SPECIAL PROBLEMS IN LINEAR PROGRAMMING

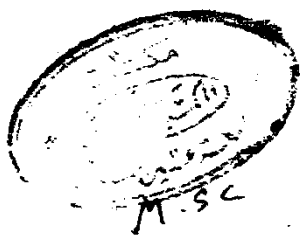
THESIS SUBMITTED IN PARTIAL

FULFILLMENT OF THE REQUIREMENTS FOR THE

AWARD OF THE M.SC. DEGREE

BY

RAFAT RIAD RIZKALLA



SUBMITTED AT

AIN SHAMS UNIVERSITY

FACULTY OF SCIENCE

DECEMBER 1982

2

M.SC. COURSES

STUDIED BY THE AUTHOR (1980 - 1981)

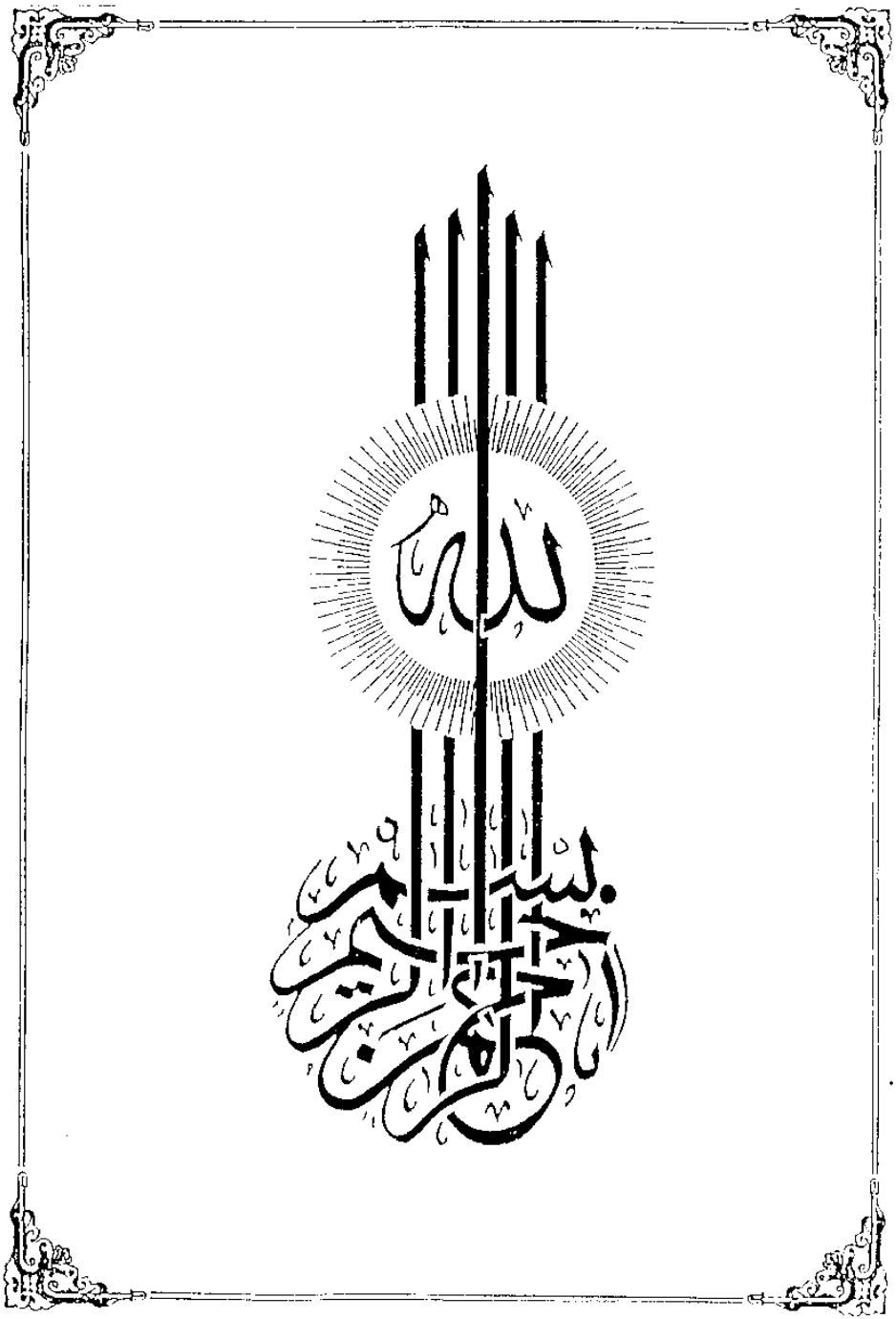
AT AIN SHAMS UNIVERSITY

FACULTY OF SCIENCE

1. Abstract Algebra.
2. Algebriac Topology.
3. Functional Analysis.
4. Numerical Analysis.
5. Spectral Theory.

Rg

20/12/1982



ACKNOWLEDGMENT

I wish to express my sincerest gratitude to Professor Emeritus Dr.Ragy H.Makar, Faculty of Science, Ain Shams University, for his constant encouragement and kind help.

I would like to acknoweldge my deepest gratitude and thankfulness to Dr. Afaf Fouad Nakhla, Expert of Center of Planning Techniques, Institute of National Planning, for suggesting the topic of the thesis, for her kind supervision and for her invaluable help during the preparation of the thesis.

December 1982

CONTENTS

INTRODUCTION

Linear programming has become one of the most important and effective tools in use today in modern management analysis . It is concerned with solving a very special type of problem-one in which all relations among the variables are linear both in the constraints and the function to be optimized. The general linear programming problem can be described as follows: Given a set of m linear inequalities or equations in r variables, we wish to find nonnegative values of these variables which will satisfy the constraints and maximize or minimize some linear function of the variables.

A certain class of linear programming problems, known as transportation type problems, arises very frequently in practical applications. The properties that it possesses enable one to solve the problem by methods that are considerably more efficient than the regular simplex method [13] .

The basic transportation problem was first formulated, along with a constructive solution , by Frank L.Hitchcok [14] . His paper, " the distribution of a product from several sources to numerous localities", sketched out the partial theory of a technique foreshadowing the simplex method ; it did not exploit special properties of the transportation problem except in finding starting solutions. The transportation problem is discussed in detail by Koopmans [16].

The importance of this problem lies in the fact that many problems having nothing to do with transportation fit the transportation model and can, therefore, be solved by one of its simple solution procedures.

In some transportation problems starting from the optimal primal and dual

solutions, we can obtain a new updated problem in which we transport a larger amount for less total cost, on the same network, with unaltered unit costs. The main purpose of this study is to develop a method for updating the transport ation problem.

The thesis is divided into five chapters. In chapter I, we present the basic concepts, theorems and algorithms concerning the graph theory and network flows. Chapter II deals with the standard transportation problem . A novel approach is used to derive the transportation algorithm from the simplex method. Degeneracies in the solution are also covered in this chapter. At the end of this chapter we explain a simple technique to solve the transprotation problem as a minimum cost flow problem using the algorithms given in chapter I. Chapter III deals with the variants of the transportation problem . For 54 unimodular linear programming problems it is shown that all the 54 problems can be solved as standard transportation problems. In most cases very little analysis is required to convert a problem to a standard transportation problem or to show that the problem is insoluble. But in other cases transformations had to be devised, based on the properties of optimal solutions. A generaliz ation of the standard transportation problem in which the origin and destination constraints consist not only of equality but also inequality constraints is considered. In chapter IV, we introduce a new method for updating the transportation problem. In the updated problem we can transport a larger amount for less total cost, on the same network, with unaltered unit costs. Lastly, in chapter V, we design a new computer program for the transportation algorithm described in chapter II. The program involves also the new algorithm described in chapter IV to update the transportation problem. Computer output of some examples is presented. The program is coded in Fortran language which is the more widly used language. The program was compiled and tested on the IBM computer 1130 at the computer centre, Ain shams University.

CHAPTER I

GRAPH THEORY AND NETWORK FLOWS

Introduction

This chapter deals with the graph theory and network flows. Some special properties of graphs are presented in section 1. Section 2 deals with the problem of finding shortest paths between pairs of vertices and also deals with the determination of negative cost circuits in graphs. In this section we shall describe two algorithms to solve this problem for the general case $c_{ij} \gtrless 0$.

The next two sections are concerned with maximum flow and minimum cost-maximum flow problems in graphs with arc capacities and costs . In section 3 we shall discuss the basic (s to t) maximum flow problem . The incremental graph and the decomposition of a flow pattern are also presented. In section 4 we shall describe two algorithms to find the minimum cost flow from s to t.

1.1. Theory of graphs

A graph G is a collection of points or vertices $x_1, x_2, \ldots, x_n$ (denoted by the set X), and a collection of lines $a_1, a_2, \ldots, a_m$ (denoted by the set A) joining all or some of these points. The graph G is denoted by the doublet (X,A).

If the lines in A have a direction they are called arcs and the resulting graph is called a directed graph. If the lines have no orientation they are called links and the graph is nondirected. When an arc is denoted by the pair of its initial and final vertices, its direction will be assumed to be from the first vertex to the second.

An alternative way to describe a directed graph G, is by specifying the

set X of vertices and a correspondence Γ which shows how the vertices are related to each other. Γ is called a mapping of the set X in X

$$\Gamma(x_i) = \{x_j \in X \mid (x_i, x_j) \text{ arc in } G\} \quad , \quad x_i \in X \tag{1-1}$$

and the graph is denoted by the doublet $G = (X, \Gamma)$.

In the case of a nondirected graph the correspondences Γ will be assumed to be those of an equivalent directed graph in which every link has been replaced by two arcs in opposite directions.

The relation Γ^{-1} is called the inverse correspondence,

$$\Gamma^{-1}(x_i) = \{ x_k \in X \mid (x_k, x_i) \text{ arc in } G \}, \; x_i \in X \tag{1-2}$$

Definition 1:

A path in a directed graph is any sequence of arcs where the final vertex of one is the initial vertex of the next one. A path which does not use the same vertex more than once is called an elementary path.

Definition 2:

A chain is the nondirected counterpart of the path and applies to graphs with the direction of their arcs disregarded.

Definition 3:

A circuit is a path $a_1, a_2, \ldots, a_q$ in which the initial vertex of a_1 coincides with the final vertex of a_q.

Definition 4:

A directed (nondirected) graph G = (X,A) is said to be connected if there exists a path (chain) between any two of its vertices. A connected graph without a circuit is called a tree.

A number c_{ij} may sometimes be associated with an arc (x_i, x_j). These numbers are called weights, lengths or costs and the graph is then called arc-weighted. The length (or cost) of the path p is $l(p) = \sum_{(x_i,x_j) \text{ in } p} c_{ij}$.

1.2. Shortest paths

For a given arc-weighted graph $G = (X, \Gamma)$ with arc costs given by the matrix $C = (c_{ij})$, the shortest path problem is the problem of finding the shortest path from a specific starting vertex $s \in X$ to a specific ending vertex $t \in X$, provided that such a path exists.

1.2.1. The shortest path between two specified vertices s and t

Algorithm for general cost matrix ($c_{ij} \neq 0$).

Let $l^k(x_i)$ be the label on vertex x_i at the end of the (k-1) st iteration.

Step 1. Set $S = \Gamma(s)$, $k=1$, $l^1(s) = 0$, $l^1(x_i) = c(s,x_i)$ for all $x_i \in \Gamma(s)$, and $l^1(x_i) = \infty$ for all other x_i.

Step 2. For every vertex $x_i \in \Gamma(S)$, $(x_i \neq s)$, update its label according to the expression:

$$l^{k+1}(x_i) = \min \left[l^k(x_i), \min_{x_j \in T_i} \left[l^k(x_j) + c(x_j, x_i) \right] \right] \qquad (1-3)$$

where $T_i = \Gamma^{-1}(x_i) \cap S$.

The set T_i contains those vertices for which the currently shortest paths from s are of cardinality k, (i.e. those vertices in S), and for which an arc to vertex x_i exists. Note that if $x_i \notin \Gamma(S)$, the shortest path from s to x_i cannot possibly be of cardinality k+1 and no change to the label of x_i is necessary.

For those vertices $x_i \notin \Gamma(S)$ set $l^{k+1}(x_i) = l^k(x_i)$.

$$\theta_{ij} = \begin{cases} \theta_{kj}, & \text{if } c_{ik} + c_{kj} < c_{ij} \text{ in eqn (1-5)} \\ \text{unchanged}, & \text{if } c_{ij} \leq c_{ik} + c_{kj} \end{cases}$$

At the end of the algorithm the shortest paths can be obtained immediately from the final (H) matrix . Thus, if the shortest path between any two vertices x_i and x_j is required this path is given by the vertex sequence:

$$x_i, x_v, \ldots, x_\alpha, x_\beta, x_\alpha, x_j$$

where

$x_\alpha = \theta_{ij}$, $x_\beta = \theta_{i\alpha}$, $x_\gamma = \theta_{i\beta}$ etc. until

finally $x_i = \theta_{iv}$.

If we start the algorithm so that $c_{ii} = \infty$, for all i, then the final value of c_{ii} would be the cost of the shortest circuit through the vertex x_i ,

1.3. The basic (s to t) maximal flow problem

Consider the graph G = (X, A) with arc capacities q_{ij}, a source vertex s and a terminal vertex t ; (s and t$\in$ X) . A set of numbers ξ_{ij} defined on the arcs $(x_i, x_j) \in A$ are called flows in the arcs if they satisfy the following conditions:

$$\sum_{x_j \in \Gamma(x_i)} \xi_{ij} - \sum_{x_k \in \Gamma^{-1}(x_i)} \xi_{ki} = \begin{cases} v & \text{if } x_i = s \\ -v & \text{if } x_i = t \\ 0 & \text{if } x_i \neq s \text{ or } t \end{cases} \qquad (1\text{-}6)$$

and $\xi_{ij} \leq q_{ij}$ for all $(x_i, x_j) \in A$ (1-7)

Equation (1-6) is an equation of conservation of flow and equation (1-7) states the capacity constraint for each arc of the graph G. The objective is to find a set of arc flows so that

$$v = \sum_{x^{s}\epsilon\Gamma(s)} \xi_{sj} = \sum_{x_k\epsilon\Gamma^{-1}(t)} \xi_{kt} \qquad (1\text{-}8)$$

is maximized where ξ_{sj} and ξ_{kt} are written for the flows from vertex s to x_j and from x_k to t respectively. Such a graph is called a network [4].

A cut -set $(X_o \rightarrow \bar{X}_o)$ separates s from t if $s\epsilon X_o$ and $t\epsilon \bar{X}_o$. The value of such a cut-set is the sum of the capacities of all arcs of G whose initial vertices are in X_o and the final vertices in $\bar{X}_o$; i.e.

$$v(X_o \rightarrow \bar{X}_o) = \sum_{(x_i,x_j)\epsilon(X_o \rightarrow \bar{X}_o)} q_{ij}$$

The minimum cut - set $(X_m \rightarrow \bar{X}_m)$ is then the cut-set with the smallest such value.

Definition 6:

An arc (x_i, x_j) of the chain is called a forward arc if $\xi_{ij} \geq 0$ and is called a backward arc if $\xi_{ji} > 0$.

Definition 7 :

The chain in which $\xi_{ij} < q_{ij}$ on all forward arcs and $\xi_{ji} > 0$ on all backward arcs is called a flow augmenting chain.

Theorem 1. (Maximum-flow minimum-cut theorem)[4]

The value of the maximum flow from s to t is equal to the value of the minimum cut - set $(X_m \rightarrow \bar{X}_m)$ Separating s from t .

1.3.1. *Labelling algorithm for the (s to t) maximum flow problem*

The labelling algorithm developed by Ford and Fulkerson [10] for the solution of this problem is now described.

The algorithm starts with an arbitrary feasible flow (zero flow may be used) and then tries to increase the flow value by systematically searching all possible flow-augmenting chains from s to t .

A. The labelling process

A vertex can only be in one of three possible states; labelled and scanned (i.e. it has a label and all adjacent vertices have been processed), labelled and unscanned (i.e. it has a label but not all its adjacent vertices have been processed) and unlabelled. A label on a vertex x_i is composed of two parts and takes one of the two forms $(+ x_j, \delta)$ or $(- x_j, \delta)$. The part $+ x_j$ implies that the flow along arc (x_j, x_i) can be increased. The part $-x_j$ implies that the flow along arc (x_i, x_j) can be decreased. δ represents in both cases the maximum amount of extra flow that can be sent from s to x_i along the augmenting chain being constructed.

Initially all vertices are unlabelled.

Step 1. Label s by $(+ s, \delta(s) = \infty)$. s is now labelled and unscanned and all other vertices are unlabelled.

Step 2 . Choose any labelled unscanned vertex x_i and suppose its label is $(\pm x_k, \delta(x_i))$.

(i) To all vertices $x_j \in \Gamma(x_i)$ that are unlabelled and for which $f_{ij} < q_{ij}$ attach the label $(+ x_i, \delta(x_j))$

where:

$$\delta(x_j) = \min(\delta(x_i), q_{ij} - f_{ij}) \qquad (I\text{-}9)$$

and

(ii) To all vertices $x_j \in \Gamma^{-1}(x_i)$ that are unlabelled and for which $f_{ji} > 0$ attach the label $(-x_i, \delta(x_j))$ where :

$$\delta(x_j) = \min(\delta(x_i), f_{ji}) . \qquad (I\text{-}10)$$

(The vertex x_i is now labelled and scanned and the vertices x_j labelled by (i) and (ii) above are labelled and unscanned.) Indicate that x_i is now scanned by marking it in some way.

Step 3. Repeat step 2 until either t is labelled in which case proceed to step 4 or t is unlabelled and no more labels can be placed in which case the algorithm terminates with ξ as the maximum flow vector .

B. Flow augmenting process

Step 4. Let x = t and go to step 5.

Step 5. (i) If the label on x is of the form (+z, $\delta(x)$), change the flow along the arc (z,x) from $\xi(z,x)$ to $\xi(z,x) + \delta(t)$.

(ii) If the label on x is of the form $(-z,\delta(x))$, change the flow along the arc (x,z) from $\xi(x,z)$ to $\xi(x,z) - \delta(t)$.

Step 6. If z = s erase all labels and return to step 1 to repeat the labelling process starting from the new improved flow calculated in step 5 above.

If $z \neq s$ set x=z and return to step 5.

1.3.2. The incremental graph

The process of finding a flow-augmenting chain in a graph G=(X,A) when the arc flows are given by the vector ξ, can be considered as an (s to t) path finding process in an incremental graph $G^{\xi}(\xi) = (X, A^{\xi})$ defined as follows :

$$X^{\xi} = X$$

and

$$A^{\xi} = A_1^{\xi} \cup A_2^{\xi}$$