_{10} 12$	=	13

<u>i</u>	<u><math>D_i</math></u>	<u><math>U_i</math></u>	<u><math>R_i</math></u>
1	0	2	0
2	1	0	3
3	1	5	0
4	3	0	0
5	3	0	4
6	7	0	0
7	-1	6	8
8	-1	10	9
9	-1	0	7
10	8	11	0
11	10	0	12
12	10	0	0

To determine the representation of nonbasic arc  $x_6$ , node five will be called its left end and node six its right end. Tracing down labels from node six and finding the connecting arcs defines the path:

$$\begin{aligned} n_1 &= 5 \\ n_2 = D_1 &= 3 & |N_{53}| &= 5 \\ n_3 = D_3 &= 1 & |N_{31}| &= 3 \\ D_1 &= 0 & |N_{13}| &= 1 & (5, a_5, 3, a_3, 1, a_1) \end{aligned}$$

This is the path to the root and the slack arc at the root. From the right side:

$$\begin{aligned} n_1 &= 6 \\ n_2 = D_6 &= 7 & |N_{67}| & \\ D_7 &= -1 & n_3 = R_7 &= 8 & |N_{78}| &= 8 & p &= 7 \\ n_4 = R_8 &= 9 & |N_{89}| &= 9 \\ n_5 = R_9 &= 7 & |N_{97}| &= 10 & \text{Terminate since } n_5 &= p. \end{aligned}$$

$$(6, a_7, 7, a_8, 8, a_9, 9, a_{10}, 7)$$

This is the path to the pseudoroot and the pseudoroot. These sequences of nodes and arcs are used to derive the column representing  $x_6$  in terms of the basic variables.

To trace all of the nodes above a specified node, the following algorithm is used:

Let the original node be  $k$ .

- (1) Follow the up labels until  $U_i = 0$  for some  $i$ .
- (2) If  $R_i \neq 0$ , let  $i = R_i$  and go to step 1, otherwise go to step 3.

(3) Let  $i = D_i$ . If  $R_i \neq 0$ , let  $i = R_i$  and go to step 1, otherwise go to step 4.

(4) If  $D_i \neq k$ , go to step 3 otherwise terminate.

To trace all nodes above node one in the single tree example with five nodes, the algorithm produces the following trace.

$U_1 = 2$	$i = 2$		
$U_2 = 4$	$i = 4$		
$U_4 = 0$	$R_4 = 3$	$i = 3$	
$U_3 = 0$	$R_3 = 5$	$i = 5$	
$U_5 = 0$	$R_5 = 0$	$D_5 = 2$	$i = 2$
$R_2 = 0$	$D_2 = 1$	$i = 1$	Terminate.

If the root node is used as the starting point, the algorithm traces all the nodes of a rooted tree. For a pseudorooted tree, a node in the cycle is chosen as the starting node, the previous procedure is used to trace above it; the node to the right of the current node in the cycle is obtained using the right label; and the procedure is repeated until the original node is reached. The upward trace is used to calculate the original simplex multipliers, update the simplex multipliers, and generate a row of the basis inverse, if it is required. The labeling scheme is the means for storing the current basis. The method is essentially an indirect addressing scheme in which the variables of the basis (arcs) are identified through knowledge of the equations (nodes) in which they occur.

As mentioned, this procedure is limited in that only one variable may appear in the same two equations. That is, there cannot be two arcs

between the same two nodes and only one slack or artificial variable can appear in each equation. Consider a situation where the cost for a regular variable is a piecewise linear function. Charnes and Lemke [12] have shown that this variable may be replaced by the sum of a set of variables, one for each segment of the cost function. For the GFP this means that several variables will appear in the same two equations with different costs and bounds. The labeling systems must have a mechanism to identify which basic variable corresponds to the arc connecting two nodes in a tree of the basis. Similarly, both a slack variable and an artificial variable in the same equation may be required to define an initial basic solution. The following labeling method will allow these conditions which were not allowed in the previous scheme.

Instead of using nodes to determine arcs in the basis, this scheme will use a node and attached arc to determine an adjacent node. In a sense this scheme is the opposite of the previous indirect addressing scheme. Three labels, down, up, and right, will be associated with each node. However, the label is the number of the basic arc leading to the appropriate node in the previous labeling scheme. The end of a tree is still indicated by the up label equaling zero, i.e.,  $U_i = 0$ . The right label being zero (i.e.,  $R_i = 0$ ) and less than zero (i.e.,  $R_i < 0$ ) have the same meaning. Suppose there are  $n$  two ended arcs numbered  $1, \dots, n$ , and  $s$  slack arcs numbered  $n+1, \dots, n+s$ . Then if  $D_i > n$ , node  $i$  is a root node and  $D_i$  is the associated slack arc.

Tracing down a tree from node  $s$  is accomplished by the following algorithm. Initially  $i = s$ ; there are  $n$  two ended arcs; and  $j = 0$ .

(1)  $j = j+1$ ,  $n_j = i$ ,  $k = D_i$ .

(2) If  $k > n$ , go to step 7. If  $k < 0$ , go to step 3.

Otherwise,  $a_j = k$ . If  $p_k = i$ , let  $i = q_k$ ; otherwise, let  $i = p_k$ , go to step 1.

(3) Let  $t = i$ .

(4) Let  $k = R_i$ ,  $a_j = k$ ,  $j = j+1$ .

(5) If  $p_k = i$ , let  $i = q_k$ , otherwise let  $i = p_k$ ; go to step 6.

(6)  $n_j = i$ . If  $i = t$ , terminate; otherwise, go to step 4.

(7)  $a_j = k$ , root reached, terminate.

The set of labels and arc orientations using the new scheme for the previous example is:

<u>k</u>	<u><math>p_k</math></u>	<u><math>q_k</math></u>	<u>i</u>	<u><math>D_i</math></u>	<u><math>U_i</math></u>	<u><math>R_i</math></u>
1	12	10	1	13	2	0
2	2	1	2	2	0	3
3	1	3	3	3	5	0
4	3	4	4	4	0	0
5	3	5	5	5	0	4
6	5	6	6	7	0	0
7	6	7	7	- 1	7	8
8	8	7	8	- 1	11	9
9	9	8	9	- 1	0	10
10	9	7	10	11	12	0
11	10	8	11	12	0	1
12	10	11	12	1	0	0
13	1					

Applying the down trace algorithm, from both ends of  $a_6$ :

1)  $i = p_6 = 5$

2)  $j = 1$ ,  $n_1 = 5$

3)  $k = D_5 = 5$

$0 \leq k \leq 12$       Thus  $a_1 = 5$

$$4) \quad p_6 \neq 5 \quad \text{Thus } i = p_6 = 3$$

$$5) \quad j = 2, \quad n_2 = 3$$

$$6) \quad k = D_3 = 3$$

$$0 \leq k \leq 12 \quad \text{Thus } a_2 = 3$$

$$7) \quad p_3 \neq 3 \quad \text{Thus } i = p_3 = 1$$

$$8) \quad j = 3$$

$$9) \quad k = D_1 = 13$$

$$k > 12$$

$$a_3 = 13 \quad \text{Terminate}$$

The path to the root and the slack arc at the root are:

$$(5, a_5, 3, a_3, 1, a_{13})$$

From the other end of  $a_6$ :

$$1) \quad i = q_6 = 6$$

$$2) \quad j = 1, \quad n_1 = 6$$

$$3) \quad k = D_6 = 7$$

$$0 \leq k \leq 12 \quad \text{Thus } a_1 = 7$$

$$4) \quad p_7 = 6 \quad \text{Thus } i = q_7 = 7$$

$$5) \quad j = 2, \quad n_2 = 7$$

$$6) \quad D_7 = -1 \quad \text{Pseudoroot encountered.}$$

$$7) \quad t = 7$$

$$8) \quad k = R_7 = 8, \quad a_2 = 8, \quad j = 3$$

$$9) \quad p_8 \neq 7 \quad \text{Thus } i = p_8 = 8, \quad n_3 = 8, \quad t \neq i$$

$$10) \quad k = R_8 = 9, \quad a_3 = 9, \quad j = 4$$

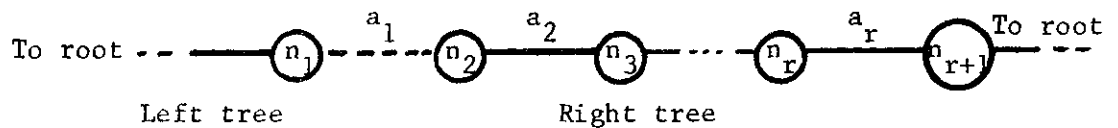
$$11) \quad p_9 \neq 8 \quad \text{Thus } i = p_9 = 9, \quad n_4 = 9, \quad t \neq i$$

- 12)  $k = R_9 = 10, a_4 = 10, j = 5$   
 13)  $p_{10} = 9$       Thus  $i = q_{10} = 7, n_s = 7$   
 14)  $i = t = 7$       Terminate.

The path to the pseudoroot and the pseudoroot are:

$$(6, a_7, 7, a_8, 8, a_9, 9, a_{10}, 7)$$

The changes required for an upward trace are analogous. The updating of the labels to reflect a basis change is essentially the same for both labeling methods. We will illustrate it for the latter scheme which is adopted in the coding of the algorithm presented. The exact details of the required changes depend on the nature of the basis change as shown in Figure 7. Generally, only the labels for the nodes between the entering arc and the leaving arc must be changed. Consider the simplest case in which neither a root nor pseudoroot is involved in a basis change. The path between the entering arc ( $a_1$ ) and the leaving arc ( $a_r$ ) is called the stem. Suppose the nodes and arcs are numbered such that this path is  $(n_1, a_1, \dots, n_r, a_r, n_{r+1})$  as shown below.



The down labels in the right tree before the change are:

$$D_i = a_i \quad i = 2, \dots, r.$$

The new down labels after the basis change are:

$$\overline{D}_i = a_{i-1} \quad i = 2, \dots, r.$$

The down labels for the left tree and the remainder of the right tree remain unchanged after the basis change.

Now consider the up labels. Before the basis change, suppose:

$$U_{n_i} = a_{i-1} \quad i = 3, \dots, r+1.$$

The new up labels after a basis change will be:

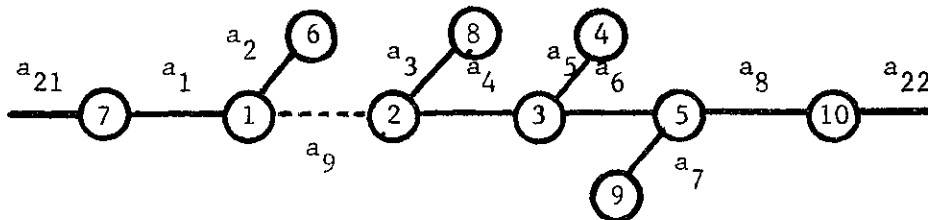
$$\bar{U}_{n_i} = R_{n_{i-1}} \quad i = 3, \dots, r+1.$$

As discussed previously in this section, the representation of the tree above a certain node is not unique. Thus, if  $U_{n_i} \neq a_{i-1}$ , then by using the right labels determine  $n_j$  so that  $R_{n_j} = a_{i-1}$ . Then let  $\bar{R}_{n_j} = R_{n_{i-1}}$ .

To impose the new up structure, let:

$$U_{n_i} = a_{i+1}, \quad R_{n_{i+1}} = U_{n_i} \quad i = 1, \dots, r-1.$$

These label changes will, in effect, graft the stem and all nodes connected to it to the tree through the entering arc and cut the leaving arc. To illustrate the scheme:



Assume  $n = 20$ .

$a_9$  enters,  $a_6$  leaves.

An appropriate set of labels is:

<u>i</u>	<u>D<sub>i</sub></u>	<u>U<sub>i</sub></u>	<u>R<sub>i</sub></u>
1	1	2	0
2	4	3	0
3	6	5	7
4	5	0	4
5	8	6	0
6	2	0	0
7	21	1	0
8	3	0	0
9	7	0	0
10	22	8	0

The stem is:

$$(n_1, a_9, n_2, a_4, n_3, a_6, n_5)$$

The down labels become:

$$D_2 = 9$$

$$D_3 = 4$$

To reverse the upward labels:

$$1) \quad T = R_2 = 0 \quad \text{Adjoin entering arc.}$$

$$R_2 = U_1 = 2$$

$$U_1 = 9$$

$$2) \quad U_3 = 4 ? \quad \text{No} \quad \text{Remove old up hierarchy.}$$

$$k = U_3 = 5 \quad \text{If } p_5 = 3 \quad n_5 = q_5 = 4$$

$$R_4 = 4 ? \quad \text{Yes} \quad R_4 = T = 0$$

$$3) \quad T = R_3 = 7$$

$$R_3 = U_2 = 3$$

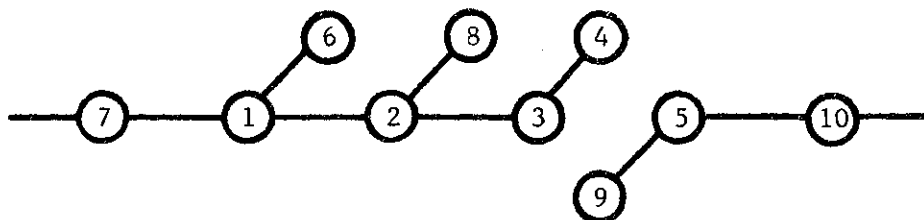
$$U_2 = 4$$

$$4) \quad U_5 = 6 ? \quad \text{Yes}$$

$$U_5 = T = 7$$

Since 5 is the cut node, terminate.

The new trees and labels are:



$i$	$D_i$	$U_i$	$R_i$
1	1	9	0
2	9	4	2
3	4	5	3
4	5	0	0
5	8	7	0
6	2	0	0
7	21	1	0
8	3	0	0
9	7	0	0
10	22	8	0

The boxed labels were the ones changed.

This is the fundamental scheme for changing labels. Only the labels indicating the hierarchy along the stem must be changed. Slight modification must be made to create or dissolve a cycle or for the entry or departure of a root.

Recent work in tree structure labeling has been done by Srinivasan and Thompson [87], Glover and Klingman [27], and Glover, Klingman, and Kearny [34]. The predecessor index method of the latter two papers is precisely the down labeling scheme for tracing to the root. The extensions mentioned by Srinivasan and Thompson [87] are more useful when the basis graph consists of only one tree and does not seem to be as attractive when several rooted components are present. The paper of Glover, Klingman, and Kearny details the label changing procedures for the ordinary

flow problem using a Johnson type scheme of the first type mentioned. Maurras [73] discusses a labeling and relabeling procedure for the GFP but it is not clear how his scheme is used to trace up or down a tree. His main idea seems to be that of identifying the order of the coefficients for an arc in a path to the root.

The nature of the GFP has been described in this chapter and methods to take advantage of this structure for solving the GFP have been outlined. These devices are organized into an algorithm for the GFP and some of its specializations in the next chapter.

## CHAPTER IV

### CONTINUOUS ALGORITHMS

#### 4.1 Introduction

In this chapter the characteristics of the generalized flow problem noted in previous chapters are used to specialize the primal simplex method into an algorithm for the GFP. The current basis is recorded using the triple labeling scheme described in Section 3.6. The column generation scheme given in Section 3.3 and simplex multiplier recalculation method described in Theorem 3.2 are the main tools for carrying out the simplex operations. A general description of the GFP algorithm is given in this chapter. Computational results are also presented.

The GFP algorithm specializes further when the same concepts are used to construct an algorithm for the ordinary flow problem. These specializations are noted in Section 4.3, and an outline of the ordinary flow algorithm and computational results are given.

Finally, the special characteristics of a similar algorithm for the transportation problem are noted and the computational results using such an algorithm are presented in Section 4.4.

#### 4.2 GFP Algorithm

##### 4.2.1 Statement of the Algorithm

The algorithm is now stated in general terms.

1. Initialize by determining a primal basic solution and computing

the corresponding simplex multipliers.

2. Select a nonbasic variable  $x_j$  to enter the basis from the set  $J = \{j: \bar{c}_j > 0 \text{ and } x_j = M_j \text{ or } \bar{c}_j < 0, x_j = 0 \text{ where } \bar{c}_i = c_i - \pi_{p_i} a_{1i} - \pi_{q_i} a_{2i}\}$ . If  $J = \emptyset$  terminate; otherwise go to step 3.

3. Determine the structure of the current basis relative to the entering variable using the labels.

4. Iteratively construct the column of the entering variable and determine the variable to leave the basis, noting the position of the leaving variable in the tree structure of the current basis.

5. Using the column  $(\underline{d}_j)$  and the change  $(\Delta)$  in the value of the entering variable from step 4, compute the new values of the basic variable affected by the basis change, i.e.,  $\underline{x}' = \underline{x} - \Delta \underline{d}_j$ .

6. Using the information about the tree structure and the entering and leaving variables from steps 3 and 4, change the labels to represent the new basis.

7. Recalculate the simplex multipliers  $(\pi)$  above the entering variable. Return to step 2.

The difference between this algorithm and the general primal simplex method is that additional logical information is used to take advantage of the special structure of the basis of the GFP. Thus, in step 4, only the nonzero entries in the column for the entering variable are calculated. Also, in step 7, only the simplex multipliers which change are recalculated. The theoretical justification for these methods were presented in Chapter III. The algebraic operations and storage requirements of maintaining the current basis inverse is avoided by using these techniques also. To use these techniques, the label representation of the

basis must be maintained and updated. This was shown to be a reasonable task and efficient means for doing it were presented in Section 3.6.

#### 4.2.2 Initialization

The initialization may be accomplished in several ways. The simple one used to obtain the computational results presented later in this section is given below. Assume the equations are in equality form (i.e., slack variables have been added).

Let there be  $m$  equations and  $n$  variables with nonzero coefficients  $(a_{1i}, a_{2i})$  in the two equations  $p_i$  and  $q_i$ . Let there be  $s$  slack variables so that there are  $s$  variables with one nonzero coefficient  $(a_{1i})$  in equation  $p_i$ . The initial right hand side is  $b = (b_1, \dots, b_m)$ . The initialization procedure is:

$$\text{Set } x_j = x_j^* \quad \text{where } x_j^* = \begin{cases} 0 & \text{if } c_j \geq 0 \\ M_j & \text{if } c_j < 0 \end{cases}$$

Adjust the right hand side so that:

$$b^* = b - Ax^*$$

For each equation  $i$  which contains a slack variable  $x_i^s$  and whose adjusted right hand side  $b_i^*$  is greater than zero let:  $x_i^s = b_i^*$ .

For each of the remaining  $r$  equations define an artificial variable  $x_i$ ,  $i = n+s+1, \dots, n+s+r$  so that

$$\text{if } b_i^* < 0, a_{1i} = -1$$

$$b_i^* \geq 0, a_{1i} = 1$$

and let  $x_i = |b_i^*|$  for all such equations.

Additionally, if  $b_i^* = 0$ , set the upper bound on the corresponding

artificial variable equal to zero. A set of labels corresponding to the above artificial and slack variables is constructed to define the initial basis.

In certain instances, a starting solution with fewer artificials may be attainable. Consider the degenerate case, i.e., when

$$R' = \{i : b_i^* = 0\} \neq \emptyset \quad (1)$$

In this case, for  $i \in R'$ , any variable  $x_j$  with a nonzero coefficient in equation  $i$  can be brought into the basis at its present value (either  $M_j$  or 0). A nonempty set  $R'$  might often occur. Consider the case where the equations for GFP can be partitioned before the addition of slack variables into the sets:

$$S = \{i : \sum_j a_{kj} x_j \leq b_i ; b_i > 0\} \quad (2)$$

$$R = \{i : \sum_j a_{kj} x_j = b_i ; b_i = 0\} \quad (3)$$

$$T = \{i : \sum_j a_{kj} x_j \geq b_i ; b_i > 0\} \quad (4)$$

In the usual terminology,  $S$  is the set of sources,  $R$  is the set of intermediate nodes, and  $T$  is the set of demands. Consider the graphical representation of the sets  $R$  and  $S$ . Associate with each arc the direction defined by the convention in Section 2.3 and the costs  $(c_i)$ . By defining a dummy node connected to all of the nodes of  $S$ , a spanning tree can be determined by finding the shortest path from the dummy node to all the nodes in  $S \cup R$ . The original variables corresponding to the spanning tree

could be used in an initial basis eliminating all artificials from the set  $R$ . Two limitations of this method are that the labels reflecting this basis must be determined, and the spanning tree takes little account of the original problem except for the per unit variable costs. For certain special cases the algorithm of Charnes and Raike [13] or Glover and Klingman [32] could be used to determine a spanning tree which would take into account the constraints of the original problem except for the bounds.

It must be remembered that all of these starting procedures are heuristic in that there is no theoretical reason that one should result in a fewer number of iterations than another. The desire to eliminate artificials seems reasonable to be sure; however, limited computational experience indicates that, while the procedures may reduce the number of simplex iterations, they may actually increase the total computation time. This is because, in general, the effort required for an iteration is proportional to the number of variables with nonzero entries in the current column for an incoming variable. In fact, the GFP algorithm presented takes advantage of the structure to compute these nonzero coefficients directly. By constructing a spanning tree of sorts, the initial path lengths are greatly increased and this could be more detrimental to computational effectiveness than the improvement obtained by eliminating artificial variables.

By whatever means, once the set of labels for the initial basic solution has been defined, the initial simplex multipliers can be calculated using the procedure of Section 3.2.

#### 4.2.3 Justification of the GFP Algorithm

As previously mentioned, the algorithm described in the first sections of this chapter is based on the primal simplex method of linear programming. Disregarding for a moment the question of degeneracy which will be considered in Section 4.5, the finiteness of the algorithm will follow if correspondences between the steps of the algorithm and the standard simplex method can be established.

First, at each iteration of the algorithm, the current column for a specified nonbasic variable is constructed and the standard ratio test is made to determine the departing basic variable. The maximum change in the incoming nonbasic variable and its current column are used to calculate the values of the basic variables associated with the basis after the limiting basic variable is replaced by the nonbasic variable. The labels are updated to reflect this change of basis. Hence, at each iteration a set of labels representing a basis is recorded and the corresponding values of the basic variables are maintained along with the values of the nonbasic variables. This is the same as the bounded variable simplex method.

If an artificial variable is in the final basis at a nonzero value, it is well known that the problem has no feasible solution. Otherwise, the stopping rule is the optimality criteria given in Eqs. (33) and (34) of Chapter I. Assuming nondegeneracy, no basis may be repeated and hence the algorithm terminates with the optimal solution if there is one.

#### 4.2.4 Computational Results

A computer code of the primal GFP algorithm was developed in

FORTRAN V for the UNIVAC 1108 computer. The performance of the code for solving a number of randomly generated problems is given in Table 2. The problems were generated by specifying the number of nodes and the arc density  $\alpha$ . Then each node  $i$  was connected to node  $i+1$  and to  $\alpha(n-2)$  randomly chosen additional nodes. Clearly, this will produce a connected network with arc density  $\alpha$ . The costs, nonzero coefficients, and upper bounds were chosen randomly from a uniform probability distribution over specified ranges. The percentages of source nodes and sink nodes were specified and these nodes were chosen randomly among all the nodes. The amount of supply or demand at a source or sink node was chosen from a uniform distribution over specified ranges.

Storage requirements for the code are approximately  $9n + 14m$  where  $n$  is the number of variables with nonzero coefficients in two equations and  $m$  is the number of equations. This is considerably less than for ordinary simplex algorithms which usually require at least  $mn$  storage locations. For example, a problem provided by Jensen [55] from a water resource application with  $m = 62$  and  $n = 186$  required approximately 2550 cells of storage for the GFP algorithm compared with  $mn = 11500$  and was solved in .78 second.

#### 4.3 Ordinary Flow Algorithm

The concepts used to construct the GFP algorithm can be used to devise an algorithm for the ordinary flow problem (OFP) usually solved by some version of the out-of-kilter algorithm of Ford and Fulkerson [21]. There are two primary areas of simplification in specializing the GFP algorithm for the OFP. First, because the nonzero coefficients in the

Table 2. Generalized Flow Computational Results

Problem Number	Nodes	Sources	Sinks	Arcs	Iterations	Time* (Sec)	Average Time
1	20	4	7	64	39	.152	
2	20	5	5	69	49	.125	
3**	20	4	4	51	25	.048	
4**	20	7	5	46	18	.032	
5**	20	4	8	60	41	.076	<u>.081</u>
6	50	6	19	288	315	1.019	
7	50	12	15	286	183	.433	
8	50	11	14	302	166	.380	
9	50	14	13	292	100	.208	
10	50	13	11	326	103	.253	<u>.475</u>

Table 2. (Continued)

Problem Number	Nodes	Sources	Sinks	Arcs	Iterations	Time* (Sec)	Average Time
11	75	16	25	605	371	1.158	
12	75	18	17	625	266	.630	
13	75	14	18	610	411	1.287	
14	75	19	23	639	261	.893	
15	75	14	26	599	339	1.113	<u>1.016</u>
16	100	21	32	1128	738	3.827	
17	100	16	39	1108	833	3.705	
18	100	15	31	1108	912	3.920	
19	100	18	27	1089	782	3.098	
20	100	13	31	1111	1140	5.790	<u>4.103</u>

Arc density- .1    Source density- .2    Sink density- .3

All supplies=200    Demand range 40-60    Cost range 1-100

Upper bound range 50-80    Lower bound=0    Left coefficient=1.

Right coefficient range -.8- -1.

\*Solution time on Univac 1108 in multiprocessing mode exclusive of input/output time.

\*\*No feasible solution.

constraint set are plus one and minus one, these may be specified implicitly. The second area of simplification results from the structure of the basis for the OFP and will be considered in detail in Section 4.3.2.

#### 4.3.1 OFP Algorithm Statement

To specify the coefficients for variables with two nonzero coefficients, note that one is a plus one and one a minus one. Using the previous notation, let  $x_j$  appear in equation  $p_j$  with coefficient plus one and equation  $q_j$  with coefficient minus one. In terms of the GFP notation  $a_{1j} = 1$ ,  $a_{2j} = -1$ , for  $j = 1, \dots, n$ .

The slack and artificial variables are specified logically. Let there be  $s^+$  slack arcs with a single nonzero coefficient of plus one and  $s^-$  slack arcs with a single nonzero coefficient of minus one. Thus, if  $n+1 \leq j \leq n + s^+$ , then  $a_{1j} = 1$ , and if  $n+s^+ + 1 \leq j \leq n+s^+ + s^-$ , then  $a_{1j} = -1$ .

The outline of the algorithm is the same as for the GFP and will not be repeated. The initialization procedure is the same with minor simplifications because the coefficients in the constraint set are plus one or minus one. The comments on improved initialization procedures still apply; however, there may be more incentive for using the spanning tree method since "good" algorithms are available and the only constraints neglected using the spanning tree are the bounds. Except for degeneracy, finiteness of the algorithm has been established, since it is a specialization of the GFP algorithm. Degeneracy is resolved for this problem in a relatively easy manner in Section 4.5.

### 4.3.2 Simplifications for the OFP Algorithm

It is well known and was demonstrated in Section 3.4 that the basis for the OFP can contain no cycles. This means that only three combinations of entering variable and associated basic component(s) can occur: 1) two nonzero coefficients in rows of different rooted components, 2) one nonzero coefficient in a row of one rooted component, and 3) two nonzero coefficients in rows of the same rooted component.

Cases one and two are handled in the same manner as for the GFP. Case three is covered by Lemma 4.1 below.

Let  $x_j$  be the entering nonbasic variable, with unique paths  $P_L$  and  $P_R$  from either end to the root. Let  $s$  be the node where the paths join.

Lemma 4.1

For the ordinary flow problem, if  $x_j$  has both nonzero coefficients in rows of the same component  $B_k$ , then the leaving variable corresponds to an arc in  $P_L$  or  $P_R$  above the node  $s$  where they join.

Proof:

Assume to the contrary, that is, assume the leaving variable is below node  $s$ . Then, the part of  $B_k$  above the leaving variable becomes a distinct component after the basis change with a cycle as its pseudoroot by Lemma 3.3. But this is not possible, since a cycle cannot be part of the basis for the OFP.

The procedure for constructing the column for a nonbasic variable is the same as for the GFP except that the condition for linear dependence in a cycle does not have to be checked since it has been established that any cycle corresponds to a set of linearly dependent variables.

Further, the actual column construction is greatly simplified for the OFP. It is necessary to consider only (a) whether the incoming variable is being increased or decreased, (b) the path the basic variable is in ( $P_L$  or  $P_R$ ), and (c) the orientation of the basic arc in the path. The definition of arc orientation in a path given in Section 2.3 is rephrased here.

An arc corresponding to the variable  $x_k$  is called a forward arc in a path if when tracing the path in a specified direction  $p_k$  is encountered before  $q_k$ . In other words, the equation containing the plus one coefficient is encountered before the one with a minus one. A reverse arc for  $x_k$  means that  $q_k$  is encountered before  $p_k$ .

Let the entering nonbasic variable be  $x_j$ . Define  $\delta$  by:

$$\delta = \begin{cases} +1 & \text{if } x_j = 0 \\ -1 & \text{if } x_j = M_j \end{cases} \quad (5)$$

$\delta$  indicates whether the entering variable is increasing ( $\delta = +1$ ) or decreasing ( $\delta = -1$ ). The determination of the nonzero entries in the current column for  $x_j$  is specified in the following theorem and two corollaries. These are equivalent to the rules given by Johnson in his simplex procedure for minimum cost flow in reference 58.

First assume that the two ends for the arc for  $x_j$  are in different rooted trees with coefficient  $a_{kj} = +1$  in equation  $p_j$  and coefficient  $a_{k',j} = -1$  in equation  $q_j$ .

Theorem 4.1

The nonzero entries in the current column for  $x_j$  are determined by the following rules:

1. For all basic variables  $x_i$ , corresponding to arcs between  $p_j$  and the root including the root, let  $d_i' = -\delta$  for forward arcs and  $d_i' = +\delta$  for reverse arcs.

2. For all basic variables  $x_i$ , corresponding to arcs between  $q_j$  and the root including the root, let  $d_i' = +\delta$  for forward arcs and  $d_i' = -\delta$  for reverse arcs.

Proof:

Let the coefficients for the basic variables as encountered in tracing a path be  $(a_{ki}, a_{k'i})$  as before. Consider the path  $P_L$  from  $p_j$  to the root including the root. Suppose  $P_L$  contains the variables  $(x_1, \dots, x_r)$ . By Eq. (26) of Chapter III the nonzero entries are:

$$d_1 = \frac{a_{kj}}{a_{kl}}, \quad d_i = \frac{-a_{k'i-1}}{a_{ki}} d_{i-1} \quad i = 2, \dots, r$$

or

$$d_i = a_{kj} \prod_{t=1}^{i-1} \left( \frac{-a_{k't}}{a_{kt}} \right) \frac{1}{a_{ki}} \quad i = 2, \dots, r \quad (6)$$

But  $a_{kj} = 1$  and  $\frac{a_{k't}}{a_{kt}} = -1$  for all  $t$ .

Thus:

$$d_i = \frac{1}{a_{ki}} \quad i = 1, \dots, r \quad (7)$$

By the rules of the bounded variable simplex method, the column is used directly if the nonbasic variable is increasing and the negative of the column is used if the nonbasic variable is decreasing. Thus, the column of the increasing nonbasic variable ( $x_j$  or its slack) is:

$$d'_i = \delta d_i$$

or

$$d'_i = \begin{cases} +\delta & \text{for forward arcs} \\ -\delta & \text{for reverse arcs} \end{cases} \quad (8)$$

This proves (1) of the theorem and (2) is proved in a similar manner.

The following corollaries specify the column construction for the case where the column for a nonbasic variable has both nonzero entries in rows of the same component and the case where the column for a nonbasic variable has only one nonzero entry.

Corollary: If the two nonzero coefficients of a nonbasic variable  $x_j$  are in the same basic component, then the column construction rules of the theorem are valid and only the basic variables in the paths  $P_L$  and  $P_R$  above the joining node  $s$  have nonzero entries.

Proof:

By Lemma 4.1 the basic variables in  $P_L$  to  $s$  and  $P_R$  to  $s$  and  $x_j$  form a dependent set, and thus they are the only basic variables with nonzero entries in the current column for  $x_j$ . The verification of the rules is the same as in the theorem.

Corollary: Suppose the original column for the nonbasic variable  $x_j$  has only one nonzero entry ( $a_{kj}$ ), i.e., it corresponds to a slack arc. If  $a_{kj} = 1$ , the nonzero entries in the current column for  $x_j$  are computed using (1) of the theorem and if  $a_{kj} = -1$ , the current column is computed using (2) of the theorem.

Proof:

The analogy to the theorem is straightforward.

The calculation of simplex multipliers after a basis change is also greatly simplified for the OFP. Before specifying the recalculation procedure, a lemma concerning the basis inverse for the OFP will be proved.

Lemma 4.2

For the OFP, the nonzero entries in the row of the basis inverse corresponding to the column  $b_i$  for basic variable  $x_i$  are all plus one if the arc for  $x_i$  is a reverse arc when tracing out from the root and are all minus one if the arc for  $x_i$  is a forward arc when tracing out from the root.

Proof:

Using the partitioning corollary of Section 2.5, partition the component  $B_p$  about the column  $b_i$ .

$$B_p = \left[ \begin{array}{c|c|c} B_p^1 & & 0 \\ \hline & b_i & \\ \hline 0 & & B_p'' \end{array} \right] \quad (9)$$

Assume that  $B_p^1$  contains the root unless  $b_i$  is the slack column. The row  $b^*$  of the inverse of  $B_p$  corresponding to  $b_i$  is the solution to:

$$[b_1^*, \dots, b_v^*, b_{v+1}^*, \dots, b_r^*] B_p = [0, \dots, 1, 0, \dots] = e^i \quad (10)$$

The  $b_i^*$  are the components of the required row of the inverse and  $b_v^*$  and  $b_{v+1}^*$  multiply the nonzero entries in column  $b_i$ . But this is the same as Eq. (46) of Chapter III, hence:

$$b_k^* = 0 \quad k = v+1, \dots, r \quad (11)$$

$$a_{ki} b_v^* + a_{k'i} b_{v+1}^* = 1 \quad (12)$$

For any two equations  $p_u$  and  $q_u$  corresponding to rows of  $B_p''$  with the associated elements of  $b^*$  being  $b_s^*$  and  $b_t^*$ , Eq. (52) of Chapter III must be satisfied.

$$a_{ku} b_s^* + a_{k'u} b_t^* = 0 \quad (13)$$

Where the arc for variable  $x_u$  connects nodes  $p_u$  and  $q_u$ . But:

$$b_{v+1}^* = 0 \quad \text{thus} \quad b_v^* = \frac{1}{a_{ki}} \quad (14)$$

The coefficient  $a_{ki} = -1$  if the for  $x_i$  is a forward arc and  $a_{ki} = +1$  if the arc is a reverse arc when tracing out from the root. Also from Eq. (13), if  $b_s^*$  has been calculated:

$$b_t^* = \frac{-a_{ku}}{a_{k'u}} b_s^*$$

But

$$\frac{a_{ku}}{a_{k'u}} = -1 \quad \text{for any } u.$$

Thus  $b_t^* = b_s^*$ .

This means that:

$$b_i^* = b_v^* \quad i = 1, \dots, v-1 \quad (15)$$

All nonzero entries in a row of the inverse are the same and the lemma is proved.

A theorem concerning the recalculation of simplex multipliers can now be proved using the lemma.

Let  $B_A$  be that portion of a component above the leaving variable as in the partitioning corollary following Theorem 2.1. Let  $\pi_1'$  be the value of the simplex multiplier associated with the node above the entering variable before the basis change and let  $\pi_1$  be the value of this simplex multiplier after the basis change. Then:

$$\pi_1 = \pi_1' + \Delta\pi_1 \quad (16)$$

where  $\Delta\pi_1$  is the change in the simplex multiplier because of a basis change. The following theorem specifies which simplex multipliers change and the exact amount of the change.

#### Theorem 4.2

For the OFP the only simplex multipliers which change when the basis changes are those in the component  $B_A$  above the departing variable, and these all change the same amount  $\Delta\pi_1$ .

Proof:

The first part of the theorem follows directly from Theorem 3.2 since the OFP is a special case of the GFP.

To prove the second part of the theorem, consider the  $p-1$  equations in  $p$  unknowns given by:

$$\pi B_A = c_A \quad (17)$$

$\pi$  is the vector of simplex multipliers associated with the equations of the component above the departing variable, and  $c_A$  is the vector of costs of the variables corresponding to the columns of  $B_A$ . Let  $\pi_1$ , the first component of  $\pi$ , be the simplex multiplier associated with the equation of the node above the entering variable. By adding the equation

$$\pi_1 = \pi_1' \quad (18)$$

the system (17) satisfied, before the change:

$$\pi \begin{bmatrix} 1 & \vdots \\ 0 & \vdots \\ \vdots & \vdots \end{bmatrix} B_A = [\pi_1' \ c_A] \quad (19)$$

After the basis change the system satisfies:

$$\pi_c = \begin{bmatrix} 1 & \vdots \\ 0 & \vdots \\ \vdots & \vdots \end{bmatrix} B_A = [\pi_1' + \Delta\pi_1 \ c_A] \quad (20)$$

Subtracting (19) from (20)

$$[\pi_c - \pi] \begin{bmatrix} 1 & \vdots \\ 0 & \vdots \\ \vdots & \vdots \end{bmatrix} B_A = [\Delta\pi_1 \ \underline{0}] \quad (21)$$

$$[\pi_c - \pi] = [\Delta\pi_1 \ \underline{0}] \begin{bmatrix} 1 & \vdots \\ 0 & \vdots \\ \vdots & \vdots \end{bmatrix} B_A^{-1} \quad (22)$$

Thus the change is specified by the first row of the basis inverse. But by Lemma 4.2 this row is all plus ones. Thus:

$$[\pi_c - \pi] = \Delta\pi_1 \underline{e} \quad (23)$$

And the theorem is proved.

Instead of tracing the tree above the entering variable and iteratively recomputing the simplex multipliers, the simplex multiplier  $\pi_1$  for the node above the arc of the entering variable is computed, and its change  $\Delta\pi_1$  added to all of the remaining simplex multipliers associated with  $B_A$ . Let  $\bar{c}_i'$  be the relative cost of nonbasic variable  $x_i$  after a basis change and  $\Delta\pi_1$  defined as above.

Corollary: The only relative costs  $\bar{c}_i$  which change when a basis changes are those associated with variables with one nonzero coefficient in a row corresponding to  $B_A$ . If the coefficient is minus one, then  $\bar{c}_i' = c_i + \Delta\pi_1$  and if the coefficient is plus one, then  $\bar{c}_i' = c_i - \Delta\pi_1$ .

Proof:

If a nonbasic variable has no nonzero coefficients in rows of  $B_A$ , then clearly the basis change does not affect the relative costs since the relevant simplex multipliers do not change. If a nonbasic variable has both nonzero coefficients in rows of  $B_A$ , then before the basis change:

$$\bar{c}_i = c_i - \pi_{p_i} + \pi_{q_i} \quad (24)$$

By Theorem 4.2, after the basis change:

$$\pi_{p_i}' = \pi_{p_i} + \Delta\pi_1 \quad (25)$$

$$\pi_{q_i}' = \pi_{q_i} + \Delta\pi_1$$

Thus the new relative cost is determined by:

$$\bar{c}_i = \bar{c}'_i - (\pi_{p_i} + \Delta\pi_1) + (\pi_{q_i} + \Delta\pi_r) = \bar{c}_i \quad (26)$$

If a nonbasic variable has one nonzero coefficient and it is a minus one, then either:

$$\bar{c}_i = c_i - \pi_{p_i} + \pi_{q_i} \quad (27)$$

becomes:

$$\bar{c}'_i = c_i - \pi_{p_i} + \pi_{q_i} + \Delta\pi_1 = \bar{c}_i + \Delta\pi_1 \quad (28)$$

Or for a slack:

$$\bar{c}_i = c_i + \pi_{p_i} \quad (29)$$

becomes:

$$\bar{c}'_i = c_i + \pi_{p_i} + \Delta\pi_1 = \bar{c}_i + \Delta\pi_1 \quad (30)$$

Likewise for a plus one coefficient:

$$\bar{c}'_i = c_i - \pi_{p_i} - \Delta\pi_1 + \pi_{q_i} = \bar{c}_i - \Delta\pi_1 \quad (31)$$

$$\bar{c}'_i = c_i - \pi_{p_i} - \Delta\pi_1 = \bar{c}_i - \Delta\pi_1$$

The corollary is proved.

Theorem 4.2 and the corollary are extensions of the result of Glover, Klingman, and Kearny [34] for the transportation problem.

The computational usefulness of the corollary is not straightforward, since the identification and recalculation of the current costs for the affected nonbasic variables could require more effort than is worthwhile.

### 4.3.3 Computational Results

The simplifications indicated in Section 4.3.2 were used to construct an algorithm for the ordinary flow problem. This algorithm was coded in Fortran V for the UNIVAC 1108. The results of solving a number of problems with the OFP code and the SHARE out-of-kilter code [80] are given in Table 3 and Figures 8 and 9. The problems were generated in a manner similar to that described for the GFP computational results. Both codes were run on the UNIVAC 1108 under identical conditions.

The results indicate that the use of the techniques and devices described in this paper lead to an effective solution procedure. The computational times for the OFP code are comparable to those of Glover, Klingman, and Barr [38] for their improved out-of-kilter code, although it is difficult to make comparisons since different problems were solved on different computers. Klingman, Napier, and Stutz [64] have indicated that a primal method is being developed which is better than the improved out-of-kilter code.

Since the code for the OFP algorithm was experimental and intended only to verify the effectiveness of the methods explored in this paper, we believe that better computational times may be obtained by improving the computer coding methods.

## 4.4 Primal Transportation Algorithm

### 4.4.1 Problem Statement

The ordinary flow algorithm can be specialized further when applied to the capacited transportation problem. Assume that the problem is in the form:

Table 3. OFP Computational Results

Problem Number	Nodes	Sources	Sinks	Arcs	Iterations OFP	Time-OFP* (Sec)	Time-OTK* (Sec)	Average Time
1	50	11	12	115	82	.112	.909	
2	50	20	9	128	52	.110	.459	
3	50	10	9	129	49	.055	.694	
4	50	13	9	124	70	.099	.572	
5	50	16	8	121	28	.036	.311	OFP-.082 <u>OTK-.589</u>
6	75	19	13	248	126	.168	1.478	
7	75	12	9	219	140	.243	1.368	
8	75	13	16	230	153	.205	2.080	
9	75	12	20	239	255	.452	2.995	
10	75	16	16	226	148	.218	1.138	OFP-.257 <u>OTK-1.913</u>
11	100	19	20	384	290	.565	2.941	
12	100	24	15	397	198	.438	2.510	

Table 3. (Continued)

Problem Number	Nodes	Sources	Sinks	Arcs	Iterations OFF	Time-OFF* (Sec)	Time-OTK* (Sec)	Average Time
24	165	47	24	956	351	.881	5.325	
25	165	36	35	995	833	2.236	9.771	OFF-2.047 <u>OTK-8.387</u>
26	185	33	37	1222	1303	4.271	13.710	
27	185	32	42	1189	1345	4.339	17.679	
28	185	49	34	1153	605	2.133	8.412	
29	185	43	35	1176	720	2.416	9.518	
30	185	40	45	1181	1309	5.172	14.067	OFF-3.666 <u>OTK-12.331</u>
31	200	29	43	1366	1516	5.449	20.536	
32	200	37	47	1384	1438	4.984	22.548	
33	200	34	44	1397	1419	4.972	21.509	
34	200	41	49	1393	1336	4.992	17.571	
35	200	33	37	1416	1197	3.578	14.435	OFF-4.795 <u>OTK-19.319</u>

Table 3. (Continued)

Problem Number	Nodes	Sources	Sinks	Arcs	Iterations OFF	Time-OFP* (Sec)	Time-OTK* (Sec)	Average Time
13	100	26	19	396	175	.338	2.443	
14	100	21	16	395	230	.599	2.931	
15**	100	18	17	391	285	.630	3.153	OFP-.514 <u>OTK-2.795</u>
16	150	32	31	837	685	1.980	8.894	
17	150	26	28	801	689	1.974	9.180	
18	150	29	33	794	750	2.516	8.832	
19	150	31	33	827	751	2.109	9.444	
20	150	35	32	800	671	1.915	6.922	OFP-2.098 <u>OTK-8.564</u>
21	165	38	24	977	670	2.033	8.795	
22	165	49	34	1023	634	2.413	8.964	
23	165	32	25	1019	871	2.672	9.083	

Table 3. (Continued)

Problem Number	Nodes	Sources	Sinks	Arcs	Iterations OFP	Time-OFP* (Sec)	Time-OTK* (Sec)	Average Time
36	300	68	52	2974	1872	11.925	L	
37	300	54	76	3096	3344	20.149	L	
38	300	55	54	2975	2543	15.155	L	
39	300	54	59	2966	2759	17.293	L	
40	300	56	57	3034	2729	15.506	L	<u>OFP-16.005</u>

Arc density-.03    Source density-.2    Sink density-.2

Source range 50-80    Sink range 30-50    Cost range 1-100

Upper bound range 40-80    Lower bound=0

\*Solution time on Univac 1108 in multiprocessing mode exclusive of input/output time

\*\*No feasible solution

L-Problem too large for code

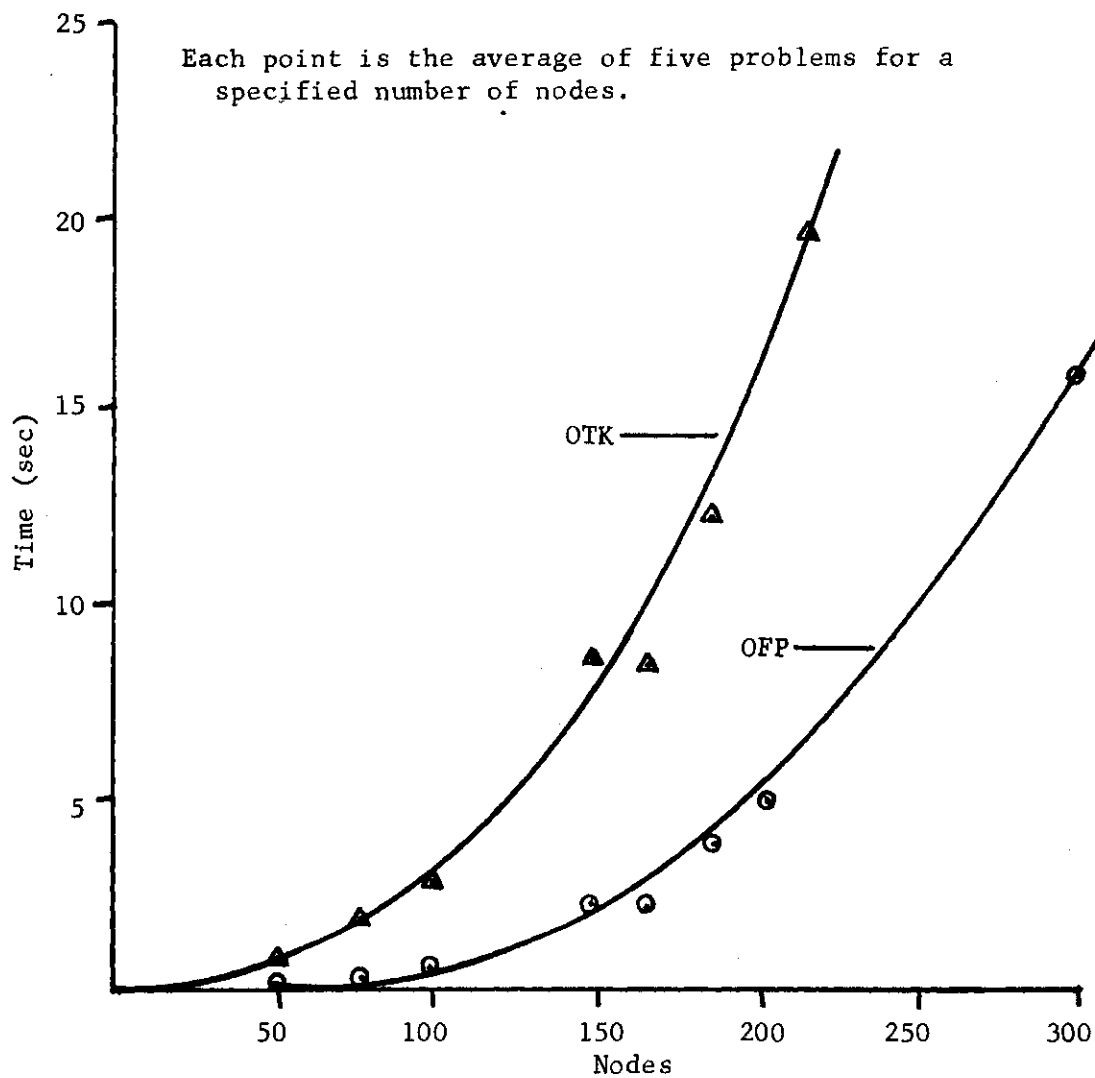


Figure 8. Comparison of OFP and SHARE OTK Algorithms (Nodes)

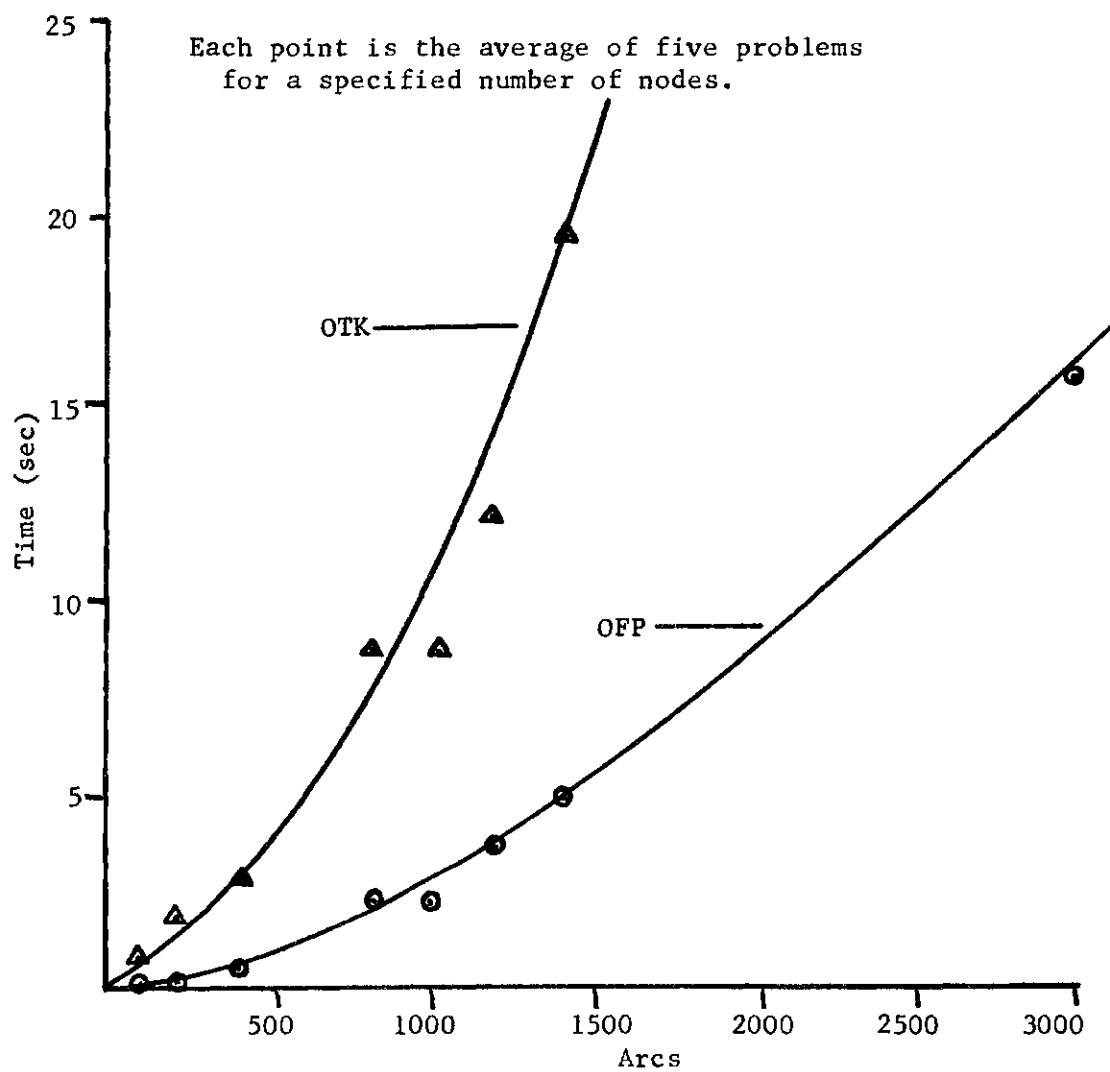


Figure 9. Comparison of OFP and SHARE OTK Algorithms (Arcs)

$$(TP) \quad \text{Minimize} \quad \sum_{i=1}^{n_s+n_t} c_i x_i \quad (32)$$

Subject to:

$$\sum_{i=(j-1)n_t+1}^{jn_t} x_i \leq a_j \quad j = 1, \dots, n_s \quad (33)$$

$$\sum_{\substack{j=i+kn_t \\ k=0, \dots, n_s-1}} -x_j = -b_i \quad i = 1, \dots, n_t \quad (34)$$

$$x_i \geq 0, a_j \geq 0, b_i \geq 0 \quad \text{for all } i \text{ and } j.$$

This can be represented as a graph with nodes  $1, 2, \dots, n_s, n_s+1, \dots, n_s+n_t$ . There is an arc starting at each node in the set  $NS = (1, \dots, n_s)$  with a plus one and ending in the set  $NT = (n_s+1, \dots, n_s+n_t)$  with a minus one. There are thus  $n_s+n_t$  equations and  $n_s*n_t$  variables each appearing in two equations. For each equation in (33) the slack variable  $x_{n_s+n_t+j} = a_j$ ,  $j = 1, \dots, n_s$  with cost  $c_{n_s+n_t+j} = 0$  are added. For each equation in (34) the artificial  $x_j = b_i$ ,  $j = 2n_s+n_t+i$ ,  $i = 1, \dots, n_t$  with cost  $c_j = R$  are added. This provides an initial starting solution with  $n_s$  slack variables and  $n_t$  artificials. This problem might be called the completely connected transportation problem. If arcs are to be excluded, they could be given the same high cost as for the artificial variables.

The structure of the transportation constraint set can be used to simplify the logic required when applying the principles of the ordinary flow algorithm. Also, since there is only one arc from any node to any other node, the node based labeling system is used to characterize the

current basis. Computationally this has advantage, since the node oriented system uses fewer operations to determine the same information as the arc based system. Experience with the GFP algorithm has shown that the code using the node labeling method is 10 to 20 percent faster than the code with the arc based system on the same problems. The savings in operations are illustrated by this example.

Suppose one is given node  $k$  and wishes to know the next node below it in the tree and the basic arc connecting the two nodes. In the arc labeling system the operations required would be:

$$(1) \quad x = k, \quad j = D_x$$

$$(2) \quad \text{If } x = p_j \text{ then } y = q_j, \text{ otherwise, } y = p_j$$

$x$ ,  $j$ , and  $y$  are the required information.

For the node labeling system there is a matrix  $N$  with elements  $N_{xy}$  such that, if arc  $j$  goes between nodes  $x$  and  $y$ , then  $N_{xy} = j$ . The same information is found by:

$$(1) \quad x = k, \quad y = D_x$$

$$(2) \quad j = N_{xy}$$

Since these types of operations are performed many times using the primal flow approach on the algorithms of this chapter, the computational savings of using the second labeling system are considerable.

There is only one slack or artificial variable associated with each node. These are identified by  $N_{k \ n_s+n_t+1} = n_s+n_t+k$ , if variable  $x_{n_s+n_t+k}$  is a slack or artificial at node  $k$ . If  $k \leq n_s$ , the variable is a slack, otherwise it is an artificial. The different labeling scheme makes it advisable to alter the methods used for the ordinary flow problem for the transportation problem. The path to the root(s) from both ends

of the entering arc are traced to determine whether the nonbasic arc connects two trees or is contained in one tree, and the two cases are considered separately.

#### 4.4.2 Simplifications for the Transportation Problem

In addition to the simplification in the logical specification of nodes and arcs already noted, one additional specialization can be used. Let  $\delta$  be defined as in Eq. (5). Let  $P_L$  and  $P_R$  be the paths from each end of the arc for the entering variable to the roots or joining node.

#### Theorem 4.3

For the ordinary transportation problem, the entries for the variables in  $P_L$  and  $P_R$  are alternately  $+\delta$  and  $-\delta$  to and including the root or to the joining node  $s$  if applicable.

Proof:

All positive coefficients appear in the equation set NS and all negative coefficients in the set NT. Starting in NS the associated arcs are alternately forward and reverse arcs starting with a forward arc, and starting in NT the arcs are alternately reverse and forward starting with a reverse arc. All slacks in  $x_s$  have a plus one coefficient and all artificials in NT have a minus one coefficient. Thus, by Theorem 4.2 this theorem is proved.

#### 4.4.3 Computational Results

The uncapacited version of the algorithm described in the previous sections was coded in Fortran V for the UNIVAC 1108 computer. The code is a direct implementation, specialization, and extension of the ideas of Johnson in [58] and [59]. It is similar to the codes of Thompson and Srinivasan [91] and Glover, Kearny, Klingman, and Napier [30,36] but was

developed independently. The original computational results obtained using artificial variables for all of the destinations in the starting solution and a first negative evaluator rule for choosing an entering variable were disappointing. The studies mentioned above reported that a better starting solution and choice rule greatly improved computation times. Using their experience, we implemented the modified row minimum start method and modified row first negative evaluator choice rule described in both [36] and [91]. The results of solving a number of 100 percent dense transportation problems with this improved version of the code are given in Table 4. These times are comparable with the times for the 1970 code of Srinivasan and Thompson [91], but are not as good as those claimed for their 1972 code or those for the code of Glover, Kearny, Klingman, and Napier in [36].

A modification was made to the triple labeling scheme of Johnson to reduce the storage requirements. In the node oriented labeling scheme in Section 3.6, a matrix entry  $N_{xy}$  was used to determine the arc between nodes  $x$  and  $y$ . For a transportation with  $n_s$  sources and  $n_t$  destination, this matrix  $N$  required  $(n_s + n_t)^2$  storage locations. If a fourth label  $A_x$  is used to identify the unique basic arc directly below node  $x$  in the basic tree, then the matrix  $N$  is not required. For example, if it is known that node  $x$  is above node  $y$  in the tree, then  $A_x$  is the basic arc which connects them. These labels are updated in the same manner as the down labels. In [36] it is mentioned that the code of Srinivasan and Thompson may be used only on 100 percent dense problems. With the additional core storage requirement of  $n_s + n_t$  plus the number of arcs, the method described here may be used to solve problems of any density in a manner similar to that described for the OFP code in the previous section.

Table 4. Transportation Computational Results\*

Problem <sub>1</sub> Set No.	Problem Size	Average Iterations	Average Time(Sec) <sup>2</sup>	Constraint Factor <sup>3</sup>
1	100 x 100	551	4.440	0 - .1
2	100 x 100	487	6.768	.1 - .2
3	100 x 100	301	4.342	.2 - .5
4	120 x 120	622	6.609	0 - .1
5	120 x 120	508	6.961	.1 - .2
6	120 x 120	316	5.516	.2 - .5
7	140 x 140	809	7.988	0 - .1
8	140 x 140	664	8.497	.1 - .2
9	140 x 140	329	4.667	.2 - .5
10	160 x 160	895	11.050	0 - .1
11	160 x 160	695	12.602	.1 - .2
12	160 x 160	369	8.629	.2 - .5

\*The results in this table were obtained by J. L. Kennington using a code developed using the methods described in this section.

<sup>1</sup>Each problem set consists of five problems.

<sup>2</sup>Solution time on Univac 1108 in multiprocessing mode exclusive of input/output time.

<sup>3</sup>The constraint factor is the excess of total supply ( $\sum S_1$ ) over total demand ( $\sum T_j$ ) divided by total supply.

$$\text{Constraint Factor} = \frac{\sum S_1 - \sum T_j}{\sum S_1}$$

#### 4.5 Resolution of Degeneracy

If the problem is not degenerate, a simplex based algorithm is finite since only basic solutions are considered which are not repeated. It remains to be shown that, when the problem is degenerate, there can be only a finite number of basis changes between decreases in the objective function. The random choice rule of Dantzig [15] or the simple rule of Azpeitia and Dickinson [2] involving choosing the variable which has been in the basis the least number of times might be adopted for theoretical completeness but neither of these has any direct relation to the problem at hand. The approach taken here will be to use a more complicated rule due to Dantzig [15] and show that for many of the conditions which can occur for a degenerate basis change this rule has a simple interpretation in terms of the basis graph. Dantzig's rule evolves from the following development.

Suppose that the right hand side of the starting tableau for the GFP be replaced by the set of  $m+1$  components row vectors  $\underline{b}_i = (b_i \ \underline{e}_i)$ ,  $i = 1, \dots, m$  where  $\underline{e}_i$  is an  $m$ -dimensional unit vector. These vectors are lexicographically positive since it is assumed  $b_i \geq 0$  for each  $i$ .

Then the vector valued variables on some subsequent iteration are given by:

$$\bar{\underline{b}}_i = [\bar{b}_i \ \beta_{i1} \ \beta_{i2}, \dots, \beta_{im}] \quad i = 1, \dots, m \quad (35)$$

where  $\underline{\beta}_i$  is the  $i^{\text{th}}$  row of the basis inverse.

If the dropping variable is chosen so that:

$$\frac{b_i}{d_{rs}} = \text{lexicomin} \left( \frac{\bar{b}_i}{\bar{d}_{is}} \right) \quad (36)$$

The corresponding vector valued objective function is lexicographically decreasing and hence finiteness is established.

To see how this rule specializes for the GFP, the various cases for basis changes will be considered. First, suppose that a nonbasic variable  $x_j$  is entering the basis and its corresponding arc is between two basis components both with roots. Define the component containing  $p_j$  as Tree One and the one containing  $q_j$  as Tree Two. Suppose that there is a tie between two basic variables  $x_s, x_t$  in the path from  $p_j$  to the root of Tree One. To use the rule of Dantzig, the rows of the inverse for these two variables are required.

In Chapter III, a method was presented showing that an iterative scheme could be used to solve for the row of the basis inverse corresponding to any basic variable. This involved solving an equation of the type  $\underline{e}_i B = \underline{e}_i$  where  $\underline{e}_i$  is the  $i^{\text{th}}$  unit vector when the  $i^{\text{th}}$  row of the inverse  $B$  is the submatrix corresponding to the tree containing  $x_i$ .  $B$  is lower triangular and all entries of  $\underline{e}_i$  are equal to zero until the  $i^{\text{th}}$  one. In terms of the graph, for variable  $x_i$  the entries of the  $i^{\text{th}}$  row of the basis inverse are all zero except corresponding to nodes above it in the basic tree. Let  $d_{ik}$  be the element in the row of the basic variable  $x_i$  for the nonbasic column for  $x_k$ . Thus, if  $x_s$  and  $x_t$  in the same tree, the rule of Dantzig says choose  $x_s$  if  $\frac{d_{ti}}{a_{k't}} > 0$  and choose  $x_t$  if  $\frac{d_{ti}}{a_{k't}} < 0$ . This follows since:

$$\frac{\bar{b}_s}{d_{si}} = \frac{\bar{b}_t}{d_{ti}} \quad \text{and} \quad T = \{i : \beta_{ti} = 0\} \subset S = \{i : \beta_{si} = 0\} \quad (37)$$

$T$  is a proper subset of  $S$ . Thus, when the first nonzero element  $\beta_{ti} \neq 0$  is encountered,  $\beta_{si} = 0$ . Thus, if  $\frac{\beta_{ti}}{d_{ti}} > 0$ , then  $\frac{\bar{b}_s}{d_{si}}$  is lexicographically smaller than  $\frac{\bar{b}_i}{d_{ti}}$  and if  $\frac{\beta_{ti}}{d_{ti}} < 0$ , the converse is true.

But the first nonzero element in  $\beta_t$  encountered is the diagonal element and it can be determined easily. Suppose that node  $r$  is below the basic arc  $x_t$  and node  $t$  is above it. The equation to determine the element  $\beta_{tt}$  is

$$a_{k't} \beta_{rt} + a_{k't} \beta_{tt} = 1 \quad (38)$$

But  $\beta_{rt} = 0$  since  $r$  is below  $x_t$ , then  $\beta_{tt} = \frac{1}{a_{k't}}$ . Thus, if  $a_{k't} > 0$ , then  $\beta_{tt} > 0$  and if  $a_{k't} < 0$ , then  $\beta_{tt} < 0$ .

The sign of the required element is determined by the coefficient of the basic arc generating the row next to the node away from the root and the sign of  $d_{ti}$  which is plus if  $x_t$  is decreasing and minus if  $x_t$  is increasing. The lexicographic rule of Dantzig can then be stated as follows.

#### Theorem 4.4

Suppose that  $x_s$  and  $x_t$  are two basic variables in the path from the entering nonbasic variable to the root which are tied for the minimum ratio to determine the blocking variable. Also suppose  $x_s$  is above  $x_t$  in the tree and let  $a_{k't}$  be the coefficient associated with  $x_t$  in the equation corresponding to the node directly above  $x_t$  in the tree. Then the

rule dropping  $x_s$  if  $\frac{a_{kt}}{d_{ti}} > 0$  and dropping  $x_t$  if  $\frac{a_{kt}}{d_{ti}} < 0$  maintains the lexicographic order which by Dantzig's proof insures finiteness.

Now suppose that the two variables  $x_t$  and  $x_s$  which are tied are in the paths between the entering variable and the root or pseudoroot in different trees, i.e.,  $x_t$  is in Tree One and  $x_s$  is in Tree Two. The submatrix for these two components is lower triangular and in block form.

$$\begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \quad (39)$$

Thus, the row of the inverse corresponding to  $x_t$  will have the first non-zero element by the same reasoning as before. The lexicographic rule for this case becomes:

Corollary: Suppose  $x_s$  and  $x_t$  are two basic variables with  $x_t$  in the path between an entering arc and the root in one tree and  $x_s$  is in a similar path in another tree. Assume that  $x_s$  and  $x_t$  are tied for leaving variable by the standard ratio test and that  $a_{k't}$  is the coefficient for  $x_t$  in the equation corresponding to the node above  $x_t$  in its tree. Then the rule which drops  $x_s$  if  $\frac{a_{kt}}{d_{ti}} > 0$  and drops  $x_t$  if  $\frac{a_{kt}}{d_{si}} < 0$  preserves the lexicographic order required for finiteness.

Proof:

The proof follows directly from the theorem.

The case where one of the dropping variables is a root instead of a variable in the path is the same as if the root variable is treated as a path variable. This follows directly from the method of generating the row of the inverse corresponding to the root.

If both ends of the entering arc are in the same rooted tree, then the path from each end to the root must join at some node  $n$  above the root. If  $x_s$  and  $x_t$  are tied for leaving variable and  $x_s$  is above  $n$  and  $x_t$  is below  $n$ , then the rule of Theorem 4.4 applies. If  $x_s$  and  $x_t$  are both above  $n$ , call the branch containing  $x_s$  branch one and the branch containing  $x_t$ , branch two. Let  $B_1$  correspond to the part of the basis above  $x_s$  and  $B_2$  the part above  $x_t$  with  $B'$  being the remainder.

$$B_t = \left[ \begin{array}{c|c|c|c|c} B' & a_{1s} & a_{1t} & 0 & 0 \\ \hline 0 & a_{2s} & 0 & B_1 & 0 \\ \hline 0 & 0 & a_{2t} & & B_2 \end{array} \right] \quad (40)$$

$$\underline{b}_s = \begin{bmatrix} 0 \\ \vdots \\ a_{1s} \\ 0 \\ \vdots \\ a_{2s} \end{bmatrix} \quad \underline{b}_t = \begin{bmatrix} 0 \\ \vdots \\ a_{1t} \\ 0 \\ \vdots \\ a_{2t} \end{bmatrix}$$

But this case reverts to the same case as in the Corollary to Theorem 4.6 where branch one is identified with Tree One and branch two is identified with Tree Two.

The cases covered so far are all that can occur for the ordinary flow problem since either a slack arc enters a rooted tree, or the nonbasic entering arc connects two trees or has both ends in the same tree. Thus, the rules above are sufficient to ensure nondegeneracy in the ordinary flow algorithm. They are restated below in terms of forward and reverse

arcs and increasing and decreasing the basic variable. Define Tree One to be the left tree and Tree Two to be the right tree in the case of two rooted components with left and right defined as in the statement of the algorithm. Assume analogous definitions for left and right branches.

$\Delta$ Nonbasic Entering Variable	$x_s$	$x_t$	Variable to Be Dropped
Increase	Above t in left tree	Forward	s
		Reverse	t
	Above t in right tree	Forward	t
		Reverse	s
Decrease	Above t in left tree	Forward	t
		Reverse	s
	Above t in right tree	Forward	s
		Reverse	t
Increase	Above n in same tree with arc t below	Forward	s
		Reverse	t
Decrease	With t below	Forward	t
		Reverse	s
Increase	Right branch with t in left	Forward	s
		Reverse	t
Decrease	Right branch with t in left	Forward	t
		Reverse	s

The rules in this chart could be implemented computationally, but it does not seem necessary since an example of cycling has not been found in practice. These rules are consistent with the rules of Maier [68] for the maximum flow problem using spanning trees.

For the GFP, consideration must be made for the possibility of a

tie occurring between a variable in a pseudoroot and some other variable. In this case there seems to be no simple interpretation of Dantzig's rule and the rows of the inverse must be generated and the standard test made. If one of the variables is in the pseudoroot and the other not, then only the row for the variable in the pseudoroot need be generated since it will have a nonzero element before the other variable does. If this were to be done in practice, the rows could be generated as previously indicated, circumventing the need for explicitly maintaining the basis inverse. Also, since the determination of the minimum ratio is done in a pairwise fashion, the transitivity of lexicographic ordering allows the resolution of ties to be made in a similar manner.

#### 4.6 Summary

Algorithms have been presented for the generalized flow problem, the ordinary flow problem, and the capacitated transportation problem based on the row and column generation techniques given in Chapter III. These algorithms take advantage of the structure of the problems to minimize the amount of computer storage required and to efficiently carry out the steps required in the operations of the primal simplex method. The computational results indicate that this is a powerful method for solving these problems. Dual or primal-dual algorithm may be developed which take advantage of the structure of these problems in a similar manner. It is conjectured that a dual algorithm may not be as effective for these problems, particularly when there are many more variables than equations.

In this case a larger number of quantities will have to be computed to determine the entering variable, and roughly the same computation

will be required to update the basic solution, the labels, and the simplex multipliers. This conjecture seems to be supported by the previously cited results of Klingman, Glover, etc. for the transportation problem. The effectiveness of a primal-dual algorithm for network problems using the concepts developed here should be investigated and is the subject of a separate study.

## CHAPTER V

### INTEGER GENERALIZED FLOW PROBLEM CHARACTERISTICS

#### 5.1 Introduction

In Section 1.9.3 it was pointed out that the group theoretic formulation of an integer programming problem most directly displays the interaction between the constraints of the underlying linear programming problem and the requirement that the variables be integer valued. In Chapters II and III characteristics of the GFP problem have been derived and a procedure has been given for solving it. The fundamental concept used to solve the GFP has been a graphical representation of its basic solutions, to identify interactions between basic and non-basic variables, and to limit the operations of the primal simplex method to only the relevant variables. If the IGFP is formulated in group theoretic terms, interactions between variables may be characterized using the graphical representation in a manner analogous to that which was done for the continuous problem.

In Section 5.2 a means of formulating the IGFP as a group theoretic integer problem is given which uses the information available when the optimal solution to the associated GFP is obtained with the algorithm given in Chapter IV. The graphical representation of the optimal GFP solution is used to identify interactions between the basic and nonbasic solutions in terms of satisfying the integrality requirement in Section 5.3. The algorithm of Estabrook for the integer generalized transporta-

tion problem, a special case of the IGFP, is discussed in Section 5.4. Finally, a summary of methods for obtaining penalties to be used in branch and bound integer programming is given in Section 5.5. Johnson's group theoretic method for the construction of inequalities from the group problems, a row at a time, is given as a means of obtaining penalties. This method is attractive since the results reported in Johnson and Spielberg indicate that the penalties are good, and the implementation of Johnson's procedure is particularly compatible with the methods used to solve the GFP.

## 5.2 IGFP Group Formulation

The statement of the general integer programming problem in terms of the optimal linear programming basis  $B$  is restated here:

$$\text{Minimize: } c_B B^{-1}b + (c_N - c_B B^{-1}A_N)x_N \quad (1)$$

Subject to:

$$x_B = B^{-1}b - (B^{-1}A_N)x_N \quad (2)$$

$$x_B, x_N \geq 0 \text{ and integer} \quad (3)$$

The statement of the problem in this manner assumes that the simple upper bound constraints have been considered explicitly. (Assume that all variables have lower bounds of zero and upper bounds  $M_j$ .) The standard group formulation can be obtained from the upper bounded tableau with some manipulation.

If a variable is at its upper bound, then in the expanded tableau

it is basic and the slack associated with the upper bound constraint is nonbasic at zero. The explicit upper bound constraint for  $x_j$  is:

$$x_j + s_j = M_j \quad \text{or} \quad x_j = M_j - s_j \quad (4)$$

All of the information needed to formulate the group IP problem is contained in the reduced upper bounded simplex tableau. The following development shows the transformation.

The extended tableau for an upper bounded linear programming problem is:

(P1)

$$\text{Minimize:} \quad cx \quad (5)$$

$$\text{Subject to:} \quad Ax = b \quad (6)$$

$$I_n x + I_n s = M \quad (7)$$

$$x, s \geq 0 \quad (8)$$

All elements in  $A$ ,  $b$ ,  $c$ , and  $M$  are integer.

The  $x$  vector is partitioned into  $x_B$ , the basic variables in the optimal upper bounded simplex problem,  $x_u$ , the nonbasic variables at their upper bound in the optimal upper bound simplex problem, and  $x_N$ , the nonbasic variables at zero in the optimal tableau. The corresponding slack variables are  $s_B$ ,  $s_u$ ,  $s_N$ . The variables  $x_B$  and  $s_B$  are  $m$  component vectors,  $x_u$  and  $s_u$  are  $p$  vectors and  $x_N$  and  $s_N$  are  $n-m-p$  vectors. The columns of the  $A$  matrix are partitioned correspondingly into  $B$ ,  $A_u$ , and  $A_N$ . Problem P1 can be rewritten as:

$$\text{Minimize:} \quad c_B x_B + c_u x_u + c_N x_N \quad (9)$$

Subject to:

(P2)

$$\begin{bmatrix} B & A_u & A_N & 0 & 0 & 0 \\ I_n & 0 & 0 & I_m & 0 & 0 \\ 0 & I_p & 0 & 0 & I_p & 0 \\ 0 & 0 & I_{n-m-p} & 0 & 0 & I_{n-m-p} \end{bmatrix} \begin{bmatrix} x_B \\ x_u \\ x_M \\ s_B \\ s_u \\ s_N \end{bmatrix} = \begin{bmatrix} b \\ M_u \\ M_N \end{bmatrix} \quad (10)$$

$$x_B, x_u, x_N, s_B, s_u, s_N \geq 0 \quad (11)$$

The optimal basis is:

$$B' = \begin{bmatrix} B & A_u & 0 & 0 \\ I_m & 0 & I_m & 0 \\ 0 & I_p & 0 & 0 \\ 0 & 0 & 0 & I_{n-m-p} \end{bmatrix} \quad (12)$$

The inverse of the optimal basis is:

$$B'^{-1} = \begin{bmatrix} B^{-1} & 0 & -B^{-1}A_u & 0 \\ 0 & 0 & I_p & 0 \\ -B^{-1} & I_n & B^{-1}A_u & 0 \\ 0 & 0 & 0 & I_{n-m-p} \end{bmatrix} \quad (13)$$

Multiplying the constraint equations of P2 by the inverse of the optimal basis results in:

$$\begin{bmatrix} x_B \\ x_u \\ s_B \\ s_N \end{bmatrix} + \begin{bmatrix} B^{-1}A_N & -B^{-1}A_u \\ 0 & I_{p-1} \\ -B^{-1}A_N & B^{-1}A_u \\ I_{n-m-p} & 0 \end{bmatrix} \begin{bmatrix} x_N \\ s_u \end{bmatrix} = \begin{bmatrix} B^{-1}b - B^{-1}A_u M_u \\ M_u \\ M_B - B^{-1}b + B^{-1}A_u M_u \\ M_N \end{bmatrix} \quad (14)$$

The values of the basic variables are substituted into the objective function as in Eqs. (1-3), and the equivalent integer programming problem is:

$$\begin{aligned} \text{Minimize:} \quad & c_B(B^{-1}b - B^{-1}A_u M_u) + c_u M_u \\ & + (c_N - c_B B^{-1}A_N)x_N + (c_B B^{-1}A_u - c_u)s_u \end{aligned} \quad (15)$$

Subject to:

(IP1)

$$\begin{bmatrix} x_B \\ x_u \\ s_B \\ s_u \end{bmatrix} = \begin{bmatrix} B^{-1}b - B^{-1}A_u M_u \\ M_u \\ M_B - B^{-1}b + B^{-1}A_u M_u \\ M_N \end{bmatrix} - \begin{bmatrix} B^{-1}A_N & -B^{-1}A_u \\ -B^{-1}A_N & -B^{-1}A_u \\ I_{p-1} \\ I_{n-m-p} \end{bmatrix} \begin{bmatrix} x_N \\ s_u \end{bmatrix} \quad (16)$$

$$x_B, x_u, x_N, s_B, s_u, s_N \geq 0 \quad \text{and integer} \quad (17)$$

Since  $M$  has integer components, from Eq. (6) if  $x$  is integer than  $s$  will be integer also and vice versa. We will therefore explicitly require  $x_N$  and  $s_u$  to be integers and impose integrality on  $x_B$  by the congruency relationship Eq. (20) below. Relaxing the nonnegativity constraints on  $x_B, s_B, x_u$ , and  $s_N$ , the group problem then becomes:

$$\text{Minimize} \quad (c_N - c_N B^{-1} A_N) x_N + (c_B B^{-1} A_u - c_u) s_u \quad (19)$$

Subject to:

$$[B^{-1} A_N \quad -B^{-1} A_u] \begin{bmatrix} x_N \\ s_u \end{bmatrix} \equiv B^{-1} b - B^{-1} A_u M_u \quad (20)$$

$$x_N, s_u \geq 0 \text{ and integer.} \quad (21)$$

The optimality conditions of the upper bounded simplex method require that:

$$c_N - c_N B^{-1} A_N \geq 0 \quad (22)$$

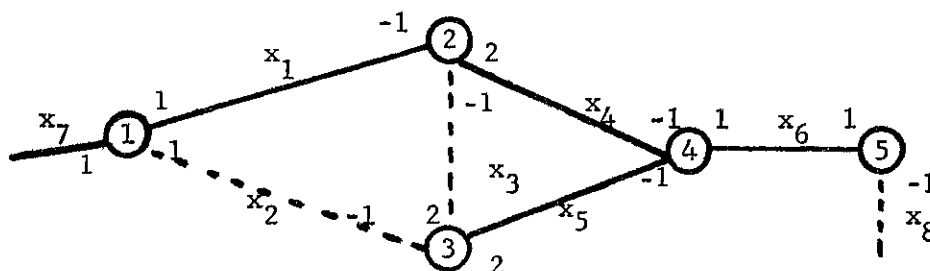
$$c_u - c_B B^{-1} A_u \leq 0 \quad (23)$$

Hence, all of the coefficients in the objective function are positive. The constraint coefficients are obtained for the nonbasic variables at zero directly from the final tableau of the upper bounded simplex methods, and the coefficients for the slack variables  $s_u$  are the negative of the ones for the corresponding variables at these upper bounds in the optimal tableau.

The algorithm for the GFP given in Chapter IV does not maintain the tableau used in the formulation of the group integer programming problem. In fact, the basis inverse is not directly available. However, either the column generation method or the row generation method given in Chapter III can be used to construct the required tableau. The row generation method is used below to illustrate this construction.

The graphical representation of the optimal LP solution to the

example problem given in Chapter II is:



The nonbasic variables are  $x_3$  and  $x_8$  at their lower bounds and  $x_2$  at its upper bound. To generate the full tableau, the row for the basic variable which is the root will be generated. Then proceeding along the tree, away from the root, the row for each basic variable in turn is generated.

The row of the basic inverse for  $x_7$  is:

$$b_1 = 1$$

$$b_2 = b_1 = 1$$

$$b_4 = 2b_2 = 2$$

$$b_3 = b_4/2 = 1$$

$$b_5 = -b_4 = -2$$

The entry for the nonbasic  $x_3$  is:

$$-1(b_2) + 2b_3 = -1 + 2 = 1$$

For the slack corresponding to  $x_2$ , it is the negative of:

$$1b_1 - 1b_3 = 1 - 1 = 0$$

For  $x_8$ :

$$-1b_5 = 2$$

The row of the basis inverse for  $x_1$  is:

$$b'_i = [0 \quad -1 \quad -1 \quad -2 \quad 2]$$

The coefficients for the nonbasic variable are:

$$x_3 : -1b'_2 + 2b'_3 = -1$$

$$s_2 : -(1b_1 - 1b_3) = -1$$

$$x_8 : -1b_5 = -2$$

Similarly, using the inverse of the row for  $x_4$ :

$$x : -1$$

$$s_2 : -\frac{1}{2}$$

$$x_8 : -1$$

For  $x_5$ :

$$x_3 : 1$$

$$s_2 : \frac{1}{2}$$

$$x_8 : 0$$

For  $x_6$ :

$$x_3 : 0$$

$$s_2 : 0$$

$$x_8 : -1$$

The updated costs are determined for the nonbasic variables:

$$\bar{c} \text{ for } x_3 : c_3 - 2\pi_3 + \pi_2 = 2 - 2(-4) + (-3) = 7$$

$$\bar{c} \text{ for } s_2 : -(c_2 - \pi_1 + \pi_3) = -(2 - 0 + (-4)) = 2$$

$$\bar{c} \text{ for } x_8 : c_8 + \pi_5 = 0 + 11 = 11$$

The integer programming problem written in terms of the final LP tableau is:

$$\text{Minimize} \quad 7x_3 + 2S_2 + 11x_8 \quad (24)$$

Subject to:

$$\begin{bmatrix} x_7 \\ x_1 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = - \begin{bmatrix} 1 & 0 & 2 \\ -1 & -1 & -2 \\ 1 & -\frac{1}{2} & -1 \\ 1 & \frac{1}{2} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_3 \\ S_2 \\ x_8 \end{bmatrix} + \begin{bmatrix} 10 \\ 5 \\ 5/2 \\ 5/2 \\ 5 \end{bmatrix} \quad (25)$$

$$\begin{aligned} x_i &\geq 0 \quad \text{for all } i \\ S_2 &\geq 0 \quad \text{and all variables integer} \end{aligned} \quad (26)$$

By dropping the nonnegativity constraints on the basic variables and taking fractional parts, the group problem is obtained.

$$\text{Minimize} \quad 7x_3 + 2S_2 + 11x_8 \quad (27)$$

Subject to:

$$\begin{bmatrix} 0 \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ S_2 \\ x_8 \end{bmatrix} \quad (28)$$

$$x_3, S_2, x_8 \geq 0 \quad \text{and integer} \quad (29)$$

The above illustrates the formulation of the group problem using the final upper bounded simplex tableau. It also illustrates the method of constructing the required information when the GFP algorithm has been

used to solve the upper bounded generalized flow problem.

### 5.3 Determinant Calculation

In general, the determinant of the final basis of the linear programming problem is a measure of the difficulty of the associated group problem. This is because the number of elements in the group is equal to or less than  $\text{Det}(B)$ . The method for calculating the determinant of the basis for a GFP is much simpler than for general linear programming problems.

In Theorem 2.1 it was shown that any basis for the GFP can be made block diagonal with blocks  $B_1, \dots, B_p$ . It is well known that the absolute value of the determinant of such a matrix is the product of the determinants of the individual blocks.

$$|\text{Det}(B)| = \prod_{i=1}^p |\text{Det}(B_i)| \quad (30)$$

Theorem 2.1 further established that the blocks were of two types: either having a column with a single nonzero entry or having a cycle. The following theorem shows the method of calculating the determinant for a block with a slack column.

Let  $a_{k,1}$  be the nonzero entry in the slack column for the variable  $x_1$ . Without loss of generality, let the variables encountered when tracing the tree above the root be in order  $x_2, \dots, x_r$  with coefficients  $(a_{k,i}, a_{k',i})$  with  $a_{k,i}$  in the equation nearer the root.

Theorem 5.1

The determinant of a block with a slack column is given by:

$$\text{Det}(B_k) = a_{k'1} \prod_{i=2}^r a_{k'i} \quad (31)$$

Proof:

Consider the arrangement of the block  $B_k$  by placing the columns in the order  $b_1, \dots, b_r$  with  $b_j$  associated with  $x_j$ . The tracing procedure implies that the new column will have one of its nonzero entries in common with a column already placed in the block while no column already in the block has a nonzero entry in common with the second nonzero entry. Consequently, the coefficient in each column in the equation farthest from the root is placed along the diagonal and the other is placed in the appropriate row above the diagonal. The resulting upper triangular matrix is:

$$B_k = \begin{bmatrix} a_{k1} & a_{k2} & & & \\ & a_{k'2} & & & \\ & & \ddots & & \\ & & & a_{k'r-1} & a_{kr} \\ & & & & a_{k'r} \end{bmatrix} \quad (32)$$

The determinant of this matrix is the product of the diagonal elements, hence the theorem is proved.

If block  $B_k$  contains a pseudoroot, it may be arranged in the following manner. Let the first  $r$  columns of  $B_k$  be the columns corresponding to the pseudoroot with the rows arranged so that the nonzero elements are in the first  $r$  rows of  $B_k$  defining block  $B_R$ . For each variable  $x_j$ ,  $j = r+1, \dots, p$  whose arc is incident on the pseudoroot, consider the



to the cycle as a root for those above it, the block arrangement below  $B_1$  follows from the same argument given for a rooted component. This leads to the following corollary.

Corollary: The absolute value of the determinant for a pseudorooted component of the basis  $B_k$  is:

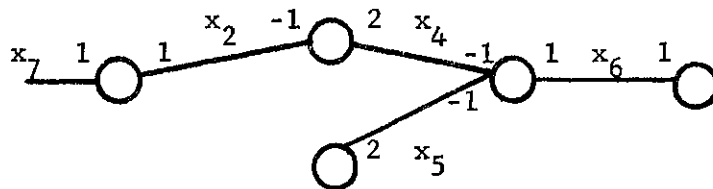
$$|\text{Det} B_k| = \left| \prod_{i=1}^r a_{ki} - (-1)^r \prod_{i=1}^r a_{k'i} \right| \left| \prod_{u=1}^P \left( \prod_{v=1}^{d_i} a_{k'j_v} \right) \right| \quad (34)$$

Proof:

That the determinant of a block diagonal matrix is the product of the determinants of the blocks along the diagonal is well known. The first expression in the equation is the determinant of a cycle given in Eq. (17) Chapter III. Each of the blocks  $B_{D_j}$  along the diagonal has the same form as in the theorem and hence the determinant is the product of the coefficients farthest from the pseudoroot when tracing out the tree.

The above theorem and corollary were proved in a different manner by Estabrook [20] for the GTP as his Theorem 2.5. The theorem and corollary can be used to calculate the determinant for each component of the basis and the absolute value of the product of the individual components gives the absolute value of the total determinant as required.

As an example of the computation of the determinant for a rooted component, consider the problem used in the last section to construct the nonbasic columns. The associated graph with the coefficients is:



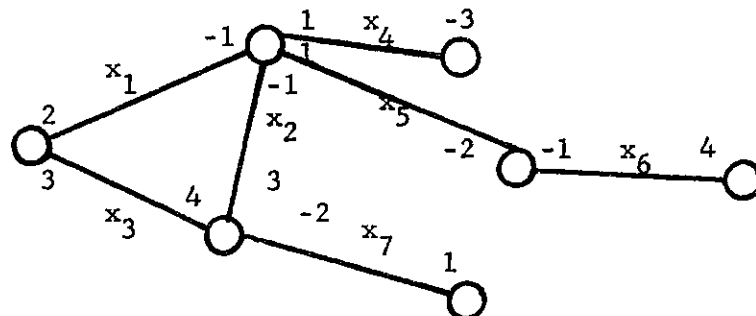
The sequence of basic variables tracing out from the root is  $(x_7, x_2, x_4, x_6, x_5)$ . The matrix of associated columns is:

$$B = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & -1 & 2 & & \\ & & -1 & 1 & -1 \\ & & & 1 & \\ & & & & 2 \end{bmatrix}$$

The absolute value of the determinant is:

$$|\text{Det } B| = |a_{17} \ a_{22} \ a_{24} \ a_{15} \ a_{26}| = |1(-1)(-1)(2)(1)| = 2$$

The graphical representation of an example with a pseudoroot is:



The sequence of basic variables generated by tracing around the pseudoroot, then out from each node in the pseudoroot is  $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ .

The matrix of associated columns is:

$$B = \begin{bmatrix} 2 & & 3 & & & & \\ -1 & -1 & & 1 & 1 & & \\ & 3 & 4 & & & -2 & \\ & & & -3 & & & \\ & & & & -2 & -1 & \\ 0 & & & & & 4 & \\ & & & & & & 1 \end{bmatrix} = \begin{bmatrix} B_R & & B_1 & & \\ & B_{D_1} & & & \\ & & B_{D_2} & & \\ & & & B_{D_3} & \end{bmatrix} \quad (35)$$

The absolute value of the determinant of the submatrix corresponding to

the pseudoroot is:

$$\begin{aligned} |\text{Det } B_R| &= |a_{11} \ a_{22} \ a_{23} \ - \ (-1)^3 \ a_{21} \ a_{12} \ a_{13}| \\ &= |(2)(-1)(4) + (-1)(3)(3)| = 7 \end{aligned}$$

For each block  $B_{D_i}$ :

$$|\text{Det } B_{D_1}| = |a_{24}| = 3$$

$$|\text{Det } B_{D_2}| = |a_{25} \ a_{16}| = (-2)(4) = 8$$

$$\text{Det } B_{D_3} = a_{17} = 1$$

The absolute value of the determinant for the component is:

$$\text{Det } B = \text{Det } B_R \text{Det } B_{D_1} \text{Det } B_{D_2} \text{Det } B_{D_3} = 408$$

Hu [54, Chapter 19] summarizes some work of Gomory concerning the relationship of the value of the determinant to the group formulation. The magnitude of the determinant of the optimal LP basis is meaningful both theoretically and computationally when considering the group integer problem.

Estabrook [20] points out that a large determinant of the optimal LP basis can cause numerical difficulties in obtaining the group representation of the problem. He then gives a method for the GTP for obtaining an equivalent representation of the group problem which may avoid these numerical difficulties. His technique is extended to the GFP below and the method for obtaining the new representation is refined. In Section 5.5 a problem involving a single row from the set of Eqs. (20)

is discussed. The device described below provides insight into the computational effort required to solve this single row cyclic group problem.

Suppose that the optimal LP basis is only one component B. Assume that B contains a slack column and that the determinant of B is D. The inverse of B can be expressed as:

$$B^{-1} = \frac{1}{D} N = \begin{bmatrix} n_{1j} \\ m_{1j} \end{bmatrix} \quad (36)$$

where  $n_{ij}$  and  $m_{ij}$  are integers.

The matrix N is called the adjoint matrix and is all integer. The coefficients of the nonbasic variables in the optimal tableau are given by:

$$B^{-1}A_N = \frac{1}{D} NA_N \quad (37)$$

Since N and  $A_N$  are integer matrices, all of the coefficients can be expressed as fractions with denominator D. The number of elements in the group generated by the columns of  $B^{-1}A_N$  is equal to or less than D and the amount of effort required for the usual techniques for solving the group IP problem Eqs. (1-3) can be expressed as a function of D. Suppose that there is a common factor d of the nonzero elements of the adjoint matrix N. Then N can be expressed as:

$$N = dN' \quad (38)$$

where  $N'$  is also an all integer matrix. If the quantity d is an integer greater than one and if d divides D, then the denominator of all the elements in  $B^{-1}A_N$  can be reduced by the factor d and numerical difficulties

caused by large  $D$  in representing Eq. (37) may be avoided. In addition, the coefficients in any row of Eq. (37) may now be considered as a set of fractions over the smaller common denominator  $D/d$ .

A method for determining such a factor  $d$  for the IGFP is shown in the following lemma and theorem. Let  $b$  be a column of  $B^{-1}$ . The column  $b$  can be generated using the procedure given in Section 3.3. Without loss of generality let the basic variables whose columns are used to generate  $b$  be in order  $x_r, \dots, x_1$ . As described in Section 3.3, these variables correspond to a path in the graph associated with  $B$  and  $x_1$  is the variable of the slack arc at the root. Let  $B$  be partitioned into  $B_p$  and  $B'_p$  where  $B'_p$  contains the columns for the variables  $x_r, \dots, x_1$ . This is the same as in Eq. (25) Chapter III.

$$B = \begin{bmatrix} B_p & B'_p \end{bmatrix} \quad (39)$$

Further, let the subblock  $B'_p$  be as follows with its rows and columns arranged so that it has the form of Eq. (32) but in reverse order:

$$B'_p = \begin{bmatrix} a_{k'r} & 0 & & & & \\ a_{kr} & a_{k'r-1} & 0 & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & a_{k'2} & 0 \\ & & & & a_{k2} & a_{k'1} \end{bmatrix} \quad (40)$$

Lemma 5.1

Let  $b$  be a column of  $B^{-1}$  as described above. If  $\frac{n_{su}}{m_{su}}$  and  $\frac{n_{iu}}{m_{iu}}$

are nonzero elements of column  $b$  in rows  $s$  and  $i$ , respectively, with row  $s$  above row  $i$ , then  $\frac{m_{su}}{m_{iu}}$  is an integer.

Proof:

We will first use the column generation scheme to generate the nonzero elements of  $b$ . Let the coefficients of the basic variables  $x_r, \dots, x_1$  in the path to the root be  $(a_{k'i}, a_{ki})$  with  $a_{ki}$  being in the equation corresponding to the node closer to the root. Equations (27) and (28) in Chapter III show that the only nonzero elements in  $b$  correspond to the columns for  $x_r, \dots, x_1$ . Using equation (26) from Chapter III, the nonzero elements  $b_r, \dots, b_1$  can be expressed as:

$$b_r = \frac{1}{a_{k'r}}, \quad b_i = \frac{\prod_{j=1}^i a_{kr-j}}{i \prod_{j=0}^{r-i} a_{k'r-j}} \quad i = 1, \dots, r-1 \quad (41)$$

If  $B$  is arranged as in Eqs. (39) and (40), then the nonzero elements in  $b$  are ordered so that  $b_1$  is in the first row, and  $b_s$  is in a row above  $b_i$  if  $s$  is less than  $i$ . But for  $b_s$  and  $b_i$  the denominators  $m_{su}$  and  $m_{iu}$  are given by:

$$m_{su} = \prod_{j=0}^{r-s} a_{k'r-j} \quad (42)$$

$$m_{iu} = \prod_{j=0}^{r-i} a_{k'r-j} \quad (43)$$

Thus for  $s$  less than  $i$ :

$$\frac{m_{su}}{m_{iu}} = \prod_{j=s}^{i-1} a_{k',j} \quad (44)$$

and this is an integer, proving the lemma.

This lemma can be used to prove the following theorem.

**Theorem 5.2**

Let  $p_{1j} = \frac{D}{m_{1j}}$ . Then:

(1)  $p_{1j}$  is an integer for each  $j$ .

(2) If  $d$  is the greatest common divisor of  $\{p_{1j}\}$ , then  $\frac{D}{d}$  is an integer.

**Proof:**

To establish (1) note that:

$$N = D \begin{bmatrix} n_{1j} \\ m_{1j} \end{bmatrix} = DB^{-1} \quad (45)$$

is an integer matrix.

Thus  $\frac{D}{m_{1j}} = p_{1j}$  must be integer for all  $i$  and  $j$ . In particular for  $i = 1$ ,

$\frac{D}{m_{1j}} = p_{1j}$  is an integer for all  $j$ .

To show (2) we know by Lemma 5.1 that  $\frac{m_{sj}}{m_{ij}}$  is an integer for  $s$  less than  $i$ . Thus:

$$\frac{p_{ij}}{p_{sj}} = \frac{\frac{D}{m_{ij}}}{\frac{D}{m_{sj}}} = \frac{m_{sj}}{m_{ij}} \quad (46)$$

is an integer for  $s$  less than  $i$ .

In particular, for  $s = 1$ ,  $q_{ij} = \frac{p_{ij}}{p_{1j}}$  is an integer for all  $i$  and  $j$ . If  $d$  is the greatest common divisor of  $\{p_{1j}\}$ , then  $\frac{p_{1j}}{d} = t_{1j}$  is an integer for all  $j$ . Furthermore,  $\frac{p_{ij}}{d} = q_{ij}t_{1j} = p'_{ij}$  is also an integer for all  $i$  and  $j$ . The matrix  $N$  can thus be expressed as:

$$N = D \begin{bmatrix} n_{ij} \\ m_{ij} \end{bmatrix} = [p_{ij}n_{ij}] = d[p'_{ij}n_{ij}] = dN' \quad (47)$$

The basis inverse is:

$$B^{-1} = \frac{dN'}{D}$$

But  $\frac{D}{d} = m_{ij}p'_{ij}$  which is also an integer and the theorem is proved.

If the factor  $d$  is greater than 1, then  $D' = \frac{D}{d}$  is smaller in absolute value than  $D$  and the coefficients of the nonbasic variables in the optimal LP tableau can be expressed as:

$$\bar{A}_N = B^{-1}A_N = \frac{1}{D'} N'A_N \quad (48)$$

All of the elements in  $\bar{A}_N$  can be expressed as fractions over the denominator  $D'$  which is smaller than  $D$ .

To determine the factor  $d$ , we generate the row of the basis inverse corresponding to the slack arc at the root, and retain the denominators  $m_{1j}$ . The determinant  $D$  can be calculated at the same time. The set  $\{p_{1j}\}$  is calculated by  $p_{1j} = \frac{D}{m_{1j}}$ . Finally,  $d$  is the greatest common divisor of  $\{p_{1j}\}$ . It is obvious that the theorem applies to each component of the basis containing a slack column if there are more than one.

For a pseudorooted component, a similar theorem can be stated.

First consider the form of such a component ordered by choosing a node, tracing around the cycle, then tracing out from each basic variable not in the cycle touching node one of the cycle; then for node two and so on. The resulting matrix will be:

$$B = \begin{bmatrix} B_c & & & & \\ \vdots & B_i & & P & \\ & \vdots & B_{n_k} & & \\ & & & & \end{bmatrix} \quad (49)$$

$B_c$  corresponds to the cycle. There is a separate block  $B_i$  for each basic arc incident on the cycle but not contained in it. This is seen by referring to the corollary for the partitioning of a block by a column.

The inverse of  $B$  is written directly:

$$B^{-1} = \begin{bmatrix} B_c^{-1} & -B_c^{-1}P & \begin{bmatrix} B_1^{-1} \\ B_2^{-1} \\ \vdots \end{bmatrix} \\ \vdots & B_i^{-1} & \vdots \\ & B_1^{-1} & B_2^{-1} \\ & & \ddots \end{bmatrix} \quad (50)$$

Let  $D = |\text{Det} B_c|$  and  $D_i = |\text{Det}(B_i)|$  for the blocks.

Theorem 5.3

The determinant of a pseudorooted component  $D_r$  may be expressed as an integer  $D'/d$  where  $d$  is the greatest common divisor of the numbers  $(D, D_1, \dots, D_r)$ .

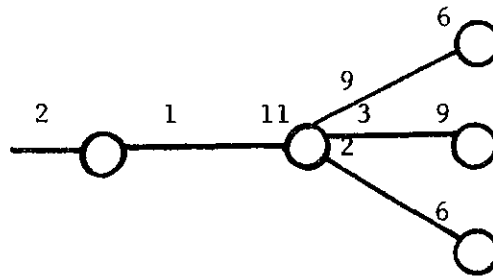
Proof:

The reasoning is similar to that for the proof of Theorem 5.2.

The details are omitted.

Theorems 5.2 and 5.3 generalize Theorem 2.6 of Estabrook [20] to the case of generalized flow and provide a more direct means of performing the reduction.

The example below illustrates the calculation of the factor  $d$  for a rooted component.



The corresponding matrix is:

$$B = \begin{bmatrix} 2 & 1 & & & \\ & 11 & 9 & 3 & 2 \\ & & 6 & & \\ & & & 9 & \\ & & & & 6 \end{bmatrix} \quad (51)$$

The determinant of  $B$  is computed to be :

$$D = 7128$$

The first row of the inverse is:

$$b'_1 = \left[ \frac{1}{2} - \frac{1}{2(11)} \quad \frac{9}{(2)(11)(6)} \quad \frac{3}{2(11)(9)} \quad \frac{2}{2(11)(6)} \right]$$

Dividing the denominators into  $D$  gives the factors  $p$ .

$$p = [(11)(6)(9)(6) \quad (9)(6)(9) \quad (9)(6) \quad (6)(6) \quad (6)(9)]$$

The greatest common denominator is 18. Thus, the problem can be represented as fractions over the common denominator 396 instead of 7128.

#### 5.4 Structure of the Nonbasic Columns and Basic Rows

The row and column generation methods of Chapter III can be used to describe the structure of  $\bar{A}$ , the updated matrix of coefficients of the nonbasic variables in the optimal LP solution. These columns play an important role in group theoretic solution procedures since they, in effect, generate the complete group of columns. Intuitively, the correct nonnegative integer combination of these columns will produce the optimal integer solution. The group based methods construct nonnegative integer combinations of these columns under a transformation, to obtain an integer solution and then verify the feasibility and/or optimality of the solution. The remainder of this section will be observations about the structure of  $\bar{A}$  and what the structure means in terms of the associated graph. The intent is to highlight the special structure of the IGFP as opposed to a general IP problem and to demonstrate the information contained in the graphical representation of the optimal continuous solution.

In Chapter III a procedure for constructing the current columns of  $\bar{A}$  for a nonbasic variable was developed. It was shown that this column contained nonzero entries only for the basic variables in the paths from the ends of the nonbasic arc to and including the respective root(s) or pseudoroot(s). Under certain conditions, it might contain nonzero entries for only a subset of these basic variables. A nonbasic variable

can affect only the integrality of those basic variables corresponding to the rows in which it has nonzero entries. Intuitively, the farther from the root that a nonbasic variable meets a tree, the greater the number of basic variables with which it will interact.

The row generation procedure of Chapter III provides similar insight for basic variables. The current row for a basic variable will have entries only for those nonbasic variables which meet the tree once above it, or twice above it in an independent cycle. This means that the set of nonbasic variables which interacts with each basic variable can be defined. The size of these sets depends in some measure upon the position of the basic variable in the tree. A root variable or a variable in a pseudoroot will have nonzero entries for all nonbasic variables which touch the tree except those which touch it twice and form dependent cycles. Intuitively, the higher a basic variable is in a tree, the fewer the number of nonbasic variables with which it interacts.

Estabrook formalizes this ordering of variables and uses it to develop his CASCADE algorithm. His procedure uses a dynamic programming method and considers one row at a time, with the row ordering based on the nearness of the associated basic variable to the top of a tree. The group problem for a row is never considered until the rows for all basic variables above it in the tree have been considered. He also takes advantage of the fact that the denominator for a row is increasing for variables which are lower in the tree as seen in the column generation procedure. He takes special advantage of the fact that the GTP constraint set has all plus one coefficients in one set of its equations to reduce the number of rows which must be explicitly considered. Estabrook's

algorithm requires checking the nonnegativity constraints on the basic variables after a solution is obtained, and he suggests means for checking or for enforcing some of these nonnegativity constraints during the group optimization. He notes, however, that enforcing these conditions during the process of the group algorithm would detract considerably from its computational effectiveness.

In this dissertation, information from the group formulation will be used to aid in the solution of the problem, but the specialized group structure outlined in this section will not be used directly.

### 5.5 Penalties

The concept of penalties in relation to branch and bound methods for integer programming was mentioned in Section 1.9.2. They will be discussed in more detail in this section and a means for using information from the group formulation in constructing penalties will be discussed. The first part of the section is based on Tomlin [93].

At a node of a branch and bound IP tree, a continuous bounded variable linear program is solved. If all the variables are integer in the solution, or if the continuous solution objective value is greater than the best integral solution so far, branching from this node is not necessary since no better integer solution is possible below the node. If some basic variable  $x_i$  is noninteger at value  $\bar{b}_i$ , then any integer solution must satisfy:

$$x_i \geq [\bar{b}_i] + 1 \quad \text{or} \quad x_i \leq [\bar{b}_i] \quad (52)$$

where  $[\bar{b}_i]$  is the greatest integer less than or equal to  $\bar{b}_i$ .

Define:

$$f_{b_i} = \bar{b}_i - [\bar{b}_i] \quad (53)$$

If  $x_i$  were chosen as the branching variable, then one of the constraints in Eq. (52) would be associated with each of the branches. If it is assumed that one of the constraints above is added to the final simplex tableau for the current node, then an estimate of the continuous objective function value at the newly created node will be the change in the objective function for the first simplex pivot. Suppose the current row for  $x_i$ , the equation constraining it to be an integer higher, and the objective row are written:

$$\bar{b}_i = x_i + \sum_j a_{ij} x_j \quad (54)$$

$$- [\bar{b}_i] - 1 = - x_i + S \quad (55)$$

$$- z_0 = \sum_j c_j x_j \quad (56)$$

To put the tableau in canonical form, Eq. (54) is added to Eq. (55).

$$\bar{b}_i = x_i + \sum_j a_{ij} x_j \quad (57)$$

$$- (1 - f_{b_i}) = \sum_j a_{ij} x_j + S \quad (58)$$

$$- z_0 = \sum_j c_j x_j \quad (59)$$

The tableau is now dual feasible and the first dual simplex pivot chooses the pivot element by finding the minimum ratio.

$$\frac{a_{ip}}{c_p} = \min_{a_{ij} < 0} \left( \frac{c_j}{-a_{ij}} \right) \quad (60)$$

The change in the objective value for the first pivot is:

$$P_{u_i} = \frac{-(1 - f_{b_i})c_p}{a_{ip}} \quad (61)$$

We will call this the up-penalty. A similar analysis for adding the constraint making  $x_i$  less than  $[\bar{b}_i]$  gives the following down-penalty.

$$P_{D_i} = \frac{(f_{b_i})c_p}{a_{ip}} \quad (62)$$

The minimum ratio in this case is found by:

$$\frac{c_p}{a_{ip}} = \min_{a_{ij} > 0} \left( \frac{c_j}{a_{ij}} \right) \quad (63)$$

This may be done for each of the basic variables which are fractional. Since any node below the current node must have one of the conditions imposed, any integer solution below this node must assume the minimum additional cost:

$$P_c = \min_{\bar{b}_i \text{ fractional}} [P_{u_i}, P_{D_i}] \quad (64)$$

These penalties are derived by imposing the restrictions that the basic variables must be integer. Each variable is considered separately with no consideration given to the interaction between basic variables

or the requirement that the nonbasic variables be integer. A simple extension to the penalties derived so far is the requirement that the nonbasic variable which comes into the basis when the newly restricted variable leaves must also be integer and hence must be at least one.

The quantity associated with the dual pivot which goes into the calculation for  $P_{u_i}$  when  $x_p$  is introduced into the basis becomes:

$$\max \left( \bar{c}_p, -\bar{c}_p \frac{(1 - f_{b_i})}{a_{ip}} \right) \quad (65)$$

And similarly for the down penalty. The motivation for these penalties is the same as before except some slight consideration has been given to the requirement that at least the nonbasic variable  $x_p \geq 1$  if  $x_p > 0$ .

Tomlin shows that a stronger penalty may be derived by considering the Gomory fractional constraint (Gomory [41]) derived from each row of the tableau associated with a noninteger basic variable. A constraint for a particular  $x_i$ :

$$-f_{b_i} = -\sum_j f_{ij}x_j + S \quad (66)$$

where:

$$f_{ij} = \begin{cases} a_{ij} - [a_{ij}] & \text{If } f_{ij} \leq f_{b_i} \\ f_{b_i} \frac{(1 - f_{ij})}{(1 - f_{b_i})} & \text{If } f_{ij} > f_{b_i} \end{cases} \quad (67)$$

where the fractional part  $f_{ij}$  is given by  $f_{ij} = a_{ij} - [a_{ij}]$ .

This constraint can be added to the tableau in the same manner as before and the change in the objective function for the first dual simplex pivot

can be used as a penalty. Tomlin shows that this penalty is greater than or equal to the previously derived penalties. This penalty utilizes more of the problem requirements since Eq. (66) is derived by considering the integrality of the basic variable  $x_i$  and all of the nonbasic variables  $\{x_j\}$ .

Gomory and Johnson [46,47] use a row of the optimal LP tableau to derive stronger constraints than Eq. (66). By relaxing the nonnegativity constraints on all of the basic variables and the integrality requirements on all of the basic variables except one particular basic variable  $x_i$ , we obtain the following problem.

$$\text{Minimize:} \quad \sum_j \bar{c}_j x_j \quad (68)$$

$$\begin{array}{l} (G_r) \\ \text{Subject to:} \end{array} \quad \sum_j f_j x_j \equiv f_o \quad (69)$$

$$x_j \geq 0 \quad \text{and integer for all } j. \quad (70)$$

Equation (69) is the congruency relationship from one row of the set of equations (20). The elements  $\{f_j\}$  and  $f_o$  are the fractional parts of the coefficients for the nonbasic variables  $x_j$  and the right hand side from the row for the basic variable  $x_i$ . The solution to  $G_r$  provides a valid penalty for it is the minimum increase in the LP objective value for satisfying the congruency relationship Eq. (69) using an integer combination of the nonbasic columns from the optimal LP tableau.

If problem  $G_r$  were solved for each row with a fractional basic

variable, then the maximum objective value from these would be a valid penalty. Unfortunately, solving the problem  $G_r$  is a difficult task in general. It is what is termed a cyclic group problem and methods of dynamic programming or a network formulation are usually used to solve it.

An attractive alternative is offered by Johnson in [62]. He presents a method which can be used to solve Eq. (68). Moreover, the method can be terminated prior to completion with information which can be used to construct a valid constraint of the form of Eq. (66) and which will provide a penalty at least as good as that gotten by using Eq. (66).

The following is a brief description of Johnson's algorithm intended to provide insight into the process of constructing the valid inequality and solving  $G_r$  at the same time.

The fractional parts  $\{f_j\}, f_0\}$  are members of a cyclic group. Positive integer combinations modulo one of the elements  $\{f_j\}$  can be formed so that they will equal (generate) each member of the group. In particular, it is desired to form the positive integer combination mod one which equals  $f_0$  and costs less than any other positive integer combination which equals  $f_0$ .

Consider the following problem  $G'_r$  which is a relaxation of problem  $G_r$ , the single row cyclic group problem.

Find  $\delta^* = \text{Minimum } \{\delta_1, \delta_2\}$

$(G'_r)$

where  $\delta_1$  and  $\delta_2$  are defined by:

$$\delta_1 = \text{Minimum} \sum_{j \in J_1} \bar{c}_j x_j \quad (71)$$

(P<sub>1</sub>)

Subject to:

$$\sum_{j \in J_1} f_j x_j = j_o$$

$$x_j \geq 0 \quad \text{for all } j \in J_1;$$

$$\delta_2 = \text{Minimum} \sum_{j \in J_2} \bar{c}_j x_j$$

(P<sub>2</sub>)

Subject to:

$$\sum_{j \in J_2} (1 - f_j) x_j = 1 - f_o; \quad (72)$$

$$x_j \geq 0 \quad \text{for all } j \in J_2$$

when the sets  $J_1 = \{j : f_j \leq f_o\}$ ,  $J_2 = \{j : f_j > f_o\}$ .

In effect, the integrality requirement on the nonbasic variables has been relaxed and  $P_1$  and  $P_2$  are single equation minimization problems called continuous knapsack problems. The constraints of  $P_1$  and  $P_2$  are such that the solution to  $G'_r$  has only one  $x_j$  nonzero ( $x_{j_1}$ ,  $x_{j_2}$ ) in each subproblem.

The optimal values of the objective values for  $P_1$  and  $P_2$  are given by:

$$\delta_1 = \frac{\bar{c}_{j_1} f_o}{f_{j_1}}, \quad \text{for some } j_1 \in J_1$$

$$\delta_2 = \frac{\bar{c}_{j_2} (1-f_o)}{(1-f_{j_2})} \quad \text{for some } j_2 \in J_2.$$

Let  $j^* = j_1$  if  $\delta_1 \leq \delta_2$  and  $j^* = j_2$  if  $\delta_2 > \delta_1$ .

The optimal value  $\delta^*$  for  $G'_r$  is given by:

$$\delta^* = \begin{cases} \frac{c_{j^*} f_o}{f_{j^*}} & \text{if } \delta_1 \leq \delta_2 \\ \frac{c_{j^*} (1-f_o)}{(1-f_{j^*})} & \text{if } \delta_2 < \delta_1 \end{cases} \quad (73)$$

Using the solution  $\delta^*$  to  $G'_r$ , Johnson defines the piecewise linear function  $\pi$  such that:

$$\pi(u) = \begin{cases} \frac{u \delta^*}{f_o} & 0 \leq u \leq f_o \\ \frac{(1-u) \delta^*}{(1-f_o)} & f_o \leq u \leq 1 \end{cases} \quad (74)$$

This function satisfies certain requirements so that the following is a valid inequality for the IP problem.

$$\pi(f_o) \leq \sum_j \pi(f_j) x_j$$

But substituting the values for  $f_o$  and  $\{f_j\}$  into Eq. (74):

$$\pi(f_o) = \frac{f_o \delta^*}{f_o} = \frac{(1-f_o) \delta^*}{(1-f_o)} = \delta^* \quad (75)$$

and

$$\pi(f_j) = \begin{cases} \frac{\delta^* f_j}{f_o} & j \in J_1 \\ \frac{\delta^* (1-f_j)}{(1-f_o)} & j \in J_2 \end{cases} \quad (76)$$

The inequality is:

$$\delta^* \leq \sum_{j \in J_1} \frac{\delta^* f_j x_j}{f_0} + \sum_{j \in J_2} \frac{\delta^* (1-f_j) x_j}{1-f_0} \quad (77)$$

But this is equivalent to:

$$f_0 \leq \sum_{j \in J_1} f_j x_j + \sum_{j \in J_2} \frac{f_0 (1-f_j) x_j}{1-f_0} \quad (78)$$

Equation (78) is the Gomory fractional cut and  $\pi(f_0) = \delta^*$  is the associated Tomlin penalty.

Now, intuitively, the price has been paid for using the group element  $f_{j*}$ ; thus, we can reach either the element  $f_0$  or  $f_0 - f_{j*} \pmod{1}$  and solve the congruency problem. It can be done with the original group elements, or the original elements plus ones of the type  $f_{j*} + e_i$  where  $e_i$  are group elements previously paid for in the sense that  $f_{j*}$  has been a limiting element. For the first iteration then the elements  $\{0,1\}$  are the  $e_i$  for which nothing was paid. The algorithm proceeds by solving sets of problems like  $G'_r$ , one for each element in the enlarged set of eligible right hand sides (i.e.,  $f_0$  and  $f_0 - f_{j*}$  for the second iteration). The algorithm generates a piecewise linear function with an increasing number of segments symmetric about the set of candidate solving elements ( $f_0, f_0 - f_{j*}, f_0 - f_{k*}, \dots$ ). At any iteration the function so generated can be used to define a valid inequality and  $\pi(f_0)$  is the associated objective function penalty. If a candidate solution element is reached

exactly by one of the eligible group elements, the cyclic group problem for that row has been solved and  $\pi(f_o)$  is the minimum cost of making the basic variable associated with that row integer with positive integer combinations of the nonbasic variables with nonzero fractional parts of their coefficients in the row. The integrality of the other basic variables has been relaxed.

This discussion has been a sketch of the method given by Johnson [62] to which the reader is referred for more details and the proof of the validity of the algorithm. The following example illustrates the calculation of all of the penalties described above. The integer programming problem with all of the constraints except for one row of the optimal LP tableau relaxed can be stated:

$$\text{Minimize} \quad 3x_1 + 2x_2 \quad (79)$$

Subject to:

$$x_3 + \frac{1}{7} x_1 - \frac{3}{7} x_2 = \frac{3}{7} \quad (80)$$

$$x_1, x_2, x_3 \geq 0 \quad \text{and integer.}$$

The penalty derived from the first dual simplex iteration after adding the constraint  $x_3 \geq 1$  is:

$$P_u = \frac{-\frac{4}{7} (2)}{-\frac{3}{7}} = \frac{7}{3} \quad (81)$$

For adding  $x_3 \leq 0$ :

$$p_d = \frac{\frac{3}{7} (3)}{\frac{8}{7}} = \frac{9}{8} \quad (82)$$

The Gomory fractional cut using Eq. (66) is:

$$\frac{3}{7} \leq \frac{1}{7} x_1 + \frac{\frac{3}{7} (\frac{3}{7})}{\frac{4}{7}} x_2 = \frac{1}{7} x_1 + \frac{9}{28} x_2 \quad (83)$$

The Tomlin penalty is:

$$\text{minimum} \left[ \frac{\frac{3}{7} (3)}{\frac{1}{7}}, \frac{\frac{3}{7} (2)}{\frac{9}{28}} \right] = \frac{8}{3} \quad (84)$$

The cyclic group problem derived from Eq. (79) and Eq. (80) is:

$$\text{Minimize} \quad 3x_1 + 2x_2 \quad (85)$$

Subject to:

$$\frac{1}{7} x_1 + \frac{4}{7} x_2 \equiv \frac{3}{7} \pmod{1} \quad (86)$$

$$x_1, x_2 \geq 0 \quad \text{and integer.}$$

The function generated by the first iteration of the Johnson algorithm is:

$$\pi(u) = \begin{cases} u (\frac{56}{9}) & 0 \leq u \leq \frac{3}{7} \\ (1-u) (\frac{14}{3}) & \frac{3}{7} \leq u \leq 1 \end{cases} \quad (87)$$

The penalty is  $\pi(\frac{3}{7}) = \frac{8}{3}$  which is the Tomlin penalty as expected. The

next iteration generates the function:

$$\pi(u) = \begin{cases} 21u & 0 \leq u \leq \frac{3}{7} \\ -\frac{133}{3}u + 28 & \frac{3}{7} \leq u \leq \frac{4}{7} \\ \frac{35}{2}u - 8 & \frac{4}{7} \leq u \leq \frac{6}{7} \\ -49u + 49 & \frac{6}{7} \leq u \leq 1 \end{cases} \quad (88)$$

The penalty is:

$$\pi\left(\frac{3}{7}\right) = 9 \quad (89)$$

The next iteration of the algorithm verifies that this is the solution of the cyclic group problem.

The maximum penalty from the single pivot analysis was  $\frac{7}{3}$ , the Tomlin penalty was  $\frac{8}{3}$ , but the penalty obtained by two iterations of Johnson's algorithm was 9. Without considering the other basic variables, an integer solution at any node below the current one will have an objective value at least 9 units greater than the objective value of the LP at the current node. The experience reported by Johnson and Spielberg [60] and Kennington [63] indicates that the penalties derived from consideration of the cyclic group problems from the rows of fractional variables is strong information for use with a branch and bound procedure.

The information required for any of the penalties is the row of coefficients for nonbasic variables corresponding to a noninteger valued basic variable. This is easily obtained from the optimal GFP solution using the row generation procedure given in Section 3.4.

The penalties described in this section are tools to aid the

decision process of a branch and bound (or perhaps an implicit enumeration) algorithm. Their use is heuristic in that using them to choose the branching variable or the next candidate problem for consideration does not assure one that the search will be shortened or that an integer solution will be found sooner. Decisions made using these penalties are based on a supposition as to the nature of the search after the decisions have been made. However, it is reasonable to conjecture that, if the computation of a penalty takes into account more of the problem restrictions, then better decisions can be made.

### 5.6 Summary

In this chapter the group formulation of an integer programming problem has been considered. The methods necessary to formulate the IGFP as a group IP problem from the optimal linear programming solution obtained by the GFP algorithm in Section 4.2 were presented. A description of this group IP problem was given and related to the graphical representation of the optimal continuous solution. The work of Estabrook was considered, and his algorithm for the IGTP was discussed. Finally, the results of several authors concerning penalties for enumeration methods were summarized.

## CHAPTER VI

### ALGORITHM FOR THE IGFP

#### 6.1 Introduction

In previous chapters the specially structured linear programming problem designated as the generalized flow problem has been characterized and an algorithm for solving it has been presented. The algorithm takes advantage of the problem characterization to eliminate some operations and improve other operations required when the simplex procedure is used to solve the problem. In Chapter V the integer generalized flow problem was characterized in terms of the solution to the associated GFP and the use of the graphical representation of the problem to identify the interaction between the variables for the integer problem was discussed. In this chapter several of the methods and procedures previously presented will be combined into a branch and bound algorithm for solving the IGFP. This algorithm is constructed to take advantage of the special properties and characterizations of both the GFP and IGFP.

#### 6.2 Selection of Method of Solution

In Section 1.9 the general approach for solving integer programming problems was described. The characterization of the GFP and IGFP did not indicate that a combinatorial approach might be used for solving the IGFP; thus, no further consideration was given to this type of solution procedure.

Implicit enumeration methods attempt to apply logical tests to the constraints of the associated linear programming problem to eliminate the need to explicitly consider certain sets of the variables as possible solutions to the problem. It appears that an implicit enumeration algorithm could be organized to take advantage of the sparsity of the IGFP constraint set. However, as indicated by Geoffrion [25] and Trotter [95], the effectiveness of an implicit enumeration algorithm is greatly enhanced by the use of information derived from the associated LP. This additional information is usually in the form of a surrogate constraint which does not have the same form as the original set of constraints and could not be treated in the same manner. Thus, although an implicit enumeration algorithm could be specialized to take advantage of the constraints of the GFP, the derived surrogate constraints would not have their special structure. Specialization of an implicit enumeration algorithm for the IGFP would be largely a matter of organizing the algorithm to take advantage of the form of the constraints and would not focus on improving the solution procedure by deriving better information from this special structure. For this reason, the implicit enumeration approach was not taken.

Although the characterization of the IGFP in Chapter V suggests that a group theoretic method might be developed for the IGFP, this approach was not used directly. The simplification used by Estabrook to develop his algorithm for the GFP does not occur for the IGFP. Additionally, Estabrook's algorithm shared the drawback of the direct group approach for the general problem, in that we are not assured that the

solution to the group problem will be feasible for the original problem. The methods which must be used to reapply the relaxed constraints have not in general been efficient. For these reasons, a group theoretic algorithm was not developed. On the other hand, the branch and bound approach has several features which permit taking advantage of the special structure of the IGFP. This is discussed in detail below.

### 6.3 Branch and Bound Procedure

In Section 1.9 the tools necessary for an efficient branch and bound procedure were listed as:

- (i) Efficient LP solution method.
- (ii) Branching variable strategy.
- (iii) Penalty calculation to provide good bounds.
- (iv) Node choosing strategy.
- (v) Efficient storage of candidate problem.
- (vi) Method for finding a "good" feasible solution.

The detailed study of the GFP and IGFP in Chapters II through V has provided the devices necessary to construct a branch and bound algorithm which incorporates the characteristics listed above. From the computational results given in Chapter IV it will be seen that the GFP algorithm given in Section 4.2 is very efficient. Further, it requires less storage space than a general simplex algorithm. The results of Johnson and Spielberg [60] and Kennington [63] indicate that the penalties derived from considering the cyclic group problem derived from the row of a noninteger basic variable facilitates a branch and bound IP algorithm. In addition, Johnson [61] indicates that the heuristic rule of choosing the branching variable as the basic variable whose row pro-

vides the maximum penalty is an effective device. Since the row generation technique given in Section 3.4 is an efficient method for generating those rows which are required, satisfactory means for accomplishing (ii) and (iii) are at hand. In conjunction with (ii) and (iii), the node choice strategy will be to choose the node with the smallest estimated objective value  $Z_E = Z_{LP} + \text{PEN}$  where  $Z_{LP}$  is the LP objective value from its predecessor node and PEN is the penalty from the single row cyclic group problem. If the penalty calculations are effective, then by choosing the least cost node, one presumes that as integer solutions are encountered a large number of nodes may be removed from the candidate list since only those with the larger penalties and hence the larger lower bounds have been included. The use of this rule is supported by Kennington's experience for the fixed charge transportation problem which is a mixed integer problem with a network structure and can be formulated as an IGFP problem. The effectiveness of the special algorithm for the GFP decreases the desirability of using a "near" node choice strategy to enhance the restart capability of the continuous solution procedure. Since an advanced start for the continuous problem is not being used, the storage requirements for the candidate problems (nodes) are reduced. A method for obtaining a good feasible solution is discussed in Section 6.5. Thus, the means are at hand for the construction of a branch and bound procedure for the IGFP. The algorithm is presented below and in Figure 10.

We will denote by C the list of candidate problems (which do not have integrality requirements) created by imposing additional bounds on a variable from some previous problem in the candidate set.

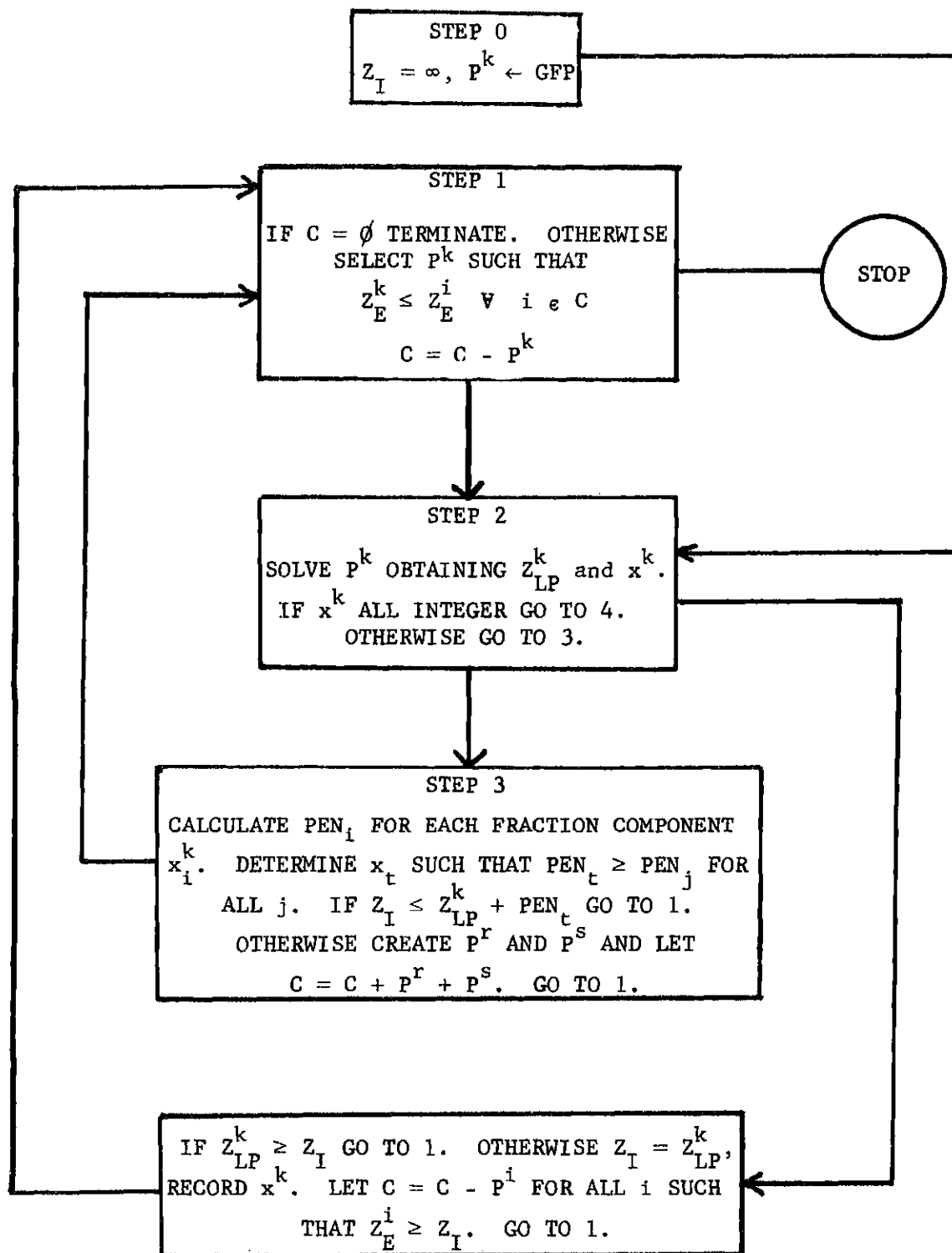


Figure 10. Branch and Bound Procedure

The solution to a problem  $P_k \in C$  will be denoted by the vector  $x^k$  with objective value  $Z_{LP}^k$ . If  $P^k$  is infeasible, we will let  $Z_{LP}^k = \infty$ . The value  $Z_E^k$  will denote the lower bound on the value of an integer solution contained in the constraint set of  $P^k$ . It is obtained by adding the penalty  $PEN$  to the objective value  $Z_{LP}^j$  where  $P^j$  is the predecessor problem to  $P^k$ .  $PEN$  is the penalty associated with the fractional variable in the solution to  $P^j$  which was restricted to construct  $P^k$ . The value  $Z_I$  will denote the current lower bound on the objective function value for the IGFP obtained from the incumbent integer solution  $x^I$ . Using this notation the branch and bound procedure can be stated.

Step 0. Set  $Z_I = \infty$ . Let  $P^k \leftarrow GFP$ . Go to Step 2.

Step 1. If  $C = \emptyset$  terminate; the incumbent solution is optimal. Otherwise select and remove problem  $P^k$  from  $C$  such that  $Z_E^k = \{Z_E^i : i \in C\}$ .

Step 2. Solve  $P^k$ , obtaining the objective value  $Z_{LP}^k$ . If the components of  $x^k$  are all integer go to step 4. Otherwise go to step 3.

Step 3. For each noninteger basic variable  $x_j$  calculate the penalty  $PEN_j$ . Choose the branching variable  $x_t$  such that  $PEN_t \geq PEN_j$  for all  $j$ . If  $Z_I \leq Z_{LP}^k + PEN_t$  go to step 1. Otherwise create two problems  $P^r$  and  $P^s$  by branching on  $x_t$  with additional constraint  $x_t \leq [x_t^k]$  for  $P^r$  and  $x_t \geq [x_t^k] + 1$  for  $P^s$  with lower bounds  $Z_E^r = Z_E^s = Z_{LP}^k + PEN_t$ . Place these problems in  $C$  and return to step 1.

Step 4. If  $Z_{LP}^k \geq Z_I$  go to step 1. Otherwise set  $Z_I = Z_{LP}^k$

and record the new incumbent solution  $x^k$ . Remove any problem  $P^i$  from  $C$  if  $Z_E^i \geq Z_I$ . Go to step 1.

The algorithm is finite since branches of the enumeration tree are terminated only if no integer solution can be found below a node which will be better than the incumbent solution. If the candidate list is empty then the optimal integer solution is the incumbent solution.

The GFP algorithm given in Section 4.2 is the algorithm used to solve the relaxed candidate problems. The penalties are calculated from Johnson's algorithm for the cyclic group problem using the row generation procedure of Section 3.4. The number of iterations in Johnson's algorithm is specified by the user and Tomlin's penalties are generated if only one iteration is specified.

#### 6.4 Computational Results

The results of solving a number of problems with the IGFP algorithm are listed in Table 5. Problems 1, 2, and 9-14 are of the IGTP type and the rest are IGFP problems. The GFP algorithm used to solve the candidate problems does not employ an advanced start and was started with an initial basis composed of all artificial and slack variables. Problems 11 and 12 are the same problem with 11 using one iteration of the Johnson algorithm to find penalties and 12 using three iterations. For this problem at least, the additional effort used to find stronger penalties resulted in three less LP problems being solved but the total solution time was increased by 30 percent. For almost all of the problems the bulk of the solution time is spent in finding the first integer solution. After one is found, the penalties seem to provide an effective fathoming device

Table 5. IGFP Computational Results

Problem Number	No. of Eqn.	No. of Var.	LP's Solved	LP Time	Continuous Sol. Value	Integer Sol. Val.	Time to First Sol.	Total Time*
1	4	8	9	.052	342.5	360	.107	.120
2	4	8	7	.027	350.	380	.092	.101
3	4	12	9	.238	16.273	17	.376	.382
4	5	21	2	.007	456.5	477	.019	.024
5	5	21	3	.015	43.	44	.072	.077
6	5	11	4	.024	267.333	278	.047	.053
7	5	13	12	.068	108.608	119	.311	.316
8	5	14	5	.052	36.283	38	.130	.135
9	5	11	8	.071	29.444	31	.118	.123
10	7	19	5	.079	104.667	106	.120	.124

Table 5. (Continued)

Problem Number	No. of Eqn.	No. of Var.	LP's Solved	LP Time	Continuous Sol. Value	Integer Sol. Val.	Time to First Sol.	Total Time*
11	8	24	29	1.0125	565.197	566	1.929	1.958
12	8	24	25	.839	565.197	566	2.530	2.563
13	8	24	9	.278	305.145	307	.571	.576
14	8	24	5	.158	563.788	**		.280
15	10	42	7	.093	113.5	115	.166	.225

\*All times in seconds on Univac 1108 in multiprocessing mode exclusive of input/output time

\*\*No feasible integer solution

for the remaining candidate problems. A method for obtaining a feasible integer solution after solving the initial GFP is given in the next section. It might be used with the IGFP branch and bound algorithm to provide a better bound at the initiation of the search.

### 6.5 Obtaining a Feasible Integer Solution

In this section a method for obtaining an integer solution which is feasible to all of the constraints binding at the LP optimal solution is given. If the solution is also feasible to all of the nonbinding constraints it is a feasible integer solution and may be used as the initial incumbent solution. It provides an initial bound on the branch and bound search tree. The method is due to Hillier [53] and Biondi and Schmid [9] who developed their procedures independently. This procedure is particularly easy to implement using the row and column construction techniques for the GFP given previously. Using these methods the solution (if one is found) is obtained at little additional cost after solving the GFP. The method tends to find a solution in the proximity of the continuous solution ( $x_{LP}^*$ ). A brief outline of the method is given below. A major limitation is the assumption that the original problem contains no equality constraints.

The intuition and geometry of the method is to construct a hypercube of edge length one inside the interior of the feasible region defined by the constraints binding at the LP optimum. The integer solution contained in this hypercube will be feasible for the binding constraints and thus is a good candidate as a feasible integer solution. After solving the GFP, for all binding constraints (not containing a basic slack),

construct the new right hand side defined by:

$$b'_i = b_i - \frac{1}{2} \sum_{j \in J_B} |a_{ij}|$$

$J_B$  is the set of basic variables and  $(a_{ij})$  are the corresponding coefficients in the  $i^{\text{th}}$  equation.

This new right hand side is used to calculate the vector:

$$x' = \langle B^{-1}b' \rangle$$

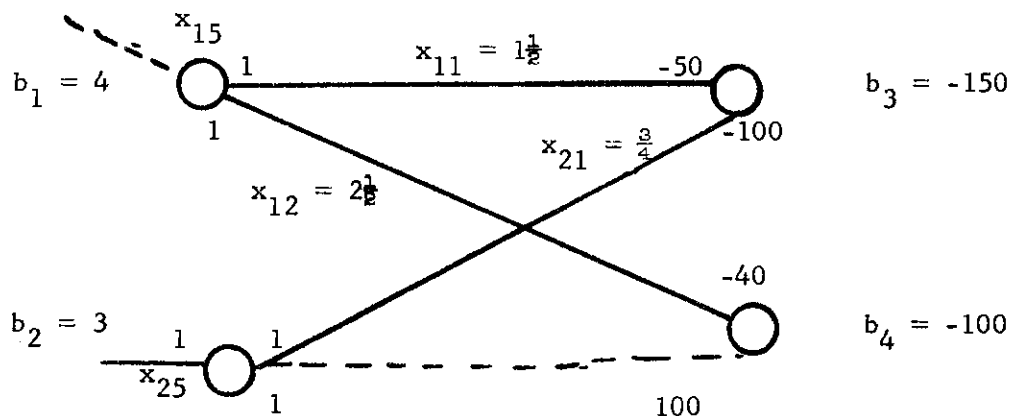
where

$\langle x \rangle$  denotes the integer vector obtained by rounding each component of  $x$  to the nearest integer value and  $B^{-1}$  is the inverse of the optimum basis.

If  $x'$  is feasible to all of the nonbinding constraints, then it is a feasible integer solution. If  $x'$  is not feasible, either the procedure may terminate and the branch and bound search may be started with no lower bound or, as suggested by Hillier [53], a search may be made between  $x_{LP}$  and  $x'$  for a feasible solution.

For the GFP,  $b'_i$  can be calculated by scanning the variables and calculating a new right hand side value if  $x_i$  is basic. The inverse of the basis is not explicitly available, but it can be generated a row at a time by the procedure in Section 3.4. The only equations to be checked after  $x'$  is calculated are the bounds 0 and  $M_i$  and the constraints corresponding to basic slack arcs in the graphical representation, since they are the only ones which are not binding at the LP optimum. An example is given below.

The optimal solution to the problem defined by Eqs. (15) and (16) in Chapter I displayed on the associated graph are:



$$b_1' = 4 - \frac{1}{2}(1+1) = 3$$

$$b_2' = b_2 = 3$$

$$b_3' = -150 - \frac{1}{2}(1-50 + 1-100) = -225$$

$$b_4' = -100 + \frac{1}{2}(1-40) = -120$$

The row of the inverse for  $x_{25}$  is:

$$\left[1 \quad \frac{1}{100} \quad \frac{1}{2} \quad \frac{1}{80}\right]$$

Thus

$$x_{25}' = \begin{bmatrix} 1 & 1/100 & 1/2 & 1/80 \end{bmatrix} \begin{bmatrix} 3 \\ -225 \\ 3 \\ -120 \end{bmatrix} = \frac{3}{4}$$

For  $x_{21}$ :

$$x_{21}' = \begin{bmatrix} 0 & -1/100 & -1/2 & -1/80 \end{bmatrix} \begin{bmatrix} 3 \\ -225 \\ 3 \\ -120 \end{bmatrix} = 2\frac{1}{4}$$

For  $x_{11}$ :

$$x'_{11} = [0 \quad 0 \quad 1 \quad 1/40] \begin{bmatrix} 3 \\ -225 \\ 3 \\ -120 \end{bmatrix} = 3$$

The rounded solution is:

$$x'_{11} = 0 \quad x'_{12} = 3 \quad x'_{21} = 2 \quad x'_{22} = 0$$

The solution is feasible with a cost of 430 which would serve as a lower bound for the branch and bound procedure.

## CHAPTER VII

### CONCLUSIONS AND RECOMMENDATIONS

#### 7.1 Conclusions

The primary motivation for the research presented in this dissertation was the search for a solution procedure for the integer generalized flow problem. For this purpose it was necessary to obtain an effective solution procedure for the continuous problem. In addition, in proceeding towards this goal, we have obtained several worthwhile results related to the area of primary investigation. The main accomplishments and results of this research are elaborated below.

As mentioned previously, the main trend of thinking in this study was guided by Johnson's characterization of a basis for a network problem in terms of a basis forest. This graphical representation has been proved in this study using a constructive approach. The approach taken is organized into a procedure for row and column generation to carry out the simplex operations. The procedure directly exploits the sparseness of the constraint matrix of the GFP and minimizes the amount of computation effort.

The procedure mentioned above directly leads to an efficient means of computing the basis inverse of the GFP. This is used to compute the dual variables and the relative cost factors. Ability to compute the inverse in this fashion is also likely to have a strong impact on the

additional research areas listed later. It may be recalled that the methods of Balas [3,4], Johnson [58], and Maurras [71,72] require the recalculation of all simplex multipliers, whereas, in this study, only those multipliers which change are identified and recomputed at each iteration.

To keep track of the basis and the simplex computations, Johnson [59] has proposed a triple labeling scheme for the OFP. The scheme is extended to the GFP in this study.

The information derived from the GFP is used to solve the IGFP. The graphical representation of the GFP is shown to provide a good means of viewing the group theoretic formulation of the IGFP and, in particular, the graph can be used to identify the interaction between variables.

A means for directly calculating the determinant of an LP basis is given, as is a method for improving the group formulation. The constructive procedure and the continuous algorithm for the GFP are used to construct a branch and bound algorithm for the IGFP.

In addition to the development of algorithms for solving the GFP and the IGFP and the associated detailed steps, the study has demonstrated that the resulting procedures are effective. The solution times given in Table 2 for the GFP are indicative of the power of the methods. The GFP algorithm is specialized for the OFP and transportation problem. The computational results for the OFP algorithm are better than those for the out-of-kilter algorithm, long thought to be the best algorithm for minimum cost flow problems. The computational results for the transportation code are comparable with recent results by other investigators.

Table 5 lists the computation times for the IGFP algorithm. There are no effective algorithms to compare with these results.

## 7.2 Recommendations

Several areas for additional research are indicated by the investigation reported here. Some of these are listed below.

(1) A primal-dual algorithm for the OFP could be constructed to take advantage of the graphical representation of basic solutions. This representation can be viewed as an efficient way to identify sets of variables and/or equations. For example, the set of variables required to represent a nonbasic variable is directly identifiable using the graph. However, in the out-of-kilter algorithm, a search must be made (labeling) for this set which is identified at a breakthrough. Similarly, the set of simplex multipliers which change at an iteration is easily identified and updated using the tree representation. The out-of-kilter method must accomplish a similar function by identifying labeled and unlabeled sets of nodes and the set of arcs between them. The main requirement for a primal-dual algorithm using the graphical representation would be a means of identifying the set of variables which forms the restricted primal problem.

(2) Based on the results for the transportation algorithm, an investigation of improved starting solutions and entering variable selection criteria should be made for both the GFP and the OFP.

(3) The graphical interpretation of the group formulation for the IGFP may provide the means for constructing a group theoretic algorithm. Since the interaction between variables is identifiable, the use of this

information in a group theoretic algorithm should be studied.

(4) The computational results for the IGFP indicate that a great deal of the effort is spent in identifying the first integer solution. The heuristic method for finding a feasible solution presented in Section 6.5 should be implemented and tested computationally.

(5) The graphical representation of the GFP and the OFP provides a means for efficiently performing parametric analysis after the original problem has been solved. For example, if the cost of a basic variable changes, the only simplex multipliers affected until the basis changes are those associated with nodes above that basic variable whose cost changed. Likewise, if a component of the right hand side changes, the only basic variables whose values change until a basis change occurs, are those in the path from the node associated with the changed right hand side component and the root (pseudoroot). Clearly, the parametric programming potential of the graphical representation should be exploited.

## BIBLIOGRAPHY

1. Arms, R. L., "An Algorithm for a Special Class of Generalized Transportation-Type Problems," Research Analysis Corp. Report TP-337 (1968).
2. Azpeitia, A. G. and Dickinson, D. J., "A Decision Rule in the Simplex Method that Avoids Cycling," Numerische Mathematik, Vol. 6, 329-331 (1964).
3. Balas, E. and P. L. Ivanescu, "On the Generalized Transportation Problem," Management Science, Vol. 11, 188-203 (1964).
4. Balas, E., "The Dual Method for the Generalized Transportation Problem," Management Science, Vol. 12, 555-568 (1966).
5. Balas, E., "Discrete Programming by the Filter Method," Operations Research, Vol. 15, 915-957 (1967).
6. Balas, E., "Integer Programming and Convex Analysis: Intersection Cuts from Outer Polars," Mathematical Programming, Vol. 2, 330-382 (1972).
7. Balinski, M. L. and K. Spielberg, "Methods for Integer Programming: Algebraic, Combinatorial, and Enumerative," Chapter 7 in Progress in Operations Research, ed. J. Aronofsky, John Wiley and Sons, 1969.
8. Balinski, M. L., "On Maximum Matching, Minimum Covering and Their Connections," in the Proceedings of the Princeton Symposium on Mathematical Programming, ed. H. W. Kuhn, Princeton University Press, 1970.
9. Biondi, E. and R. Schmid, "An Approximate Algorithm for Discrete Linear Programming," IEEE Transactions on Systems Science and Cybernetics, Vol. SSC-5 (1969).
10. Burdet, C. A., "Enumerative Inequalities in Integer Programming," Mathematical Programming, Vol. 2, 32-64 (1972).
11. Burdet, C. A., "On the Algebra and Geometry of Integer Programming Cuts," Management Science Research Report 291, Carnegie-Mellon University GSIA (1972).
12. Charnes, A. and C. Lemke, "Minimization of Nonlinear Separable Convex Functions," Naval Research Logistics Quarterly, Vol. 1, 301-312 (1954).

## BIBLIOGRAPHY (Continued)

13. Charnes, A. and W. M. Raike, "One Pass Algorithms for Some Generalized Network Problems," Operations Research, Vol. 14, 914-924 (1966).
14. Dakin, R. J., "A Tree Search Algorithm for Mixed Integer Programming Problems," Computer Journal, Vol. 8, 250-255 (1965).
15. Dantzig, G., Linear Programming and Extensions, Princeton University Press, 1963.
16. Demmy, W. S., "Multicommodity Flows in Generalized Networks," Memorandum Report No. 9, Research Division, Advanced Logistics Systems Center, Headquarters AFLC (1969).
17. Driebeck, N. J., "An Algorithm for the Solution of Mixed Integer Programming Problems," Management Science, Vol. 12, 576-587 (1966).
18. Edmonds, J. and E. Johnson, "Matching: A Well Solved Class of Integer Linear Programs," in Calagary International Conference on Combinatorial Structures and Their Applications-1969, Gordon & Breach, 1970.
19. Eisenman, K., "The Generalized Stepping Stone Method for the Machine Loading Model," Management Science, Vol. 11, 154-176 (1964).
20. Estabrook, J. R., Jr., An Integer Programming Method for the Generalized Transportation Problem (Unpublished Dissertation), Dept. of Industrial Engineering, Columbia University (1969).
21. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, 1962.
22. Forrest, J. J. H., J. P. H. Hirst, and J. A. Tomlin, "Practical Solution of Large and Complex Integer Programming Problems with UMPIRE," presented at the XIX International Meeting of TIMS (1972).
23. Fujisawa, T., "Maximum Flows in Lossy Networks," Proceedings of the First Allerton Conference on Circuit and Systems Theory, 1963.
24. Garfinkel, R. S. and G. L. Nemhauser, Integer Programming, John Wiley and Sons, 1972.
25. Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, Vol. 17, 437-454 (1969).
26. Geoffrion, A. M. and R. E. Marsten, "Integer Programming Algorithms: A Survey," Management Science, Vol. 18, 465-491 (1972).

## BIBLIOGRAPHY (Continued)

27. Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, Vol. 13, 879-919 (1965).
28. Glover, F., "Surrogate Constraints," Operations Research, Vol. 16, 741-749 (1968).
29. Glover, F. and D. Klingman, "Locating Stepping-Stone Paths in Distribution Problems Via the Predecessor Index Method," Transportation Science, Vol. 4, 220-225 (1970).
30. Glover, F., D. Klingman, A. Napier, and D. Kearny, "A Comparison of Computation Times for Various Starting Procedures, Basis Change Criteria, and Solution Algorithms for Distribution Problems," Report CS 44, Center for Cybernetic Studies, University of Texas-Austin (1971).
31. Glover, F. and D. Klingman, "On the Equivalence of Some Generalized Network Problems to Pure Network Problems," Report CS 81, Center for Cybernetic Studies, University of Texas-Austin (1972).
32. Glover, F. and D. Klingman, "Basic Dual Feasible Solutions for a Class of Generalized Networks," Operations Research, Vol. 20, 126-136 (1972).
33. Glover, F. and D. Klingman, "Dual Approximate Methods for the Distribution Problem," Management Science, Vol. 18, 574-583 (1972).
34. Glover, F., D. Klingman, and D. Kearny, "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," Transportation Science, Vol. 6, 171-179 (1972).
35. Glover, F. and D. Klingman, "A Note on Computational Simplifications in Solving Generalized Transportation Problems," Report CS 87, Center for Cybernetic Studies, University of Texas-Austin (1972).
36. Glover, F., D. Klingman, D. Kearny, and A. Napier, "A Computation Study of Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," Report CS 93, Center for Cybernetic Studies, University of Texas-Austin (1972).
37. Glover, F., D. Klingman, and D. Kearny, "A Double-Pricing Dual Feasible Start Algorithm for the Capacitated Transportation (Distribution) Problem," Report CS 105, Center for Cybernetic Studies, University of Texas-Austin (1970, Rev. 1972).

## BIBLIOGRAPHY (Continued)

38. Glover, F., D. Klingman, and R. S. Barr, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," Report CS 102, Center for Cybernetic Studies, University of Texas-Austin (1972).
39. Glover, F., "Cut Search Methods in Integer Programming," Mathematical Programming, Vol. 3, 86-100 (1972).
40. Gomory, R. E., "On the Relation Between Integer and Non-Integer Solutions to Linear Programs," Proceedings of the National Academy of Science, Vol. 53, 260-265 (1956).
41. Gomory, R. E., "An Algorithm for the Mixed Integer Problem," RM-2597, The RAND Corp. (1960).
42. Gomory, R. E., "An All-Integer Programming Algorithm," in Industrial Scheduling, ed. Muth and Thompson, Prentice-Hall, 1963.
43. Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs," in Recent Advances in Mathematical Programming, ed. Graves and Wolfe, McGraw-Hill, 1963.
44. Gomory, R. E., "Some Polyhedra Related to Combinatorial Problems," Linear Algebra and Its Applications, Vol. 2, 454-550 (1969).
45. Gomory, R. E., "Properties of a Class of Integer Polyhedra," Chapter 16 in Integer and Nonlinear Programming, ed. J. Abadie, Academic Press, 1970.
46. Gomory, R. E. and E. Johnson, "Some Continuous Functions Related to Corner Polyhedra," Mathematical Programming, Vol. 3, 23-85 (1972).
47. Gomory, R. E. and E. Johnson, "Some Continuous Functions Related to Corner Polyhedra-II," Mathematical Programming, Vol. 3, 359-389 (1972).
48. Gorry, G. A. and J. F. Shapiro, "Computational Experience with a Group Theoretic Integer Programming Algorithm," WP 603-72, Sloan School of Management, Massachusetts Institute of Technology (1972).
49. Grinold, R. C., "Calculating Maximal Flows in a Network with Positive Gains," WP CP-337, Center for Research in Management Science, University of California-Berkeley (1971).
50. Guignard, M. M. and K. Spielberg, "Mixed Integer Algorithms for the (0,1) Knapsack Problem," IBM Research and Development Journal, Vol. 16, 424-430 (1972).

## BIBLIOGRAPHY (Continued)

51. Hadley, G., Linear Programming, Addison-Wesley, 1963.
52. Hildebrand, F. B., Methods of Applied Mathematics, Prentice-Hall, 1965.
53. Hillier, F. S., "Efficient Heuristic Procedures for Integer Linear Programming with an Interior," Operations Research, Vol. 17, 600-637 (1969).
54. Hu, T. C., Integer Programming and Network Flows, Addison-Wesley, 1969.
55. Jensen, P. A., Private Communication (Oct., 1972).
56. Jewell, W. S., "Optimal Flow Through Networks with Gains," Operations Research, Vol. 10, 478-499 (1962).
57. Jezior, A. M. and J. J. Jarvis, "Maximal Flow with Gains Through a Special Network," Operations Research, Vol. 20, 678-688 (1972).
58. Johnson, E. L., "Programming in Networks and Graphs," ORC Report 65-1, University of California-Berkeley (1965).
59. Johnson, E. L., "Networks and Basic Solutions," Operations Research, Vol. 14, 619-623 (1966).
60. Johnson, E. L. and K. Spielberg, "Inequalities in Branch and Bound Programming," IBM Research Report RC-3649 (1971).
61. Johnson, E. L., Private Communication (May and Sept., 1972).
62. Johnson, E. L., "Cyclic Groups, Cutting Planes, and Shortest Paths," presented at the Mathematical Programming Seminar, Mathematics Research Center, University of Wisconsin-Madison (1972).
63. Kennington, J. L., Fixed-Charge Transportation Problem: A Group Theoretic Approach (Unpublished Dissertation), Dept. of Industrial and Systems Engineering, Georgia Institute of Technology (1973).
64. Klingman, D., A. Napier, and J. Stutz, "NETGEN--A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," Report CS 109, Center for Cybernetic Studies, University of Texas-Austin (1973).
65. Land, A. H. and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," Econometrica, Vol. 28, 497-520 (1960).

## BIBLIOGRAPHY (Continued)

66. Lasdon, L. S., Optimization Theory for Large Systems, The MacMillan Company, 1970.
67. Lourie, J. R., "Topology and Computation of the Generalized Transportation Problem," Management Science, Vol. 11, 177-187 (1964).
68. Maier, S. F., "Maximal Flows Using Spanning Trees," Report 71-14, Operations Research House, Stanford University (1971).
69. Malek-Zavarei, M. and J. K. Aggarwal, "Optimal Flow in Networks with Gains and Costs," Networks, Vol. 1, 355-365 (1972).
70. Martin, G. T., "An Accelerated Euclidean Algorithm for Integer Linear Programming," in Recent Advances in Mathematical Programming, ed. Graves and Wolfe, McGraw-Hill, 1963.
71. Maurras, J. F., "Optimization of the Flow Through Networks with Gains," Mathematical Programming, Vol. 3, 135-144 (1972).
72. Maurras, J. F., Private Communication (Feb., 1973).
73. Mayeda, W. and M. E. Van Valkenburg, "Properties of Lossy Communications Nets," IEEE Transactions on Circuit Theory, Vol. CT-12, 334-338 (1965).
74. Minieka, E., "Optimal Flow in a Network with Gains," INFOR, Vol. 10, 171-178 (1972).
75. Mitten, L. G., "Branch and Bound Methods: General Formulation and Properties," Operations Research, Vol. 18, 24-34 (1970).
76. Onaga, K., "Optimum Flows in General Communications Networks," Journal of the Franklin Institute, Vol. 283, 308-327 (1967).
77. Onaga, K., "Dynamic Programming of Optimum Flows in Lossy Communications Nets," IEEE Transactions on Circuit Theory, Vol. CT-13, 282-287 (1966).
78. Padberg, M. W., "On the Facial Structure of Set Covering Problems," Report I/72-13, International Institute of Management, Science Center Berlin (1972).
79. Rutenberg, D. P., "Generalized Networks, Generalized Upperbounding, and Decomposition of the Convex Simplex Method," Management Science, Vol. 16, 388-401 (1970).

## BIBLIOGRAPHY (Continued)

80. SHARE Programming Library Distribution No. 3536, "Out-of-Kilter Network Routine," (1967).
81. Shapiro, J. F., "Dynamic Programming Algorithms for the Integer Programming Problem-I: The Integer Programming Problem Viewed As a Knapsack Type Problem," Operations Research, Vol. 16, 103-121 (1968).
82. Shapiro, J. F., "Group Theoretic Algorithms for the Integer Programming Problem II: Extension to a General Algorithm," Operations Research, Vol. 16, 928-947 (1968).
83. Shapiro, J. F., "Turnpike Theorems for Integer Programming Problems," Operations Research, Vol. 18, 432-440 (1970).
84. Shapiro, J. F., "Generalized Lagrange Multipliers in Integer Programming," Operations Research, Vol. 19, 68-76 (1971).
85. Simonnard, M., Linear Programming, Prentice-Hall, 1962.
86. Smith, C. W., Maximal Flow at Minimal Cost Through a Special Network with Gains (Unpublished Thesis), Dept. of Ind. and Sys. Eng., Georgia Institute of Technology (1971).
87. Srinivasan, V. and G. L. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Applications to Distribution Problems," Journal of the ACM, Vol. 19, 712-726 (1972).
88. Takahashi, I., "Tree Algorithm for Solving Resource Allocation Problems," Journal of the OR Society of Japan, Vol. 8, 172-191 (1966).
89. Takahashi, I., "Tree Algorithm for Solving Network Transportation Problems," Journal of the OR Society of Japan, Vol. 8, 192-216 (1966).
90. Thiriez, H., "Airline Crew Scheduling: A Group Theoretic Approach," Report R-69, Flight Transportation Laboratory, M.I.T. (1969).
91. Thompson, G. L. and V. Srinivasan, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," Management Science Research Report WP-97-70-1, Carnegie-Mellon University (1970, Rev. Dec. 1971).
92. Tomizawa, N., "On Some Techniques Useful for Solution of Transportation Network Problems," Networks, Vol. 1, 173-194 (1971).

## BIBLIOGRAPHY (Concluded)

93. Tomlin, J. A., "Branch and Bound Methods for Integer and Nonconvex Programming," Chapter 21 in Integer and Nonlinear Programming, ed. J. Abadie, North Holland, 1970.
94. Tomlin, J. A., "An Improved Branch-and-Bound Method for Integer Programming," Operations Research, Vol. 19, 1070-1075 (1971).
95. Trotter, L. E., An Implicit Enumeration Algorithm for Integer Programming (Unpublished Thesis), Dept. of Ind. and Sys. Eng., Georgia Institute of Technology (1970).
96. Zionts, S., "Toward a Unifying Theory for Integer Linear Programming," Operations Research, Vol. 17, 359-367 (1969).

## VITA

Robert Warren Langley was born January 18, 1943 in Memphis, Tennessee. He was graduated from Huntsville High School, Huntsville, Alabama in June 1961, and he entered the United States Air Force Academy the same month. He was a Distinguished Graduate in June 1965 receiving a B.S. in Engineering Science with a major in Astronautics.

His first assignment after receiving his commission was as a graduate student at the Massachusetts Institute of Technology where in September 1966 he received a S.M. in Aeronautics and Astronautics with a major in Guidance and Control. From 1966 until 1970 Captain Langley was assigned to the Central Inertial Guidance Test Facility, Holloman AFB, New Mexico, first as a test analyst, then as a branch chief. He enrolled in the School of Industrial and Systems Engineering at the Georgia Institute of Technology in September 1970 and was awarded the Ph.D. degree in June 1973.