

AD-A173 105

DEVELOPMENT AND EVALUATION OF A CASUALTY EVACUATION

1/2

MODEL FOR A EUROPEAN (U) SOUTHERN METHODIST UNIV

DALLAS TX DEPT OF OPERATIONS RESEARCH... J L KENNINGTON

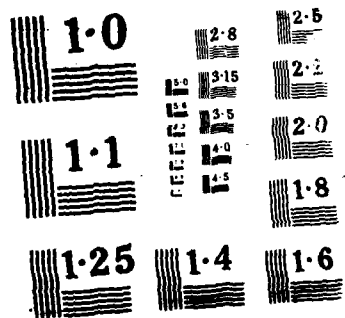
UNCLASSIFIED

DEC 85 AFOSR-IR-86-0935 AFOSR-83-0278

F/G 20/1

NL





AFOSR-TR- 86 - 0935

(2)

AD-A173 105

AN END OF THE YEAR REPORT TO

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

AND

HEADQUARTERS MILITARY AIRLIFT COMMAND

for a Grant

entitled

Approved for public release;
distribution unlimited.

DEVELOPMENT AND EVALUATION OF A CASUALTY EVACUATION MODEL
FOR A EUROPEAN CONFLICT

Prepared by:

Jeffery L. Kennington
Department of Operations Research
and Engineering Management

SOUTHERN METHODIST UNIVERSITY

Dallas, Texas 75275

December, 1985

DTIC
ELECTED
OCT 16 1986

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF CONFIDENTIALITY TO DTIC
This report has been reviewed and is
distributed in accordance with AFOSR 190-12.
MATTHEW J. HENNING
Chief, Technical Information Division

DTIC FILE COPY

86 10 16 142

UNCLASSIFIED

AD-A173105

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A			5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 86 - 0935		
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			7a. NAME OF MONITORING ORGANIZATION AFOSR		
6a. NAME OF PERFORMING ORGANIZATION Southern Methodist University		6b. OFFICE SYMBOL (If applicable)	7b. ADDRESS (City, State and ZIP Code) Building 410 Bolling AFB, D.C. 20332-6448		
6c. ADDRESS (City, State and ZIP Code) Department of Operations Research & Engineering Management Dallas TX 75275		7c. ADDRESS (City, State and ZIP Code)			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR		8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR 83-0278		
8c. ADDRESS (City, State and ZIP Code) Building 410 Bolling AFB D.C. 20332-6448		10. SOURCE OF FUNDING NOS.			
		PROGRAM ELEMENT NO. 61102F	PROJECT NO. 2304	TASK NO. A	WORK UNIT NO.
11. TITLE (Include Security Classification) DEVELOPMENT & EVALUATION OF A CASUALTY EVACUATION MODEL FOR EUROPEAN CONFLICT					
12. PERSONAL AUTHOR(S) JEFFERY L. KENNINGTON					
13a. TYPE OF REPORT Annual	13b. TIME COVERED FROM 1 Dec 85 TO 30 Nov 85	14. DATE OF REPORT (Yr., Mo., Day) December, 1985		15. PAGE COUNT 190	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GR.			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The mathematical model presented in this paper is beyond the state-of-the-art for existing mathematical programming software. Based on this working paper and discussions with LtCol McLain, Professor Kennington Proposed a research plan to refine the McLain model and computationally investigate new algorithms for solving the model. McLain and Chmielewski collected the data and developed the model generator while Kennington and his students investigated alternative solution algorithms.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐			21. ABSTRACT SECURITY CLASSIFICATION		
22a. NAME OF RESPONSIBLE INDIVIDUAL DR MARC Q. JACOBS Program Manager, Mathematical & Information Sciences		22b. TELEPHONE NUMBER (Include Area Code) 767-4939	22c. OFFICE SYMBOL NM		

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

This work was initiated by Lt. Col. Dennis R. McLain and Mr. Thomas E. Kowalsky of DCS/Plans, Headquarters Military Airlift Command at Scott AFB. Lt. Col. McLain has transferred to the Pentagon and the work at Scott is being continued by Mr. Kowalsky and Capt. Robert Chmielewski. The work originated with a November 1982 working paper by McLain entitled "Wartime Evacuation of European Theater Casualties to the United States".

↙ The mathematical model presented in this paper is beyond the state-of-the-art for existing mathematical programming software. Based on this working paper and discussions with Lt. Col. McLain, Professor Kennington proposed a research plan to refine the McLain model and computationally investigate new algorithms for solving the model. McLain and Chmielewski collected the data and developed the model generator while Kennington and his students investigated alternative solution algorithms. — > to p. 1

This model is a member of the class of optimization models known as networks with side constraints. Within this general area, Kennington managed three projects with three different students. Dr. Ellen Allen's dissertation was directed toward multicommodity problems and appears in Appendix A. A related study involving the Equal Flow Problem was undertaken by Dr. Bala Shetty. This study is reported in Appendix B. New convergence results related to both studies is presented in Appendix C. A study on the general problem of networks with side constraints was pursued by Dr. Keyvan Farhangian. The results of that report are presented in Appendix D which is scheduled to appear in The Annals of the Society of Logistics Engineers.

x -1-

During 1984, Dr. Narendra Karmarkar, of Bell Laboratories, reported that he had developed a new technique to solve linear programs based upon a projective algorithm. He claimed a worst-case bound on solution steps that was much better than the simplex method, and that his computer software was at least 50 times faster than IBM's best implementation of Dantzig's simplex algorithm. This new work could have major implications for the Casualty Evacuation Model and Kennington spent 7 months, full time, investigating this new algorithm. This investigation is reported in Appendix E. The group found that while the new method clearly worked, it was much slower than their existing software and much slower than the claims of Dr. Karmarkar. As of today, no group has been able to substantiate the claims of Dr. Karmarkar. Kennington and his team is not, at this time, investigating the projective algorithm.

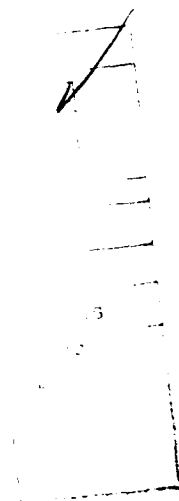
While Lt. Col. McLain and Capt. Chmielewski were completing the model generator, Professor Kennington and a former student (Mr. David Allen) worked on a long-standing problem in the area of military communications system design. The problem is to assign frequencies to nodes to minimize both co-channel and adjacent channel interference. They designed an optimization model and a heuristic solution procedure which virtually solves this problem. Problems with 600 binary decision variables were solved in only 10 seconds on a moderately-sized mainframe. This work is reported in Appendix F and this paper will appear in Naval Research Logistics Quarterly.

During the last three months, Chmielewski and Kennington have been debugging the Casualty Evacuation Model and developing the appropriate

output reports from Kennington's solution software. The work so far has been with a 4-day model. The goal is to solve a full 90-day model during the Spring of 1986.



A1



3 iii

Appendix A

USING TWO SEQUENCES OF PURE NETWORK PROBLEMS TO SOLVE
THE MULTICOMMODITY NETWORK FLOW PROBLEM

A Dissertation Presented to the Graduate
Faculty of the School of Engineering
and Applied Science

of

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

with major in

Operations Research

by

Ellen Parker Allen

B.A.S., Southern Methodist University, 1975

M.S.O.R., Southern Methodist University, 1981

May 1985

Allen, Ellen Parker

B.A.S., Southern Methodist University, 1975
M.S.O.R., Southern Methodist University, 1981

USING TWO SEQUENCES OF PURE NETWORK PROBLEMS TO SOLVE THE
MULTICOMMODITY NETWORK FLOW PROBLEM

Advisor: Professor Jeffery L. Kennington

Doctor of Philosophy degree conferred May, 1985

Dissertation completed April, 1985

This dissertation presents a new technique for solving very large scale multicommodity network flow problems. The method obtains successively better lower and upper bounds on the optimal objective function value, stopping whenever the two bounds are within a prescribed tolerance. Lower bounds are generated by partially solving a Lagrangian dual of the original problem. Upper bounds are generated by partially solving the multicommodity network problem itself, using a resource directive decomposition scheme.

One great advantage to our approach is this: in both the lower and upper bound routines, the problems decompose on commodities. As a result, only a single commodity minimum cost network flow optimizer is needed. Taking advantage of this natural decomposition also allows our technique to solve substantially larger problems than other multicommodity network codes.

A handwritten signature in black ink, appearing to read 'J. L. Kennington', with a stylized 'J' and 'K'.

TABLE OF CONTENTS

	page
ABSTRACT.	iv
LIST OF TABLES.	vi
ACKNOWLEDGEMENTS.	vii
CHAPTER I. INTRODUCTION.	1
1.1 Notation and Conventions	2
1.2 Problem Definition	4
1.3 The Casualty Evacuation Model.	6
1.4 Accomplishments of This Investigation.	8
CHAPTER II. A SURVEY OF RELATED LITERATURE	10
2.1 Pure Networks	10
2.2 Multicommodity Networks.	12
2.2.1 Partitioning Algorithms	12
2.2.2 Decomposition Algorithms.	13
2.3 Subgradient Optimization	14
CHAPTER III. THE ALGORITHM	16
3.1 Subgradient Optimization	17
3.2 Generating Lower Bounds.	23
3.3 Generating Upper Bounds.	27
3.4 The Algorithm.	34
CHAPTER IV. COMPUTATIONAL EXPERIMENTATION	36
4.1 Description of the Computer Programs	37
4.1.1 MCNF.	37
4.1.2 EVAC.	37
4.2 Description of the Test Problems	39
4.3 Summary of Computational Results	39
4.4 Analysis of Results.	41
CHAPTER V. SUMMARY AND CONCLUSIONS	47
5.1 Summary and Conclusions.	47
5.2 Areas for Future Investigation	48
LIST OF REFERENCES	50

LIST OF TABLES

Table	Page
4.1 Description of the Test Problems and Summary Comparison of Solution Times for EVAC and MCNF	44
4.2 Detailed Timing Statistics for EVAC Runs	45
4.3 Graphical Comparison of EVAC and MCNF Solution Times	46

ACKNOWLEDGEMENTS

I wish to express my gratitude to Dr. Jeff Kennington for his support and encouragement throughout my graduate studies. As my academic advisor and dissertation advisor his guidance and assistance have been invaluable.

I would like to express my deepest appreciation to my husband, Dave Allen, for his constant support, for his loving encouragement, and for continually believing in me. I also wish to thank my precious daughter, Angela Joy Allen, who, by her very arrival, provided motivation for me to finish this work.

In addition, I gratefully acknowledge my parents, Charlie and Betty Parker, for always encouraging me to pursue my dreams, for their financial support through my undergraduate years, and for their continuing emotional support.

I want to express my appreciation to Dr. Dick Helgason for his indispensable insights regarding the convergence results in this work, and for serving on my committee.

I want to thank Drs. Jay Aronson, Dick Barr, and Chuck Gartland, for graciously serving on my committee and for their helpful and constructive suggestions.

I would like to acknowledge Sheila Crain for her masterful typing of this dissertation.

I wish to acknowledge Lt. Col. Dennis R. McLain for arranging the support for this work through the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278.

Finally, I wish to thank my colleagues and friends in the Department of Operations Research and Engineering Management, especially Julie Ellis, Belinda Hargrove, Bruce Patty, and Bala Shetty, for helping to make graduate school one of the most enjoyable experiences of my life.

CHAPTER I

INTRODUCTION

This dissertation presents a new technique for solving very large multicommodity network flow problems. The specific application which motivated this work originated with the United States Air Force and was first presented to us by Lt. Col. Dennis McLain, the Assistant Director of Operations Research for the Military Airlift Command at Scott Air Force Base. The problem is an extremely large casualty evacuation model to be used by the Air Force in forming a plan for the evacuation of wartime casualties. This plan would be implemented in case of a European military conflict involving United States troops. Lt. Col. McLain was the first to model this problem as a multicommodity network flow problem where the commodities correspond to the various types of wounds. The nodes represent such entities as European bases and United States medical facilities, and the arcs represent specific aircraft flights. (A complete description of this problem is given in Section 1.3.) This problem is far too large to be solved by any known existing computer codes. In addition, since many of the data are only rough estimates (the number of casualties of various types expected at given locations), an exact technique is not called for. Instead a technique is needed to discover a guaranteed ϵ -optimum for any given $\epsilon > 0$.

This is precisely what our technique accomplishes. It generates successively better upper and lower bounds on the optimum, stopping when the two bounds are within a prescribed tolerance. We exploit the multicommodity network structure in both the lower and upper bound routines so that only a single commodity minimum cost network flow optimizer is needed. EVAC, the computer code which implements our technique, has been used to solve a series of test problems in less time and requiring less memory than MCNF, a specialized multi-commodity network flow problem solver. In addition EVAC is capable of solving very large problems which MCNF is unable to solve.

1.1 Notation and Conventions

The notational conventions employed throughout this work are described in this section. Matrices are denoted by upper case Latin letters. The element of a matrix, A , which appears in the i^{th} row and j^{th} column is indicated by A_{ij} . The symbol I is used to denote an identity matrix with dimension appropriate to the context. Lower case Latin and Greek letters are used to denote vectors. The symbol 0 is used to represent a vector of zeroes with dimension appropriate to the context. The unit vector, whose only non-zero component is a one in the j th position, is denoted e_j . Subscripts are used to indicate individual components of a vector, or as an index to indicate which of

a sequence of related vectors is meant. Superscripts on vectors correspond to individual commodities. Note that vectors are considered to be row vectors or column vectors as appropriate to the context; that is, no special notation is used to indicate the transpose of a vector. The inner product of two vectors, x and y , is denoted simply by xy . The notation $||x||$ is used to express the Euclidian norm, $(xx)^{1/2}$. Scalars are written as lower case Greek or Latin letters.

Euclidean n -dimensional space is denoted R^n . Functions are written as lower case Latin letters, and functional values have their arguments in parentheses. For example $g(y)$ is used to denote the function g evaluated at the point y . The one exception to this convention is the projection operation described in Chapter III. In this case $P[x]$ is used to express the projection of x onto the specified region.

Upper case Greek letters denote sets, with the exception that $\partial g(y)$ is used to denote the set of subgradients of a function g at a point y in the domain of g . The symbol ϵ is used as the set inclusion symbol and as a termination tolerance.

We use $\text{MAX}\{S\}$ to denote the largest element of a set S ; similarly $\text{MIN}\{S\}$ indicates the smallest element of S . The symbol ∞ is used for infinity, and $\blacksquare$ denotes the end of a proof. All other notation is standard.

1.2 Problem Definition

A network is composed of two entities: nodes and arcs. The arcs may be viewed as unidirectional means of commodity transport, and the nodes may be thought of as locations or terminals connected by the arcs and served by whatever physical means of transport are associated with the arcs. We limit our consideration to networks with finite numbers of nodes and arcs. For a given network we denote the number of nodes by m and the number of arcs by n . We impose an ordering on the nodes and arcs so as to put them in a one-to-one correspondence with the integers $\{1, \dots, m\}$ and $\{1, \dots, n\}$, respectively. The structure of a given network may be described, then, by an $m \times n$ matrix called a node-arc incidence matrix. Such a matrix, A , is defined in this way:

$$A_{ij} = \begin{cases} +1, & \text{if arc } j \text{ is directed away from node } i \\ -1, & \text{if arc } j \text{ is directed toward node } i \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, for a multicommodity network, we are concerned with more than one type of item (commodity) flowing through the arcs. We order these commodities to correspond to the integers $\{1, \dots, K\}$.

We define the following quantities to be used in the formulation of the multicommodity network flow problem:

- A is the $m \times n$ node-arc incidence matrix corresponding to the underlying network.
- x^k is an n vector of decision variables for $k = 1, \dots, K$. Note that x_j^k represents the amount of flow of commodity k on arc j .

-- c^k is an n vector of unit costs for $k = 1, \dots, K$. So c_j^k denotes the cost for one unit of flow of commodity k on arc j .

-- r^k is an m vector of requirements for $k = 1, \dots, K$, so that r_i^k denotes the required number of units of commodity k at node i . If $r_i^k < 0$ then node i is said to be a demand node for commodity k with demand = $|r_i^k|$. If $r_i^k > 0$ then node i is said to be a supply node for commodity k with supply = r_i^k . And if $r_i^k = 0$ then node i is said to be a transshipment node for commodity k .

-- u is an n vector of mutual arc capacities. That is, the total flow of all the commodities combined for arc j cannot exceed u_j .

-- v^k is a n vector of arc capacities for commodity k ($k = 1, \dots, K$). v_j^k , then, represents an upper bound on the flow of commodity k on arc j .

We sometimes refer to the entire vector of decision variables $(x^1, \dots, x^K)$ as simply x . Similarly we use c , r , and v to denote the entire vector of costs, requirements and upper bounds, respectively.

Using these ideas, we may formulate the multicommodity network flow problem for a given network with m nodes, n arcs, and K commodities as follows:

$$\text{Minimize } \sum_k c^k x^k$$

$$\text{Subject to } Ax^k = r^k, \quad k = 1, \dots, K \quad (\text{MP})$$

$$\sum_k x^k \leq u$$

$$0 \leq x^k \leq v^k, \quad k = 1, \dots, K.$$

1.3 The Casualty Evacuation Model

A large European military conflict involving U.S. Armed Forces could result in more casualties than could be effectively handled in European medical facilities. To alleviate this overcrowding, the Department of Defense plans to implement the following evacuation policy:

"During the first 30 days of a conflict, if a wounded soldier cannot be returned to duty within 15 days, then he will be evacuated to a medical facility in the United States. In the next 30 days the limit on treatment time is increased to 30 days."

Given a scenario concerning such a conflict (i.e. the number and locations of wounded and the types of wounds), this evacuation problem may be modelled as a multicommodity network flow problem. Lt. Col. Dennis McLain was the first to model the problem in this way. In Lt. Col. McLain's model the nodes correspond to 9 European recovery bases and 95 United States locations. The arcs correspond to aircraft flights connecting European and U.S. facilities. The commodities are 11 different patient types.

In order to enforce a capacity on a given facility, it is necessary to duplicate the corresponding node using the capacity as an upper bound on the arc between the duplicate nodes. For example, if node A represents a hospital with 300 beds, then we substitute two

nodes, A1 and A2, along with an arc whose capacity is 300. Further, it is necessary to include 60 copies of the entire network, one for each of the 60 one-day time periods. Additional arcs are created to link each time period to the next. The model includes a dummy "sink" node for each time period and one "super sink" node, along with capacitated arcs to allow patients who have recovered to exit from the system. These considerations produce a very large model. The dimensions of the constraint matrix are shown below:

$$\left[\begin{array}{c|c|c|c|c|c|c} A_1 & & & & & & \\ \hline & A_2 & & & & & \\ \hline & & . & & & & \\ \hline & & & . & & & \\ \hline & & & & . & & \\ \hline & & & & & A_{11} & \\ \hline I & I & . & . & . & I & I \end{array} \right] \left. \vphantom{\begin{array}{c|c|c|c|c|c|c} \end{array}} \right\} 12,541 \text{ rows}$$

where $A_1 = \dots = A_{11}$. The row dimension of this model is over 137,000, which is far beyond the scope of any known existing computer code. To put these figures in perspective, we note that Kennington reports that the largest models he has solved using his primal partitioning code, MCNF, have been on the order of 3000 rows [2].

Our plan has been to develop a specialized solution procedure which would solve a scaled-down version of Lt. Col. McLain's model. We anticipate aggregation of the data, possibly using some of the following ideas:

- (1) Aggregation of the time periods. Note that simply using 3-day time periods instead of 1-day time periods reduces the problem size to around 46,000 rows.
- (2) Aggregation of similar patient types.
- (3) Aggregation of U.S. medical facilities so that facilities which are located within a given number of miles of one another are treated as one node.

At the writing of this dissertation we have not yet received any large test problems from the Air Force. As a result, we are unable to report on the problem size limitations of our technique. However, in an attempt to test our software on a relatively large problem, we solved a randomly generated test problem with around 9,000 rows. (See Chapter 4 for the details of this problem.) This is the largest problem we have attempted so far.

1.4 Accomplishments of This Investigation

This dissertation proposes a new technique for solving extremely large multicommodity network flow problems. Our method involves generating upper bounds on the optimal objective value by partially solving the problem using a resource-directive decomposition technique, and generating lower bounds on the optimal objective value by partially solving a Lagrangian dual of the problem. Both the upper and lower bound routines exploit the network structure of the problem, decomposing it by commodities and solving the resulting pure network problems. In the limit both bounds must converge to the optimal objective function value; in practice we stop when the difference between the two bounds is within some termination tolerance.

Whether solving for lower bounds or for upper bounds a subgradient optimization technique is used. At each iteration this procedure requires the computation of a subgradient, the selection of a step size, and a projection operation. In Section 3.1, we obtain a new convergence result for a particular class of subgradient procedures. Then, in Section 3.2, we introduce a new heuristic, closely related to the subgradient optimization procedure, which has worked well for our test problems.

Our technique has been tested on randomly generated test problems and on one problem which was formulated specifically to represent the class of evacuation planning problems for which the code was developed. In addition, the same set of test problems was solved by MCNF [51], a general purpose multicommodity network flow problem solver which uses a primal partitioning scheme. Computation times for both codes are presented. Our code used an average of 68% of the time needed by MCNF, performing significantly better on the problems with fewer commodities. In addition our code required on the order of $1/K$ the amount of main memory for a K -commodity problem, so it can solve significantly larger problems than MCNF.

CHAPTER II

A SURVEY OF RELATED LITERATURE

In this chapter we present an overview of the existing work on which this dissertation is based. Section 2.1 deals with the work that has been done in the area of pure network models. Then in Section 2.2 we address the broader area of multicommodity network methods. Since our algorithm involves a subgradient optimization technique, both in the Lagrangian dual portion and in the resource-directive decomposition routine, we provide some references involving subgradient optimization in Section 2.3

2.1 Pure Networks

Network problems are linear programming problems with node-arc incidence matrices as their constraint matrices. Within this class, known formally as minimal cost network flow problems, there are several variations including transportation problems, transshipment problems, assignment problems, maximal flow problems, and shortest path problems.

Ideas for solution of network problems can be traced at least as far back as 1939, to the work of Professor Leonid Kantorovich [41]. Kantorovich, along with Professor Tjalling C. Koopmans received the Nobel Prize in Economic Science in 1975, for contributions to the theory of optimum allocation of scarce resources. Koopmans and Reiter

[54] and Frank L. Hitchcock [42], working independently, were the first to formulate the transportation problem. The mid-fifties saw a surge of interest and work in the areas of network algorithms. It was around this time at Alex Orden [59] generalized the transportation model to allow transshipment points. Lester Ford and Delbert Fulkerson [22] [20] formulated and investigated solution techniques for the maximal flow problem and the minimal cost network flow problem. The specialized algorithms that have been developed for solving network problems may be classified into two groups: primal-dual techniques, and specializations of the primal simplex algorithm. Primal-dual methods for solving networks began with Harold Kuhn's Hungarian Algorithm for the assignment problem [55] and culminated in Fulkerson's Out-of-Kilter Algorithm [23]. Primal simplex based techniques originated with the work of Professor George Dantzig [17] and continued through Ellis Johnson's 1965 paper [47]. The basis for Johnson's work can be traced to the work of Dantzig [18] and Charnes and Cooper [14].

Since that time much work has been done in the area of solution techniques, and computational advances have been made by the development of more efficient data structures. The credit for much of this work goes to Professors Fred Glover and Darwin Klingman and their colleagues at the University of Texas. This is evidenced by such papers as Barr, Glover and Klingman [9] [10], Glover, Hultz and Klingman [26] [25], Glover, Karney and Klingman [27], Glover, Karney, Klingman and Napier [28], Glover and Klingman [29] [31] [30], Glover, Klingman and Stutz [32], and Karney and Klingman [49]. Others who have contributed to the research include Srinivasan and Thompson [63] [64],

Bradley, Brown, and Graves [13], and Mulvey [57] [58]. In addition significant work has been performed by Professors Jeff Kennington, Richard Barr, and Richard Helgason of Southern Methodist University as seen in such works as [3], [41], and [52].

Today network algorithms have been demonstrated to solve linear network problems 50 times faster than general linear programming algorithms [6]. Additionally a computer implementation of such a technique may require only half the memory of the general L.P. package [6]. These advances are due to the efficient data structures which have been developed to allow a basis for a network problem to be stored as a rooted spanning tree on the nodes in the network. Using this idea all the simplex computations such as pricing, ratio test, and updates, can be performed via labelling algorithms on the basis tree. This eliminates the need to store the basis inverse in factored form.

2.2 Multicommodity Networks

Multicommodity network flow problems are problems in which several different types of items (commodities) must share arcs in a capacitated network. Each solution technique for multicommodity network models can be classified as one of two main types of algorithms: partitioning algorithms and decomposition algorithms.

2.2.1 Partitioning Algorithms

Partitioning algorithms are specializations of the simplex method which exploit the multicommodity network structure by partitioning the basis into more than one part. In one part advantage is taken of the

special network structure. Those who have studied primal partitioning algorithms include Kennington [50], Helgason and Kennington [40], Ali, Helgason, Kennington, and Lall [4], Hartman and Lasdon [36] [35], Maier [56], and Saigal [61]. Ali and Kennington [6], in their computational research, reported solution times averaging 5 times faster than general linear programming codes. A dual partitioning method was proposed by Grigoriadis and White [34]. A primal-dual partitioning scheme was developed by Jewell [46]. In addition a factorization technique was proposed by Graves and McBride [33]. MCNF, the multicommodity network code with which we compared our solution times, is a primal partitioning program.

2.2.2 Decomposition Algorithms

Decomposition schemes seek to solve the problem by decomposing it into several smaller subproblems, each of which takes the form of a pure minimum cost network flow problem. A master program coordinates the solution process. Decomposition procedures for the multicommodity network flow problem fall into two categories: price-directive schemes and resource-directive schemes.

Price-directive decomposition is based on the well-known research of Dantzig and Wolfe [19]. In a price-directive approach, the K-commodity problem is decomposed into K single commodity problems. The master program then uses the simplex method while the subproblems test for optimality and select candidates to enter the basis of the master problem. Ford and Fulkerson [21] were the first to develop this

approach for solving multicommodity network flow problems. Tomlin [67] was the first to develop a computer code implementing this technique. Others who have studied price-directive decomposition schemes are Jarvis [43], Jarvis and Keith [44], Chen and DeWald [15], and Jarvis and Martinez [45]. Price-directive approaches for generalizations of this problem have been proposed by Cremeans, Smith, and Tyndall [16], Swoveland [65] [66], Weigel and Cremeans [68], and Wollmer [69].

Resource-directive decomposition schemes decompose the problem by commodities, and the master problem systematically distributes the mutual arc capacities among the commodities. At each iteration the optimal solutions to the single commodity subproblems are used to compute a new set of allocations. Robacker [60] was the first to suggest this approach for multicommodity network problems. Research on this technique has been presented by Swoveland [65], Assad [8], Ali, Helgason, Kennington and Lall [3], and Kennington and Shalaby [53].

2.3 Subgradient Optimization

The subgradient optimization technique was first proposed by Shor [62] in 1964. Since that time subgradient algorithms have been applied to many different optimization problems. Held and Karp [37] and Held, Wolfe and Crowder [38] made use of the approach in solving the symmetric travelling salesman problem. Bazaraa and Goode [11] applied the algorithm to the asymmetric travelling salesman problem. Subgradient methods have been used to solve the assignment problem [38]. Glover, Glover and Martinson [24] applied a subgradient technique to

solve a special network with side constraints, and Ali and Kennington [7] made use of it in research involving the m -travelling salesman problem.

CHAPTER III

THE ALGORITHM

Here we present a new solution technique for the multicommodity network flow problem. This technique involves finding successively better upper and lower bounds on the optimal objective function value. The algorithm terminates whenever the two bounds are within a prescribed tolerance or when it can be shown that the current solution is an exact optimum.

Lower bounds are generated by partially solving a Lagrangian dual. At each iteration a Lagrangian relaxation of the original problem is solved; since these relaxations decompose on commodities, only a (single-commodity) minimum cost network flow optimizer is needed. A subgradient direction is used to adjust the Lagrange multipliers for the next iteration.

Upper bounds are generated using a modification of the resource-directive decomposition technique first suggested by Robacker [60]. We introduce a specialization of the subgradient direction approach which was first applied to this class of problems by Held, Wolfe, and Crowder [38].

With minor restrictions on the step sizes we show that both the upper and lower bounds converge to the optimal objective value of the original multicommodity network flow problem. Hence in the limit the algorithm will converge to an exact optimum. In practice we seek a near-optimum.

3.1 Subgradient Optimization

Let us first consider the general subgradient algorithm for optimization of convex functions; later we will present specializations of the technique for the upper and lower bound problems. Consider the nonlinear programming problem

Minimize $g(y)$

Subject to $y \in \Gamma$

where g is a real valued function that is convex over the compact, convex, nonempty set Γ . A vector n is called a subgradient of g at a point x if

$$g(y) - g(x) \geq n(y - x) \text{ for all } y \in \Gamma.$$

Note that if g is differentiable at x , the only subgradient at x is the gradient. We denote the set of all subgradients of g at x by $\partial g(x)$.

The subgradient algorithm proceeds in this manner: Given a point x in Γ , find a subgradient of g at x , obtain a new point by moving a given step size in the negative subgradient direction, and finally project this new point back onto Γ . This projection operation takes a point x and finds the point in Γ that is "closest" to x with respect to the Euclidean norm. We denote the projection of x onto Γ by $P[x]$. Using this notation we present the general subgradient optimization algorithm for minimizing a convex function g [52].

ALGORITHM 3.1 SUBGRADIENT OPTIMIZATION ALGORITHM

Step 0 (Initialization)

Let y_0 be any element of Γ . Select a set of step sizes, $s_1, s_2, s_3, \dots$, and set $i \leftarrow 0$.

Step 1 (Find Subgradient)

Let $\eta_i \in \partial g(y_i)$. If $\eta_i = 0$ terminate with y_i optimal.

Step 2 (Move to New Point)

Set $y_{i+1} \leftarrow P[y_i - s_i \eta_i]$. Set $i \leftarrow i + 1$. Return to step 1.

Let us now turn our attention to the selection of step sizes. Several ideas for choosing step sizes have been proposed. These typically involve a sequence of constants, $\{\lambda_1, \lambda_2, \lambda_3, \dots\}$ which satisfy the following conditions:

$$\begin{aligned} \lambda_i &> 0, \text{ for all } i, \\ \lim_{i \rightarrow \infty} \lambda_i &= 0, \text{ and} \\ \sum_i \lambda_i &= \infty. \end{aligned} \tag{3.1}$$

The subgradient algorithm can be shown to converge when any of the following three formulae are used for determining step sizes [52]:

$$\begin{aligned} \text{(i)} \quad s_i &= \lambda_i, \\ \text{(ii)} \quad s_i &= \lambda_i / \|\eta_i\|^2, \\ \text{(iii)} \quad s_i &= \lambda_i [g(y_i) - g^*] / \|\eta_i\|^2 \end{aligned} \tag{3.2}$$

where g^* denotes the optimal objective value.

Propositions 3.1, 3.2, and 3.3 may be found in Kennington and Helgason [52], and are given here as necessary preliminary results.

Proposition 3.1 [52]

Let $y \in \Gamma$, and let $x \in \mathbb{R}^n$. Then $(x - P[x])(y - P[x]) \leq 0$.

Proof

Choose α so that $0 < \alpha < 1$. Since Γ is convex, $\alpha y + (1 - \alpha)P[x] \in \Gamma$. By the definition of $P[x]$, $\|x - P[x]\| \leq \|x - (\alpha y + (1 - \alpha)P[x])\|$. Thus

$$\begin{aligned} \|x - P[x]\|^2 &\leq \|x - (\alpha y + (1 - \alpha)P[x])\|^2 \\ &= \|x - P[x] - \alpha(y - P[x])\|^2 \\ &= \|x - P[x]\|^2 + \alpha^2 \|y - P[x]\|^2 - 2\alpha(x - P[x])(y - P[x]). \end{aligned}$$

Then $(x - P[x])(y - P[x]) \leq \|y - P[x]\|\alpha/2$. And, since α can be taken arbitrarily close to 0,

$$(x - P[x])(y - P[x]) \leq 0. \quad \blacksquare$$

Proposition 3.2 [52]

Let $x, y \in \mathbb{R}^n$. Then $\|P[x] - P[y]\| \leq \|x - y\|$.

Proof

Case 1: Suppose $P[x] = P[y]$. Then

$$\|P[x] - P[y]\| = 0 \leq \|x - y\|.$$

Case 2: Suppose $P[x] \neq P[y]$. Then since $P[x] \in \Gamma$,

and $P[y] \in \Gamma$, from Proposition 3.1 we have that

$$(x - P[x])(P[y] - P[x]) \leq 0$$

and

$$(y - P[y])(P[x] - P[y]) \leq 0.$$

We may rewrite the above inequalities as

$$x(P[y] - P[x]) - P[x]P[y] + \|P[x]\|^2 \leq 0$$

and

$$y(P[x]-P[y]) - P[y]P[x] + ||P[y]||^2 \leq 0.$$

Adding these inequalities, we obtain

$$(x-y)(P[y]-P[x]) + ||P[y]-P[x]||^2 \leq 0.$$

Then from the Cauchy-Schwartz inequality,

$$-(x-y)(P[y]-P[x]) \leq ||x-y|| ||P[y]-P[x]||.$$

Thus

$$||P[y]-P[x]||^2 \leq ||x-y|| ||P[y]-P[x]||.$$

And since $P[x] \neq P[y]$,

$$||P[x]-P[y]|| \leq ||x-y||. \quad \blacksquare$$

Proposition 3.3 [52]

If $\eta_i \neq 0$, then, for any $y \in \Gamma$,

$$||y_{i+1}-y||^2 \leq ||y_i-y||^2 + s_i^2 ||\eta_i||^2 + 2s_i \eta_i (y-y_i).$$

Proof

Let i be any iteration of the subgradient algorithm. Suppose $\eta_i \neq 0$. Let $y \in \Gamma$. Then, by Proposition 3.2,

$$\begin{aligned} ||P[y_i - s_i \eta_i] - P[y]||^2 &\leq ||y_i - s_i \eta_i - y||^2 \\ &= ||y_i - y||^2 + s_i^2 ||\eta_i||^2 + 2s_i \eta_i (y - y_i). \end{aligned}$$

Since $P[y] = y$ and $P[y_i - s_i \eta_i] = y_{i+1}$, we have that

$$||y_{i+1} - y||^2 \leq ||y_i - y||^2 + s_i^2 ||\eta_i||^2 + 2s_i \eta_i (y - y_i). \quad \blacksquare$$

Our main convergence result is for the particular step size scheme:

$$s_i = \lambda_i [g(y_i) - \bar{g}] / ||\eta_i||^2$$

where $\bar{g}$ is a lower bound for the optimal objective and where we are at liberty to select bounds α and β for the $\{\lambda_i\}$ such that for each i , $0 < \alpha \leq \lambda_i \leq \beta < 2$.

Proposition 3.4

Let (i) $\bar{g}$ be a known lower bound for the optimal objective, g^* , with $g^* > \bar{g}$;

(ii) $\{\lambda_i\}$ be any infinite sequence such that

for all i , $0 < \alpha \leq \lambda_i \leq \beta < 2$; and

(iii) $s_i = \lambda_i [g(y_i) - \bar{g}] / ||\eta_i||^2$.

If there is a constant C such that for all i , $||\eta_i|| \leq C$, and if $\gamma > 0$ is given, then there is some n such that $g(y_n) \leq g^* + [\beta/(2-\beta)](g^* - \bar{g}) + \gamma$.

Proof

Let $\gamma > 0$ be given. Let (i), (ii), and (iii) hold. Let y^* be an optimal point, and for all i , $||\eta_i|| \leq C$. Suppose, contrary to the desired result, that for all n , $g(y_n) > g^* + [\beta/(2-\beta)](g^* - \bar{g}) + \gamma$. Then, by Proposition 3.3,

$$\begin{aligned} ||y_{i+1} - y^*||^2 &\leq ||y_i - y^*||^2 + \lambda_i^2 [g(y_i) - \bar{g}]^2 / ||\eta_i||^2 \\ &\quad + 2\lambda_i \{ [g(y_i) - \bar{g}] / ||\eta_i||^2 \} \eta_i (y^* - y_i) \\ &\leq ||y_i - y^*||^2 + \lambda_i^2 [g(y_i) - \bar{g}]^2 / ||\eta_i||^2 \\ &\quad + 2\lambda_i \{ [g(y_i) - \bar{g}] / ||\eta_i||^2 \} [g^* - g(y_i)], \end{aligned}$$

since $\eta_i \in \partial g(y_i)$.

Since $\beta \geq \lambda_i > 0$, then $\beta \lambda_i \geq \lambda_i^2$. So,

$$||y_{i+1} - y^*||^2 \leq ||y_i - y^*||^2 + \beta \lambda_i [g(y_i) - \bar{g}]^2 / ||\eta_i||^2$$

$$\begin{aligned}
& + 2\lambda_i \{ [g(y_i) - \bar{g}] / ||\eta_i||^2 \} [g^* - g(y_i)] \\
& = ||y_i - y^*||^2 + (2-\beta)\lambda_i \{ [g(y_i) - \bar{g}] / ||\eta_i||^2 \} \\
& \quad [(g^* - g(y_i)) + (\beta/(2-\beta))(g^* - \bar{g})].
\end{aligned}$$

Since $g(y_i) > g^* + (\beta/(2-\beta))(g^* - \bar{g}) + \gamma$, then $-\gamma > g^* - g(y_i) + (\beta/(2-\beta))(g^* - \bar{g})$. So,

$$||y_{i+1} - y^*||^2 < ||y_i - y^*||^2 - (2-\beta)\lambda_i [g(y_i) - \bar{g}] \gamma / ||\eta_i||^2.$$

Since $g^* \leq g(y_i)$, $\alpha \leq \lambda_i$, and $||\eta_i|| \leq C$, then

$$||y_{i+1} - y^*||^2 < ||y_i - y^*||^2 - [(2-\beta)\alpha(g^* - \bar{g})\gamma] / C^2. \quad (3.3)$$

We can choose an integer N so large that

$$C^2 ||y_1 - y^*||^2 / (2-\beta)\alpha(g^* - \bar{g})\gamma < N.$$

Thus, since $2-\beta > 0$ and $g^* - \bar{g} > 0$,

$$N(2-\beta)\alpha(g^* - \bar{g})\gamma / C^2 > ||y_1 - y^*||^2.$$

Adding together the inequalities obtained from (3.3) by letting i take on all values from 1 to N , we obtain

$$||y_{N+1} - y^*||^2 < ||y_1 - y^*||^2 - N(2-\beta)\alpha(g^* - \bar{g})\gamma / C^2 < 0,$$

a contradiction. ■

It is shown in [39] that when Γ is compact, g is continuous on some open set containing Γ , and $\partial g(y) \neq \emptyset$ for all $y \in \Gamma$, there exists a constant C such that $\|n\| \leq C$ for all $y \in \Gamma$, and $n \in \partial g(y)$, so that the boundedness condition on the subgradients in Proposition 3.4 is easily met.

3.2 Generating Lower Bounds

In this section we present a technique for generating lower bounds for the multicommodity network flow problem. This technique involves partially solving the Lagrangian dual problem using a subgradient technique to update the Lagrange multipliers at each iteration.

Recall that the multicommodity network flow problem, MP, may be stated as follows:

$$\begin{aligned}
 & \text{Minimize } \sum_k c^k x^k \\
 & \text{Subject to } Ax^k = r^k, \quad k = 1, \dots, K \quad (\text{MP}) \\
 & \sum_k x^k \leq u \\
 & 0 \leq x^k \leq v^k, \quad k = 1, \dots, K
 \end{aligned}$$

where

A is an $m \times n$ node-arc incidence matrix,
 c^k is an n vector of unit costs for $k = 1, \dots, K$,
 r^k is an m vector of node requirements for $k = 1, \dots, K$,
 u is an n vector of mutual arc capacities,
 v^k is an n vector of individual commodity bounds for $k=1, \dots, K$,
 x^k is an n vector of decision variables for $k =$
 $1, \dots, K,$

and K is the number of commodities.

Consider a Lagrangian dual problem for MP, denoted by DP:

$$\begin{array}{ll} \text{MAX} & h(\lambda) \\ \lambda \geq 0 & \end{array}$$

$$h(\lambda) = \text{MIN} \left[\sum_k c^k x^k + \lambda (\sum_k x^k - u) : \right. \quad (\text{DP})$$

$$\left. A x^k = r^k \ (k = 1, \dots, K); \ 0 \leq x^k \leq v^k \ (k = 1, \dots, K) \right]$$

where λ is an n vector of Lagrange multipliers.

First we show that any feasible solution for DP is a lower bound for MP.

Proposition 3.5 [12]

Let $\bar{x} = (\bar{x}^1, \bar{x}^2, \dots, \bar{x}^K)$ be a feasible solution for MP. Let $\bar{\lambda}$ be a feasible solution for DP. Then $h(\bar{\lambda}) \leq c\bar{x}$.

Proof

Since $h(\bar{\lambda})$ is a minimum, and since $\bar{x}$ is feasible for MP, $h(\bar{\lambda}) \leq \sum_k c^k \bar{x}^k + \bar{\lambda} (\sum_k \bar{x}^k - u)$. Further since $\bar{\lambda}$ is feasible for DP and $\bar{x}$ is feasible for MP, then $\bar{\lambda} (\sum_k \bar{x}^k - u) \leq 0$. Hence $h(\bar{\lambda}) \leq c\bar{x}$. ■

In addition to this result, Bazaraa and Shetty [12] have proved that if MP has an optimal solution, then DP has an optimal solution, and that their optimal objective function values are equal. As a result, we see that we may indeed solve (or partially solve) DP in order to obtain a lower bound for MP.

In order to justify using a subgradient optimization technique for solving DP, we must show that the objective function is concave and develop an expression for a subgradient.

Proposition 3.6

The real valued function h is concave over $\Lambda = \{\lambda; \lambda \in R^n; \lambda \geq 0\}$.

Proof

Let $\lambda^1 \geq 0$. Let $\lambda^2 \geq 0$. Let $0 < \alpha < 1$. Then

$$h(\alpha\lambda^1 + (1-\alpha)\lambda^2) = \min_k [\sum_k c^k x^k + (\alpha\lambda^1 + (1-\alpha)\lambda^2)(\sum_k x^k - u):$$

$$Ax^k = r^k (k=1, \dots, K); 0 \leq x^k \leq v^k (k=1, \dots, K)]$$

$$= \min_k [\alpha \sum_k c^k x^k + \alpha \lambda^1 (\sum_k x^k - u) + (1-\alpha) \sum_k c^k x^k + (1-\alpha) \lambda^2 (\sum_k x^k - u):$$

$$Ax^k = r^k (k=1, \dots, K); 0 \leq x^k \leq v^k (k=1, \dots, K)]$$

$$\geq \alpha \min_k [\sum_k c^k x^k + \lambda^1 (\sum_k x^k - u):$$

$$Ax^k = r^k (k=1, \dots, K); 0 \leq x^k \leq v^k (k=1, \dots, K)]$$

$$+ (1-\alpha) \min_k [\sum_k c^k x^k + \lambda^2 (\sum_k x^k - u) :$$

$$Ax^k = r^k (k=1, \dots, K); 0 \leq x^k \leq v^k (k=1, \dots, K)]$$

$$= \alpha h(\lambda^1) + (1-\alpha)h(\lambda^2). \text{ Hence } h \text{ is concave over } \Lambda. \blacksquare$$

Proposition 3.7

Let $\bar{\lambda} \geq 0$. Let $\bar{x}$ represent an optimal value of x corresponding to $h(\bar{\lambda})$. Then $d = \sum_k \bar{x}^k - u$ is a subgradient of h at $\bar{\lambda}$.

Proof

Let $\hat{\lambda}$ be any other point in Λ with corresponding optimal decision variable values $\hat{x}$. Then

$$h(\hat{\lambda}) = \sum_k c^k \hat{x}^k + \hat{\lambda} (\sum_k \hat{x}^k - u)$$

$$\begin{aligned}
&\leq \sum_k c^k \bar{x}^k + \hat{\lambda}(\sum_k \bar{x}^k - u) \quad (\text{since } \hat{x} \text{ is optimal}) \\
&= \sum_k c^k \bar{x}^k + \hat{\lambda}(\sum_k \bar{x}^k - u) + (\bar{\lambda} \sum_k \bar{x}^k - \bar{\lambda} \sum_k \bar{x}^k) + (\bar{\lambda} u - \bar{\lambda} u) \\
&= \sum_k c^k \bar{x}^k + \bar{\lambda}(\sum_k \bar{x}^k - u) + (\sum_k \bar{x}^k - u)(\hat{\lambda} - \bar{\lambda}) \\
&= h(\bar{\lambda}) + d(\hat{\lambda} - \bar{\lambda}).
\end{aligned}$$

Therefore d is a subgradient of h at $\bar{\lambda}$. ■

We now present our algorithm for computing lower bounds for MP.

Note that it is a specialization of the subgradient optimization algorithm for this problem, and its convergence follows as a maximization analog of Proposition 3.4.

ALGORITHM 3.2 LOWER BOUND ALGORITHM

Step 0 (Initialization)

Let UB be any upper bound on the solution to MP. Set $i \leftarrow 0$; $\lambda_0 \leftarrow 0$; $\alpha_0 \leftarrow 2$. Compute $y_0 \leftarrow h(\lambda_0)$ and let $x_0 = (x_0^1, \dots, x_0^K)$ be the corresponding optimal values of the decision variables.

Step 1 (Find Subgradient)

Set $\eta_i \leftarrow \sum_k x_i^k - u$. If $\eta_i = 0$, stop with y_i optimal.

Step 2 (Move to New Point)

Set $s_i \leftarrow \alpha_i(UB - y_i)/\|\eta_i\|^2$. Compute the j^{th} component of

λ_{i+1} as:

$$(\lambda_{i+1})_j \leftarrow \text{MAX} \{(\lambda_i + s_i \eta_i)_j, 0\}$$

Compute $y_{i+1} \leftarrow h(\lambda_{i+1})$ and let x_{i+1} be the corresponding optimal values of the decision variables. Set $\alpha_{i+1} \leftarrow \alpha_i/2$. Set $i \leftarrow i+1$. Return to step 1.

3.3 Generating Upper Bounds

Here we describe a procedure for generating upper bounds for the multicommodity network flow problem. This procedure is a specialization of the resource-directive decomposition (RDD) algorithm using a sub-gradient direction. First we describe the general RDD procedure; then we present our specialization.

The RDD technique produces a sequence of feasible solutions by distributing the mutual arc capacity among commodities in such a way that the solutions to the K individual subproblems provide a solution to the composite problem. At each iteration an allocation is made and the resulting K (single commodity) minimum cost network flow problems are solved. If the solution meets an optimality criterion then the procedure terminates; otherwise, a new allocation is made, and the process is repeated.

After introducing artificial variables, (a^k) , MP becomes:

$$\text{Minimize } \sum_k c^k x^k + M \sum_k \underline{1} a^k$$

$$\text{Subject to } Ax^k + a^k = r^k \quad (k = 1, \dots, K)$$

$$\sum_k x^k \leq u$$

$$0 \leq x^k \leq v^k \quad (k = 1, \dots, K)$$

$$a^k \geq 0 \quad (k = 1, \dots, K)$$

where M is a very large positive number and $\underline{1}$ is an m vector of all ones.

Let us restate the problem as:

$$\begin{aligned} &\text{Minimize } z(y^1, \dots, y^K) \\ &\text{Subject to } z(y^1, \dots, y^K) = \sum_k z^k(y^k) \quad (\text{RP}) \end{aligned}$$

$$\sum_k y^k = u$$

$$0 \leq y^k \leq v^k \quad (k = 1, \dots, K)$$

where $z^k(y^k) = \text{MIN} \{c^k x^k + M \underline{1} a^k : Ax^k + a^k = r^k; 0 \leq x^k \leq y^k; a^k \geq 0\}$ for $k = 1, \dots, K$. We shall refer to this formulation as RP. Note that $z^k(y^k) = \text{MAX}\{r^k \mu^k - y^k v^k : \mu^k A - v^k \leq c^k; \mu^k \leq M \underline{1}; v^k \geq 0\}$, by duality theory.

In order to justify using a subgradient optimization technique we must show that $z(y^1, \dots, y^K)$ is a convex function and develop an expression for a subgradient.

Proposition 3.8 [52]

The real valued function z is convex over $Y = \{y^1, \dots, y^K : y^1 \geq 0; \dots; y^K \geq 0\}$.

Proof

Let $(\bar{y}^1, \dots, \bar{y}^K) \in Y$ and $(\hat{y}^1, \dots, \hat{y}^K) \in Y$. Select α so that

$0 \leq \alpha \leq 1$. Then

$$\begin{aligned} &z[\alpha \bar{y}^1 + (1-\alpha) \hat{y}^1, \dots, \alpha \bar{y}^K + (1-\alpha) \hat{y}^K] \\ &= \sum_k z^k[\alpha \bar{y}^k + (1-\alpha) \hat{y}^k] \\ &= \sum_k \text{MAX}_{\mu^k A - v^k \leq c^k; \mu^k \leq M \underline{1}; v^k \geq 0} \{r^k \mu^k - [\alpha \bar{y}^k + (1-\alpha) \hat{y}^k] v^k\} \end{aligned}$$

$$= \sum_k \text{MAX} \{ \alpha [r_{\mu}^k \bar{y}^k v^k] + (1-\alpha) [r_{\mu}^k \hat{y}^k v^k] :$$

$$\mu^k A - v^k \leq c^k; \mu^k \leq M1; v^k \geq 0 \}$$

$$\leq \alpha \sum_k \text{MAX} \{ r_{\mu}^k \bar{y}^k v^k :$$

$$\mu^k A - v^k \leq c^k; \mu^k \leq M1; v^k \geq 0 \}$$

$$+ (1-\alpha) \sum_k \text{MAX} \{ r_{\mu}^k \hat{y}^k v^k :$$

$$\mu^k A - v^k \leq c^k; \mu^k \leq M1; v^k \geq 0 \}$$

$$= \alpha z(\bar{y}^1, \dots, \bar{y}^K) + (1-\alpha) z(\hat{y}^1, \dots, \hat{y}^K).$$

Therefore z is convex over Y . ■

Proposition 3.9 [52]

Let $\bar{y} = (\bar{y}^1, \dots, \bar{y}^K) \in Y$ be any allocation and let $(\bar{\mu}^k, \bar{v}^k)$ denote the corresponding optimal solution to $z^k(\bar{y}^k)$ for $k = 1, \dots, K$. Then $\eta = (-\bar{v}^1, \dots, -\bar{v}^K)$ is a subgradient of z at $\bar{y}$.

Proof

Let $y = (y^1, \dots, y^K) \in Y$ be any allocation and let (μ^k, v^k) denote the corresponding optimal solution to $z^k(y^k)$ for $k = 1, \dots, K$. Then:

$$\begin{aligned} z(y^1, \dots, y^K) - z(\bar{y}^1, \dots, \bar{y}^K) &= \sum_k (r_{\mu}^k y^k v^k) - \sum_k (r_{\bar{\mu}}^k \bar{y}^k \bar{v}^k) \\ &\geq \sum_k (r_{\bar{\mu}}^k y^k \bar{v}^k) - \sum_k (r_{\bar{\mu}}^k \bar{y}^k \bar{v}^k) \\ &= \sum_k (-\bar{v}^k)(y^k - \bar{y}^k). \end{aligned}$$

Hence η_i is a subgradient of z at $\bar{y}$. ■

Recall that the subgradient optimization algorithm requires a technique for projecting a point onto the feasible region. We now explore the projection operation for this problem.

Let us denote the feasible region for RP by Ω . That is, $\Omega = \{(y^1, \dots, y^K) : \sum_k y^k = u; 0 \leq y^k \leq v^k (k=1, \dots, K)\}$. Given an arbitrary allocation, $(\bar{y}^1, \dots, \bar{y}^K)$, to project it onto Ω , we solve :

$$\begin{aligned} & \text{MIN}\{|| (y^1, \dots, y^K) - (\bar{y}^1, \dots, \bar{y}^K) || : y \in \Omega\} \\ & = \text{MIN} \{ (\sum_{kj} (y_j^k - \bar{y}_j^k)^2)^{1/2} : y \in \Omega \}. \end{aligned}$$

Or, equivalently, we can solve:

$$\text{MIN}\{ \sum_{kj} (y_j^k - \bar{y}_j^k)^2 : y \in \Omega \}.$$

Note that this problem decomposes on j . Hence, for each arc j , we solve:

$$\text{MIN}\{ \sum_k (y_j^k - \bar{y}_j^k)^2 : \sum_k y_j^k = u_j; 0 \leq y_j^k \leq v_j^k; (k=1, \dots, K) \}.$$

We will denote the above projection problem by P . The following algorithm [52] is used to solve P for any arc, j .

ALGORITHM 3.3 PROJECTION ALGORITHM

Step 0 (Initialization)

If $u_j > \sum_k v_j^k$ or $u_j < 0$, terminate with no feasible

solution. Otherwise set $l \leftarrow 1$; $r \leftarrow 2K$; $L \leftarrow \sum_k v_j^k$; $R \leftarrow 0$. Compute

the breakpoints, b_i ($i=1, \dots, 2K$), as $\bar{y}_j^k$ and $\bar{y}_j^k - v_j^k$ ($k=1, \dots, K$).

Order the breakpoints so that $b_1 \leq b_2 \leq \dots \leq b_{2K}$.

Step 1 (Test for Bracketing)

If $r-l=1$ go to step 4; otherwise, set $m = \lceil (l+r)/2 \rceil$ where $[K]_I$ is the greatest integer $\leq K$.

Step 2 (Compute New Value)

$$\text{Set } C \leftarrow \sum_k \text{MAX}\{\text{MIN}[\bar{y}_j^k - y_m, v_j^k], 0\}$$

Step 3 (Update)

If $C=c$ then set $\lambda \leftarrow y_m^*$ and go to step 5. If $C > c$ then set $l \leftarrow m$; $L \leftarrow C$; and go to step 1. If $C < c$ then set $r \leftarrow m$; $R \leftarrow C$; and go to step 1.

Step 4 (Interpolate)

$$\text{Set } \lambda^* \leftarrow b_l + [(b_r - b_l)(c - L)] / (R - L).$$

Step 5

Compute the feasible (projected) allocation, y_j^k , for $k=1, \dots, K$ in this way:

$$y_j^k = \begin{cases} v_j^k, & \text{if } \lambda^* \leq \bar{y}_j^k - v_j^k \\ \bar{y}_j^k - \lambda^*, & \text{if } \bar{y}_j^k - v_j^k < \lambda^* \leq \bar{y}_j^k \\ 0, & \text{if } \lambda^* > \bar{y}_j^k \end{cases}$$

Terminate with the feasible allocation for arc j , $(y_j^1, \dots, y_j^K)$.

An upper bound algorithm using the subgradient procedure is now presented. Its convergence is a direct result of Proposition 3.4.

ALGORITHM 3.4 UPPER BOUND ALGORITHM

Step 0 (Initialization)

Let LB be any lower bound on the solution to MP. Choose a set of initial allocations, $y_0 = (y_0^1, \dots, y_0^K)$ by setting $y_0^k \leftarrow P[(1/K)(u)]$ for $k = 1, \dots, K$. Set $\lambda_0 \leftarrow 2$; $i \leftarrow 0$; $UB \leftarrow \infty$.

Step 1 (Find Subgradient)

Let (μ_i^k, v_i^k) solve $z^k(y_i^k)$ for $k = 1, \dots, K$. Let $\eta_i \leftarrow (-v_i^1, \dots, -v_i^K)$. Set $UB \leftarrow z^k(y_i^k)$. If $\eta_i = 0$, then terminate with $z(y_i)$ optimal.

Step 2 (Move to New Point)

Compute $s_i \leftarrow \lambda_i [z(y_i) - LB] / \|\eta_i\|^2$. Set $y_{i+1} \leftarrow P[y_i - s_i \eta_i]$. Set $\lambda_{i+1} \leftarrow \lambda_i / 2$; $i \leftarrow i+1$. Return to step 1.

We now introduce a heuristic modification of the upper bound algorithm, which has produced better results on our test problems. Recall that $\eta = (-v^1, \dots, -v^K)$ is a subgradient of z at $(y^1, \dots, y^K)$. Then for each arc j , the vector

$$\eta(j) = (\eta_j e_j, \eta_{n+j} e_j, \dots, \eta_{(k-1)n+j} e_j)$$

serves to isolate the components of η associated with the commodities flowing on arc j . For each such arc j we compute an individual step size at iteration i as

$$s_i(j) = \lambda_i [z(y_i^1, \dots, y_i^K) - z^*] / \|\eta_i(j)\|^2$$

where z^* is approximated by LB.

Using this idea we now present our heuristic upper bound algorithm.

ALGORITHM 3.5 HEURISTIC UPPER BOUND ALGORITHM

Step 0 (Initialization)

Let LB be any lower bound on the solution to MP. Choose a set of initial allocations, $y_0 = (y_0^1, \dots, y_0^K)$ by setting $y_0^k \leftarrow P[(1/K)(u)]$ for $k = 1, \dots, K$. Set $\lambda_0 \leftarrow 2$; $i \leftarrow 0$; $UB \leftarrow \infty$.

Step 1 (Find Subgradient)

Let (μ_i^k, v_i^k) solve $z^k(y_i^k)$ for $k = 1, \dots, K$. Let $\eta_i = (-v_i^1, \dots, -v_i^K)$. Set $UB \leftarrow \sum_k z^k(y_i^k)$. If $\eta_i = 0$, then terminate with $z(y_i)$ optimal.

Step 2 (Move to New Point)

Compute $s_i(j) \leftarrow \lambda_i [z(y_i^1, \dots, y_i^K) - LB] / \|\eta_i(j)\|^2$ for each arc j . Set $\hat{S} \leftarrow \text{diag}(s_i(1), \dots, s_i(n))$. Set

$$S \leftarrow \begin{bmatrix} \hat{S} & & & \\ & \hat{S} & & \\ & & \ddots & \\ & & & \hat{S} \end{bmatrix}.$$

Set $(y_{i+1}^1, \dots, y_{i+1}^K) \leftarrow P[(y_i^1, \dots, y_i^K) - S\eta_i]$. Set $\lambda_{i+1} \leftarrow \lambda_i/2$; $i \leftarrow i+1$.

Go to step 1.

3.4 The Algorithm

In this section we present the composite algorithm for solving MP. This procedure involves partially solving DP for successively better lower bounds and partially solving RP for successively better upper bounds on the optimal objective function value. The algorithm terminates whenever (a) the solution to DP can be shown to be an exact optimum; (b) the solution to RP can be shown to be an exact optimum; or (c) the greatest lower bound and the least upper bound generated are within a prescribed tolerance, ϵ . In case (c), the best solution to RP is presented as a guaranteed ϵ -optimal solution.

ALGORITHM 3.6 COMPLETE ALGORITHM

Step 0 (Initialization)

Let $\epsilon \leftarrow$ termination tolerance ($0 < \epsilon < 1$); NOLB $\leftarrow$ number of lower bound iterations to perform on each pass; NOUB $\leftarrow$ number of upper bound iterations to perform on each pass; LB $\leftarrow -\infty$; UB $\leftarrow \infty$.

Step 1 (Lower Bound)

Perform NOLB iterations of the lower bound algorithm (Algorithm 3.2). Let LB denote the best lower bound attained so far. If Algorithm 3.2 terminates in step 1 with an exact optimum, terminate with that solution optimal for MP.

Step 2 (Upper Bound)

Perform NOUB iterations of an upper bound algorithm (Algorithm 3.4 or 3.5). Let UB denote the best upper bound attained so far. If Algorithm 3.4 terminates in step 1 with an exact optimum, terminate with that solution optimal for MP.

Step 3 (Check for Termination)

If $\epsilon(UB) \leq LB$ then terminate with UB a guaranteed ϵ -optimum;
otherwise, go to step 1.

In this algorithm the best solutions for the lower bound and upper bound problems at each pass are retained and used as starting solutions for the respective problems on the next pass. The details of our implementation are presented in Chapter 4.

CHAPTER IV

COMPUTATIONAL EXPERIMENTATION

This chapter provides descriptions of our computer implementation of Algorithm 3.6 and of the test problems used. Our code, EVAC, uses MODFLO [1] to solve the single commodity minimum cost network flow subproblems which arise in Algorithm 3.2 and in Algorithm 3.5. MODFLO is a set of routines which may be used to solve a network flow problem or to reoptimize a previously solved problem after changes are made in some of the data. MODFLO, which is based on NETFLO [52], allows the user to change bounds, costs, and/or requirements and then reoptimize from a basis which was optimal for the original problem.

We tested EVAC on 22 randomly generated multicommodity network flow problems and on one test problem which was specially structured to be solved by EVAC. The test problems ranged in size from 22 to 754 nodes and from 53 to 1,102 arcs with from 0 to 599 linking constraints and from 3 to 20 commodities. The equivalent LP sizes are between 232 and 8,904 rows and between 470 and 12,111 columns. The 22 randomly generated problems were created using MNETGN [5], a multicommodity network problem generator. The problems were solved by EVAC and by MCNF [51], a multicommodity network flow code which uses a primal partitioning algorithm. Solution times are compared and conclusions are drawn concerning the relative effectiveness of the techniques.

4.1 Description of the Computer Programs

In this section we present a description of MCNF and EVAC, the two computer codes used in our experimentation. Both programs are written in standard FORTRAN and have been tailored to neither our equipment nor our FORTRAN compiler.

4.1.1 MCNF

MCNF was developed by Jeff Kennington at Southern Methodist University, Dallas, TX. It is an incore multicommodity network flow problem solver which uses the modification of the revised simplex method known as the primal partitioning algorithm [36]. In this algorithm the basis inverse is maintained as a set of rooted spanning trees (one for each commodity) and a working basis inverse is maintained in product form. The working basis inverse has dimension equal to the number of binding linking constraints corresponding to the current basis. The initial basis is created using a multicommodity variation of the routine used in NETFLO. A partial pricing scheme is used; the pricing tolerance is $1.E-6$ and the pivot tolerance is $1.E-8$.

4.1.2 EVAC

EVAC is our implementation of Algorithm 3.6 for solving the multicommodity network flow problem. Note that Algorithm 3.6 alternates between generating lower bounds using Algorithm 3.2 and generating upper bounds using Algorithm 3.5. Since both the lower bound problem (DP) and the upper bound problem (RP) decompose on commodities, EVAC maintains only the information concerning the current commodity in main memory. The problem data and most recent bases for all the other commodities are

kept on peripheral storage. At the user's option EVAC stores in main memory as much of the current set of allocations, $(y_i^1, \dots, y_i^k)$ and current dual variables $(-v_i^1, \dots, -v_i^k)$ as desired. All our test problems (with the exception of Problem 23) were solved with all the allocations and dual variables in core.

Both the lower bound routine and the upper bound routine use MODFLO as the optimizer for the single commodity subproblems. MODFLO uses the same partial pricing scheme as NETFLO and drives the flow on artificial arcs to zero using the Big-M method. The Big-M value that was used for our test problems, except as noted in Table 4.1, was 7 times the largest unit cost in the given problem. At subsequent iterations, initial bases for each commodity are just the optimal bases for the previous set of Lagrange multipliers. A basis for the upper-bound problem is generated by constructing a feasible basis from the previous optimal basis using the rules described in [1].

In practice we did not update the multipliers for the step sizes (α_i in Algorithm 3.2 and λ_i in Algorithm 3.5) at every iteration, but only when the improvement in the objective function was too small. As Algorithm 3.2 requires a finite upper bound (for calculation of the step size in step 2) we used an initial value of $UB \leftarrow 1.1*LB$. Thereafter for UB we used the best upper bound generated so far. The parameters and tolerance used in all our testing were these:

$$\epsilon = .90$$

$$NOLB = 5$$

$$NOUB = 5$$

$$\text{Pricing Tolerance} = 1.E-2$$

4.2 Description of the Test Problems

The multicommodity network problem generator, MNETGN, was used to create 22 random test problems. We modified the MNETGN output so that every arc appeared in every commodity's subproblem by adding arcs with upper bounds of zero where necessary. The test problem ranged in size from 22 to 754 nodes and from 53 to 1,102 arcs with from 0 to 599 linking constraints and from 3 to 20 commodities. The equivalent LP sizes are between 232 and 8,904 rows and between 470 and 12,111 columns. The number of linking constraints corresponds to a wide variety of problems from pure network problems (no linking constraints) to problems in which over 75% of the arcs are included in linking constraints.

Problem 15 was provided by Lt. Col. Dennis McLain, the Assistant Director of Operations Research at the Military Airlift Command located at Scott Air Force Base.

4.3 Summary of Computational Results

All the testing (except for Problems 15, 21, and 23) was done on a CDC 6600 at Southern Methodist University, using the FTN compiler with the optimization feature enabled. Except for Problems 7 and 23, a guaranteed ϵ -optimum was obtained for each problem with $\epsilon \geq 90\%$. Problem 7 experienced convergence difficulties when run using EVAC. Problem 8 was created from Problem 7 by increasing the linking constraint bounds by 10%. As indicated in Table 4.1, this slight modification enabled EVAC to solve the problem easily. We limited the number of lower bound iterations and upper bounds iterations to 100,

even though Problem 7 had not achieved 90% optimality by that point. Because of this the solution times for Problem 7 are given in Table 4.1 but are not included in the summary data.

Problem 23 was created to allow us to test EVAC on a relatively large problem. This problem (with 8,904 LP rows and 12,111 LP columns) was too large for MCNF to solve in the available memory, so we were not able to compare solution times for the two codes on this problem. In addition, due to the memory limitations on the CDC 6600, we were forced to use a CDC 205 to test Problem 23. For this reason the times for Problem 23 are included in Tables 4.1 and 4.2, but are not included in the totals and summary information. Since the testing on the CDC 205 involved a real-dollar expense, we were satisfied to stop when a 75% optimum was attained. The test runs for Problems 15 and 21 were made on a CDC Cyber 73. But since both the EVAC and MCNF runs for these problems were made on the Cyber 73, the totals and summary data include the times for Problems 15 and 21.

Details of the test problems are given in Table 4.1. The times are in CPU seconds and exclude the time required to input the problem data and print the solution reports. Table 4.1 also presents a comparison of the times required for MCNF and EVAC to solve each problem. In order to present a meaningful comparison of the solution times for MCNF and EVAC, we also present the solution times for EVAC exclusive of the extra I/O required to maintain the costs, bounds, and old bases for the sub-problems on peripheral storage. Since MCNF maintains all this information in main memory, this seems to be the most reasonable way of comparing timing statistics. The column titled "Guaranteed % Optimal" gives the best lower bound generated by EVAC as a percent of the best

upper bound generated by EVAC. The column titled "Actual % Optimal" presents the actual optimal objective (as obtained by MCNF) as a percent of the best upper bound generated by EVAC.

Table 4.2 provides the details of the times required by EVAC to perform various steps of the algorithm. The column titled "% of Time in Other" for the lower bound computations shows the time required for such activities as computing the Lagrange multipliers, updating the unit costs to reflect these changes, computing the resulting dual variables, and various bookkeeping activities. The corresponding column for upper bound computations reflects such activities as calculating the dual variables, testing the termination criteria, and various other short computations.

Table 4.3 summarizes the time comparisons graphically. The problems are grouped by number of commodities, as they are in Tables 4.1 and 4.2.

4.4 Analysis of Results

It seems clear from Tables 4.1 and 4.3 that EVAC severely dominates MCNF whenever the number of commodities is small. This is due to the fact that, for EVAC, quite a bit of additional overhead is involved in alternating between commodities. This overhead is not just a result of I/O, although that is a great deal of it, but is also due to the set-up time required for activities such as constructing a new feasible basis from an old basis and calculating the resulting dual variables. MCNF, on the other hand, is primarily driven by the number of binding linking constraints in the optimal solution. This is because MCNF seeks an exact optimum.

Letting $T(\text{EVAC})$ denote the average time required by EVAC (exclusive of I/O), and $T(\text{MCNF})$ denote the average time required by MCNF, we can express the following relationships:

For the 3-commodity test problems,

$$T(\text{EVAC}) = .354 * T(\text{MCNF}).$$

For the 4-commodity test problems,

$$T(\text{EVAC}) = .469 * T(\text{MCNF}).$$

For the 5-commodity test problems,

$$T(\text{EVAC}) = .666 * T(\text{MCNF}).$$

And for the test problems with 6 or more commodities,

$$T(\text{EVAC}) = .975 * T(\text{MCNF}).$$

It should also be noted that EVAC is capable of solving larger problems than MCNF. This is due to the fact that EVAC stores only one copy of the network defining data in main memory, where MCNF requires one copy for each commodity. Also, EVAC maintains in main memory the current basis, cost and bound data for only one commodity at a time. Thus, for a K-commodity problem, EVAC uses on the order of $1/K$ the main memory required by MCNF.

Note that the entries in the "Guaranteed % Optimal" and "Actual % Optimal" columns of Table 4.1 are quite close. This indicates that the sequence of lower bounds converged to values very near optimality. In addition, from Table 4.2, we see that the lower bound iterations are typically less time consuming than the upper bound iterations.

It is worth observing that EVAC was designed for very large problems which would never be solved to optimality. Even if a problem does not converge to within the requested tolerance in a prescribed number of iterations, EVAC always provides a feasible solution which is

a guaranteed ϵ -optimum for some $\epsilon > 0$. In contrast, MCNF provides only an upper bound on the optimum objective value, with no indication of how close it is to optimality until an exact optimum is actually attained.

We conclude that EVAC works extremely well in obtaining a guaranteed ϵ -optimum for the multicommodity network flow problem. While it is not as "robust" as the simplex-based MCNF, it is a good choice for the class of problems for which it was developed, the very large casualty evacuation models.

TABLE 4.1
DESCRIPTION OF THE TEST PROBLEMS AND SUMMARY COMPARISON OF SOLUTION TIMES FOR EVAC AND MCNF

PROBLEM	COMMOD- ITIES	NODES	ARCS	L.P. ROWS	L.P. COLUMNS	LINKING CONSTRAINTS	BINDING LINKING CONSTRAINTS	% IN L.B. ROUTINE (EVAC)	% IN U.B. ROUTINE (EVAC)	TOTAL EVAC TIME (EXCLUDING I/O)	TOTAL MCNF TIME	GUARANTEED % OPTIMAL	ACTUAL % OPTIMAL
1	3	253	657	1,174	2,352	412	6	81%	19%	7.97	4.22	94.3%	95.2%
2	3	142	480	721	1,678	292	11	73%	27%	5.81	3.31	91.5%	92.8%
3	3	108	336	581	1,214	254	27	22%	78%	19.46	17.12	90.0%	91.6%
TOTAL FOR 3-COMMODITY TEST PROBLEMS													
4	4	104	314	676	1,494	256	5	45%	55%	11.70	8.42	94.6%	94.9%
5	4	196	524	1,124	2,393	336	6	83%	17%	9.63	4.53	93.4%	94.3%
6	4	116	377	741	1,724	273	8	43%	57%	13.60	9.79	93.2%	93.3%
7 (*)	4	105	338	636	1,485	212	16	--	--	--	--	--	--
8	4	105	338	636	1,485	212	13	32%	68%	19.90	16.12	90.0%	91.0%
9	4	107	358	636	1,554	204	15	36%	64%	17.26	13.46	90.8%	92.4%
TOTAL FOR 4-COMMODITY TEST PROBLEMS													
10	5	101	259	510	1,214	0	0	100%	0%	5.29	2.17	100%	100%
11 (**)	5	101	246	511	1,154	1	0	100%	0%	5.20	2.18	100%	100%
12 (**)	5	104	245	584	1,212	59	7	58%	42%	9.44	6.14	93.3%	93.3%
13	5	92	295	680	1,622	215	10	32%	68%	26.78	21.16	90.6%	91.0%
14	5	85	268	626	1,439	196	18	27%	73%	31.79	26.39	90.2%	91.2%
TOTAL FOR 5-COMMODITY TEST PROBLEMS													
15	6	48	131	232	470	84	5	89%	11%	2.51	1.13	93.1%	93.3%
16	20	23	53	480	964	0	0	100%	0%	14.04	3.65	100%	100%
17	20	22	59	461	1,046	1	0	100%	0%	15.27	3.51	100%	100%
18	10	48	126	589	1,296	99	2	74%	26%	13.81	6.35	95.7%	97.0%
19	20	22	53	475	980	15	2	86%	14%	16.84	5.73	92.8%	93.3%
20	12	45	121	633	1,367	81	4	49%	51%	29.10	18.33	91.9%	91.9%
21	6	71	238	609	1,524	177	9	31%	69%	50.55	40.44	92.5%	92.7%
22 (**)	6	76	219	619	1,361	157	26	22%	78%	50.04	43.03	90.8%	91.1%
23 (*) (**)	11	754	1,102	8,904	12,111	599	--	72%	28%	327.18	--	75.3%	--
TOTAL FOR ≥ 6 COMMODITY TEST PROBLEMS													
										122.35	125.50		
TOTAL FOR ALL TEST PROBLEMS										257.36	393.78		

(*) PROBLEM NOT INCLUDED IN SUMMARY INFORMATION

(**) PROBLEM REQUIRED A BIG-M VALUE > 7 * LARGEST UNIT COST

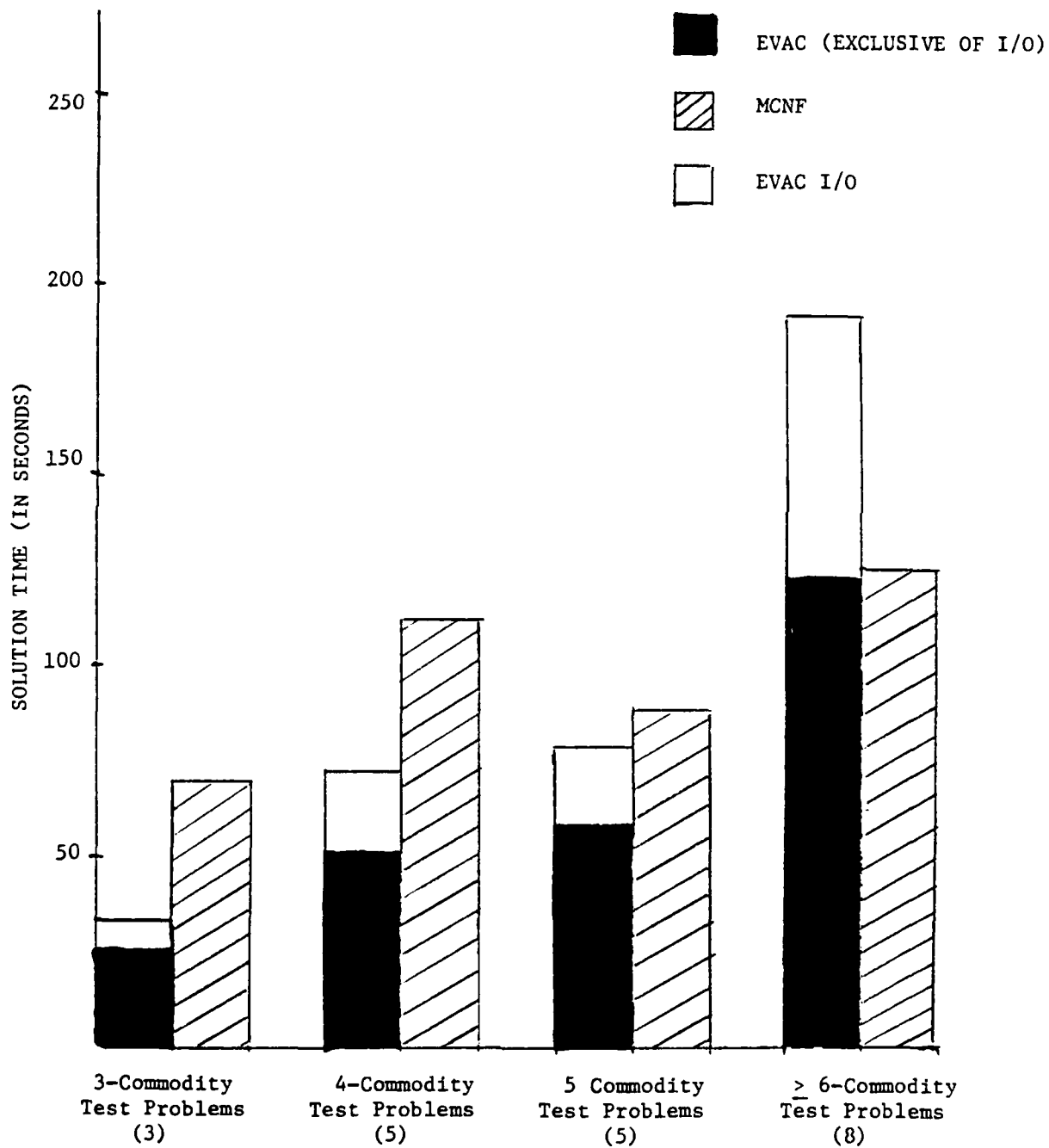
TABLE 4.2
DETAILED TIMING STATISTICS FOR EVAC RUNS

PROBLEM	COMMOD- ITIES	LOWER BOUND ROUTINE					UPPER BOUND ROUTINE						
		% TIME IN I/O	% TIME FINDING SUBGRAD.	% TIME IN L.B. ROUTINE	% TIME IN L.B. ITERA- TIONS	# L.B. ITERA- TIONS	% TIME IN I/O	% TIME FINDING SUBGRAD.	% TIME FEAS.BASIS CREATING	% TIME IN PRO- SECTION	% TIME OTHER ROUTINE	% TIME IN U.B. ITERA- TIONS	# U.B. ITERA- TIONS
1	3	46%	16%	0%	19%	81%	1%	9%	2%	2%	2%	19%	2
2	3	42%	15%	0%	16%	73%	1%	16%	2%	2%	1%	27%	2
3	3	10%	6%	0%	6%	22%	2%	4%	18%	9%	3%	78%	20
AVG FOR 3		33%	12%	0%	14%	59%	1%	12%	9%	4%	2%	41%	8
4	4	26%	9%	0%	10%	45%	2%	28%	11%	9%	1%	55%	9
5	4	53%	11%	0%	19%	83%	1%	8%	3%	2%	2%	17%	2
6	4	26%	7%	0%	10%	43%	2%	31%	9%	9%	2%	57%	10
7	(*)	--	--	--	--	--	--	--	--	--	--	--	--
8	4	17%	8%	0%	7%	32%	2%	39%	14%	9%	0%	68%	17
9	4	20%	8%	0%	8%	36%	2%	38%	13%	7%	1%	64%	12
AVG FOR 4		28%	9%	0%	11%	48%	2%	29%	10%	7%	1%	52%	10
10	5	59%	18%	0%	23%	100%	0%	0%	0%	0%	0%	0%	0
11	(**)	58%	19%	0%	23%	100%	0%	0%	0%	0%	0%	0%	0
12	(**)	33%	10%	0%	15%	58%	2%	22%	14%	2%	1%	42%	5
13	5	18%	6%	0%	8%	32%	3%	35%	13%	12%	0%	68%	23
14	5	14%	6%	0%	7%	27%	3%	36%	16%	13%	0%	73%	31
AVG FOR 5		36%	12%	0%	15%	63%	2%	19%	9%	5%	0%	37%	12
15	6	54%	14%	0%	21%	89%	1%	4%	2%	0%	3%	11%	1
16	20	74%	3%	0%	23%	100%	0%	0%	0%	0%	0%	0%	0
17	20	77%	2%	0%	21%	100%	0%	0%	0%	0%	0%	0%	0
18	10	53%	5%	0%	16%	74%	2%	8%	6%	7%	1%	26%	6
19	20	64%	3%	0%	19%	86%	2%	4%	4%	2%	1%	14%	5
20	12	33%	4%	0%	12%	49%	3%	18%	14%	12%	1%	51%	18
21	6	18%	6%	0%	7%	31%	2%	31%	14%	15%	2%	69%	22
22	(**)	11%	6%	0%	5%	22%	3%	33%	22%	14%	2%	78%	50
23	(*)(**)	8%	4%	0%	5%	17%	0%	18%	9%	5%	1%	83%	65
AVG FOR > 6		48%	5%	0%	16%	69%	2%	12%	8%	6%	1%	31%	13

(*) PROBLEM NOT INCLUDED IN SUMMARY INFORMATION

(**) PROBLEM REQUIRED A BIG-M- VALUE > 7*LARGEST UNIT COST

TABLE 4.3
GRAPHICAL COMPARISON OF EVAC AND MCNF SOLUTION TIMES



CHAPTER V

SUMMARY AND CONCLUSIONS

This chapter presents a summary of the results reported in Chapter IV and shares conclusions regarding the relative effectiveness of our technique. It also includes ideas for further investigation in the area.

5.1 Summary and Conclusions

Algorithm 3.6 describes our technique for finding an ϵ -optimal solution for the multicommodity network flow problem. Our technique differs from other approaches to the problem in that, rather than solving the multicommodity problem directly, we compute sequences of lower and upper bounds on the optimal objective function value, terminating when the bounds are within a prescribed tolerance. Both the lower and upper bound algorithms use a subgradient optimization technique and both decompose on commodities so that only a single commodity minimum cost network flow optimizer is required. At each iteration of the lower bound routine (Algorithm 3.2), an initial basis is generated from the previous optimal basis by modifying the costs to correspond to the new Lagrange multipliers, and updating the dual variables. At each iteration of the upper bound routine (Algorithm 3.5), an initial basis is constructed from the previous optimal basis

using the rules described in [1] to restore feasibility (if necessary) after changing the bounds to correspond to the new allocations.

The subgradients for the lower bounds are computed to be the sum of the flows on the mutually constrained arcs minus the associated mutual arc capacities. For the upper bounds, subgradients are computed using the dual variables obtained when solving the single commodity network problems.

Our computational work included solving each one of 23 problems twice; once using MCNF, a primal partitioning code, and once using EVAC, our implementation of Algorithm 3.6. On the average EVAC required only 65% of the time required by MCNF (ignoring I/O). EVAC's performance was far superior on the problems with fewer commodities and was not as impressive on the problems involving many commodities. In addition EVAC required on the order $1/K$ the amount of main memory as MCNF for a K -commodity problem.

5.2 Areas for Future Investigation

Algorithm 3.6 involves two more or less independent processes. That is, there is no reason why the lower bound generator (Algorithm 3.2) and the upper bound generator (Algorithm 3.5) could not proceed independently, stopping now and then to exchange their best bounds and test for optimality. Hence it appears that this procedure is well-suited to exploit the benefits of a parallel processing environment. In addition to the partitioning of the technique into two separate procedures, within each of these procedures the decomposition by commodities could take advantage of a parallel

processing scheme as well. It would seem reasonable to expect such a scheme to speed up the execution time considerably, especially when solving a very large problem.

There is also room for additional experimentation with the step sizes, specifically with the multipliers on the step sizes. Perhaps a scheme in which the multipliers were allowed to be reset to their starting values a finite number of times would speed up convergence. One might reset these multipliers whenever the improvement in the sequence of upper (lower) bounds fell below some tolerance. This would have the effect of restarting the algorithm at that point, but with a far better "starting solution".

In addition this problem has a multiperiod structure. Since the network is replicated for 60 one day time periods, it might be advantageous to exploit this structure using a forward simplex approach.

LIST OF REFERENCES

1. Ali, A., Allen, E., Barr, R., and Kennington, J., "Reoptimization Procedures for Bounded Variable Primal Simplex Network Algorithms", to appear in European Journal of Operations Research.
2. Ali, I., Barnett, D., Farhangian, K., Kennington, J., McCarl, B., Patty, B., Shetty, B., and Wong, P., "Multicommodity Network Problems: Applications and Computations," IIE Transactions, 16, 2, 127-134 (1984).
3. Ali, A. I., Helgason, R. V., Kennington, J. L., and Lall, H., "Primal-Simplex Network Codes: State-of-the-Art Implementation Technology," Networks, 8, 315-339 (1978).
4. Ali, A. I., Helgason, R. V., Kennington, J. L., and Lall, H., "Computational Comparison among Three Multicommodity Network Flow Algorithms," Operations Research, 23, 995-1000 (1980).
5. Ali, A. and Kennington, J., "MNETGN Program Documentation", Technical Report IEOR 77003, Department of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, TX, (1977).

6. Ali, A. I., and Kennington, J. L., "Network Structure in Linear Programs: A Computational Study," Technical Report No. 83-OR-1, Department of Operations Research, Southern Methodist University, Dallas, TX (1983).
7. Ali, A., and Kennington, J., "The Asymmetric M-Travelling Salesman Problem: A Duality Based Branch-And-Bound Algorithm," to appear in Discrete Applied Mathematics.
8. Assad, A. A., "Multicommodity Network Flows -Computational Experience," Working Paper OR-058-76, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, (1976).
9. Barr, R. S., Glover, F., and Klingman, D., "The Alternating Basis Algorithm for Assignment Problems," Mathematical Programming, 13, 1, 1-13 (1977).
10. Barr, R. S., Glover, F., and Klingman, D., "Enhancements of Spanning Tree Labelling Procedures for Network Optimization," INFOR, 17, 1, 16-34 (1979).
11. Bazaraa, M., and Goode, J., "The Travelling Salesman Problems: A Duality Approach," Mathematical Programming, 13, 221-237 (1977).
12. Bazaraa, M. and Shetty, C., Nonlinear Programming: Theory and Algorithms, John Wiley and Sons, New York, NY, (1979).

13. Bradley, G. H., Brown, G. G., and Graves, G. W., "Design and Implementation of Large-Scale Primal Transshipment Algorithms," Management Science, 24, 1, 1-34 (1977).
14. Charnes, A., and Cooper, W. W., Management Models and Industrial Applications of Linear Programming, Vols. 1 and 2, John Wiley and Sons, New York, NY, (1961).
15. Chen, H., and DeWald, C. G., "A Generalized Chain Labeling Algorithm for Solving Multicommodity Flow Problems," Computers and Operations Research, 1, 437-465 (1974).
16. Cremeans, J. E., Smith, R. A., and Tyndall, G. R., "Optimal Multicommodity Network Flows with Resource Allocation," Naval Research Logistics Quarterly, 17, 269-280 (1970).
17. Dantzig, G. B., "Application of the Simplex Method to a Transportation Problem," in T. C. Koopmans, Ed., Activity Analysis of Production and Allocation, John Wiley and Sons, New York, NY, (1951).
18. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, NJ (1963).
19. Dantzig, G. B., and Wolfe, P., "Decomposition Principle for Linear Programs," Operations Research 8, 101-111 (1960).

20. Ford, L. R., and Fulkerson, D. R., "Maximal Flow through a Network," Canadian Journal of Mathematics, 8, 3, 399-404 (1956).
21. Ford, L. R., and Fulkerson, D. R., "A Suggested Computation for Maximal Multicommodity Network Flow," Management Science, 5, 97-101 (1958).
22. Ford, L. R., and Fulkerson, D. R., Flows in Networks, Princeton University Press, Princeton, NJ, (1962).
23. Fulkerson, D. R., "An Out-of-Killer Method for Minimal-Cost Flow Problems," Journal of the Society of Industrial and Applied Mathematics, 9, 1, 18-27 (1961).
24. Glover, F., Glover, R., and Martinson, F., "The U.S. Bureau of Land Management's New NETFORM Vegetation Allocation System," Technical Report of the Division of Information Science Research, University of Colorado, Boulder, CO (1982).
25. Glover, F., Hultz, J., and Klingman, D., "Improved Computer-Based Planning Techniques," Research Report CCS 283, Center for Cybernetic Studies, The University of Texas, Austin, TX, (1977).
26. Glover, F., Hultz, J., and Klingman, D., "Network Versus Linear Programming Algorithms and Implementations," CCS 306, Center for Cybernetic Studies, The University of Texas, Austin, TX, (1977).

27. Glover, F., Karney, D., and Klingman, D., "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," Networks, 4,3, 191-212 (1974).
28. Glover, F., Karney, D., Klingman, D., and Napier, A., "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," Management Science, 20, 5, 793-813 (1974).
29. Glover, F., and Klingman, D., "New Advances in the Solution of Large-Scale Network and Network-Related Problems," Technical Report CCS 177, Center for Cybernetic Studies, The University of Texas, Austin, TX, (1974).
30. Glover, F., and Klingman, D., "New Advances in the Solution of Large-Scale Network and Network-Related Problems," CCS 238, Center for Cybernetic Studies, The University of Texas, Austin, TX, (1975).
31. Glover, F., and Klingman, D., "Some Recent Practical Misconceptions about the State-of-the-Art of Network Algorithms," Operations Research, 2, 370-379 (1978).
32. Glover, F., Klingman, D., and Stutz, J., "Augmented Threaded Index Method for Network Optimization," INFOR, 12, 3, 293-298 (1974).

33. Graves, G. W., and McBride, R. D., "The Factorization Approach to Large-Scale Linear Programming," Mathematical Programming, 10, 1, 91-110 (1976).
34. Grigoriadis, M.D., and White, W. W., "A Partitioning Algorithm for the Multicommodity Network Flow Problem," Mathematical Programming, 3, 157-177 (1972).
35. Hartman, J. K., and Lasdon, L. S., "A Generalized Upper Bounding Method for Doubly Coupled Linear Programs," Naval Research Logistics Quarterly, 17, 4, 411-429 (1970).
36. Hartman, J., and Lasdon, L., "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems", Networks, 1, 333-354, (1972).
37. Held, M., and Karp, T., "The Travelling Salesman Problem and Minimum Spanning Trees: Part II," Mathematical Programming, 1, 6-25 (1971).
38. Held, M., Wolfe, P., and Crowder, H., "Validation of Subgradient Optimization", Mathematical Programming, 6, 66-68, (1974).

39. Helgason, R., "A Lagrangian Relaxation Approach to the Generalized Fixed Charge Multicommodity Minimum Cost Network Flow Problem," unpublished dissertation, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, TX, (1980).
40. Helgason, R. V., and Kennington, J. L., "A Product Form Representation of the Inverse of a Multicommodity Cycle Matrix," Networks, 7, 297-322 (1977).
41. Helgason, R. V., and Kennington, J. L., "An Efficient Procedure for Implementing a Dual-Simplex Network Flow Algorithm," AIIE Transactions, 9, 1, 63-68 (1977).
42. Hitchcock, F. L., "The Distribution of a Product from Several Sources to Numerous Localities," Journal of Mathematics and Physics, 20, 224-230 (1941).
43. Jarvis, J. J., "On the Equivalence Between the Node-Arc and Arc-Chain Formulation for the Multicommodity Maximal Flow Problem," Naval Research Logistics Quarterly, 15, 525-529 (1969).
44. Jarvis, J. J., and Keith, P. D., "Multicommodity Flows with Upper and Lower Bounds," Working Paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, (1974).

45. Jarvis, J. J., and Martinez, D. M., "A Sensitivity Analysis of Multicommodity Network Flows," Transportation Science, 11, 4, 299-306 (1977).
46. Jewell, W. S., "A Primal-Dual Multicommodity Flow Algorithm," ORC 66-24, Operations Research Center, University of California, Berkeley, CA, (1966).
47. Johnson, E. L., "Programming in Networks and Graphs," Technical Report ORC 65-1, Operations Research Center, University of California at Berkeley (1965).
48. Kantorovich, L.V., "Mathematical Methods in the Organization and Planning of Production," Publication House of the Leningrad State University, 1939. 68pp. Translated in Management Science, 6, 366-422 (1960).
49. Karney, D., and Klingman, D., "Implementation and Computational Study on an In-core, Out-of-core Primal Network Code," Operations Research, 24, 1056-1077 (1976).
50. Kennington, J. L., "Solving Multicommodity Transportation Problems Using a Primal Partitioning Simplex Technique," Naval Research Logistics Quarterly, 24, 2, 309-325 (1977).

51. Kennington, J., "A Primal Partitioning Code for Solving Multicommodity Network Flow Problems", Technical Report No. 79008, Department of Operations Research, Southern Methodist University, Dallas, TX, (1979).
52. Kennington, J., and Helgason, R., Algorithms for Network Programming, John Wiley & Sons, New York, NY, (1980).
53. Kennington, J. L., and Shalaby, M., "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems," Management Science, 23, 9, 994-1004 (1977).
54. Koopmans, T. C., and Reiter, S., "A Model of Transportation," in T. C. Koopmans, Ed., Activity Analysis of Production and Allocation, John Wiley and Sons, New York, NY, (1951).
55. Kuhn, H. W., "The Hungarian Method for the Assignment Problem", Naval Research Logistics Quarterly, 2, 83-97 (1955).
56. Maier, S. F., "A Compact Inverse Scheme Applied to a Multi-commodity Network with Resource Constraints," in R. Cottle and J. Krarup, Eds., Optimization Methods for Resource Allocation, The English University Press, London, England (1974).
57. Mulvey, J. M., "Pivot Strategies for Primal-Simplex Network Codes," Journal of the Association for Computing Machinery, 25, 2, 266-270 (1978).

58. Mulvey, J., "Testing of a Large-scale Network Optimization Program," Mathematical Programming, 15, 291-314 (1978).
59. Orden, A., "The Transshipment Problem", Management Science, 2, 2, 276-285 (1956).
60. Robacker, J. T., "Notes on Linear Programming: Part XXXVII, Concerning Multicommodity Networks," Memo RM-1799, The Rand Corporation, Santa Monica, CA, (1956).
61. Saigal, R., "Multicommodity Flows in Directed Networks," ORC 67-38, Operations Research Center, University of California, Berkeley, CA, (1967).
62. Shor, N., "On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems," unpublished dissertation, Cybernetics Institute, Academy of Sciences, U.S.S.R. (1964).
63. Srinivasan V., and Thompson, G. L., "Accelerated Algorithms for Labelling and Relabeling of Trees, with Applications to Distribution Problems," Journal of the Association for Computing Machinery, 19, 4, 712-726 (1972).

64. Srinivasan, V., and Thompson, G. L., "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," Journal of the Association for Computing Machinery, 20, 194-213 (1973).
65. Swoveland, C., "Decomposition Algorithms for the Multicommodity Distribution Problem," Working Paper 184, Western Management Science Institute, University of California, Los Angeles, CA, (1971).
66. Swoveland, C., "A Two-Stage Decomposition Algorithm for a Generalized Muticommodity Flow Problem," INFOR, 11, 232-244 (1973).
67. Tomlin, J. A., "Mathematical Programming Models for Traffic Network Problems," unpublished dissertation, Department of Mathematics, University of Adelaide, Australia (1967).
68. Weigel, H. S., and Cremeans, J. E., "The Multicommodity Network Flow Model Revised to Include Vehicle per Time Period and Mode Constraints," Naval Research Logistics Quarterly, 19, 77-89 (1972).
69. Wollmer, R. D., "Multicommodity Networks with Resource Constraints: The Generalized Multicommodity Flow Problems," Networks, 1, 245-263 (1972).

Appendix B

Technical Report 85-OR-1

THE EQUAL FLOW PROBLEM

By

Iqbal Ali¹

Jeffery Kennington²

Bala Shetty²

Southern Methodist University
Dallas, Texas 75275
(214)-692-3072

April 1985

¹ Department of General Business
University of Texas at Austin

² Department of Operations Research
Southern Methodist University

ABSTRACT

This paper presents a new algorithm to solve a network problem with equal flow side constraints. The proposed solution technique is motivated by the desire to exploit the special structure of the side constraints and to maintain as much of the characteristics of pure network problems as possible. Not only has specialized software for the efficient solution of pure networks been developed, but the same computational efficacies lend themselves to the solution of sequences of minimum cost network flow problems by using reoptimization procedures. Our solution technique for the equal flow problem consists of solving two sequences of pure network problems. One sequence yields tighter lower bounds on the optimal value by considering the Lagrangean relaxation of the equal flow problem in which the side constraints are dualized. The second sequence yields upper bounds on the optimal value for the problem and maintains a feasible solution at all times. This sequence is obtained by considering a reformulation of the equal flow problem based on parametric changes in the requirements vector. The procedure has the added attractive feature that it provides a feasible solution which is known to be within a percentage of the optimal at all times. As such, the algorithm terminates when a solution with a prespecified tolerance on the objective function value is obtained. On NETGEN problems, using the first 150 arcs to form 75 equal flow side constraints, we found that the new algorithm is approximately 3 times faster than existing techniques and requires only 50% of the storage.

KEY WORDS

Linear Programming

Network Models

Networks With Side Constraints

Equal Flow Problem

ACKNOWLEDGEMENT

This research was supported in part by the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278.

I. INTRODUCTION

This paper presents a new technique to solve the equal flow problem. This problem is easily conceptualized as a minimal cost network flow problem with additional constraints on certain pairs of arcs. Specifically, given pairs of arcs are required to take on the same value. Applications of this model include crew scheduling [6], estimating driver costs for transit operations [28], and the two duty period scheduling problem [25]. The equal flow problem may be solved using a specialization of the simplex method for networks with side constraints. However, by exploiting the special structure of the side constraints, we have developed a new algorithm which results in a decrease in both computer storage and computation time.

It is well documented that pure network problems can be solved from fifty to one hundred times faster using specialized primal simplex software as compared to general linear programming systems. Motivated by this great advantage, our procedure solves the equal flow problem as a sequence of pure network problems and totally eliminates the need to deal with a basis matrix.

1.1 Problem Description

The equal flow problem is defined on a network represented by an (m,n) node-arc incidence matrix, A , in which K pairs of arcs are identified and required to have equal flow. Mathematically, this is expressed as:

$$\begin{array}{ll}
\text{Minimize} & cx \\
\text{s.t.} & Ax = b \\
& x_k = x_{k+K}, \quad k = 1, \dots, K \\
& 0 \leq x \leq u
\end{array}$$

where, c is a $1 \times n$ vector of unit costs, b is an $m \times 1$ vector of node requirements, 0 is an $n \times 1$ vector of zeroes, x is an $n \times 1$ vector of decision variables, and u is an $n \times 1$ vector of upper bounds. The above definition, henceforth referred to as $P1$, assumes that the first $2K$ arcs appear in the equal flow constraints. This assumption is in no way restrictive since, by rearranging the order of the arcs, any equal flow problem with K pairs can be expressed in the above form. Note that the K pairs of arcs are mutually exclusive, i.e. an arc appears in at most one side constraint.

1.2 Survey of Related Literature

In 1961, Charnes and Cooper [7] presented a specialized algorithm for the model:

$$\begin{array}{ll}
\text{Minimize} & cx \\
\text{s.t.} & Ax = b \\
& Cx = d \\
& x \geq 0,
\end{array}$$

where A and C are some general matrices but A has some favored structure. Their algorithm, called the double reverse method, takes advantage of the special structure of the matrix A . Variations of this algorithm may be found in [2, 8, 10, 15, 19, 24]. Specializations for multicommodity

problems may be found in [14, 20, 21].

In 1980, Shepardson and Marsten [25] showed that the two duty period scheduling problem can be reformulated as a single duty period scheduling problem with equal flow side constraints. They obtain a Lagrangean dual for this equal flow problem, by dualizing with respect to the equal flow side constraints. This Lagrangean dual is maximized using the subgradient optimization technique. In 1984, Turnquist and Malandraki [28] modeled the problem of estimating driver costs for transit operations as an integer equal flow problem. They obtain a Lagrangean dual for their problem, by dualizing with respect to the side constraints. Their algorithm is a slight modification of the subgradient optimization technique. They perform a line search between two successive solutions obtained during the subgradient optimization process.

Beck, Lasdon, and Engquist [5] transformed the equal flow problem into a quadratic programming problem which has a penalty for violating the equal flow constraints. They solved this nonlinear programming problem using the Fletcher-Reeves conjugate gradient method [9], a successive linear programming code [13], and a convex simplex code. If the penalty is sufficiently large, this approach is guaranteed to converge to the optimal solution of the equal flow problem.

1.3 Objective of the Investigation

The objective of this investigation is to develop and computationally test a new algorithm for the equal flow problem. This algorithm utilizes the subgradient optimization technique and is based on the relaxation/

restriction procedure proposed by Glover, Glover, and Martinson [11] for a generalized network model with special side constraints. We establish that the equal flow problem may be solved as two sequences of pure network problems, one sequence corresponds to computing a lower bound while the other corresponds to computing an upper bound. In the limit, both bounds will converge to the optimal objective value. Our implementation terminates when the difference between the bounds is within a prespecified tolerance.

The subgradient optimization technique requires the computation of subgradients, choice of appropriate step sizes, and the application of a projection operation. We show that the subgradients for the upper bound can be computed using the optimal dual variables obtained by solving pure network problems. We also develop theoretical results that yield an easy implementation of the projection operation. The step sizes selected are a modification of the ones proposed by Polyak [23]. For this choice of step sizes, we prove that our algorithm must necessarily obtain an iterate at which the objective value is arbitrarily close to the optimal objective value. In a computational study, comparing our code with a code that is designed to solve network problems with side constraints, we found that the new code runs approximately 3 times faster and requires 50% less core storage.

II. THE SUBGRADIENT ALGORITHM

The Subgradient Algorithm was first introduced by Shor [27] and is a general procedure for solving nonlinear programming problems. It may be viewed as a generalization of the steepest descent (ascent) method for convex (concave) problems in which the gradient may not exist everywhere. The subgradient is simply substituted in place of the gradient for those points for which the gradient does not exist. When this occurs, the algorithm may move to a point with objective value worse than the current point. Hence, the objective function does not necessarily improve at each iteration and consequently the convergence results of Zangwill [29] do not apply. Remarkably though, under fairly minor conditions on the step size, convergence can be guaranteed.

Let the nonlinear program PO be given by:

$$\begin{array}{ll} \text{Minimize} & f(y) \\ \text{s.t.} & y \in G \end{array}$$

where f is a real valued function that is convex over the compact, convex, and nonempty set G . A vector η will be called a subgradient of f at $\bar{y}$ if $f(y) - f(\bar{y}) \geq \eta(y - \bar{y})$ for all $y \in G$. For any $\bar{y} \in G$, we denote the set of all subgradients of f at $\bar{y}$ by $\partial f(\bar{y})$. The subgradient algorithm makes use of an operation called the projection operation. The projection of a point x onto G , denoted by $P[x]$, is defined to be the unique point $y \in G$ that is nearest to x with respect to the Euclidean norm. Using the projection operation, we now present the subgradient algorithm in its most general form.

ALG 1 SUBGRADIENT OPTIMIZATION ALGORITHM

Step 0 (Initialization)

Let y_0 be any element of G , select a set of step sizes $s_0, s_1, s_2, \dots$, and set $i \leftarrow 0$.

Step 1 (Find Subgradient)

Let $\eta_i \in \partial f(y_i)$. If $\eta_i = 0$, then terminate with y_i optimal.

Step 2 (Move to New Point)

Set $y_{i+1} \leftarrow P[y_i - s_i \eta_i]$, set $i \leftarrow i + 1$ and return to step 1.

Various proposals have been offered for the selection of the step sizes. Three general schema which have been suggested are:

- i) $s_i = \lambda_i$,
- ii) $s_i = \frac{\lambda_i}{\|\eta_i\|^2}$,
- iii) $s_i = \frac{\lambda_i (f(y_i) - f^*)}{\|\eta_i\|^2}$, $0 < \lambda_i < 2$,

where f^* is the optimal value of f over G . If the constants, λ_i 's, satisfy the following conditions:

$$\lambda_i \geq 0, \text{ all } i; \quad \lim_{i \rightarrow \infty} \lambda_i = 0; \quad \text{and} \quad \sum \lambda_i = \infty,$$

then the convergence of the algorithm is guaranteed using (i) or (ii)

(see Goffin [12], Helgason [17], Kennington and Helgason [21]). For the

upper bounds, we use a modification of the third step size. The following result is available for this scheme.

Proposition 1 (Polyak [23])

Let f be a real valued convex function over the compact, convex, and nonempty set G . Also, let f^* be the minimum of f and $\|\eta_i\| \leq C$ for all i and some constant C . Then there exists a $y_i^* \in G$ with $f(y_i^*) \rightarrow f^*$, if scheme (iii) is used.

Note that in (iii), f^* is the optimal value of f over G . Since the optimal objective is unknown before solving the problem, we use a lower bound on f^* in our implementation.

III. THE LOWER BOUND

Recall that the equal flow problem, which we denote by P1, is given by:

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{s.t.} && Ax = b \\ &&& x_k = x_{K+k}, \quad k = 1, \dots, K \\ &&& 0 \leq x \leq u. \end{aligned}$$

In our algorithm for P1, lower bounds on the optimal objective of P1 are used for step sizes and for termination. In this section, we describe a procedure to obtain these lower bounds.

Consider the following Lagrangean dual for P1, which we shall refer to as D1:

$$\begin{aligned} &\text{Maximize} && h(w), \text{ where } w = [w_1, \dots, w_K] \in \mathbb{R}^K, \text{ and} \\ &&& h(w) = \text{Min}\{cx + \sum_{k=1}^K w_k(x_k - x_{K+k}) : Ax = b, 0 \leq x \leq u\}. \end{aligned}$$

Proposition 2 (Shetty [26])

Let $\bar{x}$ be a feasible solution to P1 and let $w = [w_1, \dots, w_K]$ be a feasible solution to D1. Then $c\bar{x} \geq h(w)$.

Proposition 3 (Bazaraa and Shetty [4])

If P1 has a minimum, then the optimal objectives for P1 and D1 are equal.

As a consequence of Propositions 2 and 3, we may solve D1 to obtain a lower bound. We will now show that D1 may be solved using the

subgradient optimization technique for concave functions. This technique is similar to ALG 1 with a modification. Let $p_0, p_1, p_2, \dots$ denote a sequence of step sizes and let $d_i \in \partial h(w_i)$. Then step 2 is replaced by:

Step 2 (Move to New Point)

Set $w_{i+1} \leftarrow w_i + p_i d_i$, set $i \leftarrow i+1$ and return to step 1.

To use this algorithm $h(w)$ must be concave, and we need a means of generating subgradients. These two results follow:

Proposition 4 (Shetty [26])

The real valued function h is concave over R^K .

Proposition 5 (Shetty [26])

For a given $\bar{w}$, let $\bar{x}$ be an optimal solution to

$$\text{Min}\{cx + \sum_{k=1}^K \bar{w}_k (x_k - x_{K+k}) : Ax = b, 0 \leq x \leq u\}.$$

Then $d = [(\bar{x}_1 - \bar{x}_{K+1}), \dots, (\bar{x}_K - \bar{x}_{2K})]$ is a subgradient of h at $\bar{w}$.

We used scheme (i) for step sizes. Let UBND denote an upper bound and assume that the optimal objective value is positive. Our algorithm for obtaining lower bounds is presented below:

ALG 2 LOWER BOUND ALGORITHM

Step 1 (Initialization)

Initialize UBND, step size p , and tolerance ϵ .

Set $w \leftarrow 0$.

Step 2 (Find Subgradient)

Let $\bar{x} = [\bar{x}_1, \dots, \bar{x}_n]$ solve

$$h(w) = \text{Min}\{cx + \sum_{k=1}^K w_k(x_k - x_{K+k}) : Ax = b, 0 \leq x \leq u\}.$$

Set $\text{LBND} \leftarrow h(w)$.

If $(\text{UBND} - \text{LBND}) \leq \epsilon(\text{UBND})$ then terminate;

otherwise, set $d \leftarrow [(\bar{x}_1 - \bar{x}_{K+1}), \dots, (\bar{x}_K - \bar{x}_{2K})]$.

Step 3 (Move to a New Point)

3a. Set $w \leftarrow w + pd$, set $p \leftarrow p/2$.

3b. Go to 2.

IV. THE UPPER BOUND

An alternate formulation of P1, which will be referred to as P2, is as follows:

$$\begin{array}{ll}\text{Minimize} & g(y) \\ \text{s.t.} & y \in S\end{array}$$

where for any vector $y = [y_1, \dots, y_K]$,

$$g(y) = \text{Min}\{cx: Ax = b, 0 \leq x \leq u, x_k = x_{K+k} = y_k \text{ for all } k\}$$

and

$$S = \{y: 0 \leq y_k \leq \min(u_k, u_{K+k}) \text{ for all } k\}.$$

Clearly, P1 and P2 are equivalent. That is, given an optimum for one, we can construct an optimum for the other. We will now show that P2 is a special case of the nonlinear program P0 and may be solved using the Subgradient Optimization Algorithm, ALG 1.

Proposition 6 (Shetty [26])

The real valued function g is piece-wise linear convex over the compact, convex and nonempty set S .

To apply the subgradient algorithm, we need a procedure for obtaining a subgradient of g at a point y . The following proposition shows that the dual variables may be used to construct a subgradient.

Proposition 7 (Shetty [26])

Let $(\pi, v_1, v_{K+1}, \dots, v_K, v_{2K}, \mu)$ be the optimal dual variables for

$$\begin{array}{llll}
\text{Minimize} & cx & & \\
\text{s.t.} & Ax = b & (\pi) & \\
& x_1 = y_1 & (v_1) & \\
& x_{K+1} = y_1 & (v_{K+1}) & \\
& \cdot & & \\
& \cdot & & \\
& \cdot & & \\
& x_K = y_K & (v_K) & \\
& x_{2K} = y_{2K} & (v_{2K}) & \\
& 0 \leq x \leq u & (\mu). &
\end{array}$$

Then $\eta = (v_1 + v_{K+1}, \dots, v_K + v_{2K})$ is a subgradient of g at $y = (y_1, \dots, y_K)$.

As a result of Proposition 7, a subgradient at any given point

$y = (y_1, \dots, y_K) \in S$ required in our specialization of ALG 1 can be

obtained by solving $\text{Min}\{cx: Ax = b, x_1 = y_1, \dots, x_{2K} = y_K, 0 \leq x \leq u\}$, which

we shall refer to as P3. After substituting $x_1 = y_1, \dots, x_{2K} = y_K$, in

$Ax = b$, we obtain a pure network problem, which we shall refer to as

P4 and is given below:

$$\begin{array}{ll}
\text{Minimize} & \hat{c}\hat{x} \\
\text{s.t.} & \hat{A}\hat{x} = \hat{b} \\
& 0 \leq \hat{x} \leq \hat{u}.
\end{array}$$

To apply ALG 1, we need a procedure for constructing the optimal dual variables $(v_1, v_{K+1}, \dots, v_K, v_{2K})$ for P3 from the optimal dual variables

for P4. Suppose the arc corresponding to j has "From" node j_1 and "To" node j_2 . That is, arc j is the ordered pair (j_1, j_2) . Then we define $\text{FROM}(j) = j_1$ and $\text{TO}(j) = j_2$. Using this notation, the following proposition gives the required formulae:

Proposition 8 (Shetty [26])

Let $\hat{\pi}$ be the vector of optimal dual variables for P4. Then $[\hat{\pi}, v_1, v_{K+1}, \dots, v_K, v_{2K}]$ with $v_j = -\hat{\pi}_{\text{FROM}(j)} + \hat{\pi}_{\text{TO}(j)} + c_j$, $j=1, \dots, 2K$, are optimal duals for P3.

We now present two propositions that justify the projection routine used for the upper bound. The proofs may be found in Kennington and Helgason [21] and Shetty [26].

Proposition 9

Let S be a nonempty, convex set and $\hat{y} \notin S$. Then $y^* \in S$ is a projection of $\hat{y}$ on to S if $(\hat{y} - y^*)(y - y^*) \leq 0$ for all $y \in S$.

Proposition 10

Let $\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_K) \in \mathbb{R}^K$ with

$$\hat{y}_k < 0 \quad \text{for } k = 1, \dots, L$$

$$0 \leq \hat{y}_k \leq \tilde{u}_k, \quad \tilde{u}_k = \min(u_k, u_{K+k}) \quad \text{for } k = L+1, \dots, L+M$$

$$\hat{y}_k > \tilde{u}_k \quad \text{for } k = L+M+1, \dots, K$$

where L, M are integers and $0 \leq L, M \leq K$. Then

$P(\hat{y}) = y^* = (0, \dots, 0, \hat{y}_{L+1}, \dots, \hat{y}_{L+M}, \tilde{u}_{L+M+1}, \dots, \tilde{u}_K)$ is
a projection of $\hat{y}$ on S .

Following a description of the terminology used, our algorithm for obtaining an upper bound for P_1 is presented below. Let RFREQ denote the frequency at which the constant λ in step size (iii) is reset to its initial value, λ_0 , LBND denote a lower bound on P_1 , UBND denote an upper bound on P_1 , P denote the projection routine described in Proposition 10, ϵ denote the termination tolerance and Q denote the iteration count for the upper bound.

ALG 3 UPPER BOUND ALGORITHM

Step 1 (Initialization)

Choose $y \in S$.

Initialize LBND, RFREQ, ϵ , and λ_0 .

Set $\tilde{u} \leftarrow (\min(u_1, u_{K+1}), \dots, \min(u_K, u_{2K}))$.

Set $Q \leftarrow 0$, set $\bar{\lambda}_k \leftarrow \lambda_0$ for $k=1, \dots, K$.

Step 2 (Find Subgradient and Step Size)

For allocation y , let $\bar{x}$ and $\hat{\pi}$, respectively, be the vectors of optimal primal and dual variables for

$\text{Min}\{\hat{c}\hat{x} : \hat{A}\hat{x} = \hat{b}, 0 \leq \hat{x} \leq \hat{u}\}$. Construct x from $\bar{x}, y$.

Set $UBND \leftarrow cx$.

If $(UBND - LBND) \leq \epsilon(UBND)$, then terminate with x optimal;

otherwise,

set $v_j \leftarrow -\hat{\pi}_{FROM(j)} + \hat{\pi}_{TO(j)} + c_j$, $j = 1, \dots, 2K$,

set $\eta \leftarrow (v_1 + v_{K+1}, \dots, v_K + v_{2K})$.

If $Q = RFREQ$, then set $Q \leftarrow 0$, set $\lambda \leftarrow \lambda_0$, set $\bar{\lambda}_k \leftarrow \lambda_0$

for $k = 1, \dots, K$, and go to 3;

otherwise,

compute $\hat{\lambda}_k$ such that $0 \leq y_k - \hat{\lambda}_k \eta_k \leq \tilde{u}_k$, $k = 1, \dots, K$,

set $\bar{\lambda}_k \leftarrow \min\{\bar{\lambda}_k/2, \hat{\lambda}_k\}$, $k = 1, \dots, K$,

set $\lambda \leftarrow \min\{\bar{\lambda}_k, k=1, 2, \dots, K\}$.

Step 3 (Move to New Point)

3a. Set $y \leftarrow P[y - \lambda \frac{(UBND - LBND)}{\|\eta\|^2} \eta]$, set $Q \leftarrow Q+1$.

3b. Go to 2.

Note that the step size (iii) presented before may be rewritten for our function g as follows:

$$s_1 = \frac{\lambda_1 (g(y_1) - g^*)}{\|\eta_1\|^2}, \quad 0 < \lambda_1 < 2,$$

where $\eta_1 \in \partial g(y_1)$ and g^* is the optimal value of g .

In our implementation, we use $\bar{g}$, a lower bound on g , in place of g^* . The following propositions demonstrate that for $\bar{g}$'s close to g^* , our procedure must necessarily obtain an iterate at which g is arbitrarily close to g^* .

Proposition 11 (Kennington and Helgason [21])

If $\eta_i \neq 0$,

$$\|y_{i+1} - y_i\|^2 \leq \|y_i - y\|^2 + s_i^2 \|\eta_i\|^2 + 2s_i \eta_i (y - y_i) \text{ for any } y \in S$$

and step size s_i .

Proposition 12

Let g^* be the optimal value of g , and also let

- i) $\alpha g^* \leq \bar{g} \leq g^*$, $0 \leq \alpha \leq 1$,
- ii) $s_i = \lambda_i (g(y_i) - \bar{g}) / \|\eta_i\|^2$, and
- iii) $0 < \epsilon \leq \lambda_i \leq \beta < 2$ for all i .

If there is a constant C such that $\|\eta_i\| \leq C$ for all i , then there exists some i such that $g(y_i) \leq M\delta + g^* \left(\frac{2}{2-\beta} - \frac{\beta}{2-\beta} \alpha \right)$ for any $\delta > 0$ and for some constant M .

Proof

First, we assert that there is some i such that

$$g(y_i) \leq \frac{\lambda_i^2 \bar{g}}{(\lambda_i^2 - 2\lambda_i)} - \frac{2\lambda_i g^*}{(\lambda_i^2 - 2\lambda_i)} - \frac{\delta}{(\lambda_i^2 - 2\lambda_i)}, \text{ for any } \delta > 0.$$

Suppose that for all i ,

$$g(y_i) > \frac{\lambda_i^2 \bar{g}}{(\lambda_i^2 - 2\lambda_i)} - \frac{2\lambda_i g^*}{(\lambda_i^2 - 2\lambda_i)} - \frac{\delta}{(\lambda_i^2 - 2\lambda_i)}, \text{ where}$$

$\delta > 0$ is given. Let $y^* \in S$ be an optimal point. By Proposition 11,

$$\|y_{i+1} - y^*\|^2 \leq \|y_i - y^*\|^2 + \frac{\lambda_i^2 (g(y_i) - \bar{g})^2}{\|\eta_i\|^2} + \frac{2\lambda_i (g(y_i) - \bar{g}) \eta_i (y^* - y_i)}{\|\eta_i\|^2}$$

Since $\eta_i \in \partial g(y_i)$, $\eta_i (y^* - y_i) \leq g^* - g(y_i)$.

Thus,

$$\begin{aligned} \|y_{i+1} - y^*\|^2 &\leq \|y_i - y^*\|^2 + \frac{\lambda_i^2 (g(y_i) - \bar{g})^2}{\|\eta_i\|^2} + \frac{2\lambda_i (g(y_i) - \bar{g}) (g^* - g(y_i))}{\|\eta_i\|^2} \\ &= \|y_i - y^*\|^2 + (g(y_i) - \bar{g}) \left[\frac{g(y_i) (\lambda_i^2 - 2\lambda_i) - \lambda_i^2 \bar{g} + 2\lambda_i g^*}{\|\eta_i\|^2} \right] \\ &\leq \|y_i - y^*\|^2 - \frac{(g(y_i) - \bar{g}) \delta}{\|\eta_i\|^2} \\ &\leq \|y_i - y^*\|^2 - \frac{(g^* - \bar{g}) \delta}{c^2} \end{aligned} \tag{1}$$

We can choose an integer N large enough that

$$\frac{c^2 \|y_1 - y^*\|^2}{(g^* - \bar{g})\delta} < N.$$

Adding together the inequalities obtained from (1) by letting i take on all values from 1 to N , we obtain

$$\|y_N - y^*\|^2 \leq \|y_1 - y^*\|^2 - \frac{N(g^* - \bar{g})\delta}{c^2} < 0,$$

a contradiction. This justifies our assertion.

By simplifying our assertion further we get,

$$\begin{aligned} g(y_1) &\leq \frac{\delta}{(2\lambda_1 - \lambda_1^2)} - \frac{\bar{g}\lambda_1}{(2 - \lambda_1)} + \frac{2g^*}{(2 - \lambda_1)} \\ &\leq \frac{\delta}{(2\lambda_1 - \lambda_1^2)} - \frac{\alpha g^*\lambda_1}{(2 - \lambda_1)} + \frac{2g^*}{(2 - \lambda_1)} \\ &= \frac{\delta}{(2\lambda_1 - \lambda_1^2)} + g^* \left[\frac{2 - \alpha\lambda_1}{2 - \lambda_1} \right] \\ &= \frac{\delta}{(2\lambda_1 - \lambda_1^2)} + g^* \left(1 + \frac{\lambda_1}{(2 - \lambda_1)} (1 - \alpha) \right) \\ &\leq \frac{\delta}{(2\lambda_1 - \lambda_1^2)} + g^* \left(1 + \frac{\beta}{2 - \beta} (1 - \alpha) \right) \end{aligned}$$

$$= \frac{\delta}{(2\lambda_1 - \lambda_1^2)} + g^*\left(\frac{2}{2-\beta} - \frac{\beta}{2-\beta} \alpha\right)$$

$$\leq \frac{1}{(2\epsilon - \epsilon^2)} \cdot \left(M\delta + g^*\left(\frac{2}{2-\beta} - \frac{\beta}{2-\beta} \alpha\right) \right), \text{ where } M \text{ is a constant less than}$$

This completes the proof of Proposition 12.

V. THE ALGORITHM

In this section, we present our new algorithm for solving the equal flow problem. Let ITERL denote the number of iterations spent in step 1a in computing the lower bound before returning to the upper bound, and ITERU denote the number of iterations spent in computing the upper bound before returning to the lower bound. Also, let T denote the iteration count for the lower bound, R denote the iteration count for upper bound, and p_0 denote the initial step size for the lower bound.

ALG 4 SUBGRADIENT OPTIMIZATION ALGORITHM FOR THE EQUAL FLOW PROBLEM

Step 0 (Initialization)

Initialize ITERL, ITERU, REREQ, λ_0 , p_0 , and tolerance ϵ .

Set $T \leftarrow 0$, set $Q \leftarrow 0$, set $R \leftarrow 0$, set $w \leftarrow 0$, and set IFLAG $\leftarrow 0$.

Set UBND $\leftarrow +\infty$, set LBND $\leftarrow -\infty$, set $p \leftarrow p_0$, and set $\bar{\lambda}_k \leftarrow \lambda_0$ for $k=1, \dots, K$.

Set $\tilde{u} \leftarrow (\min(u_1, u_{K+1}), \dots, \min(u_K, u_{2K}))$.

Step 1 (Compute Lower Bounds)

1a. Call ALG 2 (steps 2 and 3a).

1b. Set $T \leftarrow T+1$

If $T \leftarrow \text{ITERL}$, then go to 1.

1c. (Initialize y)

If IFLAG $\neq 0$, then go to 2; otherwise,

set $y \leftarrow [\min(\tilde{u}_1, (\bar{x}_1 + \bar{x}_{K+1})/2), \dots, \min(\tilde{u}_K, (\bar{x}_K + \bar{x}_{2K})/2)]$.

Step 2 (Compute Upper Bounds)

2a. Set $T \leftarrow 0$, set $IFLAG \leftarrow 1$.

2b. Call Alg 3 (steps 2 and 3a)

2c. Set $R \leftarrow R+1$.

If $R < ITERU$, then go to 2b.

Step 3

Set $R \leftarrow 0$.

Set $p \leftarrow p_0$.

Go to 1.

In the above algorithm, $IFLAG$ is used in obtaining a starting y from the solutions in 1a. The bases used in steps 1 and 2 are generated from the optimal bases obtained in the previous iterations.

VI. COMPUTATIONAL EXPERIMENTATION

This section describes the computer implementation, EQFLO, and testing of our algorithm for the equal flow problem. The algorithm was tested on a set of 35 test problems randomly generated using NETGEN [22]. Computation times are compared with those of NETSIDE [2], a general purpose code for network problems with side constraints. Both NETSIDE and EQFLO are written in standard FORTRAN for an incore implementation and have not been tailored to either the machine or FORTRAN compiler used for testing.

6.1 Description of the Computer Codes

NETSIDE was developed by Barr, Farhangian, and Kennington at Southern Methodist University, Dallas, Texas. Designed to solve network problems with side constraints, it used a specialization of the revised simplex method known as the primal partitioning algorithm [15]. The basis inverse is maintained as a rooted spanning tree and a working basis inverse in product form. The reinversion routine is a modification of the work of Hellerman and Rarick [18] and uses the "spike swapping theory" of Helgason and Kennington [16]. The initial working basis consists of a combination of artificial and slack variables. The working basis is reinverted every 50 iterations. The pricing routine uses a candidate list of size 10 with a block size of 400. Both pricing and pivot tolerance are $1.E-6$.

EQFLO is our implementation of ALG 4, and makes use of MODFLO [1] to solve pure network subproblems. MODFLO is a set of subroutines which may

AD-A173 185

DEVELOPMENT AND EVALUATION OF A CASUALTY EVACUATION

2/2

MODEL FOR A EUROPEAN (U) SOUTHERN METHODIST UNIO

DALLAS TX DEPT OF OPERATIONS RESEARCH J L KENNINGTON

UNCLASSIFIED

DEC 83 AFOSR-IR-86-0533 AFOSR-83-0278

F/C 20/1

NL

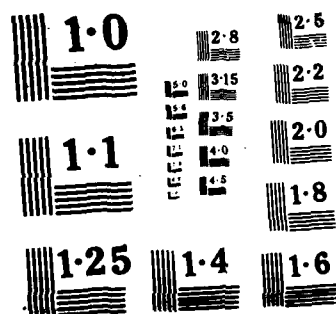
END

DATE

FILED

1986

104



be used to solve a network problem as well as reoptimize after problem data changes. Based on NETFLO [21], this code allows the user to change costs, bounds and/or requirements for a network problem and reoptimize. The tuning parameters used in all runs were as follows: ITERL = 5, ITERU = 10, REFREQ = 5, $\lambda_0 = 0.75$, $p_0 = 0.01$, and $\epsilon = 0.1$. MODFLO [1] is used to reoptimize after each change to either the costs or right-hand-sides.

6.2 The Test Problems

The program NETGEN, a generator for large-scale network test problems, was used to generate 35 test problems. The parameters used to generate these problems are described in Klingman, Napier, and Stutz [22]. The test problems have between 200 and 1500 nodes, and 1500 and 7000 arcs. For each problem, the first 150 arcs were paired to form equal flow sides constraints. The characteristics of these test problems are listed in Table 1.

Our algorithm requires upper bounds on all equal flow arcs. Though NETGEN generates bounds on some of these arcs, there were others with no upper bounds. We set the maximum of all supplies and demands to be the upper bounds on such arcs. These bounds were acceptable since the optimal solutions obtained for our pure network problems were the same as the ones listed in NETGEN. Furthermore, for all 35 test problems the first 150 arcs were used to form 75 pairs of equal flow side constraints. We were unable to experiment with more than 75 pairs due to a core storage limitation of 301K. NETSIDE required approximately 300K octal words of storage for

problems 28 through 35 with 75 pairs and any further increase in the number of pairs would exceed the storage limitation.

Table 1 About Here

6.3 Computational Results

All 35 test problems were solved on the CDC 6600 at Southern Methodist University, using the FTN compiler with OPT = 2. All 35 problems were solved twice using EQFLO; once with the same step size for every pair of equal flow arcs and the second time with different step sizes for different pairs. While using EQFLO to solve these problems, ALG 4 was followed exactly the first time, whereas, the computation of the step size for the upper bound was altered the second time. The modification was as follows:

Step 3 (Move to New Point)

$$3a. \text{ Set } y_k \leftarrow P \left[y_k - \bar{\lambda}_k \frac{(UBND-LBND)}{\| \eta \| ^2} \eta_k \right], \quad k = 1, \dots, K,$$

set $Q \leftarrow Q+1$.

Note that this modification results in different step sizes for different equal flow pairs. The details of all runs are given in Tables 2 and 3. The times are in CPU seconds and exclude input and output.

The value 0.01 used for p_0 worked well for all test problems except problem number 11. This problem experienced difficulties in converging within 10% of the optimal. However, the problem did converge within 10%

of the optimum when we changed p_0 to values between 3 and 10.

Tables 2 and 3 About Here

The computational results presented in Tables 2 and 3 are summarized in Table 4. Letting $T(\text{ALG})$ denote the CPU time required to solve the 35 test problems using code ALG, the relationship is given below:

$$T(\text{NETSIDE}) = 2.54 (\text{EQFLO}), \text{ same step size,}$$

$$T(\text{NETSIDE}) = 3.00 (\text{EQFLO}), \text{ different step sizes.}$$

Note that EQFLO performs better as the problem size increases. Although EQFLO was slightly slower than NETSIDE on problems 1 to 10, its performance increased substantially on problems 11 to 35. In particular, EQFLO ran approximately 5 to 6 times faster than NETSIDE on problems 28 through 35 and these problems are fairly large. We expect EQFLO to perform even better on much larger problems. This is attributable to the fact that the time for pricing and updating increases dramatically for NETSIDE with an increase in the size of the network, whereas, the time increase should be relatively small for EQFLO because the above operations are performed very efficiently using labelling procedures on the rooted spanning tree.

Table 4 About Here

The 75 side constraints made the problems approximately three times harder. That is, the pure networks were solved in 693 seconds while it

required 1973 seconds to solve the equal flow problem. Klingman, Napier, and Stutz [22] solved the same 35 pure network problems in approximately 200 seconds using an advance start on a CDC 6600 at the University of Texas at Austin. Richard Barr's best time on these 35 test problems is 104 seconds using ARC II [3]. This difference in time is due to the fact that EQFLO is a real code (as opposed to all-integer), uses an all artificial start, and does not use the advanced data structure or candidate list incorporated in ARC II.

These 35 problems were the largest that could be solved using NETSIDE under a core storage limitation of 301K octal words. However, EQFLO required much less storage; approximately 50% less than NETSIDE. This additional storage for NETSIDE results from the working basis inverse and the arrays required during the reinversion process.

VII. SUMMARY AND CONCLUSIONS

This paper presents a new procedure for the equal flow problem. Unlike the simplex method for the network problem with side constraints, this new procedure does not require a working basis. We have showed that using the subgradient optimization technique, the equal flow problem may be solved as two sequences of pure network problems. One sequence corresponds to a lower bound while the other corresponds to an upper bound. In the lower bound, each network differs from the previous one in that the cost vector has changed. In the upper bound, each network differs from the previous one in that the right hand side has changed. While solving the pure network problems with these changes in the problem data, a reoptimization procedure is used to obtain a good starting solution. Our technique terminates when the difference between two bounds is within a prespecified tolerance.

Subgradients for upper bounds are computed using the optimal dual variables obtained by solving the pure network problems. The subgradients for lower bounds are the difference between the flows on the equal flow arcs, obtained while solving the Lagrangean relaxation. The projection operation is easily implemented. The step sizes (i) and (iii), described in Section II, are used for lower and upper bounds, respectively. For these step sizes, we are guaranteed a solution at which the objective value is arbitrarily close to the optimal objective value.

We solved all test problems twice; once with the same step size for

all equal flow pairs, and once with different step sizes for each pair. The tests were conducted on a set of 35 randomly generated problems and a comparison was made with NETSIDE, a code that is designed to solve network problems with side constraints. On the average, our code ran approximately 3 times faster. However, it's performance improved substantially as the problem size increased. The new algorithm requires only 50% of the core storage required by NETSIDE.

Table 1 NETGEN Test Problems

Problem Number	Number of Nodes	Number of Arcs
Transportation Problems		
1	100 X 100	1511
2	100 X 100	1700
3	100 X 100	2207
4	100 X 100	2405
5	100 X 100	3100
6	150 X 150	3450
7	150 X 150	4800
8	150 X 150	5470
9	150 X 150	6395
10	150 X 150	6611
Assignment Problems		
11	200 X 200	1900
12	200 X 200	2650
13	200 X 200	3400
14	200 X 200	4150
15	200 X 200	4900
Capacitated Network Problems		
16	400	1374
17	400	2511
18	400	1374
19	400	2511
20	400	1484
21	400	2904
22	400	1484
23	400	2904
24	400	1398
25	400	2692
26	400	1398
27	400	2692
Uncapacitated Network Problems		
28	1000	3000
29	1000	3500
30	1000	4500
31	1000	4900
32	1500	4492
33	1500	4535
34	1500	5257
35	1500	5880

Table 2 Comparison of NETSIDE and EQFLO on 35 Test Problems
(Same step size for every equal flow pair)

Problem Number	NETSIDE		EQFLO					
	Optimal Objective	Total Time	Time				% of Optimal at Termination	
			Total	Pure Network	Lower Bound	Upper Bound	Lower Bound	Upper Bound
1	2694547	30	50	7	19	24	98	108
2	2350637	24	58	7	23	28	95	105
3	1939836	27	127	9	55	63	99	110
4	1612265	33	79	10	33	36	98	108
5	1480741	33	40	12	13	15	97	108
6	2472907	71	46	22	9	15	98	108
7	2236784	96	59	28	14	17	97	107
8	2223900	84	58	32	11	15	99	108
9	1839835	115	44	36	6	2	98	105
10	2291942	105	79	36	19	24	96	106
11*	4992	135	51	17	11	23	98	108
12	3573	105	93	23	20	50	95	105
13	3142	103	78	27	16	35	98	108
14	2787	118	34	31	1	2	99	101
15	2795	150	127	35	31	61	97	108
16	82161432	43	8	6	1	1	99	107
17	45601025	66	13	8	4	1	99	105
18	81600312	40	8	6	1	1	99	106
19	45601025	66	12	8	3	1	99	102
20	74065202	40	9	6	2	1	99	108
21	40137087	44	11	8	2	1	99	101
22	73429862	32	8	6	1	1	99	109
23	39354594	33	11	8	2	1	99	101
24	85926653	91	7	3	3	1	98	104
25	58203746	66	9	5	3	1	99	101
26	74267081	65	6	3	2	1	97	102
27	47295659	57	7	4	2	1	99	107
28	131316225	201	31	20	9	2	99	107
29	113594497	260	167	25	72	70	98	107
30	90569484	337	243	23	111	109	91	106
31	84943754	296	44	24	16	4	99	109
32	180390305	529	80	48	25	7	98	109
33	205246112	453	83	47	23	13	98	108
34	166247998	477	95	51	24	20	96	106
35	163964307	503	68	52	11	5	99	107

* $p_0 = 10$.

Table 3 Comparison of NETSIDE and EQFLO on 35 Test Problems
(Different step sizes for different pairs)

Problem Number	NETSIDE		EQFLO					
	Optimal Objective	Total Time	Time				% of Optimal at Termination	
			Total	Pure Network	Lower Bound	Upper Bound	Lower Bound	Upper Bound
1	2694547	30	47	7	16	24	97	108
2	2350637	24	50	7	19	24	94	105
3	1939836	27	105	9	42	54	99	109
4	1612265	33	74	10	29	35	98	108
5	1480741	33	37	12	10	15	95	106
6	2472907	71	46	22	9	15	98	107
7	2236784	96	57	28	13	16	97	105
8	2223900	84	42	32	4	6	98	109
9	1839835	115	44	36	6	2	98	105
10	2291942	105	76	36	19	21	96	105
11*	4992	135	53	17	9	27	94	104
12	3573	105	79	23	14	42	95	105
13	3142	103	63	27	11	25	98	108
14	2787	118	34	31	1	2	99	101
15	2795	150	108	35	24	49	98	107
16	82161432	43	8	6	1	1	99	107
17	45601025	66	13	8	4	1	99	105
18	81600312	40	8	6	1	1	99	106
19	45601025	66	12	8	3	1	99	102
20	74065202	40	9	6	2	1	99	108
21	40137087	44	11	8	2	1	99	101
22	73429862	32	8	6	1	1	99	109
23	39354594	33	11	8	2	1	99	101
24	85926653	91	7	3	3	1	98	104
25	58203746	66	9	5	3	1	99	101
26	74267081	65	6	3	2	1	97	102
27	47295659	57	7	4	2	1	99	107
28	131316225	201	31	20	9	2	99	107
29	113594497	260	81	25	28	28	97	107
30	90569484	337	148	23	62	63	93	104
31	84943754	296	44	24	16	4	99	109
32	180390305	529	80	48	25	7	98	109
33	205246112	453	78	47	22	9	98	108
34	166247998	477	86	51	23	12	97	107
35	163964307	503	68	52	11	5	99	107

* $p_0 = 10$.

Table 4 Summary of Computational Results

Problems	Time(NETSIDE)	Time(EQFLO)									
		Same step size					Different step sizes				
		Total	Pure Network	Lower Bound	Upper Bound	Total	Pure Network	Lower Bound	Upper Bound	Total	Pure Network
1 - 10	618	640	199	202	239	578	199	167	212		
11 - 15	611	383	133	79	171	337	133	59	145		
16 - 27	643	109	71	26	12	109	71	26	12		
28 - 35	3056	811	290	291	230	616	290	196	130		
Total	4928	1943	693	598	652	1640	693	448	499		

REFERENCES

1. Ali, A., E. Allen, R. Barr, and J. Kennington, "Reoptimization Procedures for Bounded Variable Primal Simplex Network Algorithms", Technical Report 83-OR-2, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, Texas, 75275, (1983).
2. Barr, R., K. Farhangian, and J. Kennington, "Networks with Side Constraints: An LU Factorization Update", Technical Report 83-OR-4, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, Texas, 75275, (1983).
3. Barr, R., F. Glover, and D. Klingman, "Enhancements of Spanning Tree Labelling Procedures for Network Optimization", INFOR, 17, 16-34, (1979).
4. Bazaraa, S., and C. Shetty, Nonlinear Programming: Theory and Algorithms, John Wiley & Sons, New York, N.Y., (1978).
5. Beck, P., L. Lasdon, and M. Engquist, "A Reduced Gradient Algorithm for Nonlinear Network Problems", ACM Transactions on Mathematical Software, 9, 57-70, (1983).
6. Carraraesi, P., and G. Gallo, "Network Models for Vehicle and Crew Scheduling", European Journal of Operations Research, 16, 139-151, (1984).
7. Charnes, A., and W. Cooper, Management Models and Industrial Applications of Linear Programming, Volume II, John Wiley & Sons, New York, N.Y., (1961).

8. Chen, S., and R. Saigal, "A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints", Networks, 7, 59-79, (1977).
9. Fletcher, R., and C. Reeves, "Function Minimization by Conjugate Gradients", Computer Journal, 7, 149-154, (1964).
10. Glover, F., and D. Klingman, "The Simplex SON Algorithm for LP/Embedded Network Problems", Mathematical Programming, 15, 148-176, (1981).
11. Glover, F., R. Glover, and F. Martinson, "The U. S. Bureau of Land Managements's New NETFORM Vegetation Allocation System", Technical Report of the Division of Information Science Research, University of Colorado, Boulder, Colorado, 80309, (1982).
12. Goffin, J., "On Convergence Rates of Subgradient Optimization Methods", Mathematical Programming, 13, 329-347, (1977).
13. Griffith, R., and R. Stewart, "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", Management Science, 7, 379-392, (1964).
14. Grigoriadis, M., and W. White, "A Partitioning Algorithm for the Multicommodity Network Flow Problem", Mathematical Programming, 3, 157-177, (1972).
15. Hartman, J., and L. Lasdon, "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems", Networks, 1, 333-354, (1972).

16. Helgason, R., and J. Kennington, "Spike Swapping in Basis Reinversion", Naval Research Logistics Quarterly, 4, 697-702, (1980).
17. Helgason, R., "A Lagrangean Relaxation Approach to the Generalized Fixed Charge Multicommodity Minimal Cost Network Flow Problem", Unpublished Dissertation, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, Texas, 75275, (1980).
18. Hellerman, E., and D. Rarick, "Reinversion with the Preassigned Pivot Procedure", Mathematical Programming, 1, 195-216, (1971).
19. Kaul, R., "An Extension of Generalized Upper Bounded Techniques for Linear Programming", ORC Report No. 65-27, Department of Operations Research, University of California, Berkeley, California, (1965).
20. Kennington, J., "Solving Multicommodity Transportation Problems Using a Primal Partitioning Simplex Technique", Naval Research Logistics Quarterly, 24, 309-325, (1977).
21. Kennington, J., and R. Helgason, Algorithms for Network Programming, John Wiley & Sons, New York, N.Y., (1980).
22. Klingman, D., A. Napier, and J. Stutz, "NETGEN: A Program for Generating Large Scale Minimum Cost Flow Network Problems", Management Science, 20, 814-821, (1974).
23. Poljak, B., "Minimization of Unsmooth Functionals", U.S.S.R. Computational Mathematics and Mathematical Physics, 9, 14-29, (1969).

24. Sakarovitch, M., and R. Saigal, "An Extension of Generalized Upper Bounding Techniques for Structured Linear Programs", SIAM Journal of Applied Mathematics, 15, 906-914, (1967).
25. Shepardson, F., and R. Marsten, "A Lagrangean Relaxation Algorithm for the Two Duty Period Scheduling Problem", Management Science, 26, 274-281, (1980).
26. Shetty, B., "The Equal Flow Problem", unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, Texas, 75275, (1985).
27. Shorn, N., "On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems", Dissertation, Cybernetics Institute, Academy of Sciences, U.S.S.R., (1964).
28. Turnquist, M., and C. Malandraki, "Estimating Driver Costs for Transit Operations Planning", Presented at the Joint National Meeting of ORSA/TIMS, Dallas, (1984).
29. Zangwill, W., Nonlinear Programming: A Unified Approach, Prentice Hall, Englewood Cliffs, New Jersey, (1969).

Appendix C

Technical Report 85-OR-7

A GENERALIZATION OF POLYAK'S CONVERGENCE
RESULT FOR SUBGRADIENT OPTIMIZATION

By

Ellen Allen¹

Richard Helgason¹

Jeffery Kennington¹

Bala Shetty²

August 1985

¹Department of Operations Research
Southern Methodist University
Dallas, Texas 75275
(214) 692-3072

²Department of Business Analysis
Texas A & M University
College Station, Texas 77843
(409) 845-0810

Comments and criticisms from interested readers are cordially invited.

ABSTRACT

This paper generalizes a practical convergence result first presented by Polyak. This new result presents a theoretical justification for the step size which has been successfully used in several specialized algorithms which incorporate the subgradient optimization approach.

KEY WORDS

Subgradient Optimization

Nonlinear Programming

Convergence

ACKNOWLEDGMENT

This research was supported in part by the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278. We wish to thank David Anderson of the Department of Mathematics of Southern Methodist University for helpful suggestions concerning the form of Proposition 7.

I. THE SUBGRADIENT ALGORITHM

Let g be a finite convex functional on R^n . For each $y \in R^n$, define the subdifferential of g at y by:

$$\partial g(y) = \{\eta : \eta \in R^n \text{ and for all } z \in R^n, g(z) \geq g(y) + \eta \cdot (z - y)\}.$$

Any $\eta \in \partial g(y)$ is called a subgradient of g at y . It is well known that if y is a point at which g is differentiable, then $\partial g(y) = \{\nabla g(y)\}$, a singleton set.

Let $G \neq \emptyset$ be a closed and convex subset of R^n . For each $y \in R^n$, define the projection of y on G , denoted by $P(y)$, to be the unique point of G such that for all $z \in G$, $\|P(y) - y\| \leq \|z - y\|$. It is well known that the projection exists in this case and that for all $x, y \in R^n$ $\|P(x) - P(y)\| \leq \|x - y\|$.

Consider the nonlinear programming problem given by:

$$\text{minimize } g(y)$$

(NLP/SD)

$$\text{subject to } y \in G,$$

where we assume that for all $y \in G$, $\partial g(y) \neq \emptyset$ and that the set of optimal points $\Gamma \neq \emptyset$. We denote the optimal objective value by γ .

The subgradient optimization algorithm for the solution of NLP/SD was first introduced by Shor [11] and may be viewed as a generalization of the steepest descent method in which any subgradient is substituted for the gradient at a point where the gradient does not exist. This algorithm uses a sequence of positive step sizes $\{s_i\}$, which in turn depend on a predetermined sequence of fixed constants $\{\lambda_i\}$ and (in some cases) certain other quantities.

SUBGRADIENT OPTIMIZATION ALGORITHM

Step 0 (Initialization)

Let $y_0 \in G$ and set $i \leftarrow 0$.

Step 1 (Find Subgradient and Step Size)

Obtain some $\eta_i \in \partial g(y_i)$.

If $\eta_i = 0$, terminate with y_i optimal; otherwise, select a step size s_i .

Step 2 (Move to New Point)

Set $y_{i+1} \leftarrow P(y_i - s_i \eta_i)$, $i \leftarrow i+1$, and return to step 1.

Unfortunately, the termination criterion in step 1 may not hold at any member of Γ and is thus computationally ineffective. Hence, some other stopping rule must be devised. In practice this is often a limit on the number of iterations. The functional values produced by the algorithm will be denoted by $g_i = g(y_i)$.

Various proposals have been offered for the selection of the step sizes. Four general schema which have been suggested are:

$$s_i = \lambda_i, \quad (1)$$

$$s_i = \lambda_i / \|\eta_i\|, \quad (2)$$

$$s_i = \lambda_i / \|\eta_i\|^2, \quad (3)$$

$$s_i = \lambda_i (g_i - \rho) / \|\eta_i\|^2, \quad (4)$$

where ρ , the target value, is an estimate of γ and all $\lambda_i > 0$.

The papers of Polyak [9] and Held, Wolfe, and Crowder [6] have provided the major impetus for widespread practical application of the algorithm. Schema (4) has proven to be a particularly popular choice

among experimenters. Theorem 4 of Polyak [9] is the most often quoted convergence result justifying use of this schema. For many mathematical programming models, the target value is a lower bound on the optimum (i.e., [1, 2, 3, 4, 7, 8, 10]). For this case Polyak's Theorem 4, using schema (4), requires that $\lambda_i = 1$ for all i . For all the above studies, a decreasing sequence of λ 's was found to work better than $\lambda_i = 1$ for all i . Hence, the existing theory did not justify what we had found to work well in practice. The objective of this paper is to present new theoretical results which help to explain what has been found to work well in practice. Specifically, we have generalized Polyak's result for a decreasing sequence of λ 's. In addition, we also loosen slightly the restrictions imposed on the sequence $\{\lambda_i\}$ when the target value is larger than γ .

II. POLYAK'S CONVERGENCE RESULTS

Some of the convergence results for the subgradient optimization algorithm appear unusual in that they specify only that a functional value within a given tolerance of the optimal value γ will eventually be produced. The results of Theorem 4 of Polyak [9] use the following general restrictions on the sequence $\{\lambda_i\}$ used with schema (4):

$$0 < \alpha \leq \lambda_i \leq \beta < 2, \quad (5)$$

where α and β are fixed constants.

The results contained in this theorem include:

under (4), (5), and (essentially) the assumption that there is some $\kappa > 0$ such that $\|\eta_i\| < \kappa$,

(A) if $\rho > \gamma$, either

(a) there is some n such that $g_n < \rho$,

or

(b) all $g_n \geq \rho$ and $\lim g_n = \rho$;

and

(B) if $\rho < \gamma$ and all $\lambda_n = 1$, given $\delta > 0$,

there is some n such that $g_n \leq \gamma + (\gamma - \rho) + \delta$.

In the next section we will relax condition (5) to the following:

$$0 < \lambda_i \leq \beta < 2 \text{ and } \sum \lambda_i = \infty, \quad (6)$$

where β is a fixed constant, and we will present a generalization of (B) for a decreasing sequence $\{\lambda_i\}$.

III. NEW CONVERGENCE RESULTS

The main results in this section appear in Propositions 5 and 7.

Proposition 5 corresponds to part A of Polyak's Theorem 4 with slightly weaker conditions on the sequence $\{\lambda_i\}$, and Proposition 7 is a generalization of part B of Theorem 4.

Proposition 1

If $y \in G$, then

$$\|y - y_{i+1}\|^2 \leq \|y - y_i\|^2 + s_i^2 \|\eta_i\|^2 + 2s_i (g(y) - g_i).$$

Proof

Let $y \in G$.

$$\begin{aligned} \|y - y_{i+1}\|^2 &= \|y - P(y_i - s_i \eta_i)\|^2 \\ &= \|P(y) - P(y_i - s_i \eta_i)\|^2 \\ &\leq \|y - y_i + s_i \eta_i\|^2 \\ &= \|y - y_i\|^2 + s_i^2 \|\eta_i\|^2 + 2s_i \eta_i \cdot (y - y_i) \\ &\leq \|y - y_i\|^2 + s_i^2 \|\eta_i\|^2 + 2s_i (g(y) - g_i). \end{aligned}$$

Proposition 2

If $y \in \Gamma$, then under (4),

$$\|y - y_{i+1}\|^2 \leq \|y - y_i\|^2 + \lambda_i (g_i - \rho) [\lambda_i (g_i - \rho) - 2(g_i - \gamma)] / \|\eta_i\|^2.$$

Proof

Let $y \in \Gamma$. Substituting in Proposition 1 for s_i from (4) and using $g(y) = \gamma$, we obtain

$$\begin{aligned} \|y - y_{i+1}\|^2 &\leq \|y - y_i\|^2 + \lambda_i^2 (g_i - \rho)^2 / \|\eta_i\|^2 + 2\lambda_i (g_i - \rho) (\gamma - g_i) / \|\eta_i\|^2 \\ &= \|y - y_i\|^2 + \lambda_i (g_i - \rho) [\lambda_i (g_i - \rho) - 2(g_i - \gamma)] / \|\eta_i\|^2. \end{aligned}$$

Proposition 3

If $y \in \Gamma$, $\rho \geq \gamma$, and $g_i \geq \rho$, then under (4),

$$\|y - y_{i+1}\|^2 \leq \|y - y_i\|^2 + \lambda_i (\lambda_i - 2) (g_i - \rho)^2 / \|\eta_i\|^2.$$

Proof

Let $y \in \Gamma$, $\rho \geq \gamma$, and $g_i \geq \rho$.

Now, $\lambda_i (g_i - \rho) - 2(g_i - \gamma) \leq \lambda_i (g_i - \rho) - 2(g_i - \rho) = (\lambda_i - 2) (g_i - \rho)$.

Thus, $\lambda_i (g_i - \rho) [\lambda_i (g_i - \rho) - 2(g_i - \gamma)] / \|\eta_i\|^2 \leq \lambda_i (\lambda_i - 2) (g_i - \rho)^2 / \|\eta_i\|^2$.

The result now follows from Proposition 2.

Proposition 4

If $y \in \Gamma$, $\rho \geq \gamma$, and all $g_i \geq \rho$, then under (4) with all $\lambda_i < 2$, there is some ψ such that $\lim \|y - y_i\|^2 = \psi$.

Proof

Let $y \in \Gamma$, $\rho \geq \gamma$, all $\lambda_i < 2$, and all $g_i \geq \rho$. Since each $\lambda_i < 2$, then also each $\lambda_i (\lambda_i - 2) (g_i - \rho)^2 / \|\eta_i\|^2 \leq 0$, and from Proposition 3, $\{\|y - y_i\|^2\}$ is a monotone nonincreasing sequence. This sequence is bounded below by zero and thus converges to some value, say ψ .

Proposition 5

If $\rho \geq \gamma$ and there is some $\kappa > 0$ such that all $\|\eta_i\| < \kappa$, then under (4) and (6), given $\delta > 0$, there is some M such that $g_M \leq \rho + \delta$.

Proof

Let $\delta > 0$ be given, with $\rho \geq \gamma$, and all $\|\eta_i\| < \kappa$. Suppose, contrary to the desired result, that all $g_i > \rho + \delta$. Take any $y \in \Gamma$. Then from Proposition 3,

$$\lambda_i (2 - \lambda_i) (g_i - \rho)^2 / \|\eta_i\|^2 \leq \|y - y_i\|^2 - \|y - y_{i+1}\|^2.$$

Since $\lambda_1 \leq \beta < 2$, $\|\eta_1\| < \kappa$, and $g_1 - \rho > \delta$,

$$\lambda_1 (2-\beta) \delta^2 / \kappa^2 \leq \|y-y_1\|^2 - \|y-y_{i+1}\|^2. \quad (7)$$

Adding together the inequalities obtained from (7) by letting i take on all values from 0 to n , we obtain

$$(\lambda_0 + \dots + \lambda_n) (2-\beta) \delta^2 / \kappa^2 \leq \|y-y_0\|^2 - \|y-y_{n+1}\|^2. \quad (8)$$

As n goes to ∞ , the left side of (8) goes to ∞ , whereas, by Proposition 4, the right side of (8) goes to $\|y-y_0\|^2 - \psi^2$, a contradiction.

Proposition 5 gives a practical convergence result when the target exceeds the optimal value. At worst we eventually obtain an objective value arbitrarily close to the target value.

Proposition 6

If $y \in \Gamma$, $g_1 \geq \rho$, and $\lambda_1 \leq \beta \neq 2$, then under (4),

$$\begin{aligned} \|y-y_{i+1}\|^2 &\leq \|y-y_i\|^2 \\ &\quad + \lambda_1 (g_1 - \rho) (2-\beta) [(\gamma - g_1) + (\beta/(2-\beta)) (\gamma - \rho)] / \|\eta_1\|^2. \end{aligned}$$

Proof

Let $y \in \Gamma$, $g_1 \geq \rho$, and $\lambda_1 \leq \beta \neq 2$.

$$\begin{aligned} \text{Now, } \lambda_1 (g_1 - \rho) - 2(g_1 - \gamma) &\leq \beta(g_1 - \rho) - 2(g_1 - \gamma) \\ &= \beta(g_1 - \rho) - (2-\beta)(g_1 - \gamma) - \beta(g_1 - \gamma) \\ &= \beta(\gamma - \rho) + (2-\beta)(\gamma - g_1) \\ &= (2-\beta) [(\gamma - g_1) + (\beta/(2-\beta)) (\gamma - \rho)]. \end{aligned}$$

$$\begin{aligned} \text{Thus, } \lambda_1 (g_1 - \rho) [\lambda_1 (g_1 - \rho) - 2(g_1 - \gamma)] / \|\eta_1\|^2 \\ \leq \lambda_1 (g_1 - \rho) (2-\beta) [(\gamma - g_1) + (\beta/(2-\beta)) (\gamma - \rho)] / \|\eta_1\|^2. \end{aligned}$$

The result now follows from Proposition 2.

Proposition 7

If $\rho < \gamma$ and there is some $\kappa > 0$ such that all $\|\eta_i\| < \kappa$, then under

(4) and (6), given $\delta > 0$, there is some M such that $g_M \leq \gamma + (\beta/(2-\beta)) (\gamma - \rho) + \delta$.

Proof

Let $\delta > 0$ be given, with $\rho < \gamma$, and all $\|\eta_i\| < \kappa$. Suppose, contrary to the desired result, that all $g_i > \gamma + (\beta/(2-\beta))(\gamma - \rho) + \delta$, or $(\gamma - g_i) + (\beta/(2-\beta))(\gamma - \rho) < -\delta$. Since $\beta < 2$ and $g_i > \rho$, then

$$\begin{aligned} & \lambda_i (g_i - \rho) (2 - \beta) [(\gamma - g_i) + (\beta/(2 - \beta)) (\gamma - \rho)] / \|\eta_i\|^2 \\ & < -\delta \lambda_i (g_i - \rho) (2 - \beta) / \|\eta_i\|^2. \end{aligned} \quad (9)$$

Take any $y \in \Gamma$. Then by (9) and Proposition 6, we have that

$$\delta \lambda_i (g_i - \rho) (2 - \beta) / \|\eta_i\|^2 < \|y - y_i\|^2 - \|y - y_{i+1}\|^2.$$

Since $\|\eta_i\| < \kappa$ and $g_i \geq \gamma > \rho$, then also

$$\lambda_i \delta (\gamma - \rho) (2 - \beta) / \kappa^2 < \|y - y_i\|^2 - \|y - y_{i+1}\|^2. \quad (10)$$

Adding together the inequalities obtained from (10) by letting i take on all values from 0 to n , we obtain

$$(\lambda_0 + \dots + \lambda_n) \delta (\gamma - \rho) (2 - \beta) / \kappa^2 < \|y - y_0\|^2 - \|y - y_{n+1}\|^2. \quad (11)$$

As n goes to ∞ , the left side of (11) goes to ∞ , whereas, by Proposition 4, the right side of (11) goes to $\|y - y_0\|^2 - \psi^2$, a contradiction.

The above is our generalization of Polyak's Theorem 4 Part B.

At worst we eventually obtain an objective value whose error is arbitrarily close to $\beta/(2-\beta)$ times the error present in the target value estimate of γ .

IV. CONCLUSIONS

Proposition 5 gives the convergence result obtained under (4) and (6) for a target value at or above the optimal value. It is readily apparent that Proposition 5 is compatible with Polyak's result (A). Proposition 7 gives the corresponding result for a target value under the optimal value. We have found this to be a more practical result (see e.g., [1, 2, 3, 4, 7, 8, 10]). Taking $\beta=1$, we have Polyak's result (B) as a special case of Proposition 7. Proposition 7 shows more clearly the dependence of the demonstrably attainable error on the upper bound β for $\{\lambda_i\}$. This paper has not addressed the question of any convergence rate associated with the use of (4) and (6). Goffin [5] has provided such results when schema (2) is used.

REFERENCES

1. Ali, I., "Two Node-Routing Problems," unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1980).
2. Ali, I., and J. Kennington, "The Asymmetric M-Travelling Salesman Problem: A Duality Based Branch-and-Bound Algorithm," (to appear in Discrete Applied Mathematics).
3. Ali, I., J. Kennington, and B. Shetty, "The Equal Flow Problem," Technical Report 85-OR-1, Operations Research Department, Southern Methodist University, Dallas, Texas, (1980).
4. Allen, E., "Using Two Sequences of Pure Network Problems to Solve the Multicommodity Network Flow Problem," unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1985).
5. Goffin, J., "On Convergence Rates of Subgradient Optimization Methods," Mathematical Programming, 13, 329-347, (1977).
6. Held, M., P. Wolfe, and H. Crowder, "Validation of Subgradient Optimization," Mathematical Programming, 6, 66-68, (1974).
7. Helgason, R., "A Lagrangean Relaxation Approach to the Generalized Fixed Charge Multicommodity Minimal Cost Network Flow Problem," unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1980).

8. Kennington, J., and M. Shalaby, "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems," Management Science, 23, 9, 994-1004, (1977).
9. Polyak, B., "Minimization of Unsmooth Functionals," U.S.S.R. Computational Mathematics and Mathematical Physics, 9, 14-29, (1969).
10. Shetty, B., "The Equal Flow Problem," unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1985).
11. Shor, N., "On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems," dissertation, Cybernetics Institute, Academy of Science, U.S.S.R., (1964).

Appendix D

Technical Report 83-OR-4

NETWORKS WITH SIDE CONSTRAINTS:
AN LU FACTORIZATION UPDATE

By

Richard S Barr¹

Keyvan Farhangian²

Jeffery L. Kennington¹

Southern Methodist University
Dallas, Texas 75275
(214)-692-3072

Revised

September 1985

¹ Department of Operations Research
School of Engineering and Applied Science
Southern Methodist University
Dallas, Texas

² Consilium Associates, Inc.
Palo Alto, California

ABSTRACT

An important class of mathematical programming models which are frequently used in logistics studies is the model of a network problem having additional linear constraints. A specialization of the primal simplex algorithm which exploits the network structure can be applied to this problem class. This specialization maintains the basis as a rooted spanning tree and a general matrix called the working basis. This paper presents the algorithms which may be used to maintain the inverse of this working basis as an LU factorization, which is the industry standard for general linear programming software. Our specialized code exploits not only the network structure but also the sparsity characteristics of the working basis. Computational experimentation indicates that our LU implementation results in a 50% savings in the nonzero elements in the eta file, and our computer codes are approximately twice as fast as MINOS and XMP on a set of randomly generated multicommodity network flow problems.

ACKNOWLEDGEMENT

This research was supported in part by the Department of Defense under Contract Number MDA903-82-C-0440 and the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278.

KEY WORDS

Linear Programming

Network Models

Networks With Side Constraints

Logistics

NOTE TO EDITOR

PLEASE SET ALL NUMERALS UNDERLINED AND LETTERS UNDERLINED IN BOLDFACE.

DO NOT SET UNDERLINES.

I. INTRODUCTION

Good software for solving linear programming models is one of the most important tools available to the logistics engineer. For logistics studies, these linear programs frequently involve a very large network of nodes and arcs, which may be duplicated by time period. For example, nodes may represent given cities at a particular point in time while arcs represent roads, railways, and legs of flights connecting these cities. Some nodes are designated as supply nodes, others demand nodes, while some may simply represent points of transshipment. The mathematical model characterizes a solution such that the supply is shipped to the demand nodes at least cost while not violating either the upper or lower bounds on the flow over an arc.

If the main structure of a logistics problem can be captured in a network model, then the size of solvable problems becomes enormous. Hence, more realistic situations can be modelled that would otherwise lie outside the domain of general linear programming techniques. For example, one current logistics planning model involves 200 nodes and (365 days/yr) (30 years) = 10,950 time periods to give over 2,000,000 constraints. Network problems having 20,000 constraints and 20,000,000 variables are solved routinely at the U. S. Treasury Department.

Unfortunately, the pure network structure may require simplification of the problem to the point that key policy restrictions must be omitted. The work presented in this study builds upon existing large-scale network solution technology to allow for the inclusion of arbitrary additional

constraints. Typical constraints include capacities on vehicles carrying different types of goods, restrictions on the total number of vehicles available for assignment, and budget restrictions. The addition of even a few non-network constraints can greatly enhance the realism and usability of these models. Our approach exploits--to as great an extent as possible--the traditional network portion of the problem while simultaneously enforcing any additional restrictions imposed by the practitioner.

For general linear programming systems, the most important component is the algorithm used to update the basis inverse. Due to the excellent sparsity and numerical stability characteristics, an LU factorization with either a Bartels-Golub or Forrest-Tomlin update has been adopted for modern linear programming systems. For pure network problems, the basis is always triangular and corresponds to a rooted spanning tree. The modern network codes which exploit this structure have been found to be from one to two orders of magnitude faster than the general linear programming systems. In this paper, we have combined these two powerful techniques into an algorithm for solving network models having additional side constraints.

Let A be an $\bar{m} \times \bar{n}$ matrix, let $\underline{c}$ and $\underline{u}$ be $\bar{n}$ -component vectors, and let $\underline{b}$ be an $\bar{m}$ -component vector. Without loss of generality, the linear program may be stated mathematically as follows:

$$\text{minimize} \quad \underline{c} \cdot \underline{x} \quad (1)$$

$$\text{subject to:} \quad A \underline{x} = \underline{b} \quad (2)$$

$$\underline{0} \leq \underline{x} \leq \underline{u} . \quad (3)$$

The network with side constraint model is a special case of (1) - (3)

in which A takes the form

$$A = \left[\begin{array}{c|c} M & \\ \hline S & P \end{array} \right] \left. \vphantom{\begin{array}{c|c} M & \\ \hline S & P \end{array}} \right\} \begin{array}{l} n \\ m \end{array}$$

where M is a node-arc incidence matrix.

1.1 Applications

There are numerous applications of the network with side constraint model. Professor Glover and his colleagues have solved a large passenger-mix model for Frontier Airlines and a large land management model for the Bureau of Land Management (see [7, 8]). A world grain export model has been solved to help analyze the port capacity of U. S. ports during the next decade (see [2]). A cargo routing model is being used by the Air Force Logistics Command to assist in routing cargo planes for the distribution of serviceable spares (see [1]). Lt. Col. Dennis McLain, has developed a large model to assist in the development of a casualty evacuation plan in the event of a European conflict (see [14]). A National Forest Management Model has been developed to aid forest managers in long term planning for national forests

(see [10]). In addition, work is currently underway which attempts to convert general linear programs into the network with side constraint model (see [4, 16]).

1.2 Objective of Investigation

Due to both storage and time considerations, the basis inverse is maintained as an LU factorization in modern LP software (see [3,5, 15]). The objective of this investigation is to extend these ideas to the primal partitioning algorithm when applied to the network with side constraints model.

1.3 Notation

The i^{th} component of the vector $\underline{a}$ will be denoted by a_i . The $(i,j)^{\text{th}}$ element of the matrix A is denoted by A_{ij} . $A(i)$ and $A[i]$ denotes the i^{th} column and i^{th} row of the matrix A , respectively. $\underline{0}$ denotes a vector of zeroes, $\underline{1}$ denotes a vector of ones, and $\underline{e}^k$ denotes a vector with a 1 in the k^{th} position and zeroes elsewhere. Sigma is used to denote the scalar signum function defined by

$$\sigma(y) = \begin{cases} 1, & \text{if } y > 0 \\ 0, & \text{if } y = 0 \\ -1, & \text{if } y < 0 \end{cases}$$

The identity matrix is given by "I".

II. THE PRIMAL SIMPLEX ALGORITHM

We assume that A has full row rank and that there exist a feasible solution for (1) - (3). Given a basic feasible solution, we may partition A , $\underline{c}$, $\underline{x}$, and $\underline{u}$ into basic and nonbasic components, that is, $A = [B \mid N]$, $\underline{c} = [\underline{c}^B \mid \underline{c}^N]$, $\underline{x} = [\underline{x}^B \mid \underline{x}^N]$, and $\underline{u} = [\underline{u}^B \mid \underline{u}^N]$. Using the above partitioning, the primal simplex algorithm may be stated as follows:

PRIMAL SIMPLEX ALGORITHM

0. *Initialization.* Let $[\underline{x}^B \mid \underline{x}^N]$ be a basic feasible solution.
1. *Pricing.* Let $\underline{\Pi} = \underline{c}^B B^{-1}$. Define

$$\psi_1 = \{i : x_i^N = 0 \text{ and } \underline{\Pi} N(i) > c_i^N\},$$

$$\psi_2 = \{i : x_i^N = u_i^N \text{ and } \underline{\Pi} N(i) < c_i^N\}.$$
 If $\psi_1 \cup \psi_2 = \emptyset$, terminate with $[\underline{x}^B \mid \underline{x}^N]$ optimal; otherwise, select $k \in \psi_1 \cup \psi_2$ and set $\delta \leftarrow 1$ if $k \in \psi_1$ and $\delta \leftarrow -1$, otherwise.

2. *Ratio Test.* Set $\underline{y} \leftarrow B^{-1} N(k)$. Set

$$\Delta_1 \leftarrow \min_{\sigma(y_j)=\delta} \left\{ \frac{x_j^B}{|y_j|}, \infty \right\}$$

$$\Delta_2 \leftarrow \min_{-\sigma(y_j)=\delta} \left\{ \frac{u_j^B - x_j^B}{|y_j|}, \infty \right\}$$

Set $\Delta \leftarrow \min \{\Delta_1, \Delta_2, u_k^N\}$.

If $\Delta \neq \infty$, then go to 3; otherwise, terminate with the conclusion that the problem is unbounded.

3. *Update Values.* Set $x_k^N \leftarrow x_k^N + \Delta \delta$ and $\underline{x}^B \leftarrow \underline{x}^B - \Delta \delta \underline{y}$. If $\Delta = u_k^N$, return to step 1.

4. *Update Basis Inverse.* Let

$$\psi_3 = \{j : x_j^B = 0 \text{ and } \sigma(y_j) = \delta\}$$

$$\psi_4 = \{j : x_j^B = u_j^B \text{ and } -\sigma(y_j) = \delta\}.$$

Select any $l \in \psi_3 \cup \psi_4$. In the basis, replace $B(l)$ with $N(k)$, update the inverse of the new basis, and return to step 1.

III. THE PARTITIONED BASIS

The network with side constraint model may be stated as follows:

$$\text{minimize} \quad \underline{c}^1 \underline{x}^1 + \underline{c}^2 \underline{x}^2 \quad (4)$$

$$\text{subject to:} \quad M \underline{x}^1 = \underline{b}^1 \quad (5)$$

$$S \underline{x}^1 + P \underline{x}^2 = \underline{b}^2 \quad (6)$$

$$0 \leq \underline{x}^1 \leq \underline{u}^1 \quad (7)$$

$$0 \leq \underline{x}^2 \leq \underline{u}^2 \quad (8)$$

We may assume without loss of generality that,

- (i) The graph associated with M has n nodes and is connected (i.e., there exists an undirected path between every pair of nodes).
- (ii) $[S \mid P]$ has full row rank (i.e., $\text{rank } [S \mid P] = m$).
- (iii) Total supply equals total demand (i.e., $\sum \underline{b}^1 = 0$).

Since the rank of system (5) is one less than the number of rows, we add what has been called the root arc to (5) to obtain

$$M \underline{x}^1 + \underline{e}^p a = \underline{b}^1$$

where $0 \leq a \leq 0$ and $1 \leq p \leq n$.

Then the constraint matrix for the network with side constraints model becomes

$$A = \left[\begin{array}{c|c|c} M & & \underline{e}^p \\ \hline S & P & \end{array} \right] .$$

It is well-known that every basis for A may be placed in the form

$$B = \left[\begin{array}{c|c} T & C \\ \hline D & F \end{array} \right] \quad (9)$$

where T corresponds to a rooted spanning tree and

$$B^{-1} = \left[\begin{array}{c|c} T^{-1} + T^{-1} C Q^{-1} D T^{-1} & -T^{-1} C Q^{-1} \\ \hline -Q^{-1} D T^{-1} & Q^{-1} \end{array} \right] \quad (10)$$

where $Q = F - D T^{-1} C$. The objective of this paper is to give algorithms which maintain Q^{-1} as an LU factorization.

IV. THE INVERSE UPDATE

Recall that the partitioned basis takes the form

$$B = \begin{array}{c} \begin{array}{cc} \text{key} & \text{nonkey} \end{array} \\ \left[\begin{array}{cc|cc} & & & \\ T & & C & \\ \hline & & & \\ D & & F & \end{array} \right] \end{array}.$$

Let

$$L = \left[\begin{array}{cc|cc} & & & \\ T^{-1} & & -T^{-1}C & \\ \hline & & & \\ & & I & \end{array} \right]$$

and let

$$\bar{B} = B L = \left[\begin{array}{cc|cc} & & & \\ I & & & \\ \hline & & & \\ DT^{-1} & & Q & \end{array} \right].$$

The inverse update requires a technique for obtaining a new Q^{-1} after a basis exchange. Let $\bar{B}_i$, L_i , B_i , and Q_i denote the above matrices at iteration i . Then we want an expression for Q_{i+1}^{-1} in terms of Q_i^{-1} .

The transformation takes the form

$$B_{i+1}^{-1} = E B_i^{-1} \quad (11)$$

where E is either an elementary column matrix or a permutation matrix.

Let E be partitioned to be compatible with B . That is,

$$E = \left[\begin{array}{cc|cc} & & & \\ E_1 & & E_2 & \\ \hline & & & \\ E_3 & & E_4 & \end{array} \right] \left. \begin{array}{l} n \\ m \end{array} \right\}$$

$\underbrace{\hspace{1.5cm}}_n \quad \underbrace{\hspace{1.5cm}}_m$

By examining the (2,2) partition of $\bar{B}_{i+1}^{-1}$, we obtain

$$Q_{i+1}^{-1} = (E_4 - E_3 T^{-1} C) Q_i^{-1} \quad (12)$$

In determining the updating formulae, we must examine two major cases with subcases.

Case 1. The leaving column is nonkey. For this case, E takes the form

$$\begin{bmatrix} I & E_2 \\ & E_4 \end{bmatrix}.$$

and (12) reduces to $Q_{i+1}^{-1} = E_4 Q_1^{-1}$.

Case 2. The leaving column is key.

Let $\underline{\gamma} = \underline{e}^j T^{-1} C$. If $\gamma_k \neq 0$, then the k^{th} column of C can be interchanged with the j^{th} column of T and the new T will be nonsingular.

Subcase 2a. $\underline{\gamma} \neq \underline{0}$. Suppose $\gamma_k \neq 0$.

Then $E_4 = E_3 T^{-1} C$ reduces to

$$R = \begin{bmatrix} I & & \\ & \underline{e}^j T^{-1} C & \\ & & I \end{bmatrix} \leftarrow \text{row } j \quad (13)$$

, and

$Q_{i+1}^{-1} = R Q_1^{-1}$. Case 1 is applied to complete the update.

Subcase 2b. $\underline{\gamma} = \underline{0}$. For this case no interchange is possible, the entering column becomes key, and $Q_{i+1}^{-1} = Q_1^{-1}$.

V. AN LU UPDATE

Let

$$U^i = \begin{array}{|ccc|} \hline I & \begin{array}{c} u_1 \\ \vdots \\ u_{i-1} \end{array} & 0 \\ \hline & 1 & \\ \hline 0 & & I \\ \hline \end{array},$$

and

$$L^i = \begin{array}{|cc|} \hline I & 0 \\ \hline & \begin{array}{c} l_i \\ l_{i+1} \\ \vdots \\ l_m \end{array} \\ \hline 0 & I \\ \hline \end{array}.$$

Matrices of the form given by U^i and L^i are called upper etas and lower etas, respectively. Suppose we have a factorization of Q^{-1} in the form

$$Q^{-1} = U^1 U^2 \dots U^m F^S F^{S-1} \dots F^1, \quad (14)$$

where $F^1, \dots, F^S$ are a combination of row and column etas. The right side of (14) is referred to as the eta file where only the non-identity rows and columns are stored. Suppose that the k^{th} column of Q is replaced by $\hat{Q}(k)$ to form the new m by m working basis $\hat{Q}$. This section presents algorithms which may be used to update (14) to produce $\hat{Q}^{-1}$ in the same form.

5.1 Nonkey Column Leaves The Basis

If $k = m$, then let $\underline{\beta} = F^S \dots F^1 \hat{Q}(k)$, let

$$\underline{\lambda}_L^m = \left[\begin{array}{c|c} I & \\ \hline & 1/\beta_m \end{array} \right],$$

and let

$$\underline{\lambda}_U^m = \left[\begin{array}{c|c} I & \begin{array}{c} -\beta_1 \\ \vdots \\ -\beta_{m-1} \end{array} \\ \hline & 1 \end{array} \right].$$

We will show that $\hat{Q}^{-1} = U^1 \dots U^{m-1} \tilde{U}^m \tilde{L}^m F^S \dots F^1$.

If $k < m$, then let $R^k = I$ and

$$Q^{-1} = U^1 \dots U^k R^k U^{k+1} \dots U^m F^S \dots F^1. \quad (15)$$

We next define a new upper eta, $\tilde{U}^k$, and a new row eta, R^{k+1} , such that

$$R^k U^{k+1} = \tilde{U}^k R^{k+1}. \quad (16)$$

Substituting (16) into (15) yields

$$Q^{-1} = U^1 \dots U^k \tilde{U}^k R^{k+1} U^{k+2} \dots U^m F^S \dots F^1. \quad (17)$$

We again define two new eta's, $\tilde{U}^{k+1}$ and R^{k+2} , such that

$$R^{k+1} U^{k+2} = \tilde{U}^{k+1} R^{k+2}. \quad (18)$$

Substituting (18) into (17) yields $Q^{-1} = U^1 \dots U^k \tilde{U}^k \tilde{U}^{k+1} R^{k+2} U^{k+3} \dots U^m F^S \dots F^1$.

Repeating this process eventually yields

$$Q^{-1} = U^1 \dots U^k \tilde{U}^k \dots \tilde{U}^{m-1} R^m F^S \dots F^1. \quad (19)$$

Let $\gamma = R^m F^S \dots F^1 \hat{Q}(k)$, let

$$\tilde{L}^m = \begin{bmatrix} I & & & \\ & 1/\gamma_k & & \\ & -\gamma_{k+1}/\gamma_k & & \\ & \vdots & & \\ & -\gamma_m/\gamma_k & & I \end{bmatrix},$$

and let

$$\hat{U}^m = \begin{bmatrix} I & \begin{matrix} -\gamma_1 \\ \vdots \\ -\gamma_{k-1} \end{matrix} & \\ \hline & 1 & \\ & \hline & & I \end{bmatrix}.$$

Then $\hat{U}^m \hat{L}^m \underline{Y} = \underline{e}^k$ and we will show that $\hat{Q}^{-1} = U^1 \dots U^{k-1} \hat{U}^k \dots \hat{U}^m$
 $\hat{L}^m R^m F^s \dots F^1$.

We now present the algorithm which updates the LU representation of Q^{-1} when the leaving column is nonkey. Assume that $\hat{Q}(k)$ is replacing $Q(k)$ in the working basis.

ALG 1: LU UPDATE FOR NONKEY LEAVING COLUMN

1. Set $\underline{\beta} \leftarrow F^S \dots F^1 \hat{Q}(k)$.
2. If $k \neq m$, set $\ell \leftarrow k$, $R^\ell \leftarrow I$, go to 4.
3. Set $\tilde{L}^m \leftarrow I$, where I is m by m .
 Set $\tilde{L}_{mm}^m \leftarrow 1/\beta_m$.
 Set $\tilde{U}^m \leftarrow I$, where I is m by m .
 Set $\tilde{U}_{jm}^m \leftarrow -\beta_j$, for $1 \leq j < m$.
 Stop with $\hat{Q}^{-1} = U^1 \dots U^{m-1} \tilde{U}^m \tilde{L}^m F^S \dots F^1$.
4. Set $\alpha \leftarrow R^\ell[k] U^{\ell+1}(\ell+1)$.
 Set $R^{\ell+1} \leftarrow R^\ell$.
 Set $R_{k,\ell+1}^{\ell+1} \leftarrow \alpha$.
 Set $\tilde{U}^\ell \leftarrow U^{\ell+1}$.
 Set $\tilde{U}_{k,\ell+1}^\ell \leftarrow 0$.
 ($R^\ell U^{\ell+1} = \tilde{U}^\ell R^{\ell+1}$)
 Set $\ell \leftarrow \ell + 1$.
5. If $\ell < m$, go to 4.
 ($U^{k+1} \dots U^m = \tilde{U}^k \dots \tilde{U}^{m-1} R^m$.)
 Set $\underline{\beta} \leftarrow R^m \underline{\beta}$.
6. Set $\tilde{L}^m \leftarrow I$, where I is m by m .
 Set $\tilde{L}_{kk}^m \leftarrow 1/\beta_k$.
 Set $\tilde{L}_{jk}^m \leftarrow -\beta_j/\beta_k$, for $k < j \leq m$.
 Set $\tilde{U}^m \leftarrow I$, where I is m by m .
 Set $\tilde{U}_{jk}^m \leftarrow -\beta_j$, for $1 \leq j \leq k$.
 Set $\tilde{U}_{kk}^m \leftarrow 1$.
 Stop with $\hat{Q}^{-1} = U^1 \dots U^{k-1} \tilde{U}^k \tilde{U}^{k+1} \dots \tilde{U}^m \tilde{L}^m R^m F^S \dots F^1$.

We now present the justification for step 3 of ALG 1. For $k = m$, we claim that $\hat{Q}^{-1} = U^1 \dots U^{m-1} \hat{U}^m \hat{L}^m F^s \dots F^1$. Note that $\hat{Q}^{-1} \hat{Q}(m) = U^1 \dots U^{m-1} \hat{U}^m \hat{L}^m \underline{\beta}$. But by construction $\hat{U}^m \hat{L}^m \underline{\beta} = \underline{e}^m$. Consider

Proposition 1.

Let $\underline{\beta}$ be any m -vector and E^i be any column eta. If $\beta_i = 0$, then $E^i \underline{\beta} = \underline{\beta}$.

By Proposition 1, $U^1 \dots U^{m-1} \underline{e}^m = \underline{e}^m$. Therefore, $\hat{Q}^{-1} \hat{Q}(m) = \underline{e}^m$. For $1 \leq \ell < m$, let $\underline{\gamma} = F^s \dots F^1 Q(\ell)$. By construction $\gamma_j = 0$ for $\ell < j \leq m$ and $\gamma_\ell = 1$. By Proposition 1, $U^{\ell+1} \dots U^{m-1} \hat{U}^m \hat{L}^m \underline{\gamma} = \underline{\gamma}$. By the construction of $U^1 \dots U^\ell$, we have $U^1 \dots U^\ell \underline{\gamma} = \underline{e}^\ell$. Therefore, if the leaving column is $Q(m)$, then step 3 of ALG 1 produces $\hat{Q}^{-1}$.

We now present a theoretical justification for step 4 of ALG 1.

Proposition 2.

Let

$$U^{p+1} = \left[\begin{array}{c|c|c} I & & \\ \hline & \eta & \\ \hline & & I \end{array} \right] \quad \text{and} \quad R^p = \left[\begin{array}{c|c} I & \\ \hline & \gamma \\ \hline & & I \end{array} \right] \leftarrow \text{row } \ell^*$$

$\uparrow$
 column ℓ

where $\ell \neq \ell^*$.

If

$$U^p = \left[\begin{array}{c|c|c} I & & \\ \hline & \alpha & \\ \hline & & I \end{array} \right] \quad \text{and} \quad R^{p+1} = \left[\begin{array}{c|c} I & \\ \hline & \beta \\ \hline & & I \end{array} \right] \leftarrow \text{row } \ell^*$$

$\uparrow$
 column ℓ

where

$$\alpha_i = \begin{cases} 0, & \text{if } i = \ell^* \\ \eta_i, & \text{otherwise, and} \end{cases}$$

$$\beta_i = \begin{cases} \eta_i \gamma, & \text{if } i = \ell \\ \gamma_i, & \text{otherwise,} \end{cases}$$

$$\text{then } R^P U^{P+1} = \tilde{U}^P R^{P+1}.$$

Proposition 2 is a theoretical justification for step 4 of ALG 1.

The proposition to follow shows the precise structure of $R^m F^s \dots F^1 Q$.

Consider

Proposition 3.

Let $U^* = F^s \dots F^1 Q$. If $\tilde{U}^* = R^m U^*$, then

$$\tilde{U}^*[i] = \begin{cases} U^*[i], & i \neq k \\ \underline{e}^k, & \text{otherwise} \end{cases}.$$

We now present the results to prove that $\hat{Q}^{-1} = U^1 \dots U^{k-1} \tilde{U}^k \dots \tilde{U}^m \tilde{L}^m R^m F^s \dots F^1$.

Proposition 4.

$$U^1 \dots U^{k-1} \tilde{U}^k \dots \tilde{U}^m \tilde{L}^m R^m F^s \dots F^1 \hat{Q}(k) = \underline{e}^k.$$

Proposition 5.

$$U^1 \dots U^{k-1} \tilde{U}^k \dots \tilde{U}^m \tilde{L}^m R^m F^s \dots F^1 \hat{Q}(i) = \underline{e}^i \text{ for } i \neq k.$$

By Propositions 4 and 5, we have

Corollary 6.

$$\hat{Q}^{-1} = U^1 \dots U^{k-1} \tilde{U}^k \dots \tilde{U}^m \tilde{L}^m R^m F^s \dots F^1.$$

Hence, ALG 1 produces the updated working basis inverse.

5.2 Key Column Leaves The Basis

In this section, we present an algorithm for updating the working basis inverse to accomplish a switch between a key column and a nonkey column. That is, $\hat{Q} = R Q^{-1}$ where R is given by (13) and

$$Q^{-1} = U^1 \dots U^m F^s \dots F^1. \quad (20)$$

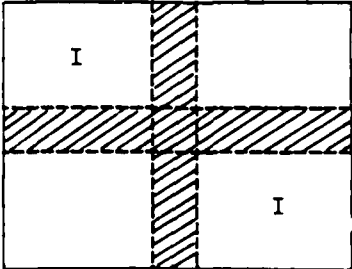
We wish to obtain $\hat{Q}^{-1}$ in the same form as (20).

To accomplish this update, we begin with $\hat{Q}^{-1} = R U^1 \dots U^m F^s \dots F^1$. We apply Proposition 2 to $R U^1$ creating the factorization $\hat{Q}^{-1} = \tilde{U}^1 R^2 U^2 \dots U^m F^s \dots F^1$. We continue with the application of Proposition 2 until we obtain $\hat{Q}^{-1} = \tilde{U}^1 \dots \tilde{U}^{k-1} R^k U^k \dots U^m F^s \dots F^1$. Proposition 2 does not apply to $R^k U^k$. However, a simple update would be to let $\tilde{U}^k = \dots = \tilde{U}^m = I$ and use the below factorization:

$$\hat{Q}^{-1} = \underbrace{\tilde{U}^1 \dots \tilde{U}^m}_{\text{LEFT FILE}} \underbrace{R^k U^k \dots U^m F^s \dots F^1}_{\text{RIGHT FILE}}.$$

This update simply involves application of Proposition 2 until it does not apply ($\ell = \ell^*$) and then shifting the remainder of the left file to the right file. We call this update the *TYPE 1 UPDATE*.

We will now give an update in which $R^k U^k \dots U^m$ is modified as opposed to moving them to the right file. Let

$$E^k = R^k U^k =$$


+ row k

Then we define matrices $\bar{U}^{k+1}$ and E^{k+1} such that $E^k \bar{U}^{k+1} = \bar{U}^{k+1} E^{k+1}$.

Following this procedure, $R^k \bar{U}^k \dots \bar{U}^m$ can be replaced by $\bar{U}^{k+1} \dots \bar{U}^m E^{m+1}$

so that

$$\hat{Q}^{-1} = \bar{U}^1 \dots \bar{U}^{k-1} \bar{U}^{k+1} \dots \bar{U}^m E^{m+1} F^S \dots F^1.$$

Further, we define a row eta $\tilde{R}$ and a column eta $\tilde{F}$ such that $E^{m+1} = \tilde{R} \tilde{F}$.

Therefore,

$$\hat{Q}^{-1} = \underbrace{\bar{U}^1 \dots \bar{U}^{k-1} \bar{U}^{k+1} \dots \bar{U}^m}_{\text{LEFT FILE}} \underbrace{\tilde{R} \tilde{F} F^S \dots F^1}_{\text{RIGHT FILE}}.$$

We call this update the *TYPE 2 UPDATE*.

We now present a set of propositions which justify the *TYPE 2 UPDATE*.

Proposition 7.

Let

$$U^{p+1} = \begin{array}{|c|c|c|} \hline I & \begin{array}{c} \eta_1 \\ \vdots \\ \eta_{\ell-1} \end{array} & 0 \\ \hline 0 & \begin{array}{c} \eta_\ell \\ \vdots \\ \eta_n \end{array} & I \\ \hline \end{array} \quad \text{and } E^p = \begin{array}{|c|c|c|} \hline I & \begin{array}{c} \mu_1 \\ \vdots \\ \mu_{\ell-1}^* \end{array} & 0 \\ \hline \gamma_1 \dots \gamma_{\ell-1}^* & \gamma_\ell^* & \gamma_{\ell+1}^* \dots \gamma_n \\ \hline 0 & \begin{array}{c} \mu_{\ell+1}^* \\ \vdots \\ \mu_n \end{array} & I \\ \hline \end{array},$$

where $\ell \neq \ell^*$ and $\mu_\ell = 0$.

If

$$\bar{U}^{p+1} = \begin{array}{|c|c|c|} \hline I & \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_{\ell-1} \end{array} & O \\ \hline O & \begin{array}{c} \alpha_\ell \\ \alpha_{\ell+1} \\ \vdots \\ \alpha_n \end{array} & I \\ \hline \end{array} \quad \text{and} \quad E^{p+1} = \begin{array}{|c|c|c|} \hline I & \begin{array}{c} \mu_1 \\ \vdots \\ \mu_{\ell-1}^* \end{array} & O \\ \hline \lambda_1 \dots \lambda_{\ell-1}^* & \lambda_\ell^* & \lambda_{\ell+1}^* \dots \lambda_n \\ \hline O & \begin{array}{c} \mu_{\ell+1}^* \\ \vdots \\ \mu_n \end{array} & I \\ \hline \end{array},$$

where

$$\lambda_i = \begin{cases} \underline{\gamma} \, \underline{\eta}, & \text{if } i = \ell, \\ \gamma_i, & \text{otherwise,} \end{cases}$$

$$\alpha_i = \begin{cases} 0, & \text{if } i = \ell^*, \\ \eta_i + \mu_i \, \eta_{\ell^*}, & \text{otherwise,} \end{cases}$$

$$\text{then } E^p U^{p+1} = \bar{U}^{p+1} E^{p+1}.$$

The following proposition is used to replace the cross matrix E^{m+1} with a row eta $\tilde{R}$ and a column eta $\tilde{F}$.
Proposition 8.

Let

$$E = \begin{array}{|c|c|c|} \hline I & \begin{array}{c} \mu_1 \\ \vdots \\ \mu_{\ell-1} \end{array} & O \\ \hline \gamma_1 \cdots \gamma_{\ell-1} & \gamma_\ell & \gamma_{\ell+1} \cdots \gamma_n \\ \hline O & \begin{array}{c} \mu_{\ell+1} \\ \vdots \\ \mu_n \end{array} & I \\ \hline \end{array}$$

If

$$\tilde{R} = \begin{array}{|c|c|c|} \hline I & & O \\ \hline \gamma_1 \cdots \gamma_{\ell-1} & X & \gamma_{\ell+1} \cdots \gamma_n \\ \hline & & \\ \hline O & & I \\ \hline \end{array}$$

and $\tilde{F} =$

$$\begin{array}{|c|c|c|} \hline I & \begin{array}{c} \mu_1 \\ \vdots \\ \mu_{\ell-1} \end{array} & O \\ \hline & Y & \\ \hline O & \begin{array}{c} \mu_{\ell+1} \\ \vdots \\ \mu_n \end{array} & I \\ \hline \end{array},$$

where X and Y are such that $XY = \gamma_\ell - \sum_{\substack{i=1 \\ i \neq \ell}}^n \gamma_i \mu_i$,

then $E = \tilde{R} \tilde{F}$.

We now present the update algorithm for the case in which the ℓ^{th} column of T is being switched with the k^{th} column of C . Let $\underline{\gamma} = \underline{e}^\ell T^{-1} C$.

ALG 2: LU UPDATE FOR A KEY LEAVING COLUMN

1. Set $R^1 \leftarrow I$.
Set $R^1[k] \leftarrow \gamma$.
Set $i \leftarrow 1$.
2. If $i = k$, go to 4.
Set $\alpha \leftarrow R^i[k] U^i(i)$.
Set $R^{i+1} \leftarrow R^i$.
Set $R_{ki}^{i+1} \leftarrow \alpha$.
Set $\tilde{U}^i \leftarrow U^i$.
Set $\tilde{U}_{ki}^i \leftarrow 0$.
3. Set $i \leftarrow i + 1$ and go to 2.
4. Set $\tilde{U}^k \leftarrow I$.
Set $E^k \leftarrow R^k U^k$.
5. Apply Proposition 7 to $E^i U^{i+1}$ to form $\tilde{U}^{i+1} E^{i+1}$.
Set $i \leftarrow i + 1$.
6. If $i < m$, go to 5.
7. Apply Proposition 8 to E^m to obtain $\tilde{R} \tilde{F}$ where $X = 1$.

At the completion of step 7 we have $\hat{Q}^{-1} = \tilde{U}^1 \dots \tilde{U}^m \tilde{R} \tilde{F} F^s \dots F^1$.

VI. COMPUTATIONAL EXPERIMENTATION

Three test problems were selected for the experiment.

SC205 is a staircase linear program which was generated by Ho and Loute [12] and transformed into a network with side constraints, Gifford-Pinchot is a model of the Gifford-Pinchot National Forest [10] which has also been transformed into a network with side constraints. RAN is a randomly generated problem.

These problems were first solved and the pivot agenda was saved. That is, entering and leaving columns for each pivot were saved on a file. This file was then used by each code so that all three basis updates follow the same path to the optimum. The number of nonzeros required to represent Q^{-1} at various points in the solution process is illustrated in Figures 1 and 2. For both problems, the LU Type 2 update dominated both the LU Type 1 update and the product-form code in terms of nonzeros in the inverse. The average core storage required for Q^{-1} using the product-form update is approximately double that required for the best LU update.

Figures 1 and 2 About Here

Given the above results, we developed three specialized network with side constraints codes and computationally compared them with three general in-core LP systems and a special system for multicommodity network flow problems. All codes are written in FORTRAN and have not been tailored to either our equipment or our FORTRAN compiler. None of the codes were tuned for our problem set. A brief description of each code follows.

NETSIDE1, NETSIDE2 AND NETSIDE3 are our specialized network with side constraints systems. The first maintains Q^{-1} in product form, while the second and third maintain Q^{-1} in LU form using a Type 1 and Type 2 update, respectively. All use the Hellerman and Rarick [11] reinversion routine. The working basis is reinverted every 60 iterations. The pricing routine uses a candidate list of size 6 with block size of 200.

MINOS [15] stands for "a Modular In-Core Nonlinear Optimization System" and is designed to solve problems of the following form:

$$\begin{array}{ll} \text{minimize} & f(\underline{x}) + \underline{c}\underline{x} \\ \text{subject to:} & \underline{A}\underline{x} = \underline{b} \\ & \underline{l} \leq \underline{x} \leq \underline{u} \end{array}$$

where $f(\underline{x})$ is continuously differentiable in the feasible region.

For this study $f(x) = 0$ at all $\underline{x}$ and therefore none of the nonlinear subroutines were used for problem solution.

For linear programs, MINOS uses the revised simplex algorithm with all data and instructions residing in core storage. The basis inverse is maintained as an LU factorization using a Bartels-Golub update. The reinversion routine uses the Hellerman-Rarick [11] pivot agenda algorithm.

XMP is a library of FORTRAN subroutines which can be used to solve linear programs. The basis inverse is maintained in LU factored form. The pricing routine uses a candidate list of size 6 with two hundred columns being scanned each time the list is refreshed. The basis is reinverted every 50 iterations.

LISS stands for "Linear In-Core Simplex System" and is an in-core LP solver with the basis inverse maintained in product form. The reinversion routine is a modification of the work of Hellerman and Rarick [11]. The basis inverse is refactored every 50 iterations. A partial pricing scheme is used with 20 blocks.

MCNF stands for "Multicommodity Network Flow". MCNF uses the primal partitioning algorithm also. The basis inverse is maintained as a set of rooted spanning trees (one for each commodity) and a working basis inverse in product form. This working basis inverse has dimension equal to the number of binding GUB constraints. A partial pricing scheme is used. Our computational experience is given in Table 1.

The row entitled GUB Constraints, gives the number of LP rows which correspond to "GUB Constraints". The row, entitled "Binding GUB Constraints", gives the number of GUB constraints met as equalities at optimality using MCNF. All runs were made on the CDC 6600 at Southern Methodist University using the FTN compiler with the optimization feature enabled.

Based on these results, we conclude that for lightly constrained multicommodity network flow problems

- (i) XMP and MINOS run at approximately the same speed,
- (ii) NETSIDE1, NETSIDE2 and NETSIDE3 run at approximately the same speed, and
- (iii) the three NETSID codes are approximately twice as fast as XMP and MINOS.

REFERENCES

1. Ali, A., R. Helgason, and J. Kennington, "An Air Force Logistics Decision Support System Using Multicommodity Network Models", Technical Report 82-OR-1, Department of Operations Research, Southern Methodist University, Dallas, Texas 75275, (1982).
2. Barnett, D., J. Binkley, and B. McCarl, "The Effects of U. S. Port Capacity Constraints on National and World Grain Shipments", Technical Paper, Purdue Agricultural Experiment Station, Purdue University, West Lafayette, Indiana, (1982).
3. Bartels, R., and G. Golub, "The Simplex Method of Linear Programming Using LU Decomposition", Communications of ACM, 12, 266-268, (1969).
4. Bixby, R. E., "Recent Algorithms for Two Versions of Graph Realization and Remarks on Applications to Linear Programming", Technical Report,
5. Forrest, J. J. H., and J. A. Tomlin, "Updated Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method", Mathematical Programming, 2, 3, 263-278, (1972).
6. Glover, F., and D. Klingman, "The Simplex Son Algorithm for LP/Embedded Network Problems", Technical Report CCS 317, Center for Cybernetic Studies, The University of Texas, Austin, Texas, (1977).
7. Glover, F., R. Glover, J. Lorenzo, and C. McMillan, "The Passenger-Mix Problem in the Scheduled Airlines", Interfaces, 12, 3, 73-80, (1982).
8. Glover, F., R. Glover, and F. Martinson, "The U. S. Bureau of Land Management's New Netform Vegetation Allocation System", Technical Report, Division of Information Science Research, University of Colorado, Boulder, Colorado, (1982).

9. Graves, G. W., and R. D. McBride, "The Factorization Approach to Large-Scale Linear Programming", Mathematical Programming, 10, 1, 91-110, (1976).
10. Helgason, R., J. Kennington, and P. Wong, "An Application of Network Programming for National Forest Planning", Technical Report OR 81006, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1981).
11. Hellerman, E., and D. Rarick, "Reinversion With the Preassigned Pivot Procedure", Mathematical Programming, 1, 195-216, (1971).
12. Ho, J. K., and E. Loute, "A Set of Staircase Linear Programming Test Problems", Mathematical Programming, 20, 2, 245-250, (1981).
13. Kennington, J. L., and R. V. Helgason, Algorithms for Network Programming, John Wiley and Sons, New York, New York, (1980).
14. McLain, D. R., "A Multicommodity Approach to a Very Large Aeromedical Transportation Problem", (working paper) Operations Research Division, Military Airlift Command, Scott Air Force Base, Illinois, (1983).
15. Murtagh, B., and M. Saunders, "MINOS User's Guide", Technical Report 77-9, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, (1977).
16. Wagner, D. K., "An Almost Linear-Time Graph Realization Algorithm", unpublished dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, (1983).

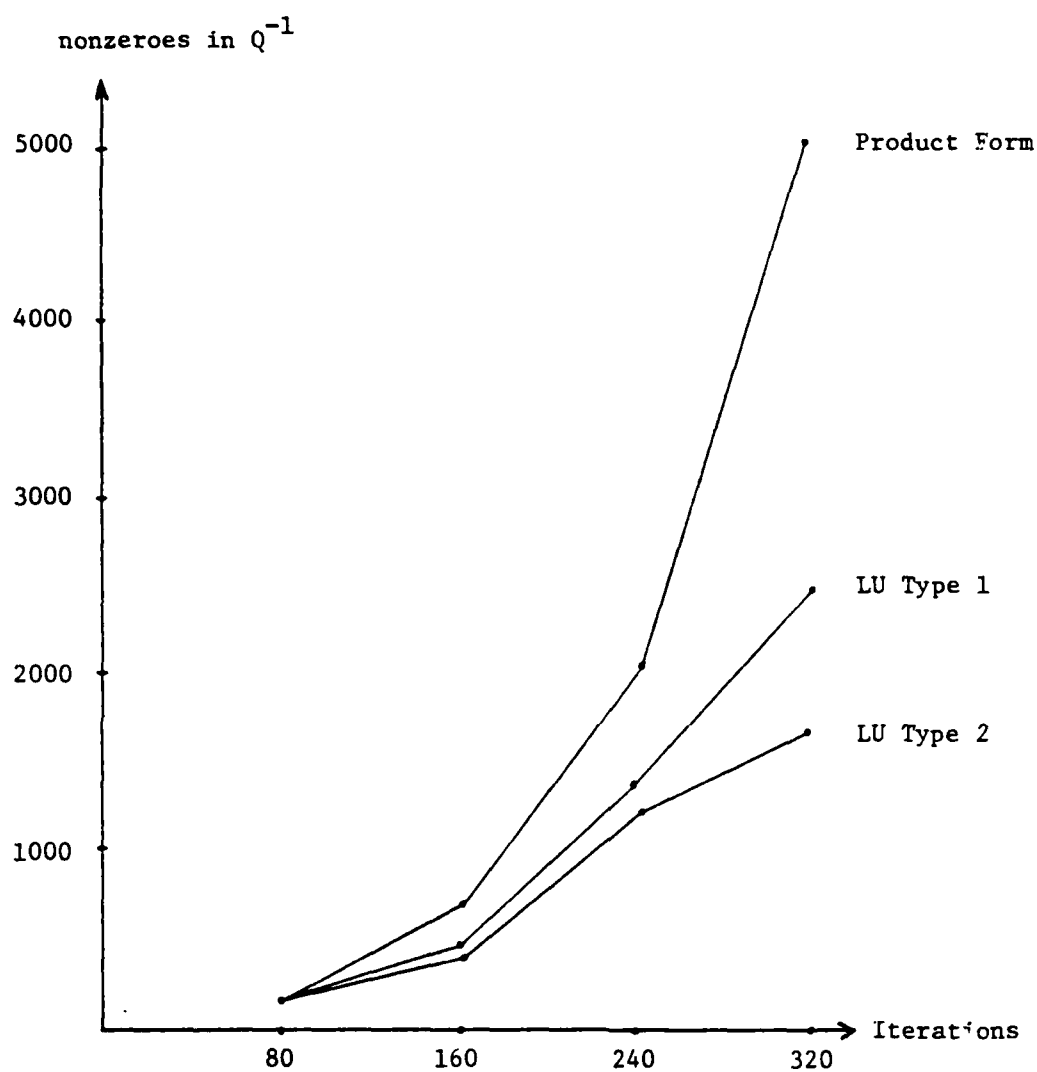


Figure 1. Nonzero Buildup In The Working Basis Inverse On SC205 [22].
(317 columns, 119 nodes, 87 side constraints)

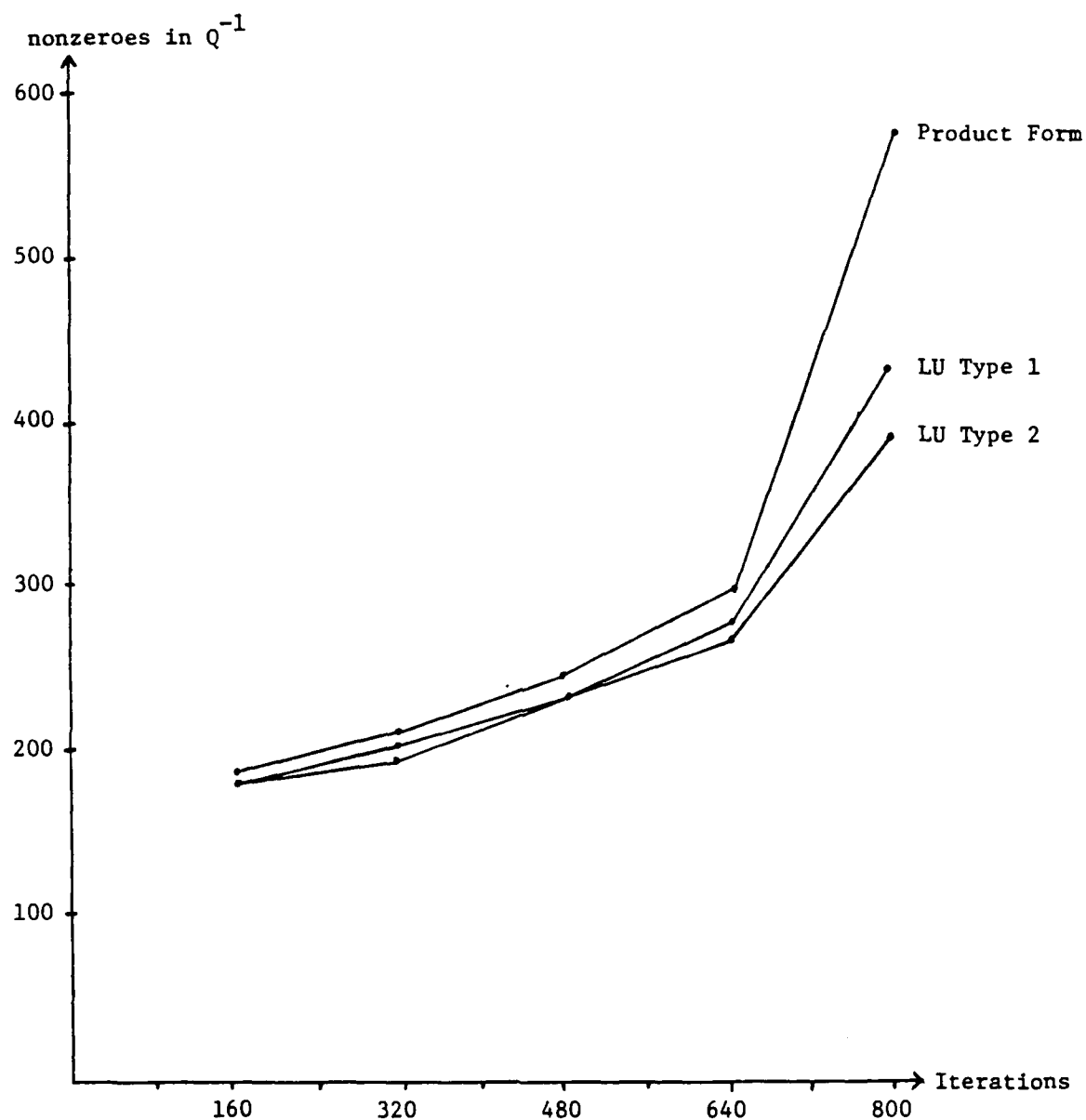


Figure 2. Nonzero Buildup In The Working Basis Inverse On Gifford Pinchot [20].
(1160 columns, 533 nodes, 84 side constraints)

Table 1 Comparison of Codes for Solving Multicommodity Network Flow Problems
(All Times Exclude Input and Output)

PROB DESC.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Total
Number	500	400	401	501	499	415	576	561	547	496	559	477	513	490	532	552	544	571	544	577	
LP Rows	995	720	841	896	851	732	850	972	901	900	934	842	843	840	840	848	900	841	872	841	
LP Coils	100	100	99.4	99.4	80	96	56	86	25	32	89	63	47	56	45	62	63	66	54	73	
X Network Rows	0	0	1	1	99	15	256	81	412	336	59	177	273	215	292	212	204	196	254	157	
CUB Conet.	0	0	0	0	2	3	5	3	6	6	7	9	8	11	13	17	21	23	26	31	
Binding CUB Conet.	1910	1440	1701	1794	2553	1746	2550	2916	2703	2700	2104	2526	2329	2510	2394	2414	2700	2523	2616	2523	
Number Nonzeros	5	20	20	5	10	20	4	12	3	4	5	6	4	5	3	4	4	5	3	6	
Number Commodities																					
MINOS [31]																					
Scaled Time	984	810	839	862	563	688	386	633	484	517	451	361	329	231	356	281	198	130	172	154	
Time (sec.)	99.40	43.07	44.29	82.61	56.70	41.86	61.76	83.66	29.09	31.70	79.56	44.63	42.55	44.19	45.28	55.86	62.92	63.35	50.72	67.39	1131
Pivots	1207	692	684	1068	739	655	750	978	376	427	986	618	564	592	594	712	781	793	692	831	
20P [28]																					
Scaled Time	632	582	559	620	426	528	356	498	452	507	435	376	269	272	381	329	207	189	222	275	
Time (sec.)	63.81	30.90	29.49	59.43	42.94	32.12	57.06	65.78	27.17	31.10	76.70	46.48	34.90	52.11	48.43	67.53	65.85	92.11	65.73	120.38	1110
Pivots	1198	751	720	1109	806	737	945	1135	499	619	1243	906	661	945	877	1099	1117	1375	1085	1687	
LISS [2]																					
Scaled Time	398	376	337	364	433	454	565	622	1125	1503	400	1223	675	627	709	588	360	380	334	493	
Time (sec.)	40.20	17.34	17.79	34.88	43.43	27.61	90.55	82.24	67.62	92.13	70.53	151.15	87.36	120.04	90.19	116.87	114.78	105.46	98.85	215.86	1765
Pivots	1267	766	769	1133	1319	1084	2087	2091	1799	2220	1767	2416	2048	2309	2137	2233	2400	2660	2120	2762	
MCNF [26]																					
Scaled Time	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	337
Time (sec.)	10.10	5.32	5.28	9.58	10.08	6.08	16.02	13.22	6.01	6.13	17.63	12.36	12.95	19.15	12.72	19.89	31.84	48.84	29.54	43.81	
Pivots	722	493	451	717	466	443	595	610	212	246	745	530	515	677	461	662	989	1272	885	1191	
NETSIDE1																					
Scaled Time	230	240	235	245	177	209	185	201	285	243	159	167	146	145	203	143	108	74	126	111	
Time (sec.)	23.18	12.79	12.41	23.43	17.87	12.68	29.64	26.51	17.11	14.87	28.06	20.63	18.96	27.85	25.87	28.50	34.25	36.51	37.33	48.65	720
Pivots	885	602	570	687	628	577	662	852	298	309	917	613	465	716	547	703	853	885	816	1180	
NETSIDE2																					
Scaled Time	229	239	230	242	182	209	191	200	296	253	161	167	152	148	209	148	111	76	134	110	
Time (sec.)	23.17	12.72	12.16	23.23	18.34	12.72	30.61	26.42	17.77	15.48	28.44	20.61	19.71	28.40	26.59	29.46	35.49	37.23	39.49	48.40	731
Pivots	885	602	570	687	628	577	662	852	298	309	917	613	465	716	547	703	853	885	823	1177	
NETSIDE3																					
Scaled Time	232	247	234	247	179	212	195	202	292	255	163	171	155	158	212	150	115	79	136	117	
Time (sec.)	23.41	13.16	12.38	23.69	18.00	12.86	31.19	26.73	17.36	15.65	28.78	21.17	20.03	30.25	27.01	29.74	36.74	38.37	40.22	51.58	816
Pivots	885	602	570	687	628	577	662	852	298	309	917	613	465	716	547	703	853	885	823	1180	

Appendix E

Technical Report 85-OR-3

THE PROJECTIVE TRANSFORMATION ALGORITHM BY KARMARKAR:
A COMPUTATIONAL EXPERIMENT WITH ASSIGNMENT PROBLEMS

By

J. Aronson¹

R. Barr¹

R. Helgason¹

J. Kennington¹

A. Loh²

H. Zaki¹

¹ Department of Operations Research
Southern Methodist University

² Department of Industrial Engineering
University of Houston

Revised August 1985

ABSTRACT

This paper describes a computational experiment comparing a pure network code, IBM's MPSX/370, and our implementation of a heuristic version of the projective transformation algorithm first suggested by N. Karmarkar. On five randomly generated dense assignment problems, we found that the pure network code was 18 times faster than MPSX which was 14 times faster than our projective transformation code.

KEY WORDS

PROJECTIVE ALGORITHM

LINEAR PROGRAMMING

NETWORKS

ACKNOWLEDGEMENT

This research was supported in part by the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278.

I. INTRODUCTION

This paper describes a pure network implementation of the new projective transformation algorithm for linear programs developed by N. Karmarkar [2]. The projection of the gradient in the transformed space is accomplished by solving a least squares problem using the LSQR routine of Paige and Saunders [4]. On dense assignment problems, we found that a pure network code NETFLO [3] is approximately 250 times faster than the new code and MPSX/370 is 14 times faster than the new code. Other computational experience with an early version of the algorithm may be found in Tomlin [5].

II. THE ALGORITHM

Let the linear program be given by

$$\min \quad cx \quad (1)$$

$$\text{s.t. } Ax = b \quad (2)$$

$$\ell \leq x \leq u. \quad (3)$$

The algorithm which we implemented may be stated as follows:

PROJECTIVE TRANSFORMATION ALG

0. Initialization

Let z^* denote the optimal objective value of (1)-(3) and let $\bar{x}$ be a starting point such that $A\bar{x} = b$ and $\ell < \bar{x} < u$. Select the step size β with $0 < \beta < 1$.

1. Form Transformation Matrix

$$d_j \leftarrow \min(u_j - \bar{x}_j, \bar{x}_j - \ell_j), \quad D = \text{diag}(d_1, \dots, d_n).$$

2. Transform Constraints

$$B = AD$$

3. Project Gradient

$$\hat{c} = Dc - B'(BB')^{-1}BDc$$

4. Transform To Original Space

$$h = D\hat{c}$$

5. Determine Max Step Size

$$\alpha_1 = \min_{h_j > 0} \left[\frac{\bar{x}_j - \ell_j}{h_j} \right]$$

$$\alpha_2 = \min_{h_j < 0} \left[\frac{\bar{x}_j - u_j}{h_j} \right]$$

$$\alpha = \min (\alpha_1, \alpha_2)$$

6. Move To New Point

$$\bar{x} = \bar{x} - \alpha \beta h$$

7. Check For Termination

If $.9c\bar{x} < z^*$, stop with $\bar{x}$ a near optimum;
otherwise, go to 1.

III. THE CODE

We developed a FORTRAN code, called PTANET, for the projective transformation algorithm which was specialized for pure network problems. That is, A is assumed to be a node-arc incidence matrix, less one row. Hence, A has full row rank. Step 3 was performed using the subroutine LSQR developed by Paige and Saunders [4]. That is, we solve the following least squares problem,

$$\min \| B^T x - Dc \|_2$$

to obtain $\hat{c}$. We used the 1978 version of LSQR since that version returns the residual. The only two calculations involving B , $p = Bv$ and $p = u^T B$, were performed by special routines which exploited the network structure of B .

LSQR uses an input parameter, EPS, for termination of the least squares solution. An EPS of 1.E-8 was found to be too large. That is, an $\bar{x}$ was generated in which at least one component of $|A\bar{x} - b|$ was greater than 1.E-6. Similar problems were encountered when we ran LSQR in single precision. Hence, all arrays are double precision and we used the following tolerances and input limits for LSQR:

```
EPS      = 1.0E-12
ATOL     = EPS*1000.
BTOL     = EPS*1000.
CONLIM   = 1./(10.*DSQRT(EPS))
ITNLIM   = 1000.
```


The core storage comparison between PTANET and the pure network code NETFLO [3] is as follows:

<u>Code</u>	<u>Arc Length Arrays</u>	<u>Node Length Arrays</u>
NETFLO	3	6
PTANET	10	5

The 1982 version of LSQR requires fewer arrays. The above implementation of PTANET does not have the minimum number of arrays that can be achieved.

IV. THE EXPERIMENT

Due to the fact that our algorithm requires a starting point $\bar{x}$ such that $A\bar{x} = b$ and $\ell < \bar{x} < u$, we restricted our test problems to dense assignment problems. That is, an assignment problem with $N = 10$ has 20 nodes and 100 arcs. The costs were randomly generated integers on the interval (1, 100). For a problem of size N , the starting solution was $\bar{x}_j = 1/N$ for all j .

We solved all problems using NETFLO first. The optimal objective value was then fed to PTANET to use for termination. We ran PTANET with $\beta = 0.9$. We also ran PTANET which rounded to the nearest feasible solution whenever N arcs had flow of at least 0.5.

The same problems were also run on MPSX/370 using the default parameter settings. Since two runs on the same problem may take a different number of iterations, we ran each problem three times and reported the average of these runs.

Our results are given in Table 1. All runs were made on the IBM 3081-D24 at Southern Methodist University. NETFLO and PTANET are written in FORTRAN and were run using FORTVS with $OPT = 3$. NETFLO solved all 5 problems in less than 1 second, MPSX took 18 seconds, while PTANET required 255 seconds. The final three iterations for each run with PTANET required approximately seventy percent of the total computational time. This is due to the ill-conditioning of $B = AD$. As the flows approach their bounds, the components of D become quite small. At optimality, every flow is either at its upper or lower bound.

V. THE NULL SPACE MATRIX

It appears that the main computational problem with the Projective Transformation ALG, as stated in Section II, is that if there is error in the calculation of d in Step 4, then $\bar{x}$ becomes infeasible. That is, $A\bar{x} \neq b$. Therefore, many iterations are required by LSQR to obtain a sufficiently accurate d so that feasibility is maintained. In order to overcome this numerical problem, we modified the algorithm to use the null space matrix to accomplish the projection.

Recall that the direction is obtained by the following steps:

2. $B = AD$.
3. $\hat{c} = (I - B'(BB')^{-1}B)Dc$.
4. $d = D\hat{c}$.

Suppose A is $m \times n$ and let Q' be the $(n \times n - m)$ null space matrix corresponding to A . The null space matrix corresponding to $B = AD$ is $D^{-1}Q'$. By the property of the null space matrix,

$$I - B'(BB')^{-1}B = D^{-1}Q'(QD^{-1}D^{-1}Q')^{-1}QD^{-1}.$$

Hence, 2, 3, and 4 can be replaced by the following:

2. Construct Q' .
3. $\hat{c} = D^{-1}Q(QD^{-1}D^{-1}Q')^{-1}QD^{-1}Dc$.
4. $d = Q'(QD^{-1}D^{-1}Q')^{-1}QD^{-1}Dc$.

By applying LSQR to

$$\min \| D^{-1}Q'x - Dc \|_2,$$

we obtain

$$x^* = (QD^{-1}D^{-1}Q')^{-1}QD^{-1}Dc.$$

Then d is simply $Q'x^*$.

For pure network problems, Q can be generated from A and any basis (rooted spanning tree). The columns of Q' may be constructed by tracing cycles in the basis tree after a nonbasic arc is appended to the tree. We modified PTANET to use Q and developed special routines to calculate $D^{-1}Q'x$ and $y'QD^{-1}$ required by the 1982 version of LSQR.

Computationally, we found that an inaccurate x^* from LSQR still produced a feasible direction d . However, these directions would not necessarily guarantee that optimality to the original problem could be obtained. Our experience with a 20×20 assignment problem is presented in Table 2. With a tolerance of $1.E-10$ and smaller, convergence was achieved. However, with a tolerance of $1.E-8$, the objective function stalled at an objective value 13% above the true optimum. That is, we were trading a worse direction in exchange for a guaranteed feasible direction. Since the dimension of Q' was very large and $D^{-1}Q'$ becomes ill-conditioned after a few major iterations, this trade-off does not pay off. NETFLO solved this problem in 0.02 seconds.

VI. CONCLUSIONS

Based on our computational tests, we were unable to confirm the original claims which were reported concerning this algorithm. The biggest difficulty appears to be that both AD or $D^{-1}Q'$ become very ill-conditioned as components of the solution vector approach either their upper or lower bounds. Two approaches have been suggested to help alleviate this problem. When a variable gets close to either its upper or lower bound, fix it to the appropriate bound and drop it from the problem. Scaling should also assist in this difficulty. We also found that skipping steps 2, 3, and 4 and using the same direction two successive major iterations can reduce the computational time by up to 25%. However, to make our present code competitive with MPSX/370 these ideas would have to perform spectacularly.

The problem of a feasible interior starting point can be solved by a two-phase approach. The problem of a satisfactory stopping rule can be solved by iterating between the primal and the dual. When the two bounds are within a given tolerance, then the algorithm terminates. These procedures could increase the computational times by a factor of four.

REFERENCES

1. Ali, A. I. and J. L. Kennington, "Network Structure in Linear Programs: A Computational Study", Technical Report 83-OR-1, Operations Research Department, Southern Methodist University, Dallas, Texas (1983).
2. Karmarkar, N., "A New Polynomial-Time Algorithm for Linear Programming", AT&T Bell Laboratories, Murray Hill, New Jersey 07974 (undated).
3. Kennington, J. L. and R. V. Helgason, Algorithms For Network Programming, John Wiley and Sons, New York, New York (1980).
4. Paige, C. C. and M. A. Saunders, "Algorithm 583 LSQR: Sparse Linear Equations and Least Squares Problems", ACM Transactions on Mathematical Software, Vol. 8, No. 2., (1982), 195-209.
5. Tomlin, J. A., "An Experimental Approach to Karmarkar's Linear Programming Algorithm", Ketron, Inc., Mountain View, California 94040 (1984).

Table 1. Computational Experience With Dense Assignment Problems

(Arcs = $N \times N$, Nodes = $2N$)

CODE	ROW DESC.	PROBLEM SIZE N					
		40	50	60	70	80	TOTAL
NETFLO	Iterations	296	410	603	613	938	
	Time (secs.)	.07	.11	.18	.23	.38	.97
MPSX/370	Iterations	213	320	467	483	659	
	Time (secs.)	1.2	1.8	3.2	4.4	7.4	18
PTANET* ($\beta = .9$)	Major Iter	9	10	10	10	11	
	Iter in LSQR	734	1218	1162	1258	1751	
	% Optimality	94	95	93	92	94	
	Time (secs.)	12	31	42	61	109	255
PTANET* With Rounding Heuristic ($\beta = .9$)	Major Iter	9	10	10	10	11	
	Iter in LSQR	734	1218	1162	1258	1751	
	% Optimality	94	95	93	92	94	
	Time (secs.)	12	31	42	61	109	255

* The data in these rows is identical; the heuristic was unable to produce an early termination.

Table 2. Solution of a 20 x 20 Assignment Problem Using the Null Space Matrix

EPS	MAJOR ITER	ITER IN LSQR	% OPTIMALITY	TIME (SECS.)
1.E-13	7	6931	93	118
1.E-10	9	3138	93	55
1.E-8 ¹	-	-	-	-

¹Did not converge in 32 major iterations and the objective value stalled at 166. Optimal objective value is 147.

Appendix F

Technical Report 85-OR-5

THE FREQUENCY ASSIGNMENT PROBLEM:
A SOLUTION VIA NONLINEAR PROGRAMMING

By

J. David Allen

Switching Systems Division
Rockwell International
P.O. Box 10462
Dallas, TX 75207
(214)-996-5701

Richard V. Helgason

and

Jeffery L. Kennington

Operations Research Department
Southern Methodist University
Dallas, TX 75275
(214)-692-3072

Revised November 1985

Comments and criticisms from interested readers are cordially invited.

ABSTRACT

This paper gives a mathematical programming model for the problem of assigning frequencies to nodes in a communications network. The objective is to select a frequency assignment which minimizes both co-channel and adjacent channel interference. In addition, a design engineer has the option to designate key links in which the avoidance of jamming due to self-interference is given a higher priority. The model has a nonconvex quadratic objective function, generalized upper bounding constraints, and binary decision variables. We developed a special heuristic algorithm and software for this model and tested it on five test problems which were modifications of a real-world problem. Even though most of the test problems had over 600 binary variables, we were able to obtain a near optimum in less than 12 seconds of CPU time on a CDC Cyber-875.

ACKNOWLEDGEMENT

This research was supported in part by the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278.

KEY WORDS

Nonlinear Programming

Integer Programming

Communication Networks

NOTE TO EDITOR

PLEASE SET ALL LETTERS UNDERLINED IN BOLDFACE. DO NOT SET UNDERLINES.

I. INTRODUCTION

One of the most critical design problems in a radio communication network is the assignment of transmit frequencies to stations (nodes) so that designated key communication links will not be jammed due to self-interference. In this investigation, we describe a novel new optimization model and a solution technique which can be used to assist design engineers in this process.

1.1. Problem Description

A radio communications network consists of radio stations, each equipped with one or more transmitters and receivers. When a given station has the ability to receive information intelligibly from a transmitting station, a link is said to exist from the transmitting station to the receiving station. The interconnection of these stations and links may be viewed graphically as a set of nodes, representing the radio stations, joined together by directed arcs, representing the links.

We assume in our model that one transmitter and several receivers are located at each radio station (node). The transmitter is tuned to a specified center frequency, and the receivers are tuned to the transmit frequencies of the neighboring stations to which the station is to be linked. A channel is associated with each center frequency in a way similar to the way channels and frequencies are associated in a television set. When a TV is tuned to channel 4, for example, it is really being tuned to receive video signals being broadcast at 67.25 MHz.

For our model, a given center frequency will be associated with each channel number. Using this definition, the frequency assignment problem may be defined as follows:

"Given N transmitting stations (nodes), assign 1 of F transmit channels to each node in such a way as to minimize the number of designated key links jammed due to co-channel and adjacent channel interference."

We say that a link is jammed if either of the following conditions occurs:

- (i) a node receives two signals on the same channel that are less than α dB apart in signal strength, or
- (ii) a node receives a signal on a given channel while a neighboring node transmits on an adjacent channel. If the neighbor's signal strength exceeds the signal strength of the current node by more than β dB, then the incoming signal will be garbled.

The constants α and β are functions of the hardware used in the network. Some of the determining factors are the receiver selectivity, the type of signal modulation, and the purity of the signal.

We now introduce the notation used to describe the mathematical model. Let $f \in \{1, \dots, F\}$ denote a channel and $n \in \{1, \dots, N\}$ denote a node. $\underline{e}_i$ will denote a vector whose entries are 0 except for the i^{th} which is 1. Let

$x_{fn} = 1$ if channel f is assigned to node n
and 0 otherwise,

$\underline{x}_f$ = the row vector $[x_{f1}, \dots, x_{fN}]$, and

$g(\underline{x}_1, \dots, \underline{x}_F)$ = a weighted number of jammed links with
assignment $(\underline{x}_1, \dots, \underline{x}_F)$.

Using the above notation, the mathematical model of the frequency
assignment problem is

$$\min \quad g(\underline{x}_1, \dots, \underline{x}_F) \quad (1)$$

$$\text{s.t.} \quad \sum_f x_{fn} = 1, \text{ all } n \quad (2)$$

$$x_{fn} \in \{0,1\}, \text{ all } f,n. \quad (3)$$

For this application, $g(\cdot)$ is a nonconvex quadratic function and
therefore (1) - (3) is a member of the class of binary nonconvex cost
nonlinear programs.

1.2 Related Literature

A heuristic procedure for solving a similar problem using a graph
coloring algorithm has been evaluated by Zoellner and Beall [7]. Closely
related models have been investigated by Morito, Salkin, and Williams
[5] and by Mathur, Salkin, Nishimura, and Morito [4]. Their models
are general linear integer programs with a single constraint. Using
a special branch-and-bound algorithm, they successfully solved their
model with up to fifty channels.

1.3 Accomplishments of the Investigation

We developed a novel new mathematical model of the frequency

assignment problem which takes the form of a binary nonconvex quadratic cost nonlinear program. The model incorporates weighting constants that allow a design engineer to tune the model to a particular application. We present an elegant specialization of the convex simplex algorithm to obtain a local optimum for this model. In addition, specialized software has been developed for this model and tested on five versions of a real-world problem. The software works quite well requiring less than a minute of computer time for all five test problems.

II. THE OBJECTIVE FUNCTION

In this section, we define the weighted interference function, $g(\underline{x}_1, \dots, \underline{x}_F)$. This function is generated from a set of signal strength matrices, $(A_1, \dots, A_F)$, two weighting matrices, and a set of critical values α , β , and $\delta_1, \dots, \delta_N$. Let a_{ij}^f denote the received signal strength in dBu/m of a signal which originates at node i and is received by node j , and let A_f denote the matrix whose elements are a_{ij}^f . Let the weighting matrices P and W be determined as follows:

$$p_{ij} = \begin{cases} \bar{p}_1, & \text{if } (i,j) \text{ is a designated key link} \\ \bar{p}_2, & \text{otherwise} \end{cases}$$

and

$$w_{ij} = \begin{cases} \bar{w}_1, & \text{if } (i,j) \text{ is a designated key link} \\ \bar{w}_2, & \text{otherwise.} \end{cases}$$

The constants $\bar{p}_1$, $\bar{p}_2$, $\bar{w}_1$, and $\bar{w}_2$ are tuning parameters which are used to provide weights in the interference function for the key links.

γ is used to denote the scalar function, defined by

$$\gamma(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Using $\gamma(\cdot)$, we define the three matrices

$$q_{ij}^f = \begin{cases} \sum_{\substack{k \neq i,j \\ a_{ik}^f > \delta_k}} \gamma(\alpha - |a_{jk}^f - a_{ik}^f|) w_{ik} + p_{ij}, & i \neq j \\ 0, & \text{otherwise;} \end{cases}$$

$$r_{ij}^f = \begin{cases} \sum_{\substack{k \neq i,j \\ a_{ik}^f > \delta_k}} \gamma(a_{jk}^{f+1} - a_{ik}^f - \beta) w_{ik}, & i \neq j \\ 0, & \text{otherwise;} \end{cases}$$

and

$$s_{ij}^f = \begin{cases} \sum_{\substack{k \neq i,j \\ a_{ik}^f > \delta_k}} \gamma(a_{jk}^{f-1} - a_{ik}^f - \beta) w_{ik}, & i \neq j \\ 0, & \text{otherwise.} \end{cases}$$

Using these matrices, the interference function is given by

$$g(\underline{x}_1, \dots, \underline{x}_F)$$

$$= \underbrace{\sum_{f=1}^{f=F} \underline{x}_f' Q_f \underline{x}_f}_{\text{co-channel interference}} + \underbrace{\sum_{f=1}^{f=F-1} \underline{x}_f' R_f \underline{x}_{f+1}}_{\text{adjacent channel interference from channel above}} + \underbrace{\sum_{f=2}^{f=F} \underline{x}_f' S_f \underline{x}_{f-1}}_{\text{adjacent channel interference from channel below}}.$$

In addition it is often desirable to use all of the channels. Therefore, we appended the function

$$\bar{w}_3 \sum_f \sum_{i=1}^{i=N-1} x_{fi} \sum_{j=i+1}^{j=N} x_{fj}$$

to $g(\cdot)$ so that in the absence of self-interference, the channels would be equally distributed among the nodes. The scalar $\bar{w}_3$ is also a tuning parameter.

Using the above formulae, we now give an example which presents the matrices required to define $g(\cdot)$. Let $\alpha = 2$, $\beta = 3$, $\bar{w}_3 = 0$, $\delta_n = 0$ for all n , and $p_{ij} = w_{ij} = 1$ for all i, j .

If

$$A_1 = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 1 & 0 & 3 & 3 \\ 2 & 3 & 0 & 2 \\ 4 & 3 & 2 & 0 \end{bmatrix}, \text{ and } A_2 = \begin{bmatrix} 0 & 2 & 3 & 5 \\ 2 & 0 & 5 & 5 \\ 3 & 5 & 0 & 1 \\ 5 & 5 & 1 & 0 \end{bmatrix}, \text{ then}$$

$$Q_1 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{ and } S_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

III. THE ALGORITHM

Let $\underline{x}' = [\underline{x}'_1, \dots, \underline{x}'_F]$. Then the frequency assignment problem takes the general form:

$$\min \quad g(\underline{x}) = \underline{x}' C \underline{x} \quad (4)$$

$$\text{s.t.} \quad \sum_f x_{fn} = 1, \text{ all } n \quad (5)$$

$$x_{fn} \in \{0,1\}, \text{ all } f,n \quad (6)$$

where the diagonal elements of C are 0 and all other elements are positive.

The continuous relaxation of (4) - (6) is obtained by replacing (6) with

$$0 \leq x_{fn} \leq 1, \text{ all } f,n. \quad (7)$$

The model (4), (5), (7) is a nonconvex quadratic program and a local optimum can be efficiently obtained by application of the convex simplex algorithm as described in Zangwill [6]. Suppose we begin with a feasible integer solution $\bar{\underline{x}}' = [\bar{\underline{x}}'_1, \dots, \bar{\underline{x}}'_F]$. We assume that all nonbasic variables have a value of zero. Let $\ell_1, \dots, \ell_N$ denote the subscript such that $\bar{x}_{\ell_1 1} = \dots = \bar{x}_{\ell_n n} = 1$. Then a nonbasic variable x_{fn} with a value of zero, prices favorably if $[\nabla g(\bar{\underline{x}})]' (\underline{e}_i - \underline{e}_j) < 0$ where $i = (f-1)N + n$ and $j = (f-1)N + \ell_n$. The line search for this problem requires that we solve the problem

$$\min_{0 \leq \Delta \leq 1} g(\bar{\underline{x}} + (\underline{e}_i - \underline{e}_j)\Delta). \quad (8)$$

$$\text{But } \left. \frac{dg(\bar{\underline{x}} + (\underline{e}_i - \underline{e}_j)\Delta)}{d\Delta} \right|_{\Delta=1}$$

$$\begin{aligned} &= (\underline{e}_i - \underline{e}_j)' \nabla g(\bar{\underline{x}} + \underline{e}_i - \underline{e}_j) \\ &= (\underline{e}_i - \underline{e}_j)' (C + C') (\bar{\underline{x}} + \underline{e}_i - \underline{e}_j) \end{aligned}$$

$$= [\nabla g(\bar{x})]' (\underline{e}_1 - \underline{e}_j) + (\underline{e}_1 - \underline{e}_j)' (C + C') (\underline{e}_1 - \underline{e}_j).$$

Since x_{fn} priced favorably, then $[\nabla g(\bar{x})]' (\underline{e}_1 - \underline{e}_j) < 0$.

$$\text{Also, } (\underline{e}_1 - \underline{e}_j)' (C + C') (\underline{e}_1 - \underline{e}_j)$$

$$= \underline{e}_1' (C + C') \underline{e}_1 + \underline{e}_j' (C + C') \underline{e}_j - \underline{e}_1' (C + C') \underline{e}_j - \underline{e}_j' (C + C') \underline{e}_1.$$

But, the diagonal elements of $(C + C')$ are 0 and all other elements are nonnegative. Hence, the solution to (8) is $\Delta^* = 1$ and the exact change to the objective function will be $\nabla g(\bar{x}')' (\underline{e}_1 - \underline{e}_j) - \underline{e}_1' (C + C') \underline{e}_j - \underline{e}_j' (C + C') \underline{e}_1$, a strict decrease. Therefore, in the new solution x_{fn} is set to 1 and x_{l_n} is set to 0. Since this holds for every iteration of the convex simplex algorithm, integrality is maintained and a local optimum for (4) - (6) can be obtained by finding a local optimum for (4), (5), (7).

Let $\bar{x}$ be any initial assignment for the frequency assignment problem. Using this initial assignment, the algorithm may be stated as follows:

For $f = 1, \dots, F$.

For $n = 1, \dots, N$.

$$l_n := k \text{ where } \bar{x}_{kn} = 1.$$

$$i := (f - 1) N + n.$$

$$j := (f - 1) N + l_n.$$

$$p := [\nabla g(\bar{x})]' (\underline{e}_1 - \underline{e}_j).$$

If $p < 0$

then

$$\bar{x}_{l_n} := 0$$

$$\bar{x}_{fn} := 1$$

Repeat as long as $p < 0$ for some f and some n .

IV. COMPUTATIONAL EXPERIENCE

We implemented the frequency assignment algorithm in a FORTRAN code. All data, including the matrices Q_f , R_f , and S_f , are stored in high speed core. Special subroutines were written to evaluate both $g(\cdot)$ and $Vg(\cdot)$ at a point. The code begins with F different starting solutions and stops when a local optimum is found. The initial assignment for run $r \in \{1, \dots, F\}$ is to assign frequency $\{(n + r - 2) \text{ modulo } F\} + 1$ to node n . The best solution obtained from all F runs is the output.

Five test problems were generated from the real-world 43 node network illustrated in Figure 1. The lines connecting nodes are the designated key links. The problems all have the same topology but differ in the selection of the critical values and the weighting constants. A random assignment was generated and the matrices were modified so that this assignment produced a cost of zero. Hence, the optimal objective value for each problem is zero.

Our computational experience is reported in Table 1. All runs were made on a CDC Cyber-875 using the FTN5 compiler with $OPT = 2$. The "Initial Obj Value" row is the average objective value for the F initial solutions. Note that all five problems were run in less than 1 minute of CPU time and the "Final Obj Values" were quite close to the optimum as compared to the initial assignments.

Figure 1 Table 1
About Here

V. CONCLUSIONS

Our optimization model and computer software provide a practical approach to assist communication network designers in obtaining near optimal solutions for the frequency assignment problem. The fact that the diagonal elements of C in the quadratic objective function $\underline{x}^T C \underline{x}$ are zero, allows a very efficient implementation of the convex simplex method which maintains integrality. Hence, if we begin with an integer assignment, the convex simplex algorithm follows a sequence of integer points until a local minimum is obtained. This procedure is so fast that very large problems can be easily handled by this approach.

REFERENCES

1. Collins, M., L. Cooper, R. Helgason, J. Kennington, and L. LeBlanc, "Solving the Pipe Network Analysis Problem Using Optimization Techniques", Management Science, 24 (7), 747-760, (1978).
2. Kennington, J. L., and R. V. Helgason, Algorithms for Network Programming, John Wiley and Sons, New York, New York, (1980).
3. Kennington, J. L., "A Convex Simplex Code For Solving Nonlinear Network Flow Problems", Technical Report 82-OR-6, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1982).
4. Mathur, K., H. Salkin, K. Nishimura, and S. Morito, "The Design of an Interactive Computer Software System for the Frequency-Assignment Problem", IEEE Transactions on Electromagnetic Compatibility, Vol. EMC-26, No. 4, 207-212, (1984).
5. Morito, S., H. Salkin, and D. Williams, "Two Backtrack Algorithms for the Radio Frequency Intermodulation Problem", Applied Mathematics and Optimization, 6, 221-240, (1980).
6. Zangwill, W. I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, (1969).
7. Zoellner, J. A., and C. L. Beall, "A Breakthrough in Spectrum Conserving Frequency Assignment Technology", IEEE Transactions on Electromagnetic Compatibility, Vol. EMC-19, No. 3, 313-319, (1977).

Table 1. Computational Results With 43 Node Model

Row Description	Problem				
	1	2	3	4	5
α dB	10	10	12	10	10
β dB	25	25	25	25	30
F (channels)	10	12	14	14	14
Binary Variables	430	516	602	602	602
Iterations	525	341	497	540	526
Solution Time (secs)	5	7	10	11	11
Initial Obj Value	3153	1561	1875	1671	1670
Final Obj Value	164	103	79	5	4
Jammed Key Links	8	4	3	0	0

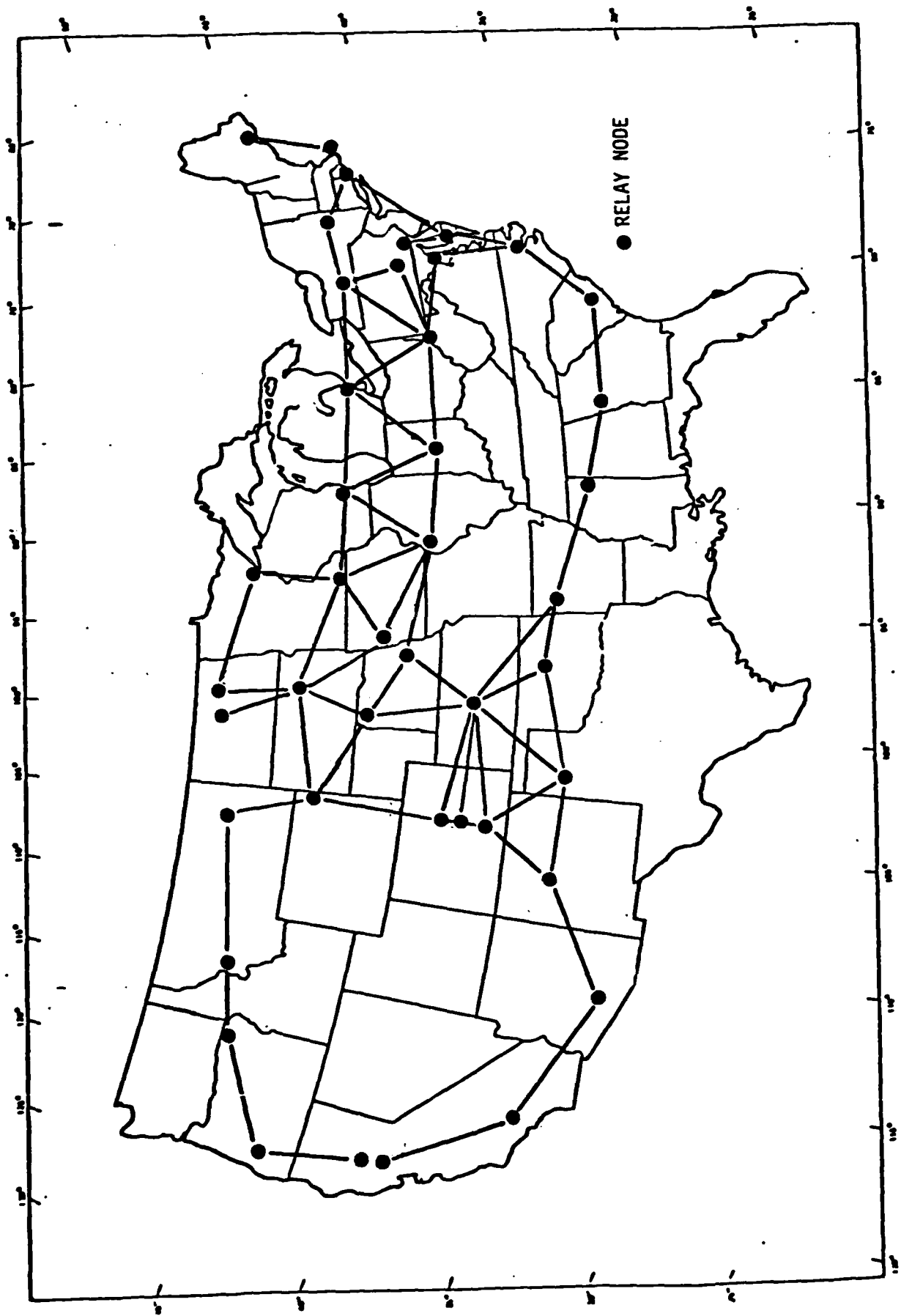


Figure 1. Forty-three Node Communication Network With Designated Key Links

ATE
LMED
=8