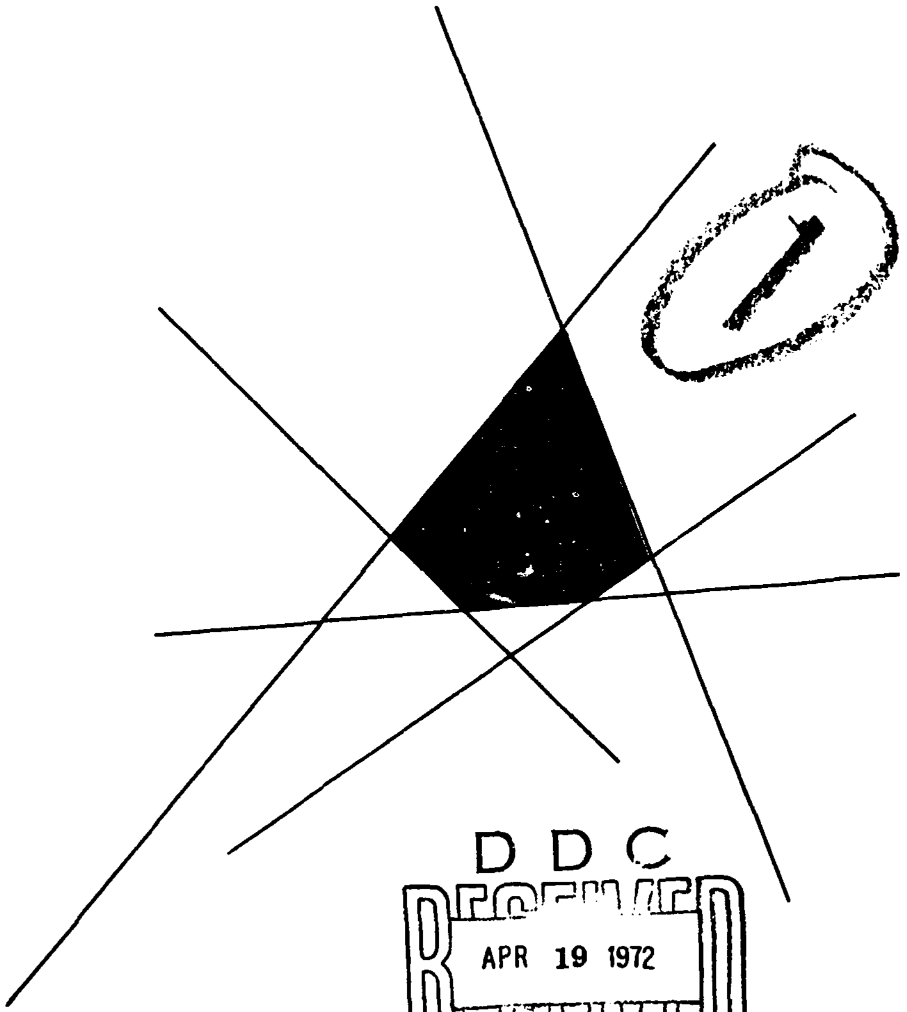
ORC 72-5
FEBRUARY 1972

A MULTI-COMMODITY CONCAVE COST MINIMIZATION PROBLEM FOR COMMUNICATION NETWORKS

by
Subhabrata Sen



OPERATIONS RESEARCH CENTER

COLLEGE OF ENGINEERING
UNIVERSITY OF CALIFORNIA · BERKELEY

132

A MULTI-COMMODITY CONCAVE COST MINIMIZATION PROBLEM FOR COMMUNICATION NETWORKS

by

Subhabrata Sen
Operations Research Center
University of California, Berkeley

FEBRUARY 1972 ORC 72-5

This research has been supported by the U.S. Army Research Office-Durham under Contract DA-31-124-ARO-D-331 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) University of California, Berkeley	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP	
3. REPORT TITLE A MULTI-COMMODITY CONCAVE COST MINIMIZATION PROBLEM FOR COMMUNICATION NETWORKS		
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Research Report		
5. AUTHOR(S) (First name, middle initial, last name) Subhabrata Sen		
6. REPORT DATE February 1972	7a. TOTAL NO. OF PAGES 125	7b. NO. OF REFS 48
8a. CONTRACT OR GRANT NO. DA-31-124-ARO-D-331 b. PROJECT NO. 20014501B14C c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ORC 72-5 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.		
11. SUPPLEMENTARY NOTES NONE	12. SPONSORING MILITARY ACTIVITY U.S. Army Research Office-Durham Box CM, Duke Station Durham, North Carolina 27706	
13. ABSTRACT SEE ABSTRACT.		

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6811

Unclassified
Security Classification

A-31408

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Concave Programming						
Minimization Problem						
Multi-Commodity Flows						
Communication Networks						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0101-807-6821

Unclassified
Security Classification

A-31409

TABLE OF CONTENTS

PAGE

ACKNOWLEDGEMENTS

I would like to express my gratitude to the members of my thesis committee: Stuart E. Dreyfus, Chairman, Richard M. Karp and Richard C. Grinold. Professor Dreyfus stimulated and guided the research by repeatedly posing provocative questions and his criticism of the manuscript was most helpful. Professor Karp and Professor Grinold made valuable suggestions and thoroughly examined the manuscript.

I am also indebted to Mr. Harry M. Ardley for suggesting the problem to me and giving me the benefit of his long experience with the communication industry; and the Pacific Telephone Company for providing me with financial support while this research was conducted.

I must thank Professor Rabindra N. Bhattacharya for his criticism and help in developing some of the material in Chapter 3.

I would like to thank Noreen Revelli, Linda Betters and Kathy Price for typing the manuscript.

ABSTRACT

In this network synthesis problem a matrix giving flow requirements between each pair of points is specified, and the cost of flow in each arc is a concave function of the amount of flow. A flow pattern which fulfills the requirements at minimum cost is sought. This problem is formulated as a concave programming problem with linear constraints. All the practical difficulties of formulation and theoretical difficulties of identifying the globally minimal solution while avoiding locally minimal solutions are discussed. Three procedures are developed. The first procedure is inefficient. The second procedure is quite efficient but produces locally minimal solutions. Therefore different heuristics and sequential sampling plans are suggested to obtain the globally minimal solution (or estimate it) from a number of locally minimal solutions. The third procedure is a global search procedure of branch and bound type where each subproblem is easy to solve. This procedure has limitations because computer core size requirement grows very rapidly with the size of the network. Different ways of circumventing this limitation are discussed. Finally, a post optimization procedure that approximates the solution of nonconcave problems with cost specified by step functions defined only at integral points is discussed. In conclusion, a more general problem and future research directions are mentioned.

CHAPTER 1

1.1 Introduction

This thesis studies the techniques of planning a communication network. It is a synthesis problem; the required number of channels connecting each source-sink pair is specified and the layout of the channels of communication must be determined. The problem is sometimes known as the "TELPAK" problem which refers to packing more channels together to take advantage of economies of scale.

This synthesis problem arises most often in two situations. The first is in planning an intercity network such that the cost incurred is minimum given an estimate of the traffic between cities. The traffic volume data are used in C.C.S.[1] calculations to determine the number of channels required between cities for a certain prespecified probability of blocking and certain assumptions about the alternate routing procedure. The second is in determining the most economical way of leasing, from a telephone company, trunk groups between the offices of a corporation given an estimate of the traffic volume between the offices and taking advantage of high volume discounts.

In both the above cases, the planning problem would be fairly simple, as we explain later in this chapter, if the cost per channel were constant or increasing with the number of channels. But in this case, we face a situation where improved technology can be used when the number of channels increases. Cost per channel decreases with increasing capacity. This trend is evident in the past developments (open wire, N2 carrier, microwave radio, analog transmission over coaxial cable) and is expected to continue in the foreseeable future (waveguides).

[1]C.C.S. means hundred call second which is a unit used in measuring traffic in a telephone network.

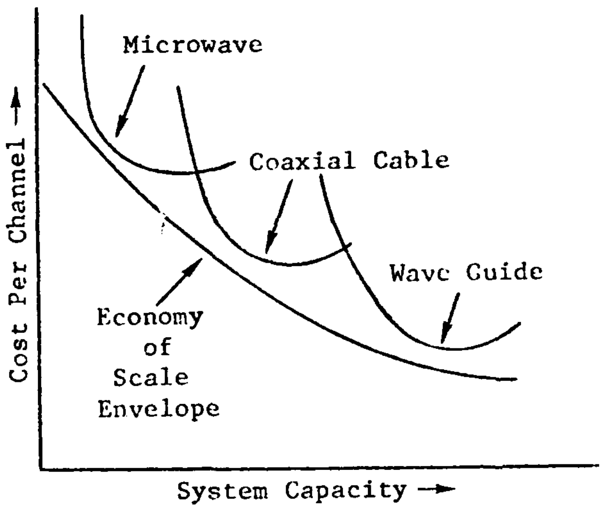

Microwave
Coaxial Cable
Wave Guide
Economy of Scale Envelope
Cost Per Channel →
System Capacity →

FIGURE 1.1

When the cost per channel is a decreasing function, we have a "concave type" (explained in the note on Page 8) function as the total cost objective function which has to be minimized. A small example of such a problem should be useful. Assume that a single telephone line costs one dollar per mile, but if we build one hundred lines between two points, the cost is 75 dollars per mile. If we build two hundred lines between two cities, it costs, say, 120 dollars per mile. Because the cost is concave type, it is cheaper to pack the lines into major routes than to build all telephone lines directly. To illustrate the problem, suppose that we have six locations A , B , C , D , E , and F as shown in Figure 2. The numbers in Figure 2 are the mileages between the points. Let there be 50 telephone lines required between points A and C , and also between points B and D . No lines are required between any other pair of points. The total cost of building a direct line between A and C and also between B and D is

$$2 \times (20 \times 1 \times 50) = 2{,}000 \text{ dollars.}$$

If we do not build direct lines between B and D but build the 50 lines by way of A and C , then the total number of lines to be built between A and C is 100, and the cheaper rate is available. Thus, the total cost is

$$2 \times 4 \times 1 \times 50 + 100 \times .75 \times 20 = 1{,}900 \text{ dollars.}$$

If we do not build direct lines between either A and C or between B and D but build 50 lines between A and E , B and E , C and F , D and F , and also 100 lines between E and F , then the total cost is

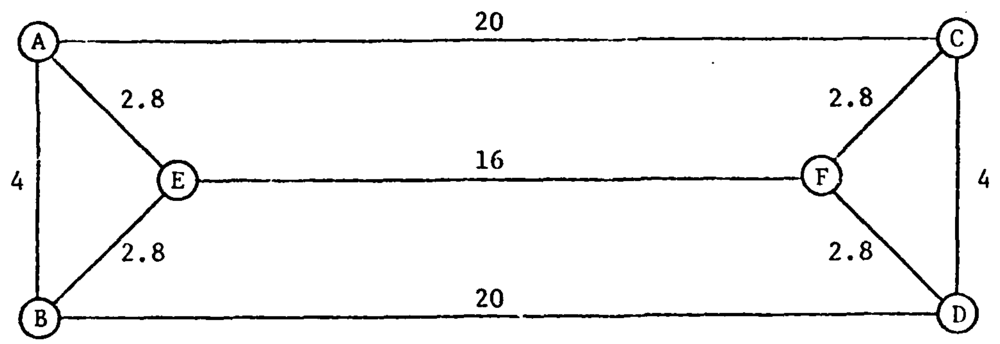


FIGURE 1.2

$$4 \times 2.8 \times 1 \times 50 + 100 \times .75 \times 16 = 1{,}760 \text{ dollars.}$$

So, the natural question is: what is the cheapest network that satisfies all the requirements. In the above example, the third type is the cheapest so far obtained and to get that we had to enumerate the cost of the direct routing and the cost of using other possible intermediate nodes. This enumeration becomes difficult as the number of sources and sinks increases or as the total number of nodes increases, and becomes almost impossible when the network configuration gets more and more complex. An attempt is made in this thesis to develop a systematic procedure for solving moderate-sized problems of this type.

1.2 Difficulties Faced in Solving the Problem

The difficulties encountered in this problem can be roughly divided into two categories. The first type of difficulty arises in building (from a real world communication network situation) a network model which can be analyzed. This will be illustrated with the network description that follows but not discussed further in this thesis. The second type of difficulty arises in the analysis of this problem. Because of a certain mathematical subtlety involved in this problem, a theoretically elegant solution procedure appears difficult to find, if one exists at all. This mathematical difficulty will be described at the end of this section to justify the use of a heuristic solution method in Chapter 3 and an elaborate enumerative solution method in Chapter 4.

1.2.1 Network Description and Practical Difficulties

Telephone calls are generated by the individual subscriber. The subscriber's demands on the network can occur whenever he chooses, can be of any duration, and can be directed to any other subscriber. Calls may be routed on one or more of three overlapping networks. The network connecting each subscriber to a central switching office is called the local network. The network connecting central offices within an area of one or more states is called the statewide network. The network which connects the main cities is called the long-haul network. These three classes of networks cover the country. The synthesis methods developed in this thesis are applicable mainly to the long-haul network and statewide networks. In the case of statewide networks, the main stations are the nodal points of the system. In the case of the long-haul network, cities are the nodal points of the system.

To calculate the number of channels required between nodal points, a projection is first made about traffic demand between two points. It is very difficult to project these traffic demands ahead of time. Demand can be affected by a change in rate structures, new service offerings, or conditions imposed by regulatory agencies. Moreover, new and unexpected technologies may significantly change the demand, as, for example, a communication satellite which could stimulate long distance calling and on-the-spot T.V. coverage, or an improved time sharing computer system which could increase data transmission through communication cables. However, despite all these uncertainties, projections are made for traffic requirements and these uncertainties are taken into account. Once traffic projections are made, the C.C.S. calculations are used to find the number of channels required for a certain probability of blocking. Usually, the blocking probability is .01 and standard traffic engineering practices are available for such calculations.

At this point, two complexities which enter into trunk requirement calculations must be taken into account. The first is the alternate routing plan. The alternate route is the path offered to a call if the most direct path is blocked. Alternate routes are always specified. If an alternate route from point A to point B goes through point C, then this fact must be taken into account in determining trunk requirements between points A and C, and between points B and C. The second complexity is what is known as diversity routing.[2] The trunks between two points are split into two or more physically different paths for better transmission reliability. That is, two points should not be completely disconnected if a part of the network is destroyed by a natural catastrophe or for any other reason. This is similar to alternate routing requirements. The above discussions show how difficult it is for a planner to say exactly how many channels are needed between each pair of points. However, for our synthesis problem, we shall assume that this figure has been determined.

Cost Estimation

There is, however, another set of difficulties, even given the knowledge of these channel requirements. Each traffic generating and terminating point, and all traffic handling points, are defined as nodes, and the links or connecting arcs between two nodes are transmission facilities. The graph is undirected because telephone conversations go both ways. Several facilities may comprise a link. This type of mapping by means of nodes and arcs is not always simple--a lot of judgemental factors are involved in deciding these nodes and arcs. For instance, at certain nodes some transmission lines pass directly through and some go via a switching machine. In this situation, the cost is different for the

[2] See "The Design of Minimum Cost Survival Networks," K. Steiglitz, P. Weiner, D. J. Kleitman, Technical Memorandum TM-105, National Resource Analysis Center, System Evaluation Division. This paper gives a heuristic treatment.

two sets, due to the extra cost of switching. At other nodes, all calls must be switched. In any case, however, the switching cost can be taken into account by using a technique called node splitting which means that one traffic handling point is split into two nodes and the cost of the arc between them is the switching cost. So, with imagination some of the difficulties can be taken care of. But a large number of practical difficulties cannot be tackled. Experience and common sense must be used in such situations.

Last, but not the least, of the first category of difficulties is determining the cost functions which relate cost to the number of channels. Cost is the main objective in our design considerations. But cost to whom, company or subscriber? And how much is the cost incurred? These are natural questions which arise when we are talking about cost. In the TELPAK problem where interoffice facilities for a large corporation are considered, the cost referred to is cost to the customer, and it could be different from the actual cost incurred by the telephone company. But in other cases cost refers to the cost incurred by the telephone company. In both situations, an element of uncertainty is involved in these costs. It is affected by the interest rate, available funds and the production cost of facilities and designs.

The possible kinds of cost functions encountered in practice in this problem are illustrated in Figures 1.3, 1.4, 1.5, 1.6, and they all have one thing in common--economy of scale. All the cost functions are what we will call "concave type"[3] and they are nondecreasing functions of flow value.

[3] That is, they are not necessarily concave because concave functions are continuous everywhere except at the end points and the following functions may not have that continuity property. Also, concave functions satisfy $\lambda \cdot f(x_1) + (1 - \lambda) \cdot f(x_2) \leq f(\lambda x_1 + (1 - \lambda)x_2)$, $0 \leq \lambda \leq 1$, which only the third and fourth of our functions satisfies (assuming λ be such that $(\lambda x_1 + (1 - \lambda)x_2)$ is a point at which f is defined).

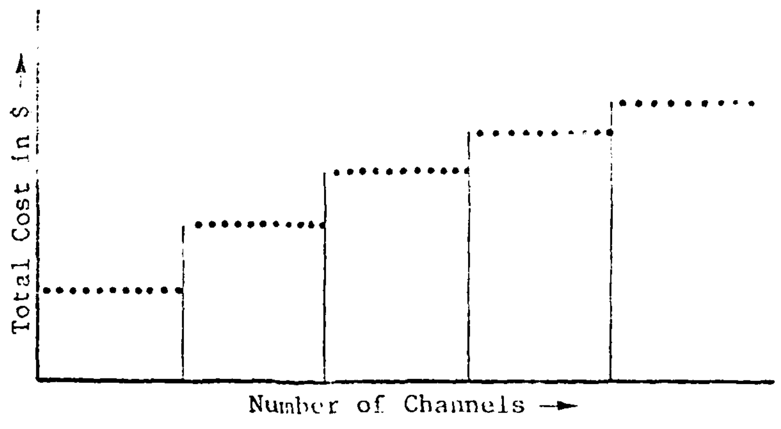


FIGURE 1.3

The step function with decreasing step size--this type of cost structure may be encountered because of the uses of different types of facilities for different ranges of the number of channels. A more sophisticated carrier can be used when the number of channels is large, resulting in a lesser cost per channel and hence smaller step sizes. The change of type of carrier can only occur at certain volumes of channel requirements resulting in jumps occurring in the cost curve.

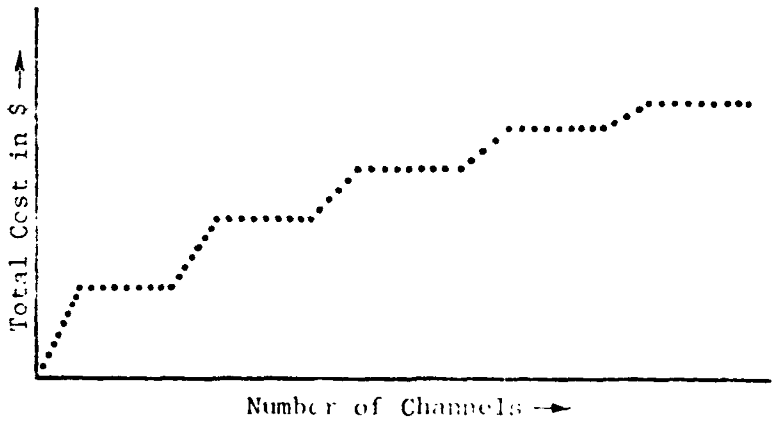


FIGURE 1.4

Semi-step function--the steps are connected by slanted lines. The steps of decreasing size occur due to the same reason as above. Here a variable cost is incurred in transitional capacity levels of different facilities, i.e., a facility can be overloaded at some additional cost per channel.

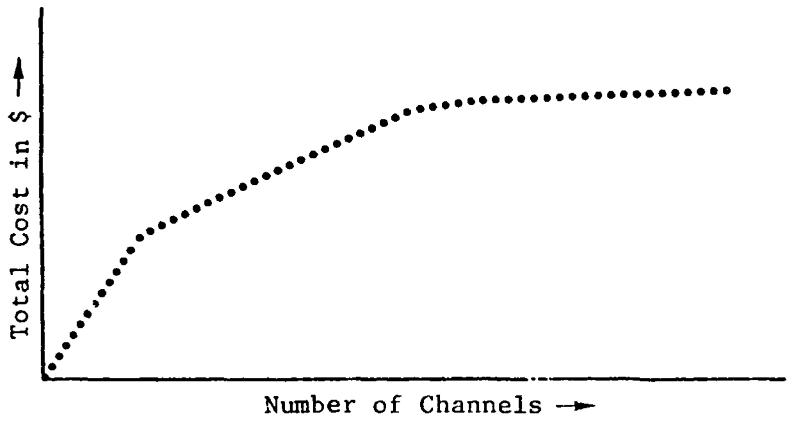


FIGURE 1.5

Piecewise linear concave function--here each part is linear but the slope decreases with increasing flow value. Here a variable cost is incurred for each additional channel but this variable cost depends on the range it is in.

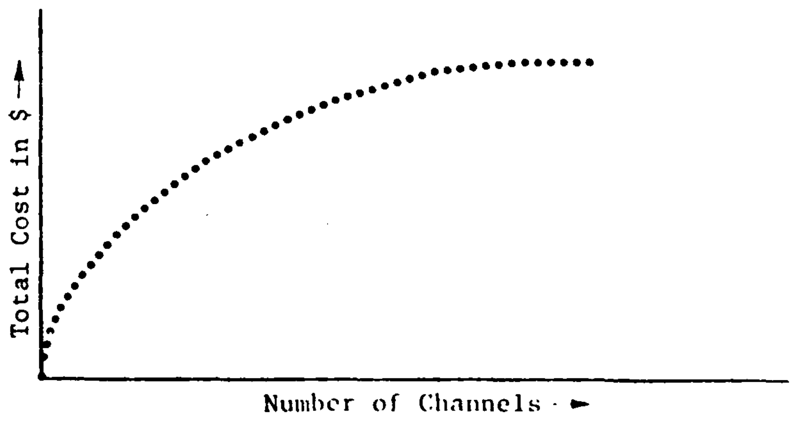


FIGURE 1.6

The final type has decreasing forward differences, i.e., if $f(n)$ is the total cost of n channels then $f(n + 1) - f(n) > f(n + 2) - f(n + 1) > \ldots > 0$. Here total cost increases with the addition of each channel, but cost per channel decreases. This is strictly a concave function of discrete type. In cases where only approximate cost data is available and some guess work is involved in getting the cost values, such functions are used.

This concludes our discussion of the practical difficulties which arise in the problem formulation stage. We assume that these difficulties can be surmounted and that the cost function, like the channel requirements, is available to the network synthesizer.

Discrete vs. Continuous Values

The cost data discussed above is discrete, i.e., values are only available for integral points. For the purpose of methods used in Chapters 3 and 4, we will employ a continuous concave representation of the data, obtained by using a continuous curve through the discrete points. However, such concave curves cannot be drawn through the discrete points for Figures 1.3 and 1.4. Hence, some approximation is necessary. Use of such a continuous concave version is justified (except Figures 1.3 and 1.4) because the answer we get is always integral (as shown in Appendix C), and the values of the data at those points match the original data. For Figures 1.3 and 1.4, some post optimization procedure is needed. Except in Chapter 5, we will always be concerned with continuous variables and concave functions. Another alternative way of joining the discrete points is by piecewise linear curves. The algorithm described in Chapter 3 is applicable to such curves with only minor modifications, as mentioned in that chapter.

1.2.2 Theoretical Difficulties

We are now in a position to discuss the second type of difficulties, those which arise in the analysis of the problem. Here we will consider a network with N nodes and M arcs (M can be different from $\frac{N(N-1)}{2}$ because some arcs between nodes are not possible) joining the nodes through which calls are channeled. The requirement matrix $R = \{r_{ij}\}$ is specified where r_{ij} = number of channels required between node i and j . Also the function $f_m(y_m)$ is given for all M arcs. This function gives the cost of y_m channels (amount of flow, in network terminology) in arc m of the network. With all this information, we can set up the above problem as a programming problem.

In the discussion that follows, the concept of extreme points of a set is very important.

Definitions:

A convex combination of two points X_1 and X_2 in a vector space is a point X given by $X = \{\lambda X_1 + (1 - \lambda)X_2\}$ where λ is a scalar and $0 \leq \lambda \leq 1$.

An extreme point X of a set is a point in the set such that there exist no two distinct points X_1 and X_2 in the set and $1 > \lambda > 0$ such that the point X is a convex combination of X_1 and X_2 .

Extreme flow is similarly defined as a feasible flow which cannot be expressed as a convex combination of two other distinctly different feasible flows.

In the arc-chain formulation that follows, two types of vector spaces, X and Y , are used. The X-space is defined as "chain space" and each of its element $\left\{x_{ij}^{k}\right\}$ indicate the amount of flow in the kth chain between source and sink pair i j . The Y-space is defined as "arc-space" and each of its elements $\{y_m\}$ indicates the amount of flow in arc m. Y-space is a linear transform of X-space by means of arc-chain incidence matrices A_{ij} defined below.

<u>Arc-Chain Formulation</u>

P_{ij} = Number of different loopless chains between node i and j.

p_{ij}^k = The set of loopless chains connecting node i and j, where $k = 1,2, \ldots, P_{ij}$.

X_{ij} = A vector of dimension P_{ij} where element $\{x_{ij}^k\}$ represents x_{ij}^k amount of flow between i and j through chain p_{ij}^k.

A_{ij} = $M \times P_{ij}$ arc-chain incidence matrix, whose (m,k) element is 1 if the chain p_{ij}^k traverses arc m, or 0 if it does not.

$Y = \sum_{i,j \ni i>j} A_{ij}X_{ij}$ = M vector $\{y_m\}$ which is the total flow in the arc m.

So, here we have to minimize the cost Z, i.e.,

(1.1a) $$\text{Minimize } Z = \sum_{m=1}^{M} f_m(y_m) \text{ ,}$$

subject to the constraints

(1.1b) $$\sum_{k=1}^{P_{ij}} x_{ij}^k \geq r_{ij} \quad \text{for all source and sink pairs } i\ j \text{ ,}$$

(1.1c) $$x_{ij}^k \geq 0 \quad \text{for all } k \text{ and all source and sink pairs } i\ j \text{ .}$$

The linear inequalities (1.1b) and (1.1c) define a polyhedron in X-space. Since Y-space is a linear transform of X-space, the above polyhedron can be transformed into another polyhedron in Y-space. The objective function Z is the positive sum of concave functions of elements y_m of the vector Y. So, it is a concave function in Y-space.

$Z(Y) = Z\left(\sum_{i,j \ni i>j} A_{ij}X_{ij}\right) \overset{\text{definition}}{=} \bar{Z}(X)$, where $\bar{Z}(X)$ is the objective function in X-space.

$\bar{Z}(X)$ is a concave function in X-space as evident from the following inequality. Consider any two points X^1, X^2, in X-space and a scalar λ such that $0 \le \lambda \le 1$. Y^1 and Y^2 are defined as the images of X^1 and X^2 in Y-space, i.e., $Y^1 = \sum_{i,j \ni i>j} A_{ij}X^1_{ij}$ and $Y^2 = \sum_{i,j \ni i>j} A_{ij}X^2_{ij}$.

$$\bar{Z}(\lambda X^1 + (1-\lambda)X^2) = Z\left(\sum_{i,j \ni i>j} A_{ij}\left(\lambda X^1_{ij} + (1-\lambda)X^2_{ij}\right)\right) \text{ by above definition}$$

because λ is a scalar and A_{ij}'s are matrices.

$$= Z\left(\lambda \sum_{i,j \ni i>j} A_{ij}X^1_{ij} + (1-\lambda) \sum_{i,j \ni i>j} A_{ij}X^2_{ij}\right)$$

$$= Z(\lambda Y^1 + (1-\lambda)Y^2) \text{ by definition}$$

$$\ge \lambda Z(Y^1) + (1-\lambda)Z(Y^2) \text{ because } Z(Y) \text{ is concave in Y-space}$$

$$= \lambda Z\left(\sum_{i,j \ni i>j} A_{ij}X^1_{ij}\right) + (1-\lambda)Z\left(\sum_{i,j \ni i>j} A_{ij}X^2_{ij}\right) \text{ by definition}$$

$$= \lambda\bar{Z}(X^1) + (1-\lambda)\cdot\bar{Z}(X^2) \text{ by definition}$$

The following lemma is useful in determining extreme points of the polyhedral set in X-space.

Lemma 1.1:

A point is an extreme point of the polyhedral set in X-space if and only if it corresponds to a solution with a single flow-chain between each source-sink pair.

Proof:

Suppose there are n source-sink pairs with $r_{ij} > 0$. Then there are n-equations of the form (1.1b).

Any basic feasible solution of the convex polyhedron defined by equations (1.1b) and (1.1c) can have at most n nonnegative variables (because there are n-equations of the form (1.1b)). No variable appears in more than one equation of the set (1.1b) and all the n-equations have to be satisfied. So, any feasible solution must have at least n-positive variables. Thus, every basic feasible solution has exactly n-positive variables, one for each source-sink pair. This establishes that any basic feasible solution corresponds to a single flow-chain between each source-sink pair.

If there is a single chain between each source-sink pair, then only one variable for each equation (1.1b) is positive and each of these positive variables are different from the other. So, they form an independent feasible set. Thus, they are a basic feasible solution.

So, a solution is a basic feasible solution if and only if it corresponds to the solution with a single flow-chain between each source-sink pair. A solution is an extreme point if and only if it is a basic feasible solution (Page 100, Reference H-1). Hence, a point is an extreme point of the polyhedral set if and only if it corresponds to a solution with a single flow-chain between each source-sink pair.||

In this problem, a concave function has to be minimized over a convex polyhedral set. The following lemma substantially reduces the number of points to be searched in any direct enumeration procedure.

Lemma 1.2:

The minimum of a concave function on the convex polyhedral set D, if it exists, is attained by at least one of the extreme points (vertices).

Proof:

Since the set is a polyhedral set, any point Y in the set can be expressed as a convex combination of its extreme points $Y_1, \ldots, Y_m$. Suppose the minimum is attained at Y_o where $Y_o = \sum_{i=1}^{m} \alpha_i Y_i$ and $\sum_{i=1}^{m} \alpha_i = 1$, $\alpha_i \geq 0 \ \forall \ i$. Since Y_o is the minimum point,

$$f(Y_o) \leq f(Y) \quad \text{for any} \quad Y \in D$$

but $f(Y_o) = f\left(\sum_{i=1}^{m} \alpha_i Y_i\right) \geq \sum_{i=1}^{m} \alpha_i f(Y_i)$ because $f(Y)$ is a concave function.

Let $f(Y_s) = \operatorname*{Min}_i f(Y_i)$. So,

$$f(Y_o) \geq \sum_{i=1}^{m} \alpha_i f(Y_i) \geq \sum_{i=1}^{m} \alpha_i f(Y_s) \quad (\text{since} \quad \alpha_i \geq 0)$$

$$= f(Y_s) \quad \left(\text{since} \quad \sum_{i=1}^{m} \alpha_i = 1\right).$$

Since $Y_s \in D$, $f(Y_o) = f(Y_s)$. Thus, the extreme point Y_s must also be a minimum point.||

So, a direct method of finding the minimum of the concave function in a polyhedral set is to enumerate all the extreme points and evaluate the function at all these extreme points to determine the minimum. But this is not a very practical method because the number of extreme points in the polyhedra defined by (1.1b) and (1.1c) is very large and grows very rapidly with an increase in the number of arcs and source-sink pairs. From the Lemma 1.2, the minimum is attained at one of the extreme points in Y-space. However, the extreme points in Y-space are very difficult to determine. And it is easy to determine an extreme point in X-space by using a single chain for each source and sink pair (from Lemma 1.1). Therefore, it is useful to determine the relationship between extreme points in Y-space and extreme points in X-space, which is done in Lemma 1.3.

The image of a point P of X-space in Y-space is the point in Y-space which is obtained by transforming the coordinates of the point P by using the matrices A_{ij} .

The following small example reveals that more than one point in X-space can have the same image in Y-space, and it also shows that the image of an extreme point in X-space can be a nonextreme point in Y-space.

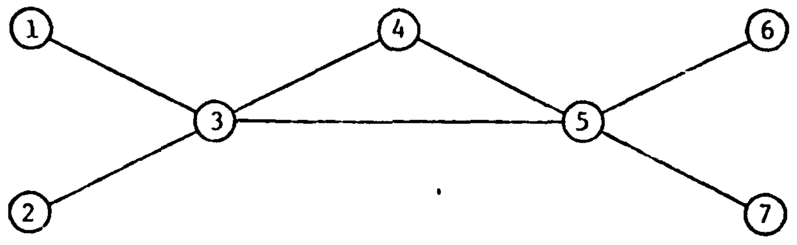


FIGURE 1.7

Example:

The required flows are $r_{17} = 2$ and $r_{26} = 2$. Let $X^T = \left(x_{17}^{1357}, x_{17}^{13457}, x_{26}^{23456}, x_{26}^{2356}\right)$ and $Y^T = (y_{13}, y_{23}, y_{34}, y_{35}, y_{45}, y_{56}, y_{57})$.

The vector Y_1 , where $Y_1^T = (2,2,2,2,2,2,2)$, is the image of both X_1 and X_2 , where $X_1^T = (2,0,2,0)$ and $X_2^T = (0,2,0,2)$. Both in X_1 and X_2 there is a single path between each source and sink pair, so they are extreme points. However, Y_1 is a convex combination of two feasible points $Y_{11}^T = (2,2,4,0,4,2,2)$ and $Y_{12}^T = (2,2,0,4,0,2,2)$, since $Y_1 = \frac{1}{2} Y_{11} + \frac{1}{2} Y_{12}$. So, Y_1 is not an extreme point.

Lemma 1.3:

For a point to be an extreme point of the polyhedron in the Y-space, it is necessary, but not sufficient, that it be the image of an extreme point of the polyhedron in X-space.

Proof:

Suppose the lemma is not true, and that there exists $\bar{Y}$ which is an extreme point of the polyhedron in Y-space and it is an image of $\bar{X} \varepsilon$ polyhedron in X-space, which is not an extreme point (i.e., $\bar{Y}$ is the image of nonextreme point in X-space). So, $\bar{X} = \sum_i \alpha_i X^i$ where $\sum_i \alpha_i = 1$, $\alpha_i \geq 0 \ \forall \ i$, and at least two α_i's are nonzero, and X^i's are extreme points of the polyhedron in X-space. Let the image of X^i in Y-space by Y^i . Since X to Y is linear translation, $\bar{Y} = \sum_i \alpha_i Y^i$. Now if all Y^i's are not equal, then by definition $\bar{Y}$ is not an extreme point, a contradiction. However, if they are all the same, then $\bar{Y}$ is also image of X^i which is an extreme point in X-space. So, the necessary part is valid. The above example shows that it is not sufficient.||

The above discussion shows that if we search all the extreme points in X-space we cover all the extreme points in Y-space. Since some extreme points in X-space may correspond to nonextreme points in Y-space, this search procedure may involve some unnecessary work, but the ease of generating extreme points in chain space justifies the emphasis on X-space in the subsequent algorithms.

Another major difficulty arises in pursuing a solution to the above problem due to the fact that the problem has quite a few local minimum points which are not necessarily global minimum points.

Definition:

By a local minimum of $Z(Y)$ on D (where $Y \varepsilon R^n$) , we shall mean a vector $\bar{Y} \varepsilon D$ for which there exists a positive scalar ϵ and a corresponding ϵ-neighborhood $N(\bar{Y},\epsilon) = \{Y \varepsilon R^n : ||Y - \bar{Y}|| < \epsilon\}$, such that $Z(\bar{Y}) \leqq Z(Y)$ for all $Y \varepsilon D \cap N(\bar{Y},\epsilon)$. If the latter inequality holds for all $Y \varepsilon D$, then $\bar{Y}$ is called the global minimum of Z in D .

So, any gradient search method which looks for an improvement of the objective function in an ϵ-neighborhood might result in a local minimum point (one such method is discussed in Chapter 3). The most difficult aspect of this is the identification of the globally minimum point when it is obtained. Considerable search in this direction convinces the author that there is no general method available which identifies a global optimal point for such optimization problems, though such identification in some special cases may be possible, as will be discussed in Chapter 2.

Though there is only one commodity, telephone calls, the problem must be treated as a multi-commodity problem because calls are distinguished by their starting and ending points, and we assume that there are many different pairs of starting and ending points specified. So, the difficulties of multi-commodity flow problems also arise here.

If (as in the synthesis problem we are studying) there is no capacity restriction and if (as is not the case in our problem) the cost is a linear function of the flow, then the solution can be easily obtained by Floyd's shortest path procedure. Here the linear cost coefficients are used as the lengths of the arcs and the entire flow between each source-sink pair is passed through the shortest path between them. In most linear cost problems, however, there is a capacity constraint. In such cases, a very efficient algorithm exists only for the case of 2 source-sink pairs [H-3]. For more than two pairs, a certain large scale programming method using a column generation technique is useful. This linear cost method can be extended successfully to *convex* cost functions because each arc can be split into a number of arcs, which transforms the problem into a linear cost problem with an enlarged number of arcs. For instance, the convex cost on an arc in the Figure 1.8 can be approximated by 3 arcs having linear cost coefficient a_1 , a_2 , a_3 , and capacity c_1 , c_2 , c_3 , respectively.

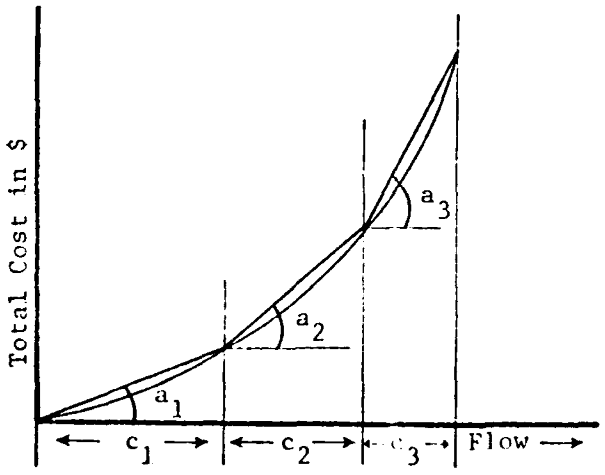


FIGURE 1.8

Here the proper order of choice of arcs is guaranteed (i.e., arc (a_1,c_1) (cost coefficient a_1 and capacity c_1) is completely filled before (a_2,c_2) is used and when (a_2,c_2) is filled (a_3,c_3) is used) because the cost coefficient increases with flow value. However, for concave cost, the cost coefficient decreases with the flow value so the proper order will not be maintained. Hence, such simplification is not possible and no procedure of finding the global optimal point, which does not involve the risk of total enumeration, is available. The methods proposed in succeeding chapters are local search, and branch and bound (here, in the worst possible case, total enumeration may take place) procedures.

CHAPTER 2

The last chapter exposed the practical and theoretical difficulties involved in seeking a solution procedure for the problem considered in this thesis. Before discussing the specific solution procedures in succeeding chapters, we will survey in this chapter the available results. The relevant areas of research which must be covered are multi-commodity flow problems and concave cost minimization problems. We will survey the important results in these two fields, and we will discuss a simple variation of this problem which is relatively easy to solve.

The pioneering work in multi-commodity flow problems was done by, among others, L. R. Ford, D. R. Fulkerson, T. C. Hu, R. E. Gomory and J. A. Tomlin[1].

Ford and Fulkerson [F-4] initiated the analysis of the maximum flow problem where there is more than one source and sink pair. They showed that if the flows are from a set of sources to a set of sinks, then the maximum sum of flows between the two sets can be obtained by solving a maximum flow problem between a super source and a super sink in an augmented network, where there are extra arcs of infinite capacity from the super source to the set of all sources and from the set of all sinks to the super sink. They also indicated [F-3] a solution procedure for the more general problem discussed below, gave the formulation shown in equations (2.2a, b, c) and indicated the solution method discussed there.

The more general multi-terminal, multi-commodity, maximum flow problem for a capacitated network was formulated by Hu and Gomory [G-4] as follows. Let there be sources N_s and sinks $N_{s'}$, $(s = 1, \ldots, q,$ $s' = 1', \ldots, q')$ where flow s is from N_s to $N_{s'}$. Let x_{ij}^{s} be the

[1]The names of R. T. Chien [C-2] and W. Mayeda [M-1] also should be mentioned in this connection. But for various reasons I will not be discussing their work.

flow s in the arc ij , and f(s,s') be the value of the flow s from N_s to $N_{s'}$. The first type of problem is to

(2.1a) Maximize $\sum_{s=1}^{q} f(s,s')$

(2.1b) Subject to $\sum_i x^s_{ij} - \sum_k x^s_{jk} = \begin{cases} -f(s,s') & \text{if } j = s \\ 0 & \text{if } j \neq s , s' \\ f(s,s') & \text{if } j = s' \end{cases}$

(2.1c) $\sum_{s=1}^{q} |x^s_{ij}| \leq b_{ij}$ (for all i,j)

(2.1d) $x^s_{ij} \geq 0$ (for all s,i,j) .

Unlike the one commodity max-flow problem, the constraint matrix here is not unimodular,[2] so an integral solution is not guaranteed. Hu [H-3] solved a special case of the above problem, the case of two commodities in an undirected network where capacities b_{ij} are even integers by using his max bi-flows min-cut theorem. His solution procedure involves the following steps. First the labelling algorithm is used to find max-flow of the 1st commodity. Then a cycle of flow of the 1st commodity is determined in such a way that by introducing some flow in the cycle, the flow values in certain arcs are decreased, so that a flow augmenting chain of the 2nd commodity is obtained to introduce the 2nd flow at the maximal possible level without changing the flow value of the 1st commodity. Rothschild and Whinston [R-4] have extended this two commodities flow result for a pseudo-symmetric[3] network. The linear programming solution of the

[2] A matrix is unimodular if all the subdeterminants of the matrix have values 0 or 1. It has been shown (p. 125, [H-3]) that with unimodular matrix D and integral vector b , the convex polyhedron $DX \geq b$ has all integral extreme points, and also if all the extreme points are integral and b is integral vector then D is unimodular.

[3] Definition: A network is Pseudo-symmetric if it has all even nodes $\left(\text{a node } i \text{ is even if } \sum_i b_{ij} \text{ is an even integer}\right)$.

max-flow problem in the multi-commodity case is also studied by Gomory and Hu [G-3]. Here the arc-chain formulation is used. Consider an M arc network with capacities $b_1, \ldots, b_M$. A chain in the network can be represented by an M vector with 1 in a component if the arc is used, and 0 if the arc is not used in the chain. Let us define an arc chain incidence matrix $A = [a_{ij}]$ as follows

$$a_{ij} = \begin{cases} 1 & \text{if the arc } i \text{ is in the chain } j \\ 0 & \text{otherwise .} \end{cases}$$

If x_j is the amount of flow in chain j, the multi-commodity max-flow problem is given by

(2.2a) Maximize $\sum_j x_j$

(2.2b) Subject to $\sum_j a_{ij} x_j \le b_i \quad (i = 1, \ldots, M)$

(2.2c) $x_j \ge 0$

where b_i is the capacity of arc i.

The matrix $[a_{ij}]$ has a very large number of columns (defined as a_j), one column for each possible chain for each commodity. Any basis has only M columns and at each step one has to consider one additional column as a candidate for the basis. So at any time we need to consider only a $(M + 1) \times (M + 1)$ matrix. Suppose we have M columns to start the algorithm and get the dual price vector $\pi = (\pi_1, \ldots, \pi_M)$ where each π_i corresponds to a specific row. Notice that $c_j = 1$ for all j here. The relative cost vector of every nonbasic column a_j is given by $\bar{c}_j = 1 - \pi a_j$. If all $\bar{c}_j \le 0$ then the present basis is optimal. If some $\bar{c}_j > 0$ we need to bring that column into the basis. All of the

π_i's can be made nonnegative by introducing into the basis columns corresponding to slack variables in rows with negative π_i's . Then the problem of determining which nonbasic column to bring into the basis is simply solved by the column generation scheme as follows. Use π_i as the length of arc i and find the shortest chain (a_j) connecting any source-sink pair. If all $\pi a_j \geqq 1$, stop since all $\bar{c}_j < 0$ and we have the optimal solution. If not, bring column a_j into the basis.

Hu and Gomory also considered the feasibility problem where certain flow requirements have to be satisfied, which is very similar to the previous problem. Sometimes the problem of feasibility is coupled with the necessity to minimize the cost of building the capacity. If c_i is the cost of building a unit capacity in arc i , we would like to minimize $\sum_i c_i b_i$. This problem is discussed in more detail by Tomlin. The most general case of the above problem, which Hu solved, is the case in which requirements vary over different time periods. Here a set of requirements has to be satisfied for each of T periods. The objective is to minimize the cost of building a sufficient capacity. An example of multi-commodity minimum cost flow problem with time varying requirements is solved in Hu's book [H-3].

Tomlin [T-1] has formulated explicitly the problem of finding minimum cost flow in a capacitated network which satisfies certain flow requirements for a directed network using both node-arc and arc-chain matrices, and for an undirected network using only the arc-chain matrix. In both formulations he obtains a large scale linear program having a special structure for the constraint matrix so that the Dantzig-Wolfe decomposition principle can be applied. After an application of the decomposition principle, a subproblem (finding the shortest route problem between two terminals) must be solved in order to generate columns. Many efficient algorithms can be applied to solve these subproblems because, with proper manipulations, we can restrict these problems so that they have only nonnegative arc cost (shortest route problems with all nonnegative entries in distance matrix are easy to solve).

More recently, Richard C. Grinold [G-5] has indicated a polyhedral game approach (similar to Kornai and Liptak [K-3]) to the multi-commodity max-flow problem in a directed network. The solution procedure involves allocating flow capacity to the various commodities, solving a one-commodity max-flow problem for each commodity and a very trivial linear program at each step. The method is easy to code and involves simple computation but is recommended only for suboptimization because of its poor convergence property. He also indicated an extension of this procedure for the multi-commodity min-cost flow problem, but with the restriction that in order to make the problem feasible for certain allocations of flow capacity we need to introduce high cost by-pass arcs, parallel to the original arcs, which can effectively double the problem size.

So most of the work in multi-commodity flow is restricted to max-flow or to linear cost cases involving a capacitated network. The problem considered in this thesis has no capacity restrictions - so a major set of constraints is avoided, thus making the problem comparatively easier to tackle. However, the nonlinearity of cost structure, particularly concave cost, introduces substantial difficulties. A look at the literature on the concave cost minimization problem is appropriate at this point.

Philip B. Zwart [Z-3] has made an interesting observation for a class of nonconvex programming problems of the form:

$$\begin{aligned} &\text{Minimize} \quad F(X) \\ &\text{Subject to} \quad G_i(X) \leq 0 \end{aligned}$$

where $F(X), G_i(X)$'s are not necessarily convex functions. If X^* is a local minimum and λ_i's are the corresponding lagrange multipliers for the problem, then, if the modified objective function $F(X) + \sum \lambda_i G_i(X)$ becomes a convex function, X^* is also a global minimum for the problem. By this one can sometimes recognize when a local minimum point is a global minimum point. Unfortunately, if we do not have convexity then we cannot tell whether the point is local or global minimum.

Willard I. Zangwill [Z-1] has considered a special type of flow problem in which the cost of flow in any arc is a concave function of the amount of flow. This special type is when there is a single source and a number of sinks or a single sink and a number of sources. He defined the concept of extreme flow as one which is not a convex combination of two other flows. The extreme flow corresponds to the extreme point of the convex polyhedron generated by flow conservation conditions in node-arc or arc-chain space. He characterized the extreme points in such problems and showed that extreme flow corresponds to arborescence flow in multi-sink single source or multi-source single sink problems. Based on this observation he developed a dynamic programming algorithm to solve such problems. Consider a network with source node 1 and sinks a and b, where $b = a + 1$. A flow from node 1 of r_a units to a and r_b units to b is required at minimum cost. We know in arborescence flow that some arcs will have $r_a + r_b$ flow, some will have r_a flow, and some will have r_b flow. Furthermore we know that if the flow $r_a + r_b$ is separated at a node into r_a on one arc and r_b on another, the flow will never be $r_a + r_b$ on any subsequent arc. Define $V_e(a)$ to be the minimum cost of shipping r_a units from node e to node a, $V_e(b)$ to be the minimum cost of shipping r_b units from node e to node b, and

$V_e(a,b)$ be the minimum cost of shipping r_a units from e to a and r_b units from e to b. To insure that no material is shipped from b to node a, define $V_b(a)$ and $V_b(a,b)$ as very large numbers. $A(e)$ is defined to be all nodes reachable from node e using the existing arcs of the network. $C_{ij}(x)$ represents the cost of x amount of flow in arc (i,j), a specified function. Then the recursive relations are:

$$V_e(a) = \min_{f \in A(e)} \{C_{ef}(r_a) + V_f(a)\} \quad \text{for all } e \neq a, b,$$

$$V_e(b) = \min_{f \in A(e)} \{C_{ef}(r_b) + V_f(b)\} \quad \text{for all } e \neq b$$

$$V_e(a,b) = \min_{f,g \in A(e)} \{C_{ef}(r_a + r_b) + V_f(a,b),$$

$$C_{ef}(r_a) + V_f(a) + C_{eg}(r_b) + V_g(b)\} \quad \text{for all } e \neq a, b$$

where $V_a(a) = V_b(b) = 0$, and $V_a(a,b) = V_a(b)$.

These relations are solved recursively until $V_1(a,b)$ is obtained, which gives the minimum cost for the required flow. This method can be generalized for more than 2 destinations. However, with n sinks the number of recursive relations to be evaluated at each node is $2^n - 1$. Hence, the calculations required become prohibitive for large n.

Arthur F. Veinott [V-1,2] has considered the characterization of the extreme points of Leontief[4] substitution system. The special network model considered by Zangwill corresponds to a transhipment Leontief substitution model. So, Veinott's approach puts Zangwill's algorithm in a more general setting. However, except for special cases (e.g., the one considered by Zangwill), the amount of computational effort required to search the extreme points to find one that is optimal increases exponentially with the size of the problem and so tends to be enormous.

[4]The definition of Leontief matrix, as well as transhipment Leontief substitution model is discussed in the paper by A. F. Veinott [V-1,2].

B. Rothfarb and M. Goldstein [G-1] in a recent paper considered the one terminal Telpack problem. This is similar to Zangwill's one sink, multiple source network problem, but they considered the cost function obtained from that in Figure 1.4 by connecting the discrete points by straight lines. Assume that the nodes are sequentially numbered from 1 to n, where the node n is the sink and the others are the sources. If an arc connects nodes i and j, $j > i$, it will be denoted by $b(i,j)$. If the flow is directed toward j it is positive, and if directed towards i it will be negative. The cost curves are defined by means of their points of discontinuity in derivative (break points) for arc $b(i,j)$; let $[W_k(i,j), C_k(i,j)]$ represent the flow and cost coordinates of the kth smallest positive value of flow at which the cost has discontinuous derivative. Then $W_{-k}(i,j) = -W_k(i,j)$ and $C_{-k}(i,j) = -C_k(i,j)$. Let $C_o(i,j) = W_o(i,j) = 0$. The level of flow x in an arc can be expressed as a linear combination of the flow levels at the nearest break point above x and the nearest one below x. To accomplish this, let $\{\lambda_{k(i,j)}\}_{k=-K}^{k=+K}$ be the set of numbers called flow indices associated with arc $b(i,j)$ such that

$$0 \leqq \lambda_k(i,j) \leqq 1 \quad \text{and} \quad \sum_{k=-K}^{+K} \lambda_k(i,j) = 1 ,$$

where K is a large enough number to cover the entire required range of flow values. If $\lambda_a(i,j) > 0$, then only $\lambda_{a-1}(i,j)$ or $\lambda_{a+1}(i,j)$, but not both, can be positive and only these two indices can be nonzero. Then the flow on $b(i,j)$ can be represented as $\sum_{k=-K}^{K} \lambda_k(i,j) W_k(i,j)$. A solution of this problem is given by a linear program.

$$\text{Minimize} \quad \sum_{(i,j)} \sum_{k=-K}^{K} C_k(i,j)\cdot\lambda_k(i,j)$$

$$\text{Subject to} \quad \sum_{j|i<j} \sum_{k=-K}^{K} W_k(i,j)\cdot\lambda_k(i,j) - \sum_{j|i>j} \sum_{k=-K}^{K} W_k(j,i)\cdot\lambda_k(j,i) = r_i$$

$$\text{for} \quad i = 1,2, \ldots, n-1 ,$$

where r_i is the required flow from the source node to terminal node n. Furthermore, there is an additional constraint; the set of flow indices $\{\lambda_k(i,j)\}_{k=-K}^{k=K}$ for each arc should satisfy the constraints specified above. So the ordinary linear programming procedure for generating the solution is not enough. Rothfarb and Goldstein show that any basis of the linear program will have arcs of two kinds, T_1 and T_2. T_1 are the arcs where one flow index is 1 and the rest 0, and T_2 are the arcs with 2 nonzero flow indices. By a series of theorems they established an intricate procedure which treats arcs of type T_1 and T_2 differently and determine which arc to bring in the basis at each step. So the pivot computation is far more elaborate than usual simplex method. Also the procedure can converge to a local, but not global optimal point. Consequently, it does not appear superior to the one-terminal versions of the algorithms of Chapter 3 and Chapter 4 to follow.

At this point one of the simple variations of the problem which is easily solved should be mentioned.[5] In this case there is just a single source and a single sink with flow requirement f between them and a concave cost function. Since it has been established that there must be an extreme flow which is the minimum, there will be a single path between the source and the sink. The flow in each of the arcs in this path will be f. To find the optimal path, use the cost of f amount of flow in

[5]The procedure is cryptically mentioned by Zangwill.

each arc of the network as the length of that arc. The shortest chain between the source and the sink of this network will be the global minimum cost solution.

Hoang Tui [T-2] has described a procedure that seeks the global minimum of a concave objective function $f(X)$ with the polyhedral constraint set $D \epsilon R^n$. The procedure starts with a local minimum point X^o which is assumed to be a nondegenerate (a necessary condition) extreme point of the polyhedron D. The procedure is illustrated for $n = 2$ case in Figure 2.1, where D is OACEFG. For notational simplicity X^o (origin O in Figure 2.1) is taken to be the origin. Along its n distinct edges, n points $y^{1,1}, \ldots, y^{1,n}$ ($y^{1,1}$ and $y^{1,2}$ along OA and OG) are chosen such that $y^{1,k} = \xi^k \cdot \theta^{1,k}$ where ξ^k is the direction vector of the kth edge, $\theta^{1,k} = \text{Max} \left\{ \theta \mid F(\xi^k\theta) \geqq \alpha_1 \right\}$, $\alpha_1 = f(X^o)$, and $F(X)$ is the concave extension[6] of $f(x)$ in R^n. $y^{1,k}$'s are linearly independent vectors. Within the cone $X^o, y^{1,1}, \ldots, y^{1,n}$, (the cone $Oy^{1,1}y^{1,2}$), the value of $f(X)$, is greater than or equal to α_1. The auxiliary problem at this point is to find the most distant extreme point of the polyhedron D from the hyperplane passing through $y^{1,1}, \ldots, y^{1,n}$ (the line $y^{1,1}y^{1,2}$ in Figure 2.1) on the opposite side of the origin X^o (the right-hand side of $y^{1,1}y^{1,2}$ in Figure 2.1). This is equivalent to solving the linear program

$$\text{Maximize} \quad h(X) = \sum_{k=1}^{n} \lambda_k$$

$$\text{Subject to} \quad X \epsilon D$$

[6]Concave extension of the function $f(x)$ is any function concave on the whole space R^n and coinciding with $f(x)$ on D.

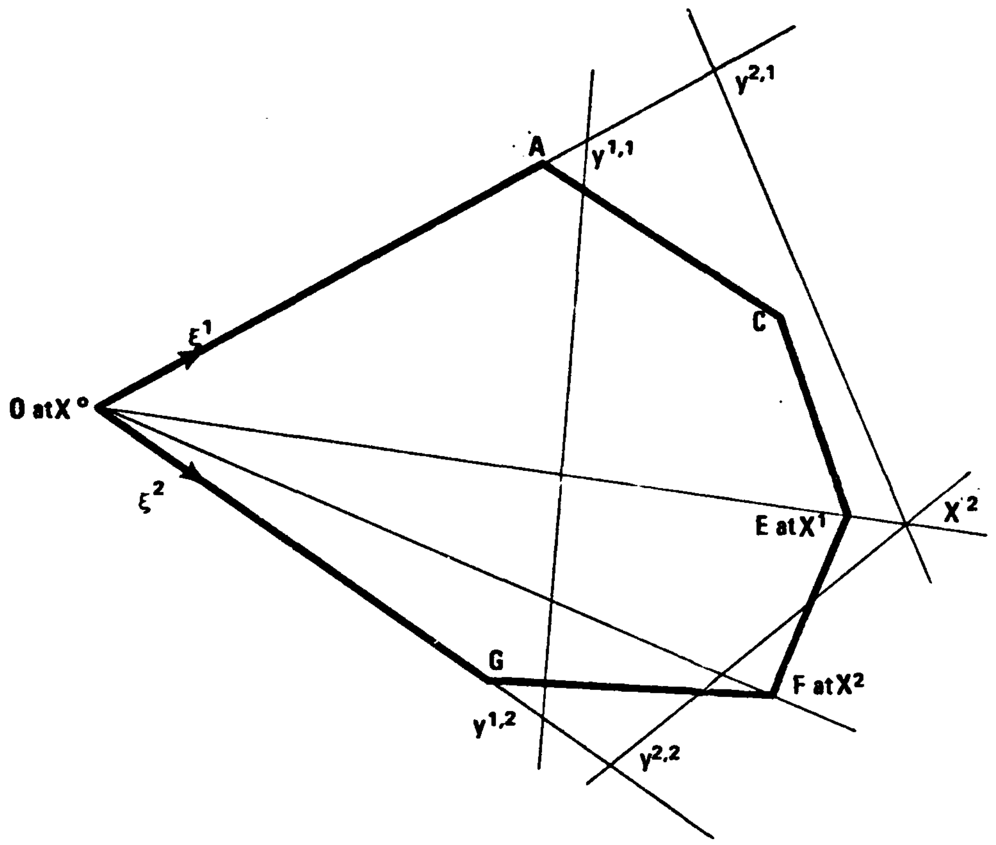


FIGURE 2.1

where λ_k's are elements of the n dimensional vector $\lambda = B^{-1} \cdot X$ and B is a matrix with columns $y^{1,1}, \ldots, y^{1,n}$.

If no solution of the above linear program exists, then X^o is the global minimum because the cone defined by $X^o y^{1,1}, \ldots, y^{1,n}$ contains the entire polyhedron. Otherwise, evaluate $f(X)$ at the new extreme point X^1 (point E furthest away from $y^{1,1}y^{1,2}$ in Figure 2.1). Determine $\alpha_2 = \text{Min}\left\{\alpha_1, f(X^1)\right\}$ ($\alpha_2 < \alpha_1$ in our example) and find $y^{2,k} k = 1,2, \ldots, n$ along n edges ($Oy^{2,1}$ and $Oy^{2,2}$ in Figure 2.1) as before, such that $F(y^{2,k}) \geqq \alpha_2$, and also determine $\bar{X}^2 = \theta_2 X^1$ such that $\theta_2 = \text{Max}\left\{\theta \geqq 0 \mid F(\theta X^1) \geqq \alpha_2\right\}$. Auxiliary problems involving smaller cones are generated by replacing one of the $y^{1,k}$ by $\bar{X}^2$ and keeping the others fixed. (Do this for $y^{1,k}$ only in case the corresponding $\lambda_k \neq 0$.) (For the problem of Figure 2.1, the two auxiliary problems are finding the furthest points of D on the right side of $y^{2,1}\bar{X}^2$ and $y^{2,2}\bar{X}^2$. No solution exist for the 1st problem because the entire polyhedron is on the left of $y^{2,1}\bar{X}^2$. For the 2nd auxiliary problem $F(X^2)$ is the solution. Now use point F in the same way that E was used previously.) In this way, more of the polyhedron is covered by each succeeding set of auxiliary problems; when the entire polyhedron is covered, we have obtained the global optimal solution. The advantage of this procedure is that all the auxiliary problems have the same constraint set D . However, the procedure is not useful in the problem of this thesis because dimension n of the polyhedral set D is very large. The number of auxiliary problems generated when k of the auxiliary problems are solved is k^n , an exponential growth. Whereas in the branch and bound procedure described in Chapter 4, the number of sub-problems becomes 2^k when the kth subproblem is solved. Also $h(X)$ is given in such a way that we cannot use any efficient method like the shortest path method to solve the auxiliary problem.

K. Ritter [R-3, C-3] has described a similar method of determining the global optimal solution for the nonconvex quadratic cost minimization problem with polyhedral constraint set. The procedure involves generating cutting planes which exclude local optimal points without excluding the global optimal point and then doing a local search (i.e., a search for a

local optimal point in the reduced set) on the reduced polyhedron. This is equivalent to a variation of Hoang Tui's method,[7] in which new local search procedure is carried out on the section of the polyhedron cut out by the hyperplanes generated by $y^{1,1}, \ldots, y^{1,n}$ on the opposite side of origin. The generation of a meaningful cutting plane in Ritter's method itself involves solving a quadratic programming problem. For our problem (assuming quadratic concave cost function), both the generation of cutting planes and a local search on the reduced polyhedron are huge quadratic programming problems. So, the procedure is not that useful. However, as we will point out in Chapter 5, some clever way of generating the cuts by solving simpler subproblems, and also a simpler local search procedure, may be a hopeful theoretical direction to pursue in order to find the global optimal solution for the problem of this thesis.

A. Victor Cabot and Richard L. Francis [C-1] have described a method of solution for the nonconvex quadratic minimization problem by ranking the extreme points. The method *per se* is not applicable for our problem, but the idea can be adapted to yield a solution procedure. Consider the problem:

$$\text{(P1)} \quad \begin{array}{ll} \text{Minimize} & f(Y) = C^T Y + Y^T D Y \\ \text{Subject to} & Y \in S \text{ , where } S = \{Y \mid AY = b, Y \geq 0\} \end{array}$$

[7]This is mentioned in the paper described in the preceding paragraph.

where Y is an n-vector of variables, the matrix A is $m \times n$, the matrix D is $n \times n$, and vectors C and b are n- and m-vectors respectively. d_j $j = i, 2, \ldots, n$ are columns of the D matrix. Solve $u_j = \text{Min} \left(d_j^T \cdot Y\right)$, subject to $Y \varepsilon S$, and assume that each of these n problems has a finite minimum. Then consider the problem

$$\text{(P2)} \qquad \text{Minimize} \quad \sum_{j=1}^{n} (c_j + u_j) y_j$$

$$\text{Subject to} \quad Y \varepsilon S$$

where y_j's are elements of the vector Y and c_j's are elements of vector C. Since $f(Y) = \sum_{j=1}^{n} \left(c_j y_j + Y^T d_j y_j\right)$, and $Y^T d_j \geq u_j$, then $g(Y) \leq f(Y)$.

If Y^o is the solution of P2, then a lower bound on the optimal value f^* of P1 is $f_\ell = g(Y^o)$ and an upper bound is $f_u = f(Y^o)$. It can be easily proved that if $\{Y^k\}$ is the set of extreme points of S such that $g(Y^k) \leqq f_u$ then the optimum solution Y^* of P1 is such that $Y^* \varepsilon \{Y^k\}$. So, the algorithm here involves searching systematically the set $\{Y^k\}$. To find the extreme points of S yielding the 2nd, 3rd, 4th, etc. smallest values of $g(Y)$, Murty's [M-4] extreme point ranking method is used. In this method only one pivoting step (for the non-degenerate case) is necessary to get each next lower ranked extreme point. Thus, this whole procedure involves:

<u>Step 0</u>:

Solve n linear programs to obtain u_j $j = 1, \ldots, n$.

<u>Step 1</u>:

Determine Y^o, the optimal solution of P2. Take $f_\ell = g(Y^o)$ and $f_u = f(Y^o)$ and Y^o as the current best solution of P1.

Step 2:

Use the extreme point ranking procedure to get the next best extreme point solution Y^k to P2. If $g(Y^k) > f_u$, then stop - the current best solution is the minimum solution to P1, and $f^* = f_u$. If $g(Y^k) \leqq f_u$, then replace f_ℓ by $g(Y^k)$.

Step 3:

If $f(Y^k) < f_u$, then replace f_u by $f(Y^k)$ and replace the current best solution of P1 by Y^k. Otherwise return to Step 2 without changing f_u or the current best solution.

Howev r, this method is not directly applicable to the problem of this thesis for the following reason. The objective function in our problem, when a quadratic approximation is made for all concave functions, is

$$f(Y) = \sum_{j=1}^{M} \left(c_j y_j + d_j y_j^2\right) \quad \text{where} \quad d_j \leqq 0$$

nd y_j is the element of the vector Y representing flow in arc j. re d_j is a scalar, or, in the notation of P1, D is a diagonal matrix. nce, $u_j = \text{Min } (d_j y_j)$ subject to the flow restriction of our problem. is subproblem is to find the minimum cost multicommodity flow in the twork where all arc costs are zero except one, arc j, which has a inear cost with negative coefficient d_j. This yields a negative loop o the objective function u_j tends to $-\infty$ (by passing a very large flow in the negative loop). Thus the method is not applicable for our problem because finite u_j values (which are unbounded here) are necessary to form the subproblem P2. However, the following variation of the procedure is applicable.

2.1 Solution Procedure by Ranking the Extreme Points

The problem considered has the objective function

$$\text{Minimize} \quad F(Y) = \sum_{j=1}^{M} f_j(y_j)$$

$$\text{Subject to} \quad Y \in S$$

where S is the feasible polyhedron in arc space Y i.e., $Y = \sum_{\substack{i,j \\ \ni i>j}} A_{ij}X_{ij}$ and X_{ij} is a P_{ij} vector $\left\{x_{ij}^k\right\}$ and satisfies the constraint equations 1.1b and 1.1c. If we can find the upper bound[8] U_j of the flow in each arc, then we consider solving the problem

$$\text{Minimize} \quad G(Y) = \sum_{j=1}^{M} C_j y_j \quad \text{where} \quad C_j U_j = f(U_j) \quad \text{(in Figure 2.2)}$$

$$\text{Subject to} \quad Y \in S$$

Then $G(Y) \leqq F(Y)$ in the entire feasible range of flow. Solving the modified linear problem is the same as solving shortest chain problems between all pairs of requirement points where arc lengths are C_i's . Let there be n positive requirement $(r_{ij} > 0)$ pairs.

Step 1:

Find the shortest chain, using C_i's as the distances of the arcs and pass the required flow through the shortest chain. Let Y^o be the arc flow vector. Then a lower bound of the optimal value F^* of the original problem $F_\ell = G(Y^o)$ and an upper bound is $F_u = F(Y^o)$. The current best solution is Y^o .

[8] A problem faced in the procedure of Chapter 4 and discussed there. A trivial upper bound is the sum of all required flows.

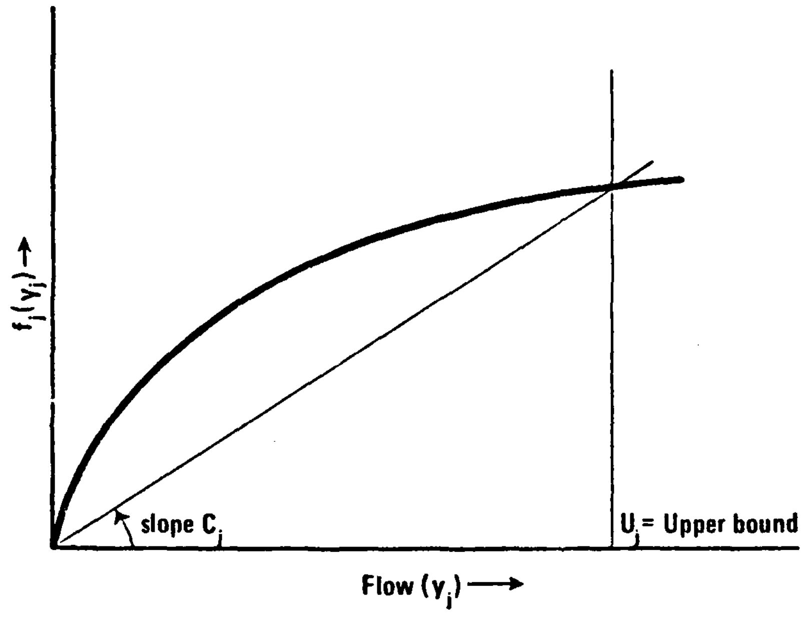

$f_j(y_j) \rightarrow$
slope C_j
U_j = Upper bound
Flow $(y_j) \longrightarrow$

FIGURE 2.2

Determine the 2nd shortest chains between all n pairs of requirement points. Let Δ_i , be the difference between the 2nd shortest chain and the shortest chain between source-sink pair ij . The list of n numbers $\{\Delta_{ij} r_{ij}\}$ is the amount of increment for the neighboring extreme points of Y^o , one (the smallest) of which corresponds to the increment of the next best extreme point of S .

Step 2:

Let s and t be the source-sink pair which gives the smallest increment (from the list) over Y^o , among all neighboring extreme points of the extreme points $(Y^o, \ldots, Y^{k-1})$ so far determined. And let it be the neighbor of $Y^m (m \leqq k - 1)$. Then displace the flow r_{st} from the chain between s and t corresponding to the point Y^m to the newly found chain between $s - t$ which gives minimum increment. Calculate the new arc flow vector Y^k . If $G(Y^k) > F_u$ stop - the current best solution is the optimal solution and $F^* = F_u$.

Step 3:

Solve the next best chain between s and t ; find Δ_{st} the difference between previously found chain and currently found chain between s and t . One neighboring extreme point of Y^k has increment $\left(G(Y^k) - G(Y^o) + \Delta_{st} \cdot r_{st}\right)$ over Y^o and remaining $(n - 1)$ neighboring extreme points have increments $\{G(Y^k) - G(Y^m) +$ increments over Y^o of neighboring extreme points of Y^m except $Y^k\}$ over Y^o . Include all these increments in the list.

Step 4:

If $G(Y^k) \leqq F_u$ then replace F_ℓ by $G(Y^k)$. If $F(Y^k) < F_u$ replace F_u by $F(Y^k)$ and current best solution of the original problem by Y^k .

Otherwise go to Step 2 without changing F_u or the current best solution.

So, in this procedure, after finding the 2nd best chain for all pairs, we need to find only the next best chain for a particular node pair at each iteration. To reduce the number of searches we could simply concentrate only on loopless chains. A chain with a loop cannot possibly be a minimum for the original problem because by taking out the flow in the loop we can reduce the cost. J. Y. Yen [Y-2] has proposed a good method of finding the K best loopless chains. However, to avoid confusion one should note that a chain with a loop, which is an extreme point in arc chain space, can also be an extreme point in node arc space. So by ignoring looped chains, we may be ignoring some extreme points, but these are obviously uninteresting ones. At any stage k in this procedure we need to store only increments of the neighboring extreme points of Y^k over Y^o such that $G(Y^o)$ + (increment of neighboring extreme points of Y^k over Y^o) $\leqq F_u$ and no others. The relative merit of this method compared to the methods described in succeeding chapters depends on the efficiency of the kth shortest chain determination procedure. For the general problem such comparisons are hard to make.

As a last note on the available literature on this TELPAK-type problem, we should mention the work of G. C. Watling and J. H. Weber [W-2]. They developed a heuristic algorithm which synthesizes, from the traffic data, the network and also the best position for switching centers. This heuristic procedure uses only the total amount of originating and terminating traffic as the input traffic data and does not consider the traffic flow between specific originating and terminating points. This is reasonable if the traffic is statistically well dispersed between all points, and this method can handle a very large problem. It is purely heuristic and can yield a solution that is not even an extreme point of the problem we are studying, and is not very mathematical in approach.

CHAPTER 3

The particular solution procedure[1] described in this chapter can be used to solve moderate sized problems (networks of up to 200 nodes) and it is very efficient (i.e., the procedure generally converges in a few iterations). However, it yields locally optimal points and one does not know how near the value is to the global optimal value. In this chapter strategies that are useful in obtaining the global optimal, using a procedure that yields local optimal points, are described. Certain statistical procedures will also be developed, including a sequential sampling plan where further search for the global optimal point is stopped when the cost of further computation becomes more than the estimated gain in reduction in the optimal value.

For the main procedure, of this chapter, to work, all the functions $f_m(y_m)$, the cost of y_m amount of flow in arc m , have to be nondecreasing concave functions in the feasible regions of y_m . In Appendix D it is shown that any concave function is continuous everywhere except perhaps at the left and right-hand end points of the interval. Since the function is nondecreasing it has to be continuous at the right-hand end point. This continuity property may not hold at the left-hand end point; there could be a jump at $y_m = 0$. Appendix D also proves that both left-hand and right-hand derivatives $D^- f_m(y_m)$ and $D^+ f_m(y_m)$ exist at all points except the end points of the interval, and the following relations hold:

[1]This procedure was suggested by Bernard Yaged [Y-1] for concave cost functions which have both first and second derivatives, but we shall show that it is valid for any nondecreasing concave cost function.

(3.1a) (i) $D^{-}f_m\left(y_m^1\right) \geqq D^{-}f\left(y_m^2\right) \quad \forall \quad y_m^2 > y_m^1$

(3.1b) (ii) $D^{+}f_m\left(y_m^1\right) \geqq D^{+}f\left(y_m^2\right) \quad \forall \quad y_m^2 > y_m^1$

(3.1c) (iii) $D^{-}f_m(y_m) \geqq D^{+}f_m(y_m) \quad \forall \quad y_m$ in the open feasible interval .

The nondecreasing property of the function guarantees that both left and right-hand derivatives are nonnegative. The following discussions and propositions characterize the optimal flow pattern.

By Lemmas 1.1 and 1.2 of Chapter 1 we have shown that, if the cost functions are concave, in the optimal flow pattern there is a single chain between each source-sink pair; i.e., for every source-sink pair st , a required flow r_{st} passes through a single chain p^*_{st} in the minimum cost (optimal) network. (However, degeneracy may occur; two chains may have the same cost.) We will show a stronger condition for flow patterns where cost functions are concave.

Proposition 3.1:

If there are two chains carrying total flow r_{st} between a single source-sink pair st then the entire flow r_{st} can be put in one of the two chains without increasing the total cost.

Proof:

Suppose there are two chains P_a and P_b between source-sink pair st carrying x_a and x_b amount of flow respectively such that $x_a + x_b = r_{st}$.

For any arc m in chain P_a with flow level $y_m + x_a$, the contribution to the cost due to flow between s and t in this arc is taken to be $g_m(x_a) = f_m(y_m + x_a) - f_m(y_m)$. A similar definition exists for the arcs in chain P_b . $g_m(\cdot)$ is also a concave nondecreasing function because $g_m(\cdot)$ is obtained from $f_m(\cdot)$ by shifting the origin to y_m . Define the functions $C_{P_a}(x_a) = \sum_{m \epsilon P_a} g_m(x_a)$ and $C_{P_b}(x_b) = \sum_{m \epsilon P_b} g_m(x_b)$. From the above definitions, the cost of r_{st} flow is taken to be $C_{P_a}(x_a) + C_{P_b}(x_b)$ and both $C_{P_a}(\cdot)$ and $C_{P_b}(\cdot)$ are positive sums of concave nondecreasing functions, hence they are also concave and nondecreasing. Therefore,

$$C_{P_a}(x_a) \geqq \frac{x_a}{r_{st}} \cdot C_{P_a}(r_{st}) + \frac{x_b}{r_{st}} \cdot C_{P_a}(0)$$

$$C_{P_b}(x_b) \geqq \frac{x_b}{r_{st}} \cdot C_{P_b}(r_{st}) + \frac{x_a}{r_{st}} \cdot C_{P_b}(0) \quad \text{because} \quad r_{st} = x_a + x_b$$

and

(3.2) $$C_{P_a}(x_a) + C_{P_b}(x_b) \geqq \frac{x_a}{r_{st}} \cdot C_{P_a}(r_{st}) + \frac{x_b}{r_{st}} \cdot C_{P_b}(r_{st})$$

since $C_{P_a}(0)$ and $C_{P_b}(0)$ are zero. The scalars $C_{P_a}(r_{st})$ and $C_{P_b}(r_{st})$ can have one of the following relations:

(i) $C_{P_a}(r_{st}) = C_{P_b}(r_{st})$

(ii) $C_{P_a}(r_{st}) > C_{P_b}(r_{st})$

(iii) $C_{P_a}(r_{st}) < C_{P_b}(r_{st})$.

If (i) occurs then putting the entire r_{st} flow in P_a or P_b results in no cost increase by the inequality (3.2).

If (ii) occurs, then putting the entire flow in P_b results in less cost because

$$C_{P_a}(x_a) + C_{P_b}(x_b) \geqq \frac{x_a}{r_{st}} \cdot C_{P_a}(r_{st}) + \frac{x_b}{r_{st}} \cdot C_{P_b}(r_{st}) > C_{P_b}(r_{st}) .$$

And if (iii) occurs, then putting the entire flow in P_a results in less cost because

$$C_{P_a}(x_a) + C_{P_b}(x_b) \geqq \frac{x_a}{r_{st}} \cdot C_{P_a}(r_{st}) + \frac{x_b}{r_{st}} \cdot C_{P_a}(r_{st}) > C_{P_a}(r_{st}) .$$

Thus the entire r_{st} flow can be put into one of the chains without any cost increase.||

Corollary 3.1:

If there are multiple (more than one) chains between a source-sink pair, they can be replaced by a single chain (one of the multiple chains) without increasing the total cost.

Notation:

$p_{ab} \subset p_{ij}$ means the arcs of the chain p_{ab} between a and b is a subset of the arcs of the chain p_{ij} between i and j .

Proposition 3.2:

In the optimal flow pattern, if the optimal chain p^*_{ij} between source i and sink j passes through nodes a and b , and there is also a flow requirement between a and b , then the optimal chain p^*_{ab} between a and b is such that $p^*_{ab} \subset p^*_{ij}$.

Proof:

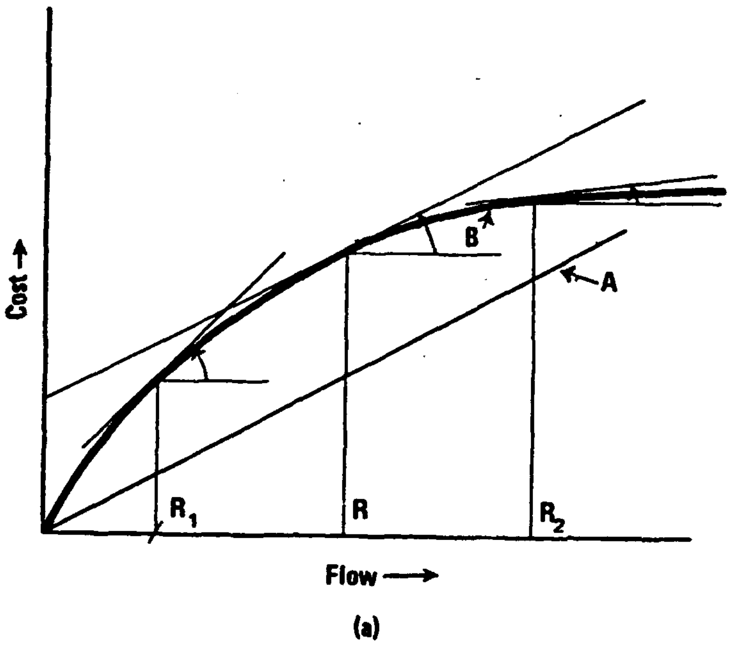
Let r_{ij} be the required flow between i and j and r_{ab} be the required flow between a and b. Suppose a precedes b in the chain from i to j. The flow pattern is unchanged if we consider requirements $r'_{ia} = r_{ij}$, $r'_{ab} = r_{ij} + r_{ab}$, and $r'_{bj} = r_{ij}$. Since there is a single chain between i and a, a and b, and b and j, chain ab coincides with chain ij between a and b, and thus $p^*_{ab} \subset p^*_{ij}$. If b precedes a in the chain from i to j then flow pattern is unchanged if flows $r'_{ib} = r_{ij}$, $r'_{ab} = r_{ij} + r_{ab}$, and $r'_{aj} = r_{ij}$. Since there is a single chain between i and b, b and a, and a and j, $p^*_{ab} \subset p^*_{ij}$. ||

Thus we can restrict our search for the optimal flow network to solutions which have the following two properties: (i) a single chain between each source and sink pair, and (ii) $p_{ab} \subset p_{ij}$ if a source and sink pair ab is contained in the chain of another source and sink pair ij.

3.1 Definition of ϵ-Optimal Routing

A routing (flow pattern) is called ϵ-optimal if it has the above two properties, and if the least cost chain for an additional flow of ϵ units (where ϵ is an arbitrarily small positive real number) between source-sink pair ij is the same as the chain taken by the ij flow of r_{ij} units.

The following example will clarify the concept of ϵ-optimality. For the network shown in Figure 3.1b, the costs of chains A and B are shown in Figure 3.1a by curves A and B respectively. R is the point where the tangent of curve B has the same slope as A. Suppose



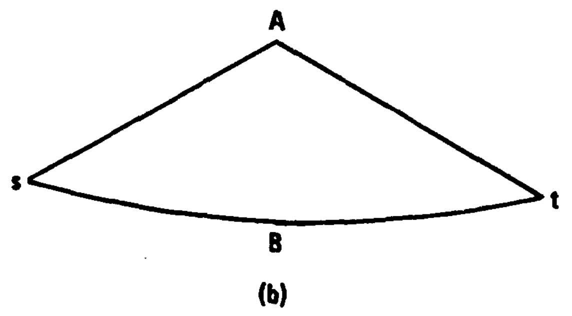


FIGURE 3.1

$r_{st} = R_1 \leqq R$ and the entire flow is in chain B . For an additional flow of amount ϵ the cheapest chain is A because the slope at 0 of A is smaller than the slope at R_1 of B . Thus chain B is not ϵ-optimal. But if the entire flow is in A then an additional flow of amount ϵ will take chain A because the slope at 0 of B is greater than the slope at R_1 of A , and chain A is ϵ-optimal. It is also globally optimal. Suppose $r_{st} = R_2 \geqq R$, and the entire flow is in B . An additional flow of amount ϵ will take path B because the slope at R_2 of B is smaller than the slope at 0 of A . Similarly, if the entire flow is in A , then an additional flow of amount ϵ will take chain A because the slope at 0 of chain B is greater than the slope at R_2 of chain A . So, both A and B are ϵ-optimal chains. However, only A is globally optimal.

The algorithm discussed below searches for an ϵ-optimal point. (The relationship between ϵ-optimality and local optimality is discussed later.) The concept of ϵ-optimality reveals a clue to the constructive approach (of the algorithm) which the conventional definition of the local optimum does not provide.

The following notations are used in the subsequent discussions. P is an M-vector where the element p_m is the amount of flow in arc m corresponding to a feasible flow pattern (i.e., a pattern satisfying (1.1) of Chapter 1). e is an M-vector of 0's and 1's. If $e_m = 1$ indicates the inclusion of arc m and $e_m = 0$ indicates its exclusion, then any chain between any source-sink pair can be represented by a vector e . C(P) is the total cost of the flow on all arcs corresponding to point P $\left(\text{i.e., } C(P) = \sum_{m=1}^{M} f_m(p_m)\right)$. δ is a scalar and $(P + \delta \cdot e^i - \delta \cdot e^j)$ is a point where δ amount of flow is taken out from chain e^i of point P and redirected to chain e^j . $F_e(P)$ represents the total cost of all the arcs in chain e of point P $\left(\text{i.e.,} \sum_{m \in \text{chain } e} f_m(p_m)\right)$, and it is a concave function of P .

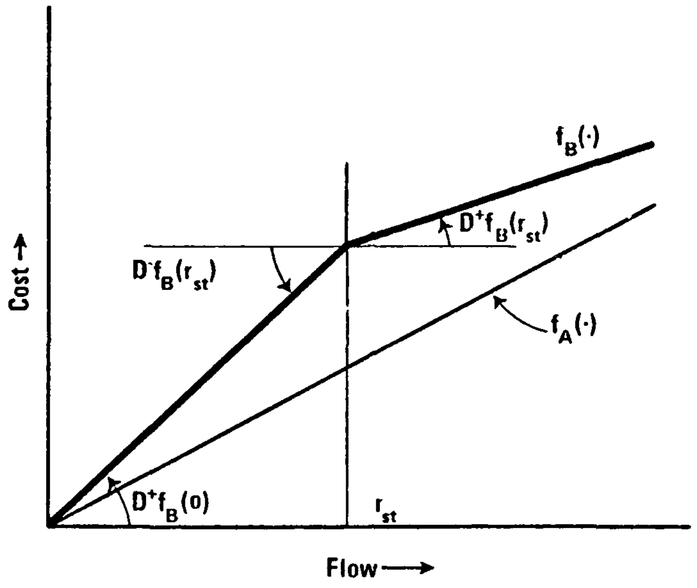
The following small example of Figure 3.2 shows a point can be ϵ-optimal but not local optimal. Figure 3.2a gives the cost function $f_A(\cdot)$ and $f_B(\cdot)$ for the flow in chains A and B respectively of the network 3.2b between source-sink pair st with flow requirements r_{st} . Suppose the entire flow is in chain B . And the left and right-hand derivatives of cost functions in their present flow levels have the following relation: $D^-f_B(r_{st}) > D^+f_A(0) > D^+f_B(r_{st})$. Then an additional flow amount ϵ will take the chain B because $D^+f_A(0) > D^+f_B(r_{st})$. Hence the present flow pattern is ϵ-optimal. But if ϵ amount of flow is displaced from B to A then total cost decreases because $\epsilon \cdot D^-f_B(r_{st}) > \epsilon \cdot D^+f_A(0)$. So, it is not locally optimal point.

Lemma 3.1:

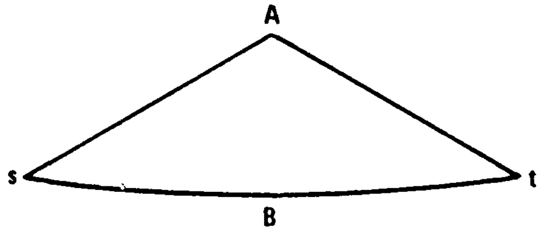
For a concave cost function a local optimal point having a single chain between each source-sink pair is an ϵ-optimal point. An ϵ-optimal point is a local optimal point only if the single path e^* between every source-sink pair also satisfies the following inequality

(3.3) $$D^-F_{e^*}(P) \leqq D^+F_e(P) \quad \forall \quad e$$

where e connects the same source-sink pair as e^* . (This inequality is satisfied by any ϵ-optimal point for problems with concave cost functions where derivatives exist at all points. Hence in this case a point is a local optimum if and only if it is ϵ-optimal.)



(a)



(b)

FIGURE 3.2

Proof:

Consider a locally optimal point P having a single chain between each source-sink pair. Consider a point Q within ϵ-neighborhood of P. Q is obtained from P by redirecting some $\delta(\leq \epsilon)$ amount of flow between a source-sink pair from the single chain e^* to some other chain e. (There could be more than one chain and could be more than one source and sink pairs. But these cases of more than one different chain can be reduced to only one different chain between a source-sink pair without any cost increase by Corollary 3.1.) $Q = P + \delta\cdot e - \delta\cdot e^*$. Let $\bar{e}^* = e^* - e^* \cap e$, and $F_{\bar{e}^*}$ and $F_{e^* \cap e}$ be defined as cost of flows in the set of arcs in $\bar{e}^*$ and $e^* \cap e$ respectively.

$$C(P + \delta\cdot e^*) = C(P) + F_{e^*}(P + \delta\cdot e^*) - F_{e^*}(P)$$

$$= C(P) + F_{\bar{e}^*}(P + \delta\cdot e^*) + F_{e^* \cap e}(P + \delta\cdot e^*) - F_{\bar{e}^*}(P) - F_{e^* \cap e}(P)$$

$$C(P + \delta\cdot e) = C(P + \delta\cdot e - \delta\cdot e^* + \delta\cdot e^*) = C(Q + \delta\cdot e^*)$$

$$= C(Q) + F_{\bar{e}^*}(Q + \delta\cdot e^*) + F_{e^* \cap e}(Q + \delta\cdot e^*) - F_{\bar{e}^*}(Q) - F_{e^* \cap e}(Q)$$

$$= C(Q) + F_{\bar{e}^*}(P) + F_{e^* \cap e}(P + \delta\cdot e^*) - F_{\bar{e}^*}(P - \delta\cdot e^*) - F_{e^* \cap e}(P) .$$

The cost value $F_{\bar{e}^*}(Q + \delta\cdot e^*)$ depends on cost of flow in the arcs in the set $\bar{e}^*$. At Q the flows are reduced from the level at P by δ on this set $\bar{e}^*$, and when δ is added back the level of flow in these arcs is the same as at P, so $F_{\bar{e}^*}(Q + \delta\cdot e^*) = F_{\bar{e}^*}(P)$. Similarly, we can show $F_{e^* \cap e}(Q + \delta\cdot e^*) = F_{e^* \cap e}(P + \delta\cdot e^*)$, $F_{\bar{e}^*}(Q) = F_{\bar{e}^*}(P - \delta\cdot e^*)$, and $F_{e^* \cap e}(Q) = F_{e^* \cap e}(P)$.

Now, $C(P + \delta\cdot e) - C(P + \delta\cdot e^*) = C(Q) - C(P) - 2\Big(F_{e^*}(P) - \frac{1}{2}\cdot F_{e^*}(P + \delta\cdot e^*) - \frac{1}{2}\cdot F_{e^*}(P - \delta\cdot e^*)\Big) \geqq C(Q) - C(P)$ (because $F_{e^*}(\cdot)$ is a concave function) $\geqq 0$, because P is a local optimal point. So,

$$C(P + \delta\cdot e^*) \leqq C(P + \delta\cdot e)$$

or,

$$\{C(P + \delta\cdot e^*) - C(P)\} \leqq \{C(P + \delta\cdot e) - C(P)\} .$$

Hence, extra $\delta(\leqq \epsilon)$ flow is passed through the optimal single chain. So, P is an ϵ-optimal point.

Let the ϵ-optimal point P satisfy the inequality (3.3). Consider a point Q in ϵ-neighborhood of P such that $Q = P + \delta\cdot e - \delta\cdot e^*$ where $\delta(\leqq \epsilon)$.

$$C(Q) = C(P + \delta\cdot e - \delta\cdot e^*) = C(P) + \delta\cdot D^+F_e(P) - \delta\cdot D^-F_{e^*}(P) \geqq C(P)$$

(because (3.3) is satisfied). Any point Q in this neighborhood will have the above relationship, so P is a local optimal point.

A point P is ϵ-optimal if the following inequality is true for $\forall$ e

$$\{C(P + \delta\cdot e^*) - C(P)\} \leqq \{C(P + \delta\cdot e) - C(P)\}$$

or,

$$\delta\cdot D^+F_{e^*}(P) \leqq \delta\cdot D^+F_e(P) \quad \Big(\text{because } C(P + \delta\cdot e^*) = C(P) + \delta\cdot D^+F_{e^*}(P)\Big)$$

or, $D^+F_{e^*}(P) \leqq D^+F_e(P)$ $\forall$ e . If the derivative exists at all points then $D^-F_{e^*}(P) = D^+F_{e^*}(P) \leqq D^+F_e(P)$. So, the ϵ-optimal point also satisfies the inequality (3.3), and therefore is a local optimal point. Thus for this case a point is ϵ-optimal if and only if it is a local optimal point.

The example of Figure 3.2 shows a counterexample if the inequality does not hold.||

3.2 Algorithm

In the description of the algorithm the derivative $Df(\cdot)$ of the cost function at a point is used. If this derivative does not exist then the right-hand derivative $D^+f(\cdot)$ is used.

Step 0:

An arbitrary positive number representing length is assigned to each arc of the network (e.g., Euclidean distance. Or an arbitrary flow level is assigned for each arc and the derivative of the cost function at the flow level is used as the length of the arc).

Step 1:

Using the lengths specified, the shortest chain is determined between each source-sink pair (if there are many pairs of sources and sinks, Floyd's algorithm is used).

Step 2:

The entire required flow between a source-sink pair is passed through the shortest chain between them. The total flow y_m in each arc m is determined. The total cost of the flow is determined. If the total cost is unchanged from the previous iteration the procedure is stopped. (Note: If the flow in each arc remains unchanged in two consecutive iterations, then the shortest chain and the total cost also remain the same. However, the flow might change between two degenerate points without having any change in the total cost. Cycling may occur if we use only the flow value to find the termination point. Such cycling is not possible if the cost function is strictly concave.)

Step 3:

Derivatives of the cost function, $Df_m(y_m)$, at the flow values y_m , are determined for each arc. $D^+f_m(y_m)$ for functions having no derivative at y_m . These derivatives are used as the new lengths of the arcs. Go to Step 1.

3.3 Remarks

When the algorithm stops, then the specified chain p^*_{ij} between a source-sink pair (ij) satisfies the inequality $\sum_{m\epsilon p^*_{ij}} Df_m(y_m) \leqq \sum_{m\epsilon p_{ij}} Df_m(y_m) \ \forall \ p_{ij}$ chain, assuming derivatives exist. Now, $Df_m(y_m) = D^+f_m(y_m) = D^-f_m(y_m)$. The inequality $\sum_{m\epsilon p^*_{ij}} D^+f_m(y_m) \leqq \sum_{m\epsilon p_{ij}} D^+f_m(y_m)$ is true even if $Df_m(y_m)$ does not exist. Therefore, the point is an ϵ-optimal point $\left(\right.$from the proof of Lemma 3.1, where $D^+F_{e^*}(P)$ is the same as $\sum_{m\epsilon p^*_{ij}} D^+f_m(y_m)$ and y_m represents point P and $\left. p^*_{ij} = e^*\right)$. The inequality (3.1) also is satisfied if $Df_m(y_m)$ exists; in that case the point is a local optimum. However, a post-optimization procedure is necessary to obtain a local optimal point for functions whose derivatives do not exist. Take each source-sink pair and change the lengths of arcs along the optimal single chain from $D^+f_m(y_m)$ to $D^-f_m(y_m)$. If the shortest route remains unchanged, then (3.3) is satisfied and the point is a local optimum. If the shortest chain changes, then an improvement (reduction) in total cost can be achieved by using this new shortest chain. The steps of the algorithm are repeated using this new point. Cycling is not possible here, because, if a different shortest chain is obtained in this procedure, the total cost strictly decreases when the flow is redirected.

The motivation for using derivatives of the cost function is the following Kuhn-Tucker necessary condition for optimality for differentiable cost function (i.e., $Df(\cdot)$ exists). The unrestricted minimization problem, equivalent to the problem (1.1) with dual variables λ_{ij} for constraints (1.1b) and β_{ij}^k for nonnegativity constraints (1.1c) is given by:

$$L = \sum_{m=1}^{M} f_m(y_m) - \sum_{\substack{\text{All } ij \\ \ni i>j}} \lambda_{ij} \left(\sum_{k=1}^{P_{ij}} x_{ij}^k - r_{ij} \right) - \sum_{\substack{\text{All } ij \\ \ni i>j}} \sum_{k=1}^{P_{ij}} \beta_{ij}^k x_{ij}^k \ .$$

Since each variable y_m is a linear combination of x_{ij}^k variables with coefficients 0 or 1,

$$\frac{\partial L}{\partial x_{ij}^k} = \sum_{m \varepsilon p_{ij}^k} Df_m(y_m) - \lambda_{ij} - \beta_{ij}^k = 0 \ .$$

The above equation and the Kuhn-Tucker necessary condition for optimality gives the following conditions.

$$\text{If } x_{ij}^k > 0 \text{ then } \beta_{ij}^k = 0 \text{ and } \lambda_{ij} = \sum_{m \varepsilon p_{ij}^k} Df_m(y_m) \text{ and}$$

$$\text{if } x_{ij}^k = 0 \text{ then } \beta_{ij}^k = \sum_{m \varepsilon p_{ij}^k} Df_m(y_m) - \lambda_{ij} \geqq 0 \ .$$

The above procedure satisfies these conditions. λ_{ij} is the length of the shortest chain between ij .

The following theorem establishes the finite convergence of the procedure. In practice a very quick convergence can be obtained.

Convergence Theorem 3.1:

For concave nondecreasing cost functions the algorithm described above converges to an ϵ-optimal point in a finite number of steps.

Proof:

We have already shown that if the above algorithm terminates, it does so at an ϵ-optimal point. Also we have shown that each iteration of the procedure involves solving a shortest chain problem where the initial distance matrix has nonnegative elements. Hence, the number of steps in each iteration is finite ($\sim N^3$ where the number of nodes is N). So, finite convergence of the algorithm is achieved if the number of iterations is finite. This is established by proving Property (a) below for any nondecreasing concave cost function.

Property (a):

If the algorithm generates two consecutive distinct points X^ℓ and $X^{\ell+1}$ ($X^{\ell+1}$ is obtained by one iteration starting from X^ℓ) then the total cost $Z(X)$ strictly decreases, i.e., $Z(X^\ell) > Z(X^{\ell+1})$. This is proved as follows.

Let the flow values in each arc m be y_m^ℓ and $y_m^{\ell+1}$ corresponding to two consecutive points, X^ℓ and $X^{\ell+1}$ (the feasible region in the chain space). $X^{\ell+1}$ is obtained from X^ℓ by the change of a certain set of flow chains between source-sink pairs. Let one such chain be changed from p_{ij} to p_{ij}^* . Since p_{ij}^* is the shortest chain using y_m flows in each arc m :

$$\sum_{m\epsilon p_{ij}} D^+f_m\left(y_m^\ell\right) \geqq \sum_{m\epsilon p_{ij}^*} D^+f_m\left(y_m^\ell\right)$$ (If the derivative $Df_m\left(y_m^\ell\right)$ exists, then the inequality is also true when $D^+f_m\left(y_m^\ell\right)$ is replaced by $Df_m\left(y_m^\ell\right)$).

If ϵ amount of flow is redirected from p_{ij} to p_{ij}^* then the resulting saving is:

$$\epsilon\cdot\left\{\sum_{m\epsilon \bar{p}_{ij}} D^-f_m\left(y_m^\ell\right) - \sum_{m\epsilon \bar{p}_{ij}^*} D^+f_m\left(y_m^\ell\right)\right\} \geqq 0$$

(because from concavity $D^-f_m\left(y_m^\ell\right) \geqq D^+f_m\left(y_m^\ell\right)$ where $\bar{p}_{ij} = p_{ij} - p_{ij} \cap p_{ij}^*$ and $\bar{p}_{ij}^* = p_{ij}^* - p_{ij} \cap p_{ij}^*$) . Also from concavity

$$\sum_{m\epsilon \bar{p}_{ij}} D^+f_m\left(y_m^\ell - \epsilon\right) \geqq \sum_{m\epsilon \bar{p}_{ij}} D^+f_m\left(y_m^\ell\right) \geqq \sum_{m\epsilon \bar{p}_{ij}^*} D^+f_m\left(y_m^\ell\right) \geqq \sum_{m\epsilon \bar{p}_{ij}^*} D^+f_m\left(y_m^\ell + \epsilon\right) .$$

So, a further saving is obtained if a further ϵ amount is redirected, and the maximum saving is obtained if entire flow r_{ij} is redirected from p_{ij} to p_{ij}^* . A similar redirecting of other flows through the shortest chains reduces the total cost value and flow values $y_m^{\ell+1}$ are obtained, hence $Z(X^\ell) \geqq Z(X^{\ell+1})$. But equality is only achieved when there is more than one shortest chain and cost functions are flat in the region involved in the transfer of flows or when flow values remain unchanged. In both cases the algorithm stops at X^ℓ . So, if X^ℓ and $X^{\ell+1}$ are distinct points then $Z(X^\ell) > Z(X^{\ell+1})$.

But in the algorithm described above the sequence generated by $\{X^\ell\}$ is the extreme points of the polyhedral set defined by (1.1b) and (1.1c). Since the polyhedral set has a finite number of hyperplanes it has a finite number of extreme points. Nondecreasing Property (a) guarantees that no two elements of X^ℓ correspond to the same extreme point. Hence, the number of elements in any sequence generated by the iterations of the algorithm is finite. Thus finite convergence is always achieved.||

3.4 Modified Procedures for Differentiable Cost Functions

For functions having derivatives at all points (i.e., differentiable), all ϵ-optimal points are local optimal points. In this case the following modified procedures are sometimes used to avoid some local optimal points and get near the global optimum. (i) In 1st modified procedure if a point $X^{\ell+1}$ is obtained starting from a point $\bar{X}^{\ell}$ then the next iteration is started from a point $\bar{X}^{\ell+1} = \beta\bar{X}^{\ell} + (1 - \beta)X^{\ell+1} (0 \leq \beta \leq 1)$. However, Property (a) may not be preserved. Similar arguments as in the proof of Property (a) can show $Z(\bar{X}^{\ell}) > Z(X^{\ell+1})$, but not necessarily $Z(\bar{X}^{\ell}) > Z(\bar{X}^{\ell+1})$. However, if β is chosen properly in each step to maintain monotonicity then only convergence Theorem 3.2 below is applicable. (ii) In 2nd modified procedure a convex combination of cost is used for new starting point $\left(\text{i.e., here cost on an arc } m = \beta \cdot Df_m(\bar{X}^{\ell}) + (1 - \beta)\cdot Df_m(X^{\ell+1})\right)$ similar arguments as in the proof of Property (a) can show $Z(\bar{X}^{\ell}) > Z(X^{\ell+1})$. Now, $Z(\bar{X}^{\ell+1}) = \beta \cdot Z(\bar{X}^{\ell}) + (1 - \beta)\cdot Z(X^{\ell+1}) < \beta \cdot Z(\bar{X}^{\ell}) + (1 - \beta)\cdot Z(\bar{X}^{\ell}) = Z(\bar{X}^{\ell})$. So, Property (a) is true here.

Convergence Theorem 3.2:

For differentiable concave nondecreasing cost functions the modified (i.e., 2nd modified procedure and a special case of 1st modified procedure where β is such that Property (a) is satisfied) algorithm converges to an ϵ-optimal (also local optimal) point.

Proof:

Here termination procedure is same as main Algorithm 3.2. So, if the algorithm terminates it does so at ϵ-optimal point. Since the cost functions are differentiable, ϵ-optimal point is also a local optimal point (by Lemma 3.1). The shortest path procedures in each iteration are finite. The sequence $\{X^\ell\}$ generated does not necessarily have finite number of elements. However, convergence (not necessarily finite) is established by proving the following Properties[2] (b) and (c) in addition to Property (a). Consider the algorithm to be a mapping M which maps from a feasible region of flow X to X itself.

<u>Property (b)</u>:

The feasible set X or, at least the subset in which the sequence $\{X^\ell\}$ generated by the algorithm lies, is compact.

<u>Property (c)</u>:

The map M is closed.[3] The properties are proved as follows.

(b) The feasible region in X-space is defined by the linear inequalities in (1.1b) and (1.1c), hence they form a convex set which is closed but which could be unbounded. In the solution procedure only the required amount of flows r_{ij} is sent through the shortest chain. Thus, the flow values on any chain in a point X^ℓ in the sequence generated by the algorithm, is bounded by $r_{st} = \underset{ij}{\text{Max}} \{r_{ij}\}$. The set is bounded (since each element of X is bounded). The set is closed and bounded, and thus compact.

(c) The closedness property of map M will be proved only for functions with derivatives, which is the case here. X and θ are two

---

[2] Properties (a), (b) and (c) are needed by Zangwill's theorem on algorithmic convergence (Page 91, Ref. Z-2). This theorem is used to prove the convergence of the map M defined below.

[3] Closedness is a property of a point to set map. Definition: The closed map M is such that $X^\ell \to X^o$, $\theta^\ell \in M(X^\ell)$, and $\theta^\ell \to \theta^o$ implies $\theta^o \in M(X^o)$.

notations used to define a feasible vector in chain-space. The polyhedron S defined in Chapter 1 for X is also defined for θ, i.e.,

$$S = \left\{ \theta = \left\{\theta^k_{ij}\right\} \;\middle|\; \sum_{k=1}^{P_{ij}} \theta^k_{ij} \geqq r_{ij} \;\; \forall \text{ source-sink pair } ij \;,\; \theta^k_{ij} \geqq 0 \right\} .$$

If there are n source-sink pairs with positive requirements, then only n elements of X or θ corresponding to an extreme point are positive. The process of finding the shortest chain by using the derivative of the cost function at the flow value on each arc at X^ℓ, corresponds to a local search operation to find a vector θ which minimizes the linear approximation of $Z(X)$ in the vicinity of X^ℓ, i.e.,

Minimize $\quad Z(X^\ell) + DZ(X^\ell)(\theta - X^\ell)$

Subject to $\theta \in S$, where $DZ(X^\ell)$ is the derivative of $Z(X)$ at X^ℓ.

Since X is constant, the above problem is the same as the linear program:

Minimize $\quad DZ(X^\ell)\cdot\theta$

Subject to $\theta \in S$.

If an extreme point θ^ℓ solves this problem, then θ^ℓ defines a new set of shortest chains. The direction vector $d^\ell = \theta^\ell - X^\ell$ defines the direction of improvement. The procedure of finding a point X of maximum improvement of the cost function in this direction is to find $X = X^\ell + \tau\cdot d^\ell$ which minimizes $Z(X^\ell + \tau\cdot d^\ell)$ subject to $0 \leq \tau \leq 1$. Since, $X = X^\ell + \tau\cdot d^\ell = (1 - \tau)X + \tau\cdot\theta^\ell$, the vector X is obtained from X^ℓ by transfer of $(\tau\cdot r_{ij})$ flow from the path corresponding to X^ℓ to the path corresponding to θ^ℓ for various ij. Because of the concavity (as discussed before) the maximum improvement occurs when $\tau = 1$. So,

$X^{\ell+1} = X^{\ell} + d^{\ell} = \theta^{\ell}$ and the map[4] $M(X)$ is defined as:

$$M(X) = \left\{\theta;\ \underset{\theta'\varepsilon S}{\text{Minimize}}\ DZ(X)\cdot\theta' = DZ(X)\cdot\theta\right\} .$$

To show $M(X)$ is a closed map one has to check: Conditions $X^{\ell} \to X^{o}$, and $\theta^{\ell} \to \theta^{o}$ where $\theta^{\ell}\ \varepsilon\ M(X^{\ell}) \Rightarrow \theta^{o}\ \varepsilon\ M(X^{o})$. This is equivalent to showing $\underset{\theta'\varepsilon S}{\text{Minimum}}\ DZ(X^{o})\cdot\theta' = DZ(X^{o})\cdot\theta^{o}$. Consider the inequality:

$$|DZ(X^{\ell})\cdot\theta' - DZ(X^{o})\cdot\theta'| \leqq |DZ(X^{\ell}) - DZ(X^{o})|\cdot|\theta'| \leqq \delta_{\ell}\cdot c$$

where $|\theta'| \leqq c$ because of the compactness of the feasible set and $\delta_{\ell} \to 0$ as $X^{\ell} \to X^{o}$.

Since the difference is uniformly bounded in θ' by $\delta_{\ell}\cdot c$
$|\underset{\theta'}{\text{Min}}\ DZ(X^{\ell})\cdot\theta' - \underset{\theta'}{\text{Min}}\ DZ(X^{o})\cdot\theta'| \leqq \delta_{\ell}\cdot c$ or,

$$|DZ(X^{\ell})\cdot\theta^{\ell} - \underset{\theta'}{\text{Min}}\ DZ(X^{o})\cdot\theta'| \leqq \delta_{\ell}\cdot c\ , \text{ (because } \theta^{\ell}\ \varepsilon\ M(X^{\ell})) .$$

aking the limit over ℓ , c being a finite number

$$|DZ(X^{o})\cdot\theta^{o} - \underset{\theta'}{\text{Min}}\ DZ(X^{o})\cdot\theta'| \leqq 0 \Rightarrow$$

$$DZ(X^{o})\cdot\theta^{o} = \underset{\theta'}{\text{Min}}\ DZ(X^{o})\cdot\theta' \Rightarrow \theta^{o}\ \varepsilon\ M(X^{o})$$

$$\Rightarrow M \text{ is a closed map.}$$

Properties (a), (b) and (c) fulfill all the conditions of Zangwill's heorem on algorithmic convergence. Here the solution set is the set of

The map M is described over the continuous variable X . However, it uses only extreme points. Hence, it goes through only discrete points of the set and corresponds to an iteration of the algorithm.

ϵ-optimal points, so either the algorithm stops at a solution or the limit of any convergent subsequence is a solution.||

3.5 Strategies to get Near the Global Optimal Point

The major drawback of this procedure is that it terminates at local optimal points and we have no idea how much less the global optimal cost might be. We have tested computationally various heuristic approaches to get near the global optimal point. The different approaches are as follows. Approach (a) is suggested by B . Yaged, and he tested it emperically. Approaches (b) and (c) are new.

(a) Using a Convex Combination of Flow Vectors or Cost Vectors

If X^ℓ and $X^{\ell+1}$ are extreme flow vectors in the ℓth and $(\ell + 1)$st iteration, then the $(\ell + 2)$nd iteration is started from $\bar{X}^{\ell+2} = \beta X^\ell + (1 - \beta)X^{\ell+1}$ where $0 \leqq \beta \leqq 1$. When β is near 1, the change of flow is small between two iterations. Sometimes this approach helps in climbing out of the valleys of the local optimals to get at the global optimum. There is no one β which gives good results. The test on different problems shows that usually the number of iterations increases when such a convex combination is used.

Sometimes a combination of marginal costs is used instead of a combination of flows; i.e., cost on arc $m = \beta \cdot Df_m(X^\ell) + (1 - \beta) \cdot Df_m(X^{\ell+1})$ is used. Bernard Yaged used different β values to get better local optimals and he reported an acceptable range of β in the interval (.8, 1.2). The use of this over-relaxation procedure (i.e., $\beta \geqq 1$) may be advantageous for some problem, but in some problems if a $\beta > 1$ is used, then some elements of the distance matrix may be negative. So some special precaution is needed to solve the shortest chain problem.

Sometimes instead of using marginal cost (i.e., $Df_m(y_m)$), average cost is used $\left(\text{i.e., } \frac{f_m(y_m)}{y_m}\right)$. Here the proof of convergence of the algorithm may not be established. However, use of the average cost value initially is sometimes helpful in getting a good starting value for using marginal cost, particularly when the cost function has a jump at 0. Sometimes a combination of average and marginal cost gives good results.

Each of the methods have been tested computationally for different problems, but it is difficult to specify which works where.

(b) Specializations of Step 0 of the Algorithm, a Systematic Search for Better Starting Points

Different strategies are employed to get better starting points. (i) Initially a local optimum is obtained using the algorithm above - then to get a new starting point for the algorithm, all the arcs in which the flow value is lower than a certain amount are made very expensive - this results in using the arcs whose flow value is large more effectively and not using arcs with low flow values. (ii) To search in a more spread out area, arcs having positive flow values for one local optimal point are sometimes blocked to get a completely different local optimum.

The procedures (i) and (ii) are utilized around the best local optimal value obtained so far, to determine whether it is better to search in the vicinity of the best local optimal value (i.e., procedure (i)) or farthest away from the best optimal value (by procedure (ii)).

From the test results it is recommended that procedure (ii) should be used initially to get a few very different local optimal points. Then procedure (i) should be used to search near the best local optimal value so far obtained.

(c) Random Starting Point Selection Strategy and a Sequential Sampling Plan

In this method (a similar method is suggested for travelling salesman problem in Ref. [R-2]) the global optimal cost is estimated from a series of observed local optimal costs. The local optimal costs are generated by selecting random starting points (not necessarily within the feasible region) for the above algorithm. Suppose that a sample S of size n of local optimal points having values $c_1, c_2, \ldots, c_n$ has been determined. We shall give a procedure for estimating the global optimal cost given this information.

We have no *a priori* knowledge of the distribution of local optimal costs, so it is assumed to be uniform between α and $\alpha + \theta$. Given $c_1, \ldots, c_n$ we wish to estimate α and θ. (This uniformity assumption in the case of a total lack of knowledge has the approval of both of the Bayesian schools [R-1], i.e., necessarists (Jeffreys [J-2]) and subjectivists (Savage [E-2, D-1]). Jeffreys argued for the legitimacy of using a uniform distribution in case of a total lack of information by quoting that "Bayes, in his great memoir, repeatedly says that the principal (i.e., assigning equal probability or assumption of uniform distribution) is to be used only in case where we have no grounds for choosing between the alternatives." Edwards, Lindman and Savage have shown in a theorem that under the assumption of a uniform distribution of parameters, the calculated approximate *a posteriori* distribution agrees closely with the actual *a posteriori* distribution. (Here the uniform distribution is assumed of the parameter rather than of the distribution function itself. However, we can think of the distribution function itself as a parameter.) So, the basis of the assumption is philosophically coherent with the Bayesian approach.)

The estimate of α is the estimate of the global optimal solution. The elements of sample S are independent, and each is from a uniform density function between α and $\alpha + \theta$. So, the likelihood function L is defined as the following function of the sample S , α and θ :

$$L(S;\alpha,\theta) = \frac{1}{\theta^n} \text{ if } \alpha \leqq c_i \leqq \alpha + \theta$$

$$= 0 \text{ , otherwise.}$$

So, the maximum likelihood estimates[5] of α and θ are respectively minimum $(c_1, \ldots, c_n)$, and {maximum $(c_1, \ldots, c_n)$ - minimum $(c_1, \ldots, c_n)$} . Let T and U be respectively, minimum and maximum elements of the sample S . Then the distribution functions of T and U , F(t) and G(u) are as follows:

$$1 - F(t) = \text{Prob } (T > t) = \left(\frac{\alpha + \theta - t}{\theta}\right)^n \quad \text{for} \quad \alpha \leqq t \leqq \alpha + \theta$$

$$\text{density function} \quad f(t) = \frac{dF(t)}{dt} = \frac{n}{\theta^n} (\alpha + \theta - t)^{n-1} \quad \text{for} \quad \alpha \leqq t \leqq \alpha + \theta$$

$$= 0 \text{ , otherwise.}$$

Expected values

[5]Definition: The maximum likelihood estimate of θ based on a random sample $S = (x_1, \ldots, x_n)$ is that value of θ which minimizes $L(S;\theta)$ $(= f(x_1,\theta) \cdot \ldots \cdot f(x_n,\theta)$ where $f(x_i,\theta)$ is the density function of x_i with parameter $\theta)$ considered as a function of θ for a given sample S .

$$E(T-\alpha) = \int_{\alpha}^{\alpha+\theta} (t-\alpha)f(t)\,dt = \int_{\alpha}^{\alpha+\theta} \frac{(t-\alpha)}{\theta}\, n\left(\frac{\theta-(t-\alpha)}{\theta}\right)^{n-1} dt =$$

$$= n\theta\int_{0}^{1} x(1-x)^{n-1}\,dx$$

$$= n\theta\cdot\beta(2,n) = \frac{\theta}{n+1}$$

where $x = \frac{t-\alpha}{\theta}$, $\beta(2,n)$ is Beta function with parameters 2 and n and $\beta(a,b) = \frac{\Gamma a\,\Gamma b}{\Gamma(a+b)}$, and $\Gamma a = a\cdot\Gamma(a-1)$. (Γ denotes a Gamma function.)

$$G(u) = \text{Prob}\ (U \leqq u) = \left(\frac{u-\alpha}{\theta}\right)^{n} \quad \text{for} \quad \alpha \leqq u \leqq \alpha+\theta$$

$$\text{density function : } g(u) = \frac{dG(u)}{du} = \frac{n}{\theta^{n}}\cdot(u-\alpha)^{n-1} \quad \text{for} \quad \alpha \leqq u \leqq \alpha+\theta$$

$$= 0 \text{ , otherwise}$$

$$E(U-\alpha) = \int_{\alpha}^{\alpha+\theta} (u-\alpha)g(u)\,du = \int_{\alpha}^{\alpha+\theta} (u-\alpha)\,\frac{n}{\theta^{n}}\,(u-\alpha)^{n-1}du =$$

$$= \int_{\alpha}^{\alpha+\theta} \left(\frac{u-\alpha}{\theta}\right)^{n}\theta\, n\,\frac{du}{\theta}$$

$$= n\theta\int_{0}^{1} x^{n}\,dx = \frac{n}{n+1}\ , \left(\text{where} \quad x = \frac{u-\alpha}{\theta}\right)$$

$$E(U-T) = \left(\frac{n}{n+1} - \frac{1}{n+1}\right)\cdot\theta = \frac{n-1}{n+1}\cdot\theta \ ; \text{ or, } \quad E\left(\frac{U-T}{n-1}\right) = \frac{\theta}{n+1}$$

$$E(T) = \alpha + \frac{\theta}{n+1} = \alpha + E\left(\frac{U-T}{n-1}\right)\ , \text{ so, } \quad E\left(T - \frac{U-T}{n-1}\right) = \alpha\ .$$

Hence, $\left(T - \frac{U-T}{n-1}\right)$ is an unbiased estimator of α . The variance of this estimator gives an idea of how good this estimator is.

$$\text{Variance}\left(T - \frac{U-T}{n-1}\right) = E\left(T - \frac{U-T}{n-1} - E\left(T - \frac{U-T}{n-1}\right)\right)^2 = E\left(\frac{n}{n-1}\cdot T - \frac{U}{n-1} - \alpha\right)^2$$

$$= \left(\frac{n}{n-1}\right)^2\cdot E(T^2) - \frac{2n}{(n-1)^2}\cdot E(U\cdot T) + \frac{1}{(n-1)^2}\cdot E(U^2) - \alpha^2$$

Now,

$$E(T^2) = E(T-\alpha)^2 + 2\cdot E(T)\cdot\alpha - \alpha^2 = \int_{\alpha}^{\alpha+\theta} (t-\alpha)^2 \frac{n}{\theta^n}(\theta+\alpha-t)^{n-1}dt$$

$$+ 2\left(\alpha + \frac{\theta}{n+1}\right)\alpha - \alpha^2$$

$$= n\theta^2\int_0^1 x^2(1-x)^{n-1}\cdot dx + \alpha^2 + \frac{2\theta\cdot\alpha}{n+1} = n\theta^2\cdot\beta(3,n) + \alpha^2 + \frac{2\theta\cdot\alpha}{n+1},$$

$$\left(\text{where} \quad x = \frac{t-\alpha}{\theta}\right)$$

$$= \frac{n\theta^2\,\Gamma 3\,\Gamma n}{\Gamma n+3} + \alpha^2 + \frac{2\theta\cdot\alpha}{n+1} = \frac{2\theta^2}{(n+2)(n+1)} + \alpha^2 + \frac{2\theta\cdot\alpha}{n+1}$$

$$E(U^2) = E(U-\alpha)^2 + 2\cdot E(U)\cdot\alpha - \alpha^2 = \int_{\alpha}^{\alpha+\theta} (u-\alpha)^2\cdot\frac{n(u-\alpha)^{n-1}}{\theta^n}\cdot du +$$

$$+ 2\left(\alpha + \frac{n\theta}{n+1}\right)\alpha - \alpha^2$$

$$= n\theta^2\int_0^1 x^{n+1}\cdot dx + \alpha^2 + \frac{2n\theta\cdot\alpha}{n+1} = \frac{n\theta^2}{n+2} + \alpha^2 + \frac{2n\theta\cdot\alpha}{n+1},$$

$$\left(\text{where} \quad x = \frac{u-\alpha}{\theta}\right).$$

The joint distribution function of T and U is given by $H(t,u)$

$$H(t,u) = \text{Prob}\ (U \leqq u, T \leqq t) = \text{Prob}\ (U \leqq u) - \text{Prob}\ (U \leqq u, T > t)$$

$$= G(u) \quad \text{if} \quad \alpha \leqq u < t \leqq \alpha + \theta \quad [\text{since,} \quad P(A) = P(A \cap B^c) +$$

$$+ P(A \cap B)]$$

$$= G(u) - \left(\frac{u - t}{\theta}\right)^n \quad \text{if} \quad \alpha \leqq t < u \leqq \alpha + \theta \ .$$

$$\text{density function} \quad h(t,u) = \frac{\partial^2 H(t,u)}{\partial t \cdot \partial u} = 0 \quad \text{if} \quad t > u$$

$$= \frac{\partial}{\partial t}\left[-\frac{n}{\theta^n}(u - t)^{n-1}\right] = \frac{n(n - 1)}{\theta^n}(u - t)^{n-2} \ , \ \alpha \leqq t < u \leqq \alpha + \theta$$

$$= 0 \ , \ \text{otherwise}$$

$$E(UT) = \int_\alpha^{\alpha+\theta}\int_\alpha^u u\cdot t\cdot h(t,u)\cdot dt\cdot du = \int_\alpha^{\alpha+\theta}\int_\alpha^u \frac{n(n - 1)}{n}(u - t)^{n-2}\ u\cdot t\cdot dt\cdot du$$

$$= \frac{n(n - 1)}{\theta^n}\int_\alpha^{\alpha+\theta} u\cdot\int_\alpha^u t\cdot(u - t)^{n-2}\cdot dt\cdot du = \frac{n(n - 1)}{\theta^n}\int_\alpha^{\alpha+\theta} u\cdot\int_0^1$$

$$[(x(u - \alpha) + \alpha)(1 - x)^{n-2}(u - \alpha)^{n-1}dx]du$$

$$= \frac{n(n - 1)}{\theta^n}\int_\alpha^{\alpha+\theta}[(u - \alpha)^n\cdot u\cdot\beta(2,n - 1) + \alpha(u - \alpha)^{n-1}\cdot u\cdot\beta(1,n - 1)]du$$

$$\left(\text{where} \quad x = \frac{t - \alpha}{u - \alpha} \ , \ t = x(u - \alpha) + \alpha \ , \ \text{and} \quad u - t = (1 - x)(u - \alpha)\right)$$

$$= n(n - 1)\cdot\left[\beta(2,n - 1)\int_\alpha^{\alpha+\theta}\left(\frac{u - \alpha}{\theta}\right)^n\cdot u\cdot du + \alpha\cdot\beta(1,n - 1)\int_\alpha^{\alpha+\theta}\right.$$

$$\left.\left(\frac{u - \alpha}{\theta}\right)^{n-1}\cdot u\cdot\frac{du}{\theta}\right]$$

$$= n(n - 1)\cdot\left[\frac{\Gamma(2)\,\Gamma(n - 1)}{\Gamma(n + 1)}\cdot\int_0^1 x^n(\theta x + \alpha)\cdot\theta dx + \frac{\alpha\,\Gamma(1)\,\Gamma(n - 1)}{\Gamma(n)}\cdot\int_0^1\right.$$

$$\left. x^{n-1}(\theta x + \alpha)\cdot dx\right] \left(\text{where} \quad x = \frac{u - \alpha}{\theta}\right)$$

$$= n(n-1)\left[\frac{1}{n(n-1)}\left(\frac{\theta^2}{n+2}+\frac{\alpha\theta}{n+1}\right)+\frac{\alpha}{n-1}\cdot\left(\frac{\theta}{n+1}+\frac{\alpha}{n}\right)\right] = \frac{1}{n+2}\cdot\theta^2 + \alpha\cdot\theta+\alpha^2 .$$

Hence,

$$\text{Variance}\left(T-\frac{U-T}{n-1}\right) = \left(\frac{n}{n-1}\right)^2\cdot\left(\frac{2\theta^2}{(n+2)(n+1)}+\alpha^2+\frac{2\theta\cdot\alpha}{n+1}\right) - \frac{2n}{(n-1)^2}\left(\frac{1}{n+2}\,\theta^2+\alpha\cdot\theta+\alpha^2\right)$$

$$+\frac{1}{(n-1)^2}\cdot\left(\frac{n\theta^2}{n+2}+\alpha^2+\frac{2n\theta\cdot\alpha}{n+1}\right)-\alpha^2 = \frac{n\theta^2}{(n-1)(n+1)(n+2)}.$$

Since

$$E(U-T)^2 = E(U^2-2UT+T^2) = E(U^2)-2E(UT)+E(T^2)$$

$$= \left(\frac{n\theta^2}{n+2}+\alpha^2+\frac{2n\theta}{n+1}\cdot\alpha\right) - 2\left(\theta\cdot\alpha+\alpha^2+\frac{\theta^2}{n+2}\right)+\left(\frac{2\theta^2}{(n+1)(n+2)}+\alpha^2+\frac{2\theta\alpha}{n+1}\right).$$

$$= \frac{n(n-1)}{(n+2)(n+1)}\cdot\theta^2 .$$

Therefore, the unbiased estimator of variance $\left(T-\frac{U-T}{n-1}\right)$ given by:

$$\frac{n(n+2)(n+1)}{n(n-1)(n-1)(n+1)(n+2)}(U-T)^2 = \left(\frac{U-T}{n-1}\right)^2$$

$\frac{U-T}{n-1}$ is the estimate of the standard error of the estimator. If this term is small, the variance of the estimator is also very small.

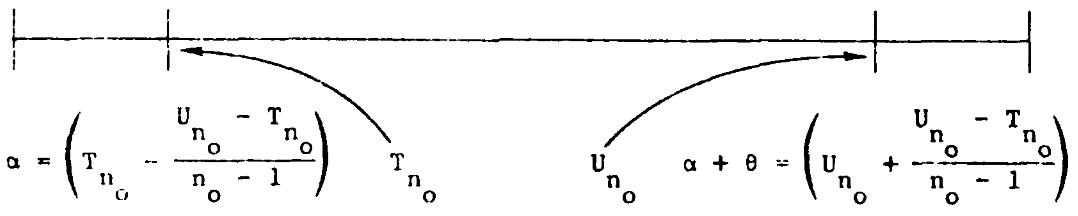
Instead of assuming a uniform distribution (i.e., $\beta(1,1)$) for the local optimals we could assume any β distribution. If we assume

$\left\{\frac{c_i - \alpha}{\theta}\right\}_{i=1}^{n}$ have $\beta(1,2)$ distribution, similar calculations can be made to show that the unbiased estimator of α (the global optimal value) is $\left\{T - \frac{U - T}{1 - C_n - \frac{1}{2n+1}} \cdot \frac{1}{2n+1}\right\}$ where $C_n = \frac{n!\,\Gamma(1/2)}{(2n+1)\,\Gamma(n + 1/2)}$.

If the distribution is assumed to be $\beta(2,1)$ then the estimator of α becomes $\left\{T - C_n \frac{U - T}{\frac{2n}{n+1} - C_n}\right\}$.

A sequential sampling procedure can be used to estimate the global optimal value where a trade-off is obtained between the cost of computation and gain due to expected reduction in global optimal value. Assume C to be the cost of finding one more local optimum. This is a function of the number of iterations (on an average) to get to a local optimum and the computing time per iteration.

The estimates of α and $\alpha + \theta$ when a sample of size n_o is obtained are $\left\{T_{n_o} - \frac{U_{n_o} - T_{n_o}}{n_o - 1}\right\}$ and $\left\{U + \frac{U_{n_o} - T_{n_o}}{n_o - 1}\right\}$ respectively (where U_n and T_n are values of U and T when the sample size is n). Assume that the $(n_o + 1)$st sample will have a uniform distribution between the estimated values of α and $\alpha + \theta$. Then the expected value of T_{n_o+1} is computed as follows:


$\alpha = \left(T_{n_o} - \frac{U_{n_o} - T_{n_o}}{n_o - 1}\right)$
T_{n_o}
U_{n_o}
$\alpha + \theta = \left(U_{n_o} + \frac{U_{n_o} - T_{n_o}}{n_o - 1}\right)$

Let $p\left(T_{n_o+1}\right)$ be the estimated density function of T_{n_o+1} . Then

$$p\left(T_{n_o+1}\right) = \frac{1}{\theta} = \frac{n_o - 1}{(n_o + 1)\left(U_{n_o} - T_{n_o}\right)} .$$

$$\begin{aligned} T_{n_o+1} &= x \quad \text{if } \alpha \leqq x \leqq T_{n_o} \\ &= T_{n_o} \quad \text{if } T_{n_o} \leqq x \leqq \alpha + \theta . \end{aligned}$$

$$E\left(T_{n_o+1}\right) = \int_{T_{n_o} - \frac{U_{n_o}-T_{n_o}}{n_o-1}}^{T_{n_o}} x \cdot dx + \int_{T_{n_o}}^{U_{n_o} + \frac{U_{n_o}-T_{n_o}}{n_o-1}} T_{n_o} \cdot dx$$

$$= T_{n_o} - \frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)} .$$

Therefore, given a sample of size n_o , one of the following plans can be used.

(i) Determine a function $G(T)$, monotonically increasing in T , which gives a measure of cost when the so far obtained minimum local optimum value is T . We would like to have the sum of the cost of computation and $G(T)$ be as small as possible. Now the estimated reduction in $G(T)$ when the $(n_o + 1)$st local optimum is found is $\left[G(T) - G\left(E\left(T_{n_o+1}\right)\right)\right]$ and the cost of taking one more sample is C . Therefore the plan should be: If

$$\left\{G\left(T_{n_o}\right) - G\left(T_{n_o} - \frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)}\right)\right\} < C ,$$

no further sample is taken. The best local optimum so far obtained is accepted. $\left(\text{Note that: } E\left(T_{n_o+1}\right) = T_{n_o} - \frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)}\right)$. If

$$\left\{G\left(T_{n_o}\right) - G\left(T_{n_o} - \frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)}\right)\right\} > C ,$$

then another sample is taken and the criterion function is recomputed. The maximum number of samples N taken is bounded because $(U_n - T_n)$ is bounded by θ. Hence, $\frac{U_N - T_N}{2(N + 1)(N - 1)}$ becomes small for large N, making the difference between values of $G(\cdot)$ at T_{n_o} and $E\left(T_{n_o+1}\right)$ small enough to be less than C.

(ii) Define a loss function $L(e)$, a monotone increasing function of e, which gives the cost incurred when the estimated expected decrease in the global optimal value by taking one more sample $\left(\text{i.e., } T_{n_o} - E\left(T_{n_o+1}\right)\right)$ is e. In this case the plan should be: If

$$L\left(\frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)}\right) < C ,$$

no further sample is taken. The best local optimum so far obtained is accepted. $\left(\text{Note that } \left[T_{n_o} - E\left(T_{n_o+1}\right)\right] = \frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)}\right)$. If

$$L\left(\frac{U_{n_o} - T_{n_o}}{2(n_o + 1)(n_o - 1)}\right) \geqq C ,$$

then another sample is taken and the criterion function is recomputed. The maximum number of samples N is bounded, because if the procedure stops at N then,

$$L\left(\frac{U_N - T_N}{2(N+1)(N-1)}\right) < C \quad \text{and} \quad L\left(\frac{U_{N-1} - T_{N-1}}{2(N)(N-2)}\right) \geq C \ .$$

So,

$$N(N-2) \leq \frac{U_{N-1} - T_{N-1}}{2L^{-1}(C)} \leq \frac{\theta}{2L^{-1}(C)} \ .$$

(Since $L(\cdot)$ is a monotone increasing function and its inverse $L^{-1}(\cdot)$ exists.) Or,

$$N(N-2) \leq \frac{n_o + 1}{n_o - 1} \cdot \frac{U_{n_o} - T_{n_o}}{L^{-1}(C)}$$

(where θ is replaced by its estimator) N is bounded because $N(N-2)$ is bounded. Hence, this procedure stops after a finite number of samplings.

This sequential sampling approach has been tested using C and L as,respectively, an increasing linear functions of t , the average number of iterations, and e , the estimated expected decrease in the value of T by one more sample (i.e., $C = at$ and $L = be$ where a and b are specified coefficients). In the solution procedure the value of a/b used is between 30 and 40. A comparison was made of this sequential procedure with the global optimal search procedure (of Chapter 4) by determining the computation time of the global search procedure and actual error and computation time of this sequential search procedure. About 25 problems of varying sizes (number of nodes between 6 and 35) have been solved using both the global search procedure of Chapter 4 (with approximately 2% error value) and the sequential search procedure of this chapter (using $a/b = 35$). The estimated error upon termination of the sequential search procedure was on the average around 3% while the actual error was about 8 - 10% (casting some doubt upon our use of a uniform distribution for the values of local minima). The computation time for the sequential search procedure was around 50 - 60% that of the global search procedure of Chapter 4. So, computationally, the sequential search procedure seems to be attractive. However, we cannot compare these two procedures for problems with a greater number of nodes because of limitations on the global search procedure caused by restricted computer memory size and computational speed.

CHAPTER 4

In this chapter, a solution procedure will be discussed which converges to a value as close as desired to the global optimum. This is a branch and bound method. The procedure is an adaptation of a general procedure due to Walkup [W-1] and Falk and Soland [F-1] to our special problem. The general procedure requires the solution of certain subproblems. These subproblems, for our problem, have a special structure and we have developed special procedures to exploit this. Each subproblem involves finding the shortest chains between all pairs of nodes. The first major difficulty faced in solving a large problem by this method is the limited size of a computer memory. Some ways to get around this problem will be described in this chapter. The second major difficulty arises in determining the upper bounds on the flow in each arc. An inefficient upper bound is the sum of all the flows. Different types of heuristic methods which yield more efficient bounds will be discussed. However, the bounds are very critical, and choice of the wrong bounds may lead to convergence to a nonglobal optimal solution.

4.1 Reformulation of the Problem

The problem can be reformulated with the superfluous extra constraint (4.1c) shown below. This constraint provides us with a range in which the concave function could be approximated to develop a solution procedure. This range I_m^o defined below is critical for the effectiveness of this procedure.

(4.1a) Minimize $Z(Y) = \sum_{m=1}^{M} f_m(y_m)$

(4.1b) Subject to $Y \in S$

(4.1c) $Y \in \prod_{m=1}^{M} I_m^o$

where

$$S = \left[Y = \{y_m\} \mid Y = \sum_{i,j \ni i>j} A_{ij} X_{ij} \ , \ \sum_{k=1}^{P_{ij}} x_{ij}^k \geq r_{ij} \ , \text{ and } \ x_{ij}^k \geq 0 \right]$$

and interval

$$I_m^o = \left[0 \ , \sum_{\substack{ij \ni \text{ a chain between} \\ ij \text{ contains} \\ \text{the arc } m \text{ and } i>j}} r_{ij} \right] .$$

At this point, the following definitions are useful.

4.2 Definitions

The linear approximation of a function $f_m(\cdot)$ in the interval $I_m = [l_m, u_m]$ is the function $f_m^{linr}(\cdot)$ given by $f_m^{linr}(y_m) = a_m + b_m(y_m - l_m)$ where $a_m = f_m(l_m)$ and $b_m = (f_m(u_m) - f_m(l_m))/(u_m - l_m)$.

This is drawn in Figure 4.1. For a concave function $f_m^{linr}(y_m) \leq f_m(y_m)$ in the interval I_m and $f_m^{linr}(y_m) \geq f_m(y_m)$ outside the interval I_m .

Linearization Error

The difference $e_m(y_m) = \left[f_m(y_m) - f_m^{linr}(y_m) \right]$ is called the linearization error of $f_m(y_m)$ at y_m . This error is nonnegative in the interval I_m and nonpositive outside the interval I_m .

4.3 Linearized Version of the Problem

In this procedure, instead of solving (4.1), a linearized version of this formulation, given in (4.2), is solved initially.

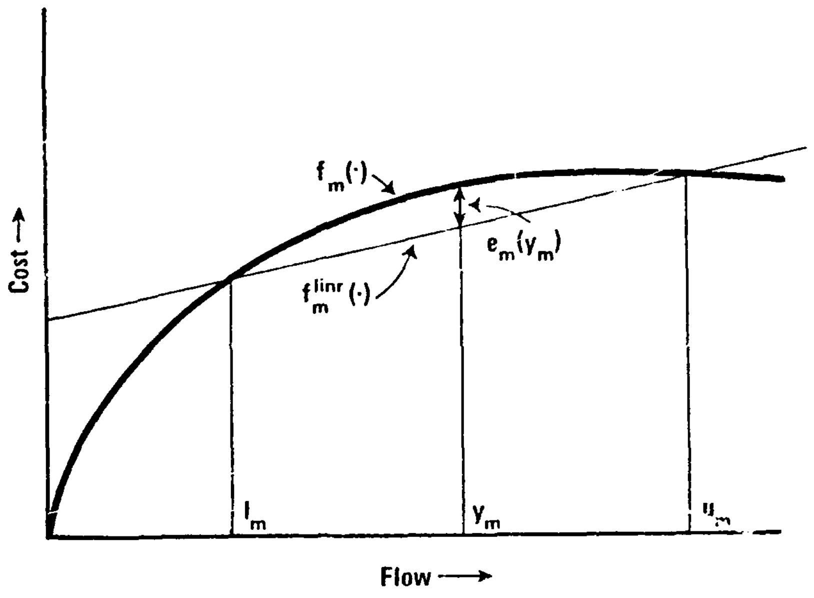


FIGURE 4.1

(4.2a) Minimize $Z'(Y) = \sum_{m=1}^{M} f_m^{linr}(y_m)$

(4.2b) Subject to $Y \varepsilon S$ where S is as defined in (4.1)

(4.2c) $Y \varepsilon \prod_{m=1}^{M} I_m^o$

where $f_m^{linr}(y_m)$ is the linear approximation of the function $f_m(y_m)$ in the interval I_m^o .

Let Y^{linr} be a solution of (4.2) and Y^{opt} be a solution of (4.1). Define $Z^{linr} = Z'(Y^{linr})$, $Z^{mixr} = Z(Y^{linr})$ and $Z^{opt} = Z(Y^{opt})$. Then the following inequality is obtained

$$Z^{linr} \leqq Z^{opt} \leqq Z^{mixr} = Z^{linr} + E \tag{4.3}$$

where $E = \sum_{m=1}^{M} e_m\left(y_m^{linr}\right)$ and $e_m\left(y_m^{linr}\right)$'s are linearization errors at $Y^{linr} = \left\{y_m^{linr}\right\}$ for intervals I_m^o . The first inequality is obtained because within I_m^o , $f_m^{linr}(y_m) \leqq f_m(y_m)$ and the answer is within I_m^o . The second inequality is obtained because Y^{linr} is any feasible solution of (4.1) and Z^{opt} is the optimal solution. The final equality is obtained because the extra amount E is the sum of all linearization errors.

If E is small, then Z^{linr} and Z^{mixr} will be a good approximation of the optimum value. The maximum value of E depends on the intervals I_m^o's ; partitioning of these intervals into shorter ones reduces the maximum possible value of E . The general procedure involves solving subproblems of the form (4.2a), (4.2b) subject to the additional constraint that the flow in each arc m is restricted to lie in a specified interval $[l_m, u_m]$. At each iteration, two new subproblems are generated from a given subproblem by dividing the admissible interval for one arc m' into a left part $[l_{m'}, y_{m'}]$ and right part $[y_{m'}, u_{m'}]$ and leaving the admissible intervals for all other arc flows unchanged. Hence, each iteration adds 1 to the size of the set T of subproblems where T has the property that any (admissible by (4.1b)) flow pattern is admissible for at least one subproblem. (If the flow happens to correspond to a point of division of any one arc, it will be admissible for two subproblems.) At iteration p , the original problem has been partitioned into p subproblems. To be able to uniquely identify a subproblem by number, we number the subproblems as follows. Assume the subproblems at iteration p are uniquely numbered from 1 to p (such is the case at iteration 1) . If now the subproblem k is partitioned into two subproblems by dividing arc m' at y , the subproblem associated with the left hand part $[l_{m'}, y]$ is numbered k and the subproblem associated with the right hand part $[y, u_{m'}]$ is numbered p + 1 . Let $I_m^{k,p}$ be the admissible interval of arc m for subproblem k at iteration p . Then $\prod_{m=1}^{M} I_m^o = \bigcup_{k=1}^{p} \prod_{m=1}^{M} I_m^{k,p}$ for all p . If we say that the kth subproblem is solved at iteration p , we mean the problem

(4.4a) Minimize $Z'(Y) = \sum_{m=1}^{M} f_{m,k,p}^{linr}(y_m)$

(4.4b) Subject to $Y \epsilon S$, S defined above

(4.4c) $Y \epsilon \prod_{m=1}^{M} I_m^{k,p}$

where $f_{m,k,p}^{linr}(y_m)$ is the linear approximation to the cost on arc m corresponding to $I_m^{k,p}$.

Bounding refers to obtaining the lower bound for each subproblem (4.4); i.e., if $Y_{k,p}^{linr}$ is the solution for (4.4), then

$Z^{linr}_{k,p} = Z'\left(Y^{linr}_{k,p}\right)$ is the lower bound. Define $Z^{mixr}_{k,p} = Z\left(Y^{linr}_{k,p}\right)$ and $Z^{linr}_{s,p} = \underset{k=1,2,\ldots,p}{\text{Minimum}} \left\{Z^{linr}_{k,p}\right\}$ when p is the number of subproblems solved.

Then the following inequalities are obtained

$$(4.5) \qquad Z^{linr}_{s,p} \leqq Z^{opt} \leqq \underset{k=1,\ldots,p}{\text{Minimum}} \left\{Z^{mixr}_{k,p}\right\} \leqq Z^{mixr}_{s,p} = Z^{linr}_{s,p} + E_{s,p}$$

where $E_{s,p} = \sum_{m=1}^{M} e_m\left(y^{linr}_{m,s,p}\right)$ and $y^{linr}_{m,s,p}$ is the mth element of the vector $Y^{linr}_{s,p}$.

The first inequality is obtained because each $Z^{linr}_{k,p}$ gives the lower bound for the problem in the interval $\prod_{m=1}^{M} I^{k,p}_m$ and the minimum gives the lower bound for the entire interval, hence the lower bound for Z^{opt} . The second inequality is obtained because each $Z^{mixr}_{k,p}$ is a feasible solution of the main problem (4.1), hence Z^{opt} is less than or equal to it. The third inequality is obtained because $Z^{mixr}_{s,p}$ is one of the elements over which minimization is done, and the final equality is obtained because $E_{s,p}$ has taken care of the linearization error. In the branch and bound process, the maximum value of $E_{s,p}$ is made small so that a good estimate of Z^{opt} is $Z^{linr}_{s,p}$. This above error can be made arbitrarily small in a finite number of steps as shown below. Moreover, the following propositions make the branch and bound procedure computationally attractive.

Proposition 4.1:

It may not be necessary to break all the rectangles of a decomposition down until the maximum of the error estimate $E_{k,p}$ for each one is as small as the desired accuracy of an answer, say ϵ .

Proof:

If any feasible solution Y^o with the value $Z^o = Z(Y^o)$ is known for the original problem (4.1), and if the solution $Z^{linr}_{k,p}$ of (4.4) is such that $Z^{linr}_{k,p} > Z^o$, then the optimal solution cannot lie in the rectangle $\prod_{m=1}^{M} I^{k,p}_m$ (because its lower bound has a greater value than the value obtained at some other point). So, no further refinement of that subproblem is necessary. In the procedure, the minimum $Z^{mixr}_{k,p}$ so far obtained is taken as Z^o and at each step all the intervals for which $Z^{linr}_{k,p} \geqq Z^o - \epsilon$ are dropped from further consideration. This procedure of rejecting some intervals altogether hastens the process of convergence.||

Proposition 4.2:

The procedure of the algorithm described below remains unchanged if the restriction (4.4c) is dropped; i.e., the kth subproblem solved at the pth iteration is

(4.4a) Minimize $Z'(Y) = \sum_{m=1}^{M} f^{linr}_{m,k,p}(y_m)$

(4.4b) Subject to $Y \varepsilon S$.

Proof:

Let Y^{lin} be the solution for this unrestricted problem and define $Z^{lin}_{k,p} = Z'(Y^{lin})$ and $Z^{mix}_{k,p} = Z(Y^{lin})$. (Notice the difference between Y^{linr}, Z^{linr}_k and Z^{mixr}_k and Y^{lin}, Z^{lin}_k, Z^{mix}_k . The last r is dropped for the unrestricted problem.) Here Y^{lin} no longer lies within the interval so inequality (4.5) is not obvious. However, in the restricted problem the constraint is $Y \varepsilon S \cap \prod_{m=1}^{M} I^{k,p}_m$ and in the unrestricted problem the constraint is $Y \varepsilon S$. Since

$S \cap \prod_{m=1}^{M} I_m^{k,p} \subset S$, it is obvious that

$$Z_{k,p}^{linr} = \underset{Y \in S \cap \prod_{m=1}^{M} I_m^{k,p}}{\text{Minimum}} Z'(Y) \geq \underset{Y \in S}{\text{Minimum}}\, Z'(Y) = Z_{k,p}^{lin} .$$

The better (smaller) minimum is obtained when the search is done on a larger set than on one of its subsets. So, if $Z_{s,p}^{lin} = \underset{k=1,\ldots,p}{\text{Minimum}} \left\{Z_{k,p}^{lin}\right\}$ where p is the number of subproblems solved, then the above inequality plus the first inequality of the chain (4.5) establishes the first of the following chain of inequalities:

$$(4.6) \qquad Z_{s,p}^{lin} \leq Z^{opt} \leq \underset{k=1,\ldots,p}{\text{Minimum}} \left\{Z_{k,p}^{mix}\right\} \leq Z_{s,p}^{mix} = Z_{s,p}^{lin} + E_{s,p} .$$

Since all $Z_{k,p}^{mix}$ are feasible, the second and third inequalities follow as before. The final equality is also valid because $E_{s,p}$ takes care of the linearization error. Now the maximum value of $E_{s,p}$ is the maximum value that the linearization error can have within $\prod_{m=1}^{M} I_m^{s,p}$. Because in this case some y_m^{lin} (mth element of Y^{lin}) may not be within the interval, the errors corresponding to those are negative. And for the ones which are within the interval the error is positive and is bounded by the maximum linearization error of the interval. So, the sum $E_{s,p}$ is bounded by the sum of the maximum linearization error within the intervals. The proposition 4.1 is also valid here because $Z_{k,p}^{lin}$ does indicate at least a lower bound of the objective function on the interval $\prod_{m=1}^{M} I_m^{k,p}$. So, both the inequality chain (4.6) and Proposition 4.1, which are necessary for the algorithm to work, are satisfied in this case.||

If the restriction of bounds on flow in each arc is dropped, then the solution of the unrestricted subproblem is accomplished by solving for the shortest chain between each pair of sources and sinks with the length of each arc equal to the slope of the linear cost curve and by passing the entire required flow through the shortest chain. The order of calculations in each step is $N^3/2$ where N is the number of nodes. Moreover, when any particular interval $I_m^{k,p} = \left[l_m^{k,p}, u_m^{k,p}\right]$ is subdivided to $\left[l_m^{k,p}, r\right]$ and $\left[r, u_m^{k,p}\right]$ where $l_m^{k,p} < r < u_m^{k,p}$, then because of the concavity of the cost function the slope of the linear curve for $\left[l_m^{k,p}, r\right]$ increases and the slope of the linear curve for $\left[r, u_m^{k,p}\right]$ decreases as shown in Figure 4.2. The two subproblems obtained by dividing the arc are as follows:

(1) Solution of a problem where one arc length has decreased. Here the modified solution procedure described in the algorithm can be used to solve the problem in $\sim N^2$ steps. Thus, considerable computation time is saved.

(2) Solution of a network problem where the length of one arc has increased. Usually, this requires solving the problem all over again in which case the calculations are $\sim N^3/2$. However, if the same arc is divided again and again, then use a very large length for the particular arc and initially calculate the shortest chain matrix and store it. Even though the arc length has increased in a particular subdivision, it has effectively decreased from the arc length used in calculating the initially stored matrix. So, the modified procedure described in the algorithm, using the stored matrix can solve this new subproblem in $\sim N^2$ steps.

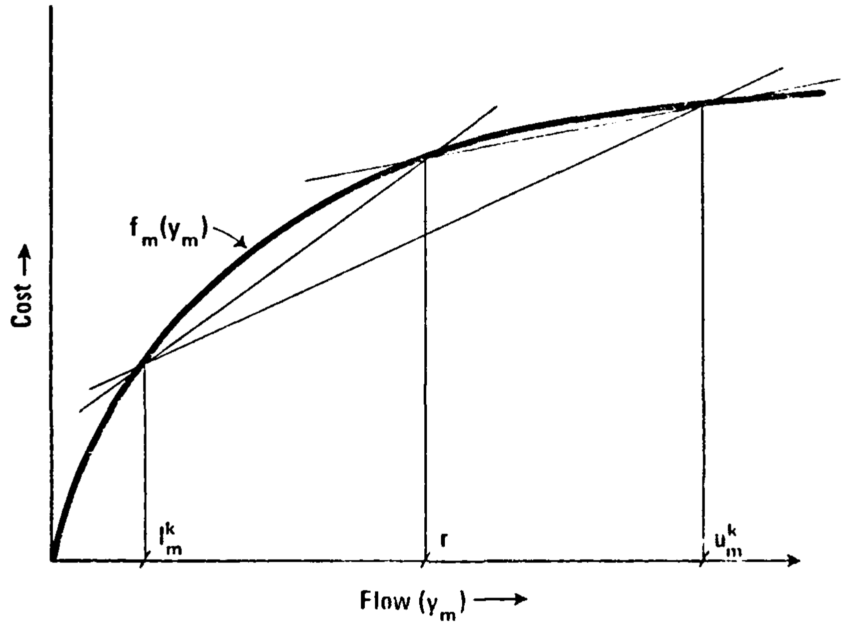


FIGURE 4.2

The algorithm is described in full detail below.

4.4 Algorithm

Step 1: (Initialization)

For each arc m, an interval $I_m^o = [0,u_m]$ is determined. (The determination of u_m, the upper bound on possible flow on arc m in the optimal solution is difficult, and the convergence of the procedure depends on how well it is obtained.) The linearized function $Z'(Y)$ for this set of intervals is formed and the unrestricted problem given by (4.4a) and (4.4b) is solved to obtain Y_1^{lin}, Z_1^{lin} (i.e., Minimum of $Z'(Y)$ without the upper and lower bound restriction on variables) and Z_1^{mix}. (The subscript indicates that this is the 1st subproblem.) The solution procedure requires solving the shortest path problem using Floyd's algorithm between all pairs of points and passing the entire flow through the shortest path between each source-sink pair. Y_1^{lin} and Z_1^{mix} is set as Y^{best} and Z^{best}, the best solution found so far. The information describing the intervals I_m^o and Z_1^{lin} is stored as the first and only item of a list. Set p, which indicates the iteration number, equal to 1.

Step 2:

Every item of the list for which the associated value of Z^{lin} is greater than or equal to $(Z^{best} - \epsilon)$ (where ϵ is the permissible error) is removed and discarded. If the list is now empty, go to Step 6. Otherwise, select the item from the list corresponding to the minimum value of Z^{lin}. Assume this is the value found when the kth subproblem is solved. From among the intervals described in the selected item, choose the interval I_s for which the error e_s at the solution point is maximum, i.e., $e_s = \operatorname*{Max}_m \left\{ f_m\left(y_m^{lin}\right) - f_{m,k,p}^{linr}\left(y_m^{lin}\right) \right\}$ and this interval $I_s = [l_s, u_s]$ is broken into two intervals $I_s^1 = \left[l_s, y_s^{lin}\right]$ and $I_s^2 = \left[y_s^{lin}, u_s\right]$. (The information about which interval should be divided for each item can be obtained in the same step when the Z^{lin} is evaluated, so that y_m^{lin} values for all arcs need not be stored.)

Step 3:

Compute the linear approximation to the arc s based on the interval I_s^1 and leave all other intervals as in the kth subproblem. (4.4a,b) is solved to obtain Y^{lin}, Z^{lin} and Z^{mix}. Call this subproblem $(k,p+1)$. If $Z_{k,p+1}^{mix}$ is less than Z^{best}, Y^{best} and Z^{best} are replaced by $Y_{k,p+1}^{lin}$ and $Z_{k,p+1}^{mix}$. Finally, the description of the intervals for subproblem $(k,p+1)$ and the value of $Z_{k,p+1}^{lin}$ is added as a new item in the list. (This step is an all-pairs shortest path problem and requires $\sim N^3/3$ additions and comparisons.) (All other subproblems ($(p-1)$ at most) except the kth one at this stage will change its second subscript to $p+1$ from p because these subproblems will be used in the next iteration only.)

Step 4:

Step 3 is repeated using I_s^2 in place of I_s^1. Call this subproblem $(p+1, p+1)$. However, here the length of arc s (suppose it connects n and l) is shortened to d_{nl}^*. (This is b_m as defined in (4.2) for interval I_m. Here $m = s$ and $I_m = I_s^2$.) The new distances between any two nodes ij is $d_{ij}^* =$ $\operatorname{Min}\left\{d_{il} + d_{nl}^* + d_{nj}, d_{ij}, d_{in} + d_{nl}^* + d_{lj}\right\}$ where d_{ij} is the shortest distance between i and j when the arc is not shortened. (The optimal solution is obtained with only $\sim N^2$ calculations.)

Step 5:

Delete $Z^{lin}_{k,p}$ and the description of the intervals used in the kth subproblem from the list. Add 1 to p and go to Step 2.

Step 6:

Stop. The vector Y^{best} is the feasible solution to the original problem and its value Z^{best} is within ϵ of the optimal value.

4.5 Remarks

Various options other than those given in the algorithm can be used. The choice of the arc to be considered for further subdivision can be arbitrary. Using the same arc again and again for a few iterations reduces the storage requirements and can be done efficiently. Also, the subdivision point, given an interval and an arc, can be the mid-point of the interval, the point x where the linearization error $\left\{f_m(x) - f^{linr}_{m,k,p}(x)\right\}$ is maximum or a point which divides the interval such that the maximum linearization error in each part is equal. The relative merits of these can only be tested empirically. However, the use of y^{lin}_m as the point of partition yields good results.

This branch and bound algorithm is equivalent to generating a tree as follows: Step 1 produces node a and puts information about the node on a list. Step 2 finds any pendant nodes of the current tree that need not be considered further, flags them, and deletes information about them from the list. If the list is empty, it stops. If not, it chooses a pendant, unflagged node (call it node x) and creates two descendants of that node. Steps 3 and 4 perform calculations on the two new nodes and add information about these two nodes to the list. Step 5 deletes information about node x from the list, flags node x and returns to Step 2. The algorithm terminates at Step 6 when all the nodes are flagged.

At any stage of the algorithm, all the intervals over which linearization is made and which are associated with the terminal nodes of the tree, cover the entire set of initial intervals I_m^o and hence the polyhedron S .

Convergence Theorem 4.1:

Given any arbitrarily small value of $\epsilon > 0$ and assuming that each function $f_m(y_m)$ is concave and nondecreasing,[1] the algorithm described above terminates in a finite number of steps and correctly produces a feasible solution with the criterion value within ϵ of the global optimal value.

Proof:

Solution of two subproblems is needed in each iteration of the algorithm. Each of these subproblems is a shortest chain problem with all nonnegative entries in the initial distance matrix. Hence, each can be solved in a finite number of steps (i.e., maximum $\sim N^3$ steps, where N = number of nodes). So, we will only have to show that number of iterations p needed before we go to Step 6 is finite for a given ϵ , and also that once we go to Step 6 we have found a feasible solution with criterion value within ϵ of the global optimal value.

Suppose in some iteration p we have reached Step 6, i.e., all the pendant nodes of the tree (all p subproblems) are flagged. According to flagging rule $Z^{best} - \epsilon \leq Z_{k,p}^{lin}$ for all values of k (i.e., $k = 1, \ldots, p$) . Since all values of k cover the entire feasible region

[1]Nondecreasing property is necessary here because it establishes the continuity of the function at the right hand boundary, which is needed in the proof. Also, this property guarantees that all the entries of the distance matrix used to solve the shortest path problem are nonnegative.

i.e., $\left(\prod_{m=1}^{M} I_m^o = \bigcup_{k=1}^{p} \prod_{m=1}^{M} I_m^{k,p}\right)$ of problem (4.1) and Z^{opt} is the global ptimal solution, then $\left(\underset{k}{\text{Minimum}} \left\{Z_{k,p}^{lin}\right\}\right) \leqq Z^{opt}$, and

$^{best} - \epsilon \leqq \left(\underset{k}{\text{Minimum}} \left\{Z_{k,p}^{lin}\right\}\right) \leqq Z^{opt}$, so we have obtained Z^{best} within ϵ of the global optimal solution.

Therefore all we will have to show is that we go to Step 6 in a inite number of iterations p for any given $\epsilon > 0$. Observe that the unctions are concave and nondecreasing, hence they are continuous except erhaps at zero flow value (where there can be a jump). So, except for ntervals in the neighborhood of zero (which we consider later) the inearization error $e_m(\cdot)$ for any interval $[y_m, y_m + I_m]$ is bounded by he following inequality:

$$e_m(\cdot) \leqq (f_m(y_m + I_m) - f_m(y_m)) \leqq \left(D^+ f_m(y_m)\cdot I_m\right) ,$$

here D^+ denotes the right hand derivative. The first inequality is due o the fact that $f_m(\cdot)$ is nondecreasing, and the second inequality is ue to the fact that $f_m(\cdot)$ is a concave function. The right hand erivative $D^+ f_m(\cdot)$ is finite except at $y_m = 0$ (shown in Appendix D). ow at each iteration one of the intervals is partitioned into two, so me I_m decreases. If we can show that by repeated partitions of an nterval, $I_m \to 0$, then $e_m(\cdot)$ will become less than ϵ/M in a finite mber of steps. Consequently, the total error satisfies

$_{,p} = \sum_{m=1}^{M} e_m(\cdot) \leqq \epsilon$ at some finite number of iteration p for all k .

nce from (4.6) $Z_{s,p}^{lin} \leqq Z^{opt} \leqq \left(\underset{k=1,\dots,p}{\text{Minimum}} \left\{Z_{k,p}^{mix}\right\}\right) = Z^{best} \leqq$

$^{ix}_{,p} = Z_{s,p}^{lin} + E_{s,p}$ and since total error $E_{s,p}$ is bounded by ϵ

$^{est} \leqq Z_{s,p}^{lin} + \epsilon$ or, $Z^{best} - \epsilon \leqq Z_{s,p}^{lin} \leqq Z_{k,p}^{lin} \quad \forall\ k = 1, \dots, p$ because

$Z^{lin}_{s,p}$ is the minimum element of all the $Z^{lin}_{k,p}$. So, in finite number of iterations p, all the pendant nodes satisfy the flagging criteria. Hence, we reach Step 6 in a finite number of steps. We now consider intervals containing 0, then to complete the proof we show that $I_m \to 0$.

The bound $e_m(\cdot) \leqq D^+f_m(y_m) \cdot I_m$ is not valid for an interval $\bar{I}_m$ with left hand endpoint 0 if $D^+f_m(0)$ does not exist. It appears that we might find ourselves reducing the length of $\bar{I}_m$ without reducing the linearization error at the point of division y_m below the amount of discontinuity at 0. This could happen if $y_m \to 0$. But the procedure guarantees the flow value on each arc is either zero or at least $r_{st} = \underset{ij \ni r_{ij} > 0}{\text{Minimum}} \{r_{ij}\}$. So, such points do not exist.

The only other possibility for the maximum linearization error to remain large for some interval I_s in subsequent partitions is if the partitions are always very near one of the boundary points (i.e., one I_s decreases very little in each iteration and approaches a nonzero limit). But, in this case, because of the continuity of the cost curve, the actual error $e_s(y_s)$ at the point where the interval is partitioned eventually becomes $< \epsilon/M$. (Because close to the boundary point where the linearization error is zero, linear approximation and actual curve are close together.) Since partition is made on an arc s such that $e_s(y_s) = \underset{m}{\text{Maximum}} \{e_m(y_m)\}$, the total actual error $E_{k,p}\left(= \sum_{m=1}^{M} e_m(y_m) \leqq M \cdot e_s(y_s)\right)$ is $< \epsilon$ where $e_s(y_s) < \epsilon/M$. The values of $Z^{mix}_{k,p}$ and $Z^{lin}_{k,p}$ for this subproblem is very close, so they will be flagged off by the rule $Z^{best} - \epsilon \leqq Z^{lin}_{k,p}$ (because $Z^{best} \leqq Z^{mix}_{k,p} = Z^{lin}_{k,p} + E_{k,p}$ or, $Z^{best} - E_{k,p} \leqq Z^{lin}_{k,p}$ and $E_{k,p} \leqq \epsilon$). Therefore, we cannot have repeated occurrence of small changes of errors. Hence, no interval of maximal error can be repeatedly subdivided, yet its length not approach 0.

Therefore, for all cases, given arbitrarily small $\epsilon > 0$ number of iterations p is finite and we get a feasible solution with criterion value within ϵ of the global optimal value.||

4.6 Difficulties in Using the Branch and Bound Procedure on Step Functions with Decreasing Step Size and Constant Interval Between Two Consecutive Steps

The rule of dividing an interval of an arc at an interior point (flow level) might, when applied to step functions, give rise to linear curves which have higher values than the original cost functions within the interval of interest, as shown in Figure 4.3. So, the solution using linear approximation does not necessarily give a lower bound on the total cost and the procedure is inapplicable.

The above difficulty is avoided if the division of intervals is made at the nearest point of discontinuity (i.e., at B instead of D in Figure 4.4). Then the linear curve does represent a lower bound within the interval. Each subsequent partition reduces the error value. But if the restriction (4.4c) is dropped, as it is for our algorithm, then for some points outside the interval on which linear approximation is based, the linear approximation curve lies below the original cost function (shown in Figure 4.4). While working with interval OB , a solution can be at flow level C and will have a positive error e_m . This point C can then be a candidate for further partition of interval OB , but this is impossible. So, for this method to be successful, the restriction of bounds needs to be maintained, and thus one of the advantages of our algorithm is lost. Even in case the interval bounds (4.4c) are maintained, the points where partition can be made are restricted to points of discontinuity. So, the reduction of error beyond a certain level will not be possible, and depending on the structure of the problem this can be quite high.

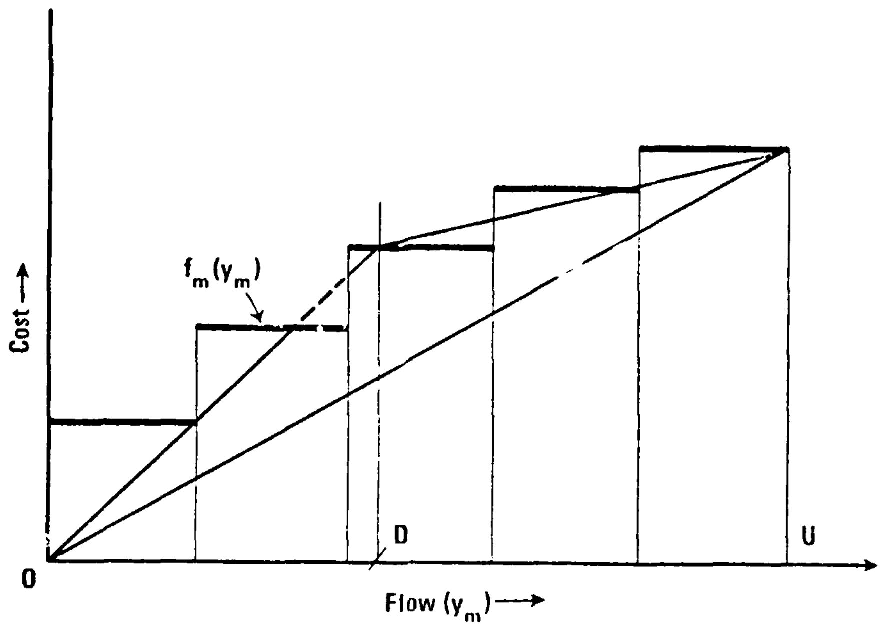


FIGURE 4.3

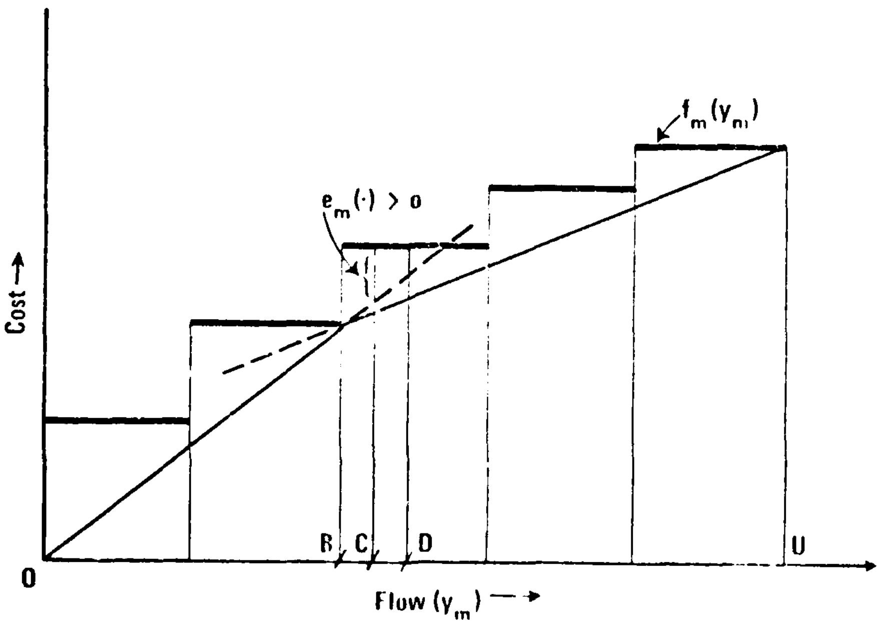


FIGURE 4.4

This problem can be solved by using any arbitrary small ϵ error by the modified method suggested by J. E. Falk and R. M. Soland. As shown in Figure 4.5, where D is the point of splitting of the interval $[0,u]$, the cost functions have open left and closed right hand intervals for each step. So, it is a lower semi-continuous[2] function. One can define a function $f_m^c(\cdot)$ for each $f_m(\cdot)$, which is a convex envelope.[3] Thus, the subproblems are convex programs and of considerable difficulty. Convergence is guaranteed for any small ϵ (not necessarily finite convergence) and proved in Reference [F-1]. Due to the unavailability of a simple convex programming code, this procedure is not tested computationally.

4.7 Limitations of the Algorithm

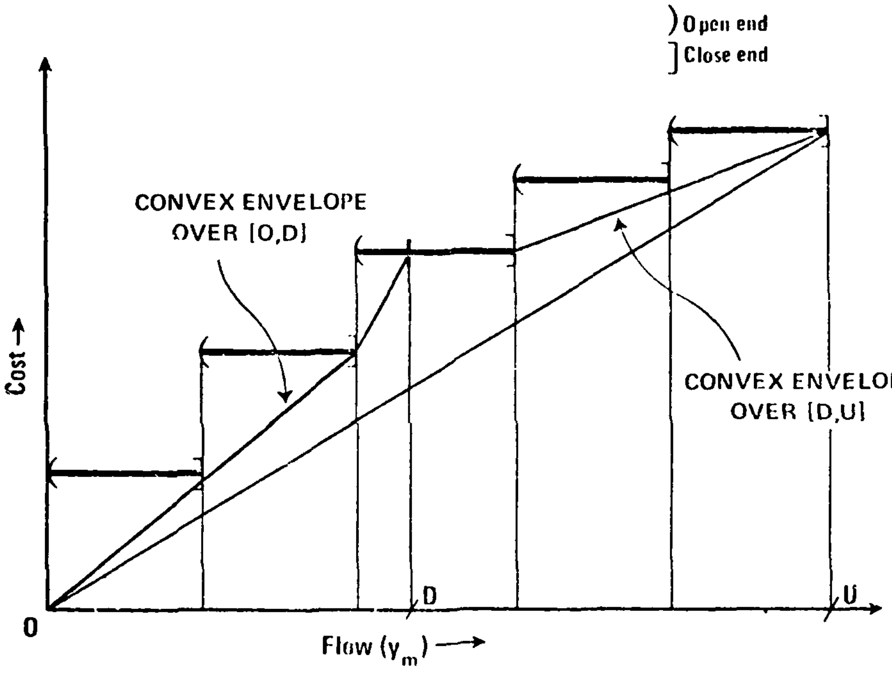
The two major limitations for this procedure are as follows:

(a) To start the procedure, we need to specify the upper bound on the flow in each arc. One of the trivial upper bounds is the sum of all the flows. This can be very large and if initially we start with this large upper bound the error value is large and it will take a considerable amount of partitioning to reduce this error to a small value. Ideally, if we could start with the upper bound at slightly more than the optimal value, we would get quick convergence. However, there is no theoretical way to determine such bounds. At the same time if we assume too low an upper bound the algorithm can yield a wrong answer, as will be evident

---

[2]Definition: Let $y^o \in C$. Then $f(\cdot)$ is lower semi-continuous at y^o if for every $\epsilon > 0$ there is a $\delta > 0$ such that if $||y - y^o|| < \delta$ and $y \in C$ then $f(y) \geq f(y^o) - \epsilon$. $f(\cdot)$ is lower semi-continuous on C if it is lower semi-continuous in each $y \in C$.

[3]Definition: The highest convex function which fits below $f_m(\cdot)$.



) Open end
] Close end
CONVEX ENVELOPE OVER [O,D]
CONVEX ENVELOPE OVER [D,U]
Cost →
0
D
U
Flow (y_m) →

FIGURE 4.5

from an example discussed below.

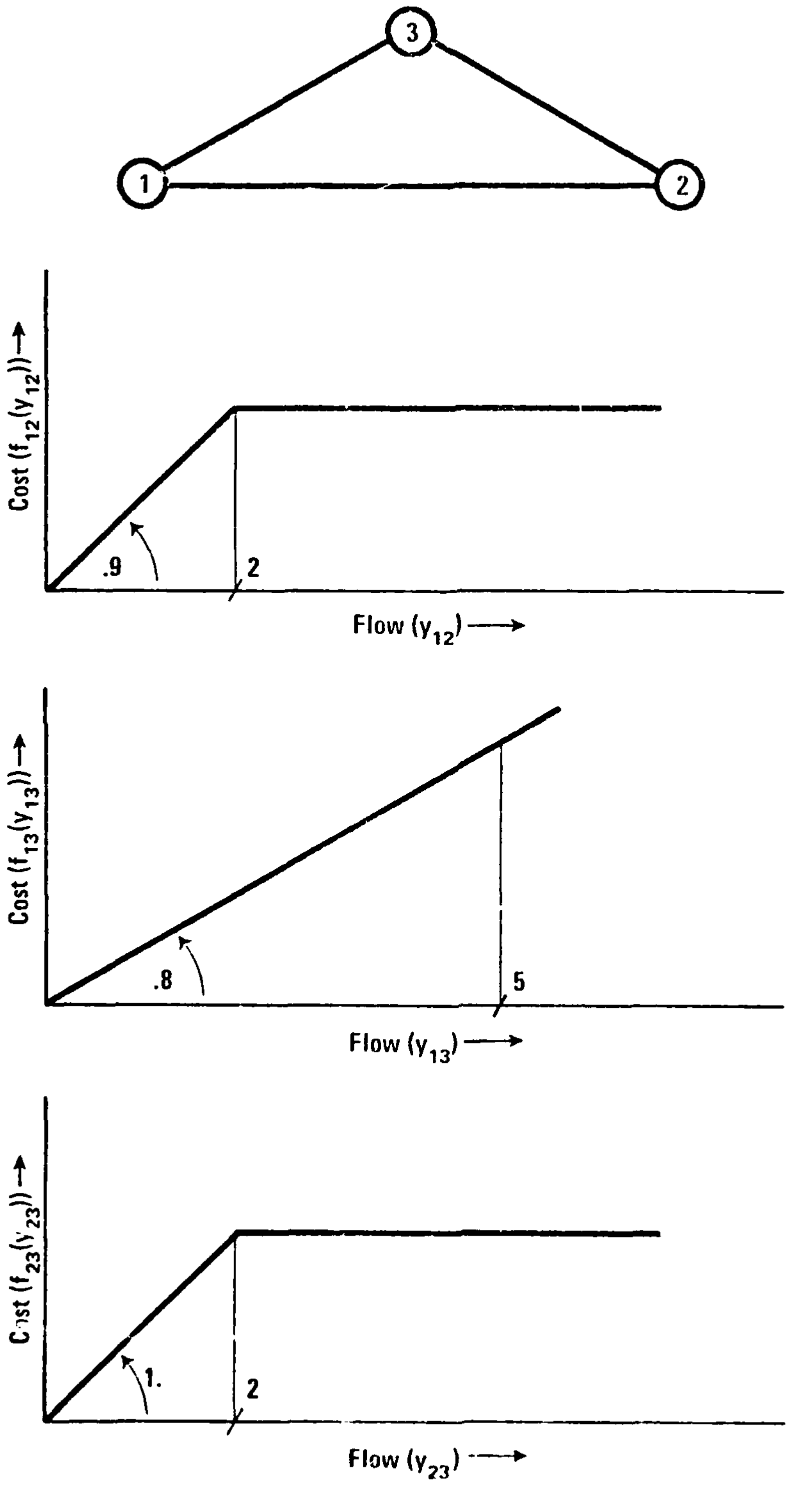
A procedure for generating upper bounds on arc flows that we have tested by computation is as follows: Let the length of the arc m' for which the upper bound is sought be zero and let the length be $f_m(r_{st})/r_{st}$ for every arc $m \neq m'$. Here r_{st} is the minimum, nonzero, specified flow requirement r_{ij}. We use r_{st} since if any flow uses an arc, the flow will be at least r_{st} and the smaller the flow the higher the cost per unit flow, i.e., length. Find the shortest chains between all pairs of nodes $i\ j$. Send flow r_{ij} along the shortest chain between i and j for all $i\ j \ni r_{ij} > 0$ and use the resultant flow on arc m' as the upper bound. While making arc m' free and every other arc as expensive as possible might appear to give maximal possible use of arc m', the following example shows that it may fail to do so. It then might be hoped that if the upper bound so generated fails to be large enough, the resulting solution given the bound will be at the bound (which can then be relaxed), but the example shows even this is false. In spite of this, it seems a reasonable heuristic.

Example:

Consider a three node network with requirements $r_{12} = 1$, $r_{13} = 4$ and $r_{23} = 1$ and cost functions as shown in Figure 4.6.

Upper bounds using the procedure are $U_{12} = 2$, $U_{13} = 5$, $U_{23} = 2$. The best solution respecting these bounds is $(y_{12}, y_{13}, y_{23}) = (1,4,1)$ with total cost = 5.1. Here none of the flow is at its upper bounds, so the bounds won't be relaxed. However, actual minimum cost solution is $(y_{12}, y_{13}, y_{23}) = (5,0,5)$ with total cost = 3.8.

Empirical evidence given by solving the problem with different sets of upper bounds shows that convergence rate for a given problem is very much dependent on how small an upper bound is used for the flows on the



3
1
2
Cost $(f_{12}(y_{12}))$
.9
2
Flow (y_{12})
Cost $(f_{13}(y_{13}))$
.8
5
Flow (y_{13})
Cost $(f_{23}(y_{23}))$
1.
2
Flow (y_{23})

FIGURE 4.6

arcs of the network. It seems that these effects are insensitive to network structure and flow pattern (i.e., disimilar networks tend to show a similar ratio of improvement when better upper bounds are used).

(b) The second limitation of this method is the core size of the computer. As the tree grows, in a large problem, we need to keep in store a large amount of information specifying the sets of intervals corresponding to pendant nodes. Moreover, to take advantage of only N^2 calculations for the arc shortening subproblem, we need the shortest distances as well as information about the paths. So, for each node of the solution tree, three matrices of size $N \times N$ (N = number of nodes in the original problem) need to be stored. Since interval bounds are integers, we can pack the information as one bit of information and use the upper half of a matrix to store it, and use the lower half to store the distances. So, the number of matrices can be reduced to two matrices of size $N \times N$. When k partitions are made, the total number of matrices is $2(k + 1)$. If nothing has been blocked, then the total number of core locations required is $2N^2(k + 1)$. This can be very large and often exceeds the core size of a fairly large machine, even for a network with only 30 to 40 nodes. Instead of storing all the above information in the core, two things can be done:

(i) As soon as any set of matrices is generated, it is kept in a disc file. In the testing, an unbuffered binary random disc file is used to store this information. By writing each matrix on a set of pages and going directly to those pages for reading it when needed, considerable time can be saved. This was attempted with a 35 node network using four pages to write each matrix. The disadvantage of this method is that considerable time is wasted in reading and writing these matrices.

(ii) Instead of storing the information in matrix form for each unchecked node in the tree, we could simply keep track of the path and keep the information about which arc is subdivided at each node and at what point. Then at any step all the interval information can be calculated by tracing the path. This requires no storage on disc files. The subproblem where a particular arc is shortened cannot use the modified algorithm because no information is stored. Moreover, when the tree grows considerably, a substantial amount of computation has to be done for generating the interval information needed. So, the saving in disc file reading and writing is offset by computing time, and this method may not be very efficient either.

For large problems, the best procedure seems to be a combination of the above methods. For a few stages, path information is used to generate a matrix and then the disc file is used to store the matrices. Thus at any point not too much calculation is needed to generate the interval information and disc reading and writing is not too frequent. (The time-shared interactive computer program developed by Stanford Research Institute named TREE (Reference [N-1]) is good for this type of work.)

A program has been written for the branch and bound procedure for a network of up to 35 nodes and a number of problems have been solved. The program and a brief summary of data on solution of the sample problems are listed in Appendix B.

CHAPTER 5

In this final chapter, a post-optimization procedure for problems with discontinuous cost functions with decreasing step sizes (Figure 1.3 of Chapter 1) will be discussed. Also, a variation of the problem in which the requirements vary in different time intervals will be formulated. Finally, the scope of possible future research will be outlined.

5.1 Post-Optimization Procedure

Both of the methods discussed in Chapters 3 and 4 assume continuity of the cost except at the left-hand extreme point. However, most of the practical problems encountered do not have continuous cost functions but functions of the form shown in Figure 1.3. To obtain a solution to this problem, the discrete points constituting the actual cost curve are approximated by means of a continuous nondecreasing concave curve. Since the approximating curve is nondecreasing and concave, it is continuous everywhere except at the left-hand extreme point (as shown in Appendix D). (Ideally, the continuous nondecreasing concave cost curve $g(y)$ should be such that, if $f(i)$ is the actual cost of i channels, and if u is the maximum number of channels possible the absolute error $\sum_{i=0}^{u} |f(i) - g(i)|$ should be minimized over all $g(y)$. However, this is itself a difficult and unsolved problem. So, the ideal case cannot be achieved, but represents an area of possible future research.) Then the method of either Chapter 3 or Chapter 4 can be applied to obtain the approximate solution of this modified problem (i.e., with nondecreasing concave cost function). Finally, the following somewhat arbitrary local improvement procedure can be used to obtain an approximate solution of the original discrete variable problem. Some reasons for choosing this particular procedure will be discussed in the remarks following the procedure.

Assume there are n source-sink requirement pairs, and number these pairs arbitrarily, from 1 to n.

Algorithm:

Step 0:

Let $i = 1$.

Step 1:

Let y_m be the flow in arc m in the current solution to the problem. Consider the ith source-sink pair. Find the chain, call it chain P, through which the entire flow of the source-sink pair i was passed in the solution of the continuous approximate problem. (Both the procedures of Chapter 3 and of Chapter 4 give rise to a solution with a single chain between each source-sink pair.) Let w_m equal the largest flow value with cost less than the cost of flow y_m. Let $t_m = f_m(y_m) - f_m(w_m)$. For each arc $m \in P$, let $Z_m = y_m - w_m$ (see Figure 5.1). (If the required flow between any source-sink pair is integral (assumed here), then the flow level y_m in every arc is also integral (as proved in Appendix C). Hence, $f_m(y_m)$ is defined. Also, C_m, defined below is integral.)

Let $Z = \underset{m \in P}{\text{Minimum}} \{Z_m\} = Z_{m'}$. Then reduce the flow in chain P for source-sink pair i by amount Z. This guarantees a reduction of the cost by at least $t_{m'}$, (possibly more because there can be more than one arc at which the minimum is attained). If Z is greater than existing flow in the chain P corresponding to ith pair, go to Step 3.

Step 2:

For the resulting flow pattern, determine, for all m, the value, called C_m, of the amount of flow that can be sent through an arc m

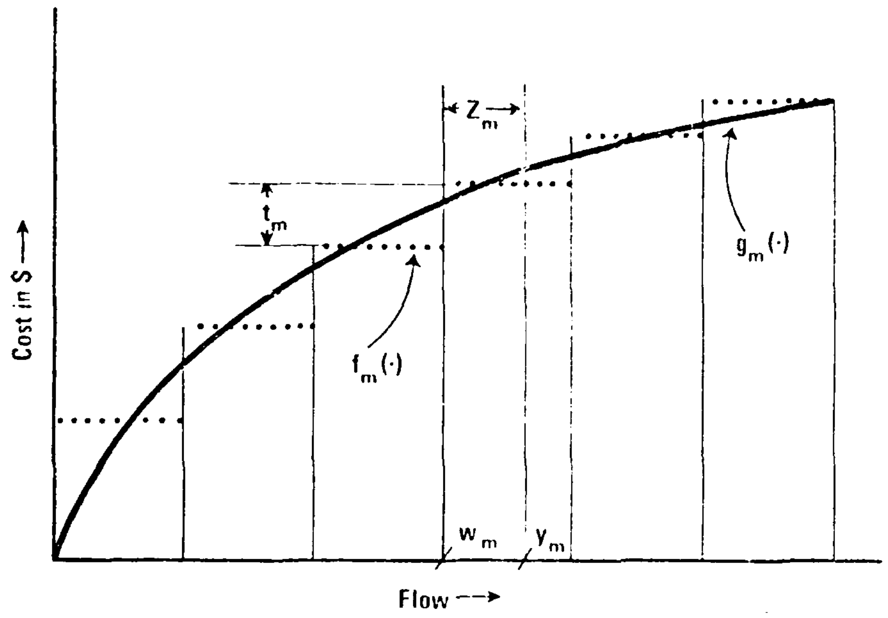


FIGURE 5.1

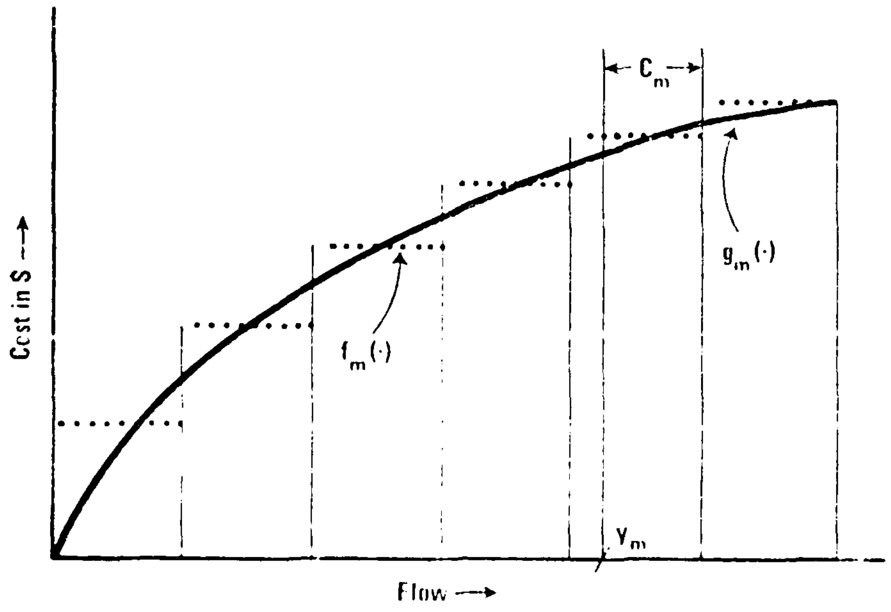


FIGURE 5.2

without incurring any further cost (see Figure 5.2).

Using C_m as the capacity for each arc m, find the maximum flow F between ith source-sink pair by the Ford and Fulkerson max-flow labelling algorithm as refined by Karp and Edmonds [E-1]. (The network considered here is undirected and the labelling algorithm mentioned above is for directed networks. So, it is necessary to convert the undirected network to a directed one by replacing each undirected arc by two directed arcs of opposite directions. From now on, all chains will be referred as paths because they are directed.)

If max flow $F < Z$, then restore Z amount of flow back to the path in Step 1.

If $F \geqq Z$, then further reduce the flow in the path determined in Step 1 by amount $(F - Z)$. Send amount F of flow through the paths determined by the max-flow algorithm.

Note:

The C_m value for at least one of the arcs in the path traced in Step 1 is zero (at m' where m' minimizes Z_m). So, the paths obtained by the max-flow algorithm do not contain the previous path.

Step 3:

Add 1 to the value of i. If $i \leqq n$, go to Step 4. If $i = n + 1$, let $i = 1$ and go to Step 4.

Step 4:

If $F < Z$ for n consecutive steps, then stop. If not, go to Step 1.

Theorem 5.1:

The algorithm described above terminates in a finite number of steps.

Proof:

The finite termination of the algorithm depends on the finite termination of any sequence of flow reduction steps and the finite termination of the max-flow algorithm in Step 2. Since all of the capacities C_m at any stage are integral, the refined procedure of Karp and Edmonds will yield a finite bound of $(1 + \log_{c/c-1} F)$ for each max-flow problem where c is the maximum number of arcs across a cut-set and F is the maximum flow.

In Step 2, if any flow change takes place then there is a decrease in the total cost by at least the amount of the smallest discontinuity in any cost function, since it does not cost anything to send the flow through the alternate paths found by the max-flow algorithm. The monotonic decrease in cost guarantees no cycling. Only a finite number of improvements is possible since the total cost is bounded below by zero (i.e., total cost $\geqq 0$) . Also, if no improvement is obtained n times consecutively, the algorithm stops. Thus, termination of the algorithm is achieved in a finite number of steps.||

5.2 Remarks

The following remarks are pertinent to the above procedure:

(i) This is a procedure to be used after a flow pattern has been found by the method of either Chapter 3 or Chapter 4 based on a continuous (except at left-hand extreme point) approximation of discrete cost data. There is no guarantee that we will obtain the global optimal point for the discrete problem. The concept of local optimal value is not applicable here because the variables (flows) are discrete.

(ii) It is not apparent whether we could have obtained more reduction in the cost by considering source-sink pairs in a different order.

(iii) For any set of source-sink pairs, some flow may be diverted from the initial unique path to one or more paths found by solving max-flow problems. Yet Step 1 considers only the original path and not the deviations determined by previous max-flow solutions. These other paths are not considered because the flow in these paths is generally small compared to the flow in the original path. Hence, the minimum reduction Z (obtained in Step 1) may be more than the flow values of these paths. Then, no cost improvement could take place.

(iv) In Step 3, if $F < Z$, then no improvement, with respect to a given source-sink pair, in the total cost is possible using our algorithm. It is possible that an improvement can be made by using flow increases greater than C_m. While the rerouting is then not free, it is possible that F can be made larger than Z at a smaller cost increase than the decrease we obtain by reducing the flow in P by the amount F. To do this, we have to consider the relative sizes of steps in the cost functions. This is a rather involved procedure and was not attempted here.

(v) The values of Z_m and C_m depend on the width of the steps in the cost curve (i.e., the difference between flow values at two consecutive steps in the cost curve). The value of t_m depends on the height of the steps in the cost function.

There is not much improvement possible if step widths and heights are very small--so, in that case, this procedure is not advisable. In case the cost functions consist of a few big steps, this procedure tends to give reasonable savings.

(vi) In the case when the cost function is similar to Figure 1.4, this procedure can be applied with the redefinition of Z_m and C_m as shown in Figure 5.3.

5.3 Future Research Directions for This Problem

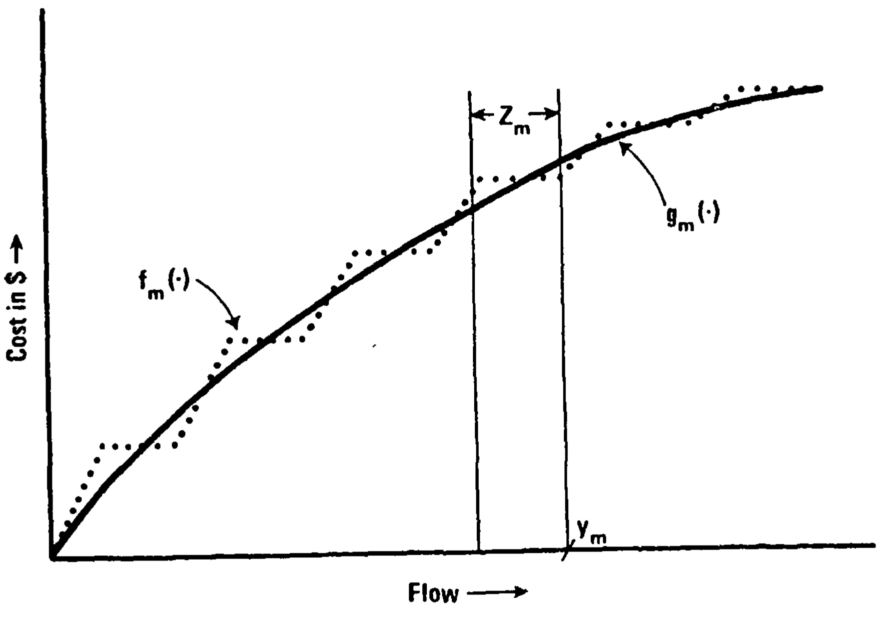
The most promising theoretical direction to pursue in solving the general network synthesis problem with continuous concave cost functions seems to be the cutting plane method proposed by Ritter. In this method, each local search procedure should be a simple subproblem (such as a shortest path problem), and the generation of cutting planes to exclude local optimal solutions without cutting out the global optimal solution, should be simple enough to be handled easily, even for large problems.

To make the procedure of Chapter 4 attractive, it is necessary to find a good upper bound of the optimal flow in each arc. Research toward finding a reasonably good method could prove fruitful.

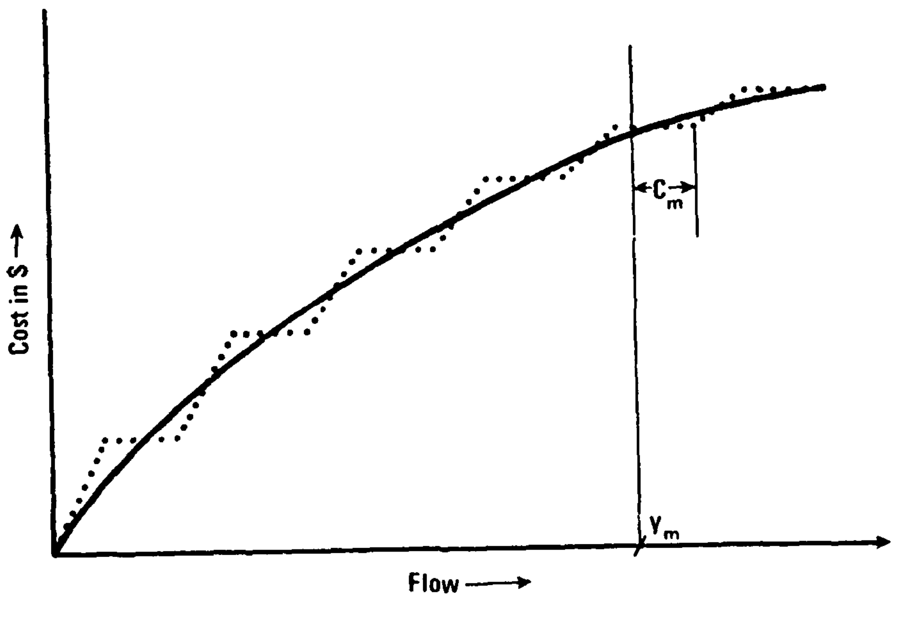
Finally, research seeking better methods of generating a good solution of the discrete cost problem, given a solution of the continuous approximate problem, may prove worthwhile. But the problem is very difficult and the procedure will almost certainly have to be heuristic.

5.4 A More General Research Problem

A somewhat more general, and realistic, situation than discussed earlier is the case where the requirement matrix changes over different time periods. The capacity should be built, at minimum cost, at the start



(a)



(b)

FIGURE 5.3

of the process so as to meet the requirements for all succeeding times, using different routing during different time periods. The programming formulation is given as follows:

$X_{ij}(t)$ = A vector of dimension P_{ij} where its elements $x_{ij}^{k}(t)$ represent the amount of flow through the kth chain between source-sink pair $i\,j$ at time t .

$A_{ij} = M \times P_{ij}$ path-edge incidence matrix, whose (m,k) element is 1 if the kth chain between source-sink pair $i\,j$ traverses edge m or 0 if it does not.

$Y(t) = \sum_{i,j \ni i>j} A_{ij}X_{ij}(t)$ = An M-vector where element $y_m(t)$ is the total flow in edge m at time t .

C = An M-vector where element C_m is the total capacity necessary in edge m such that it meets the requirements for $t = 1,2, \ldots, T$ time periods.

Here we have to minimize the total cost Z where

$$Z = \sum_{m=1}^{M} f_m(C_m) .$$

Subject to $C_m \geqq y_m(t)$ $\forall$ $t = 1,2, \ldots, T$ and for all arcs m .

$$\sum_{k=1}^{P_{ij}} x_{ij}^{k}(t) \geqq r_{ij}(t) \quad \text{for all source-sink } i\,j \text{ and all } t = 1,2, \ldots, T$$

$$x_{ij}^{k}(t) \geqq 0 \quad \text{for all } t \text{ and } i\,j .$$

In this formulation, the number of variables has increased enormously. An efficient solution procedure is not in sight and can only be found if the problem of this thesis is solved first.

However, an even more complex problem faces the telephone company. The requirement matrix changes over succeeding years and can be forecast at the beginning of the planning period. The cost of rearrangements (i.e., dismantling the facilities between certain nodes and putting them between certain other nodes in different periods) of facilities is also given. An initial network has to be synthesized which will meet the initial capacity requirements. Additions and rearrangements in succeeding years must be planned. There may also be yearly budgetary constraints. All of these must be solved in such a way that discounted total cost over several years is minimized. This problem is much more complex than the one attempted in this thesis and a heuristic solution procedure can be a challenging extension of this work.

APPENDIX A

```
AAS128      02/07/72

5C  LANGUAGE: FORTRAN FOUR, GE 635 (MARK TWO) COMPUTER
10C  132 K MACHINE, 1 MICRO-SECOND MEMORY CYCLE
15C  SIMILAR TO IBM 360-65 OR CDC 6400.
20C PROGRAM FOR A LOCAL SEARCH ALGORITHM OF SOLVING A
25C MULTI-COMMODITY FLOW PROBLEM UNDER CONCAVE COST.
30C GIVEN FLOW REQUIREMENT MATRIX R, COST MATRIX K
35C COST OF F(I,J) FLOW ON ARC (I,J)=K(I,J)*F(I,J)**Z.
40C N=NUMBER OF NODES IN THE NETWORK.
45C INPUT DATA FILES SHOULD SUPPLY VALUES OF R & K.
50 FILENAME ZDATA
55 INTEGER R(34,34),CH(34,34),C(34,34),F(35,35),S
60 INTEGER Y
65 REAL K(34,34),D(34,34),CO(34,34)
70 10 FORMAT (5X,F9.2,I5)
75 20 FORMAT (5X,F8.3)
80 30 FORMAT (I3,4X,I3,I4)
85 40 FORMAT (3X,I3)
90 50 FORMAT (2I3,I4)
95C SUPPLY THE INFORMATION ABOUT N,Z,INPUT FILE,
100C VALUE OF COEFFICIENT OF CONVEX COMBINATION B
105C AND INITIAL NUMBER OF SAMPLE NO.
110 PRINT,"N,Z,ZDATA,B,NO"
115 INPUT,N,Z,ZDATA,B,NO
120 N1=N-1
125 DO 100 I=1,N1
130 I1=I+1
135 DO 100 J=I1,N
140 100 READ (ZDATA,10) K(I,J),R(I,J)
145 PRINT,I,J,K(I,J),R(I,J)
150C STARTING FROM A RANDOM POINT
155 S=0 ; ITRAC=0
160 150 S=S+1
165 DO 170 I=1,N1 ; I1=I+1
170 DO 170 J=I1,N
175 IF(K(I,J).LT.1000) GO TO 167
180 D(I,J)=1000 ; GO TO 170
185 167 Y=RRAND(1.0) ;X=RND(1.0)
190 F(I,J)=(100*X+10*Y)/2
195 PRINT 50,I,J,F(I,J)
200 D(I,J)=(Z*K(I,J))/(F(I,J)**(1-Z))
205 170 D(J,I)=D(I,J)
210C  CREATION OF INITIAL CH(I,J) MATRIX
215 260 DO 270 I=1,N
220 DO 270 J=1,N
225 270 CH(I,J)=J
230 DO 300 I=1,N
235 300 D(I,I)=0
240C   STARTING OF ITERATIVE PROCESS.
245 ITRA=0;NCOV=0;NCO=0
250C CREATION OF C(I,J) MATRIX
```

AAS128   02/07/72

```
400 DO 450 I=1,N
DO 450 J=1,N
450 C(I,J)=J
C  FINDING THE SHORTEST PATH
DO 500 M 1,N
DO 500 I=1,N1
I1=I+1
IF(D(I,M) .GE. 1000) GO TO 500
DO 500 J=I1,N
A=D(I,M)+D(M,J)
IF (D(I,J).LE.A) GO TO 500
D(I,J)=D(I,M)+D(M,J)
  D(J,I)=D(I,J)
C(I,J)=C(I,M)
C(J,I)=C(J,M)
500 CONTINUE
C  CHECKING FOR CONVERGENCE
  ICHE=0
DO 520 I=1,N
DO 520 J=1,N
 IF (CH(I,J) .EQ. C(I,J)) GO TO 510
  CH(I,J)=C(I,J)
 GO TO 520
510 ICHE=ICHE+1
520 CONTINUE
NCHE=N*N
IF(ICHE.EQ.NCHE) NCO=2
IF(ICHE.EQ.NCHE .AND. S.EQ.1) NCOV=2
IF(ICHE.EQ.NCHE .AND. COSTT.GT.COST) NCOV=2
C UPDATING THE FLOW MATRIX
DO 550 I=1,N1
I1=I+1
DO 550 J=I1,N
550 F(I,J)=0
IF(NCOV .EQ. 0) GO TO 555
PRINT,"FROM,BT,TO,FLOW"
555 DO 600 I=1,N1
I1=I+1
DO 600 J=I1,N
IF(R(I,J).EQ.0) GO TO 600
IF (NCOV.NE.0)PRINT 30,I,J,R(I,J)
X=R(I,J)
Y=I
560 L=C(I,J)
IF (I.GT.L) GO TO 570
F(I,L)=(I,L)+X
GO TO 580
570 F(L,I)=F(L,I)+X
580 IF (L.EQ.J) GO TO 590
IF(NCOV.NE.0) PRINT 40,L
```

```
AAS128      02/07/72

505  I=L
510 GO TO 560
515 590 I=Y
520 600 CONTINUE
525 IF (NCO .EQ. 2) GO TO 750
530 IF (NCOV .EQ. 1) NCOV=0
535C CALCULATION OF NEW DISTANCE MATRIX
540 COST=0
545 DO 700 I=1,N1
550 I1=I+1
555 DO 700 J=I1,N
560 CO(I,J)=(K(I,J))*((F(I,J))**Z)
565 COST=COST+CO(I,J)
570 IF (F(I,J).EQ.0) F(I,J)=1
575 D(I,J)=((Z*K(I,J))/(F(I,J)**(1-Z)))*B+(1-B)*(D(I,J))
580 D(J,I)=D(I,J)
585 700 CONTINUE
590 PRINT,"TOTAL COST=",COST
595 ITRA=ITRA+1
600 GO TO 400
605 750 PRINT,"CONVVERGENCE AT ITERATION NO =",ITRA
610 ITRAC=ITRAC+ITRA
615 IF (S.GT.1)GO TO 780
620 COSTT=COST : COSTU=COST
625 GO TO 150
630 780 IF (COSTT .GE. COST) COSTT=COST
635 IF ( COSTU .LT. COST) COSTU=COST
640 IF (S.LT.NO) GO TO 150
645 ERR=(COSTU-COSTT)/(S-1)
650 IF (S.NE.NO) GO TO 850
655 CM=ITRAC/S
660 PRINT,"ERR=",ERR,"CM=",CM
665C SUPPLY THE COEFFICIENT OF THE LOSS FUNCTION AND
670C COMPUTATION COST FUNCTION.
675 PRINT, "E,CN"
680 INPUT, E,CN
685 850 IF((E*ERR).LT.(CN*CM)) GO TO 1000
690 NM=NO+2
695 IF (S .GT. NM) GO TO 1000
700 GO TO 150
705 1000 PRINT,COSTT,S,ERR
710 STOP ;END
```

APPENDIX B

```
AASBR1       02/03/72

C LANGUAGE: FORTRAN FOUR, GE 635 (MARK TWO) COMPUTER
C  132 K MACHINE, 1 MICRO-SECOND MEMORY CYCLE
C  SIMILAR TO IBM 360-65 OR CDC 6400.
C PROGRAM FOR A GLOBAL SEARCH (BRANCH-BOUND) ALGORITHM
C OF SOLVING A MULTI-COMMODITY FLOW PROBLEM UNDER
C CONCAVE COST. GIVEN FLOW REQUIREMENT MATRIX R,COST
C MATRIX KS, N=NUMBER OF NODES IN THE NETWORK, COST OF
C FY(I,J) FLOW IN ARC (I,J)=KS(J,I)+KS(I,J)*FY(I,J)**Z
C INPUT DATA FILE SHOULD SUPPLY VALUES OF R & KS
COMMON DY,DX,FY,R,CY,CX,N,NE,K,L
REAL DY(35,35),DX(35,35),ZA(100),ZL(100),KS(35,35)
REAL D(1260)
INTEGER S(100),KE(100),LE(100),CY(35,35),CX(35,35),RM
INTEGER C(1260)
INTEGER R(35,35),FY(35,35),LB(35,35),F(100)
FILENAME ZDATA
10 FORMAT (5X,F9.2,I5,F8.2)
20 FORMAT (I3,4X,I3,I4)
30 FORMAT (3X,I3)
32 FORMAT (I7,F10.2)
34 FORMAT (I7,2F10.2)
C SUPPLY THE VALUES OF N,Z,INPUTFILE,NUMBER OF PAGES
C FOR EACH DATA BLOCK(=NUMBER OF VARIABLES/315)
PRINT, "N,Z,ZDATA,NP"
INPUT, N,Z,ZDATA,NP
N1=N-1 ; RM=10000
DO 100 I=1,N1; I1=I+1
DO 100 J=I1,N
READ (ZDATA,10) KS(I,J),R(I,J),KS(J,I)
100 IF((R(I,J).GT.0).AND.(R(I,J).LE.RM))RM=R(I,J)
DO 150 I=1,N1; I1=I+1
DO 150 J=I1,N
IF (KS(I,J).LT. 1000) GO TO 120
DY(I,J)=1000
GO TO 150
120 DY(I,J)=((KS(J,I)+KS(I,J)*(RM**Z))/RM)
150 DY(J,I)=DY(I,J)
DO 160 I=1,N
DY(I,I)=0
160 CX(I,I)=I
NE=0
CALL SHORTEST
C OBTAINING BOUNDS ON FLOW ON EACH ARC.
DO 300 K=1,N1; K1=K+1
DO 300 L=K1,N
IF (KS(K,L).GE. 1000) GO TO 290
DX(K,L)=0
CALL ARCHANGE
R(L,K)=Y(K,L)
DY(K,L)=(KS(L,K)+(KS(K,L))*((R(L,K))**Z))/R(L,K)
```

```
AAS3R1      Ø2/Ø3/72

255 GO TO 3ØØ
26Ø 29Ø DY(K,L)=1ØØØ
265 3ØØ DY(L,K)=DY(K,L)
27ØC STARTING OF BRANCH & BOUND PROCEDURE.
275C CREATING A RANDOM BINARY FILE TO STORE THE DATAS.
28Ø OPENFILE SDATA(**,"R",1,252ØØØ)
285 CALL IF BREAK($75Ø)
29Ø NE=1 :NF=1
295 NUM=1
3ØØ CALL SHORTEST
3Ø5 DO 33Ø I=1,1ØØ
31Ø 33Ø S(I)=Ø
315 S(1)=1; II=1
32Ø DO 35Ø I=1,N1: II=I+1
325 DO 35Ø J=II,N
33Ø L2(I,J)=R(J,I)
335 35Ø L3(J,I)=Ø
34Ø GO TO 6ØØ
345 4ØØ K=KE(NF): L=LE(NF)
35Ø NUM=NUM+1
355 IF (NF.EQ.NE) GO TO 42Ø
36Ø IP=2*(NF-1)*NP+1
365 CALL UREAD ( SDATA,D,IP,NP)
37Ø IP=IP+NP
375 CALL UREAD (SDATA,C,IP,NP)
38Ø NU=N*N
385 DO 415 M=1,NU
39Ø J=INT((M-1)/N)+1
395 I=M-N*(J-1)
4ØØ CY(I,J)=C(M)
4Ø5 IF (I.LE.J) GO TO 41Ø
41Ø L3(J,I)=INT(D(M))
415 L3(I,J)=(D(M)-L3(J,I))*1ØØØ
42Ø GO TO 415
425 41Ø DY(I,J)=D(M)
43Ø DY(J,I)=DY(I,J)
435 415 CONTINUE
44Ø 42Ø DY(K,L)=KS(K,L)*(L3(K,L)**2-F(NF)**2)/(L3(K,L)-F(NF))
445 CALL ARCHANGE
45Ø L3LK=L3(L,K)
455 L3(L,K)=F(NF)
46Ø DO 43Ø I=1,II
465 43Ø IF(S(I).EQ.Ø) GO TO 44Ø
47Ø 44Ø NE=I ; S(NE)=1
475 GO TO 55Ø
48Ø 52Ø NE=NF
485 NUM=NUM+1
49Ø 5Ø5 DO 54Ø I=1,N1; II=I+1
495 DO 54Ø J=II,N
5ØØ IF (KS(I,J).GE. 1ØØ4) GO TO 53Ø
```

```
AASBRI      02/08/72

      IF(L3(J,I).EQ.0)GO TO 517
      DY(I,J)=KS(I,J)*(L3(I,J)**Z-L3(J,I)**Z)/(L3(I,J)-L3(J,I))
      GO TO 540
  517 DY(I,J)=(KS(J,I)+KS(I,J)*(L3(I,J)**Z))/L3(I,J)
      GO TO 540
  530 DY(I,J)=1000
  540  DY(J,I)=DY(I,J)
      IF(IRE.EQ.1) GO TO 548
      IF (L3LK.EQ.0) GO TO 543
      DY(K,L)=KS(K,L)*((F(NE)**Z)-(L3LK**Z))/(F(NE)-L3LK)
      GO TO 546
  543 DY(K,L)=(KS(L,K)+KS(K,L)*(F(NE)**Z))/F(NE)
  546 DY(L,K)=DY(K,L)
      L3(K,L)=(NE)
      L3(L,K)=L3LK
  548 CALL SHORTEST
      IRE=0
  550 DO 555 I=1,N1; II=I+1
      DO 555 J=II,N
      M=(J-1)*N+I
      D(M)=DY(I,J)
      IF (FY(I,J).LE.R(J,I)) GO TO 555
      IF (FY(I,J).LE.L3(I,J)) GO TO 555
      L3(I,J)=MAX0(FY(I,J),2*[illegible]J,I))
      MI=(I-1)*N+J
      D(MI)=L3(I,J)+(L3(J,I)/1000)
      IRE=1
  555 CONTINUE
      IF(IRE.EQ.1) GO TO 545
      DO 560 I=1,N
      DO 560 J=1,N
      M=(J-1)*N+I
  560 C(M)=CY(I,J)
      IP=2*(NE-1)*NP+1
      CALL UWRITE (DATA,D,IP,NP)
      IP=IP+NP
      CALL UWRITE (DATA,C,IP,NP)
C COST CALCULATIONS
  600 ZL(NE)=0; ZA(NE)=0 ; TRR=0
      DO 650 I=1,N1; II=I+1
      DO 650 J=II,N
      IF(KS(I,J) .GE. 1000) GO TO 650
      ZAA=KS(J,I)+KS(I,J)*(FY(I,J)**Z)
      IF(L3(J,I).EQ.0) GO TO 630
      ZLL= DY(I,J)*(FY(I,J)-L3(J,I))+KS(J,I)+KS(I,J)*(L3(J,I)**Z)
      GO TO 640
  630 ZLL=DY(I,J)*FY(I,J)
  640 ZA(NE)=ZA(NE)+ZAA
      ZL(NE)=ZL(NE)+ZLL
      DX(J,I)=ZAA-ZLL
```

```
AAS3RI       02/08/72

755 IF (ERR.GE.DX(J,I)) GO TO 550
760 ERR=DX(J,I); KE(NE)=I;LE(NE)=J
765 F(NE)=FY(I,J)
770 550 CONTINUE
775 IF(II.NE.1) GO TO 670
780 PRINT,ERR,ZA(NE),ZL(NE)
785 PRINT,"VALUE OF ERROR PERMITTED EP"
790 INPUT,EP
795 ZB=ZA(1)
800 670 IF(ZA(NE).GT.ZB)GO TO 680
805 ZB=ZA(NE);NB=NE ;ZC=ZB-EP
810 PRINT 32,NB,ZB
815 680 IF(NE.EQ.II) II=II+1
820 PRINT 34,NE,ZA(NE),ZL(NE)
825 IF(NE.NE.NF) GO TO 500
830 DO 700 I=1,II
835 IF (S(I).EQ.0) GO TO 700
840 IF (ZL(I).GE.ZC) S(I)=0
845 700 CONTINUE
850C CHECKING FOR TERMINATION
855 CHEC=0
860 DO 710 I=1,II
865 710 IF(S(I).EQ.0) CHEC=CHEC+1
870 IF (CHEC.EQ.II) GO TO 750
875 ZLI=ZL(1)
880 DO 720 I=1,II
885 IF(S(I).EQ.0) GO TO 720
890 IF(ZL(I).GT.ZLI) GO TO 720
895 ZLI=ZL(I); NF=I
900 720 CONTINUE
905 GO TO 400
910 750 PRINT,"LEAST COST SOLUTION=",ZB,"ERROR PERMITED=" ,EP
915 PRINT,"NUMBER OF EVALUATION = ", NUM
920 IP=2*(NB-1)*NP+NP+1
925 CALL UREAD (SDATA,D,IP,NP)
930 PRINT,"FROM,PT ,TO,FLOW"
935 DO 755 M=1,NU
940 J=INT((M-1)/N)+1
945 I=M,N*(J-1)
950 755 CY(I,J)=C(M)
955 DO 780 I=1,N;II=I+1
960 DO 780 J=II,N
965 IF(R(I,J).EQ.0) GO TO 780
970 PRINT 20, I,J,R(I,J)
975 NY=I
980 760 LO=CY(NY,J)
985 IF (LO.EQ.J) GO TO 780
990 PRINT 30, LO ; NY=LO
995 GO TO 760
1000 780 CONTINUE
```

AASBRI   02/08/72

```
1005 STOP ;END
1010 SUBROUTINE SHORTEST
1015 COMMON DY,DX,FY,R,CY,CX,N,NE,K,L
1020 REAL DY(35,35),DX(35,35)
1025 INTEGER N,NE,CY(35,35),CX(35,35),FY(35,35),R(35,35)
1030 NI=N-1
1035 DO 850 I=1,N
1040 DO 850 J=1,N
1045 850 CY(I,J)=J
1050 DO 862 M=1,N
1055 DO 862 I=1,NI;II=I+1
1060 IF(DY(I,M).EQ.1000) GO TO 862
1065 DO 860 J=II,N
1070 IF(DY(I,J).LE.(DY(I,M)+DY(M,J))) GO TO 860
1075 DY(I,J)=DY(I,M)+DY(M,J)
1080 DY(J,I)=DY(I,J)
1085 CY(I,J)=CY(I,M); CY(J,I)=CY(J,M)
1090 860 CONTINUE
1095 862 CONTINUE
1100 IF(NE.EQ.0) RETURN
1105 DO 870 I=1,NI;II=I+1
1110 DO 870 J=II,N
1115 870 FY(I,J)=0
1120 DO 900 I=1,NI;II=I+1
1125 DO 900 J=II,N
1130 IF(R(I,J).EQ.0) GO TO 900
1135 NX=R(I,J) ;NY=I
1140  880 LO=CY(NY,J)
1145 IF(NY.GT.LO) GO TO 885
1150 FY(NY,LO)=Y(NY,LO)+NX
1155 GO TO 890
1160 885 FY(LO,NY)=FY(LO,NY)+NX
1165 890 IF(LO.EQ.J) GO TO 900
1170 NY=LO
1175 GO TO 880
1180 900 CONTINUE
1185 RDTURN
1190 END
1195 SUBROUTINE ARCHANGE
1200 COMMON DY,DX,FY,R,CY,CX,N,NE,K,L
1205 REAL DY(35,35),DX(35,35)
1210 INTEGER N,K,L,CY(35,35),CX(35,35),FY(35,35),R(35,35)
1215 NI=N-1
1220 DO 920 I=1,NI;II=I+1
1225 DO 920 J=II,N
1230 AD=DY(I,K)+DX(K,L)+DY(L,J)
1235 IF (DY(I,J).LE.AD) GO TO 910
1240 DX(I,J)=AD;CX(I,J)=CY(I,K);CX(J,I)=CY(J,L)
1245 IF(I.DQ.K) CX(I,J)=L
1250 IF(J.EQ.L) CX(J,I)=K
```

```
AASBR1        02/08/72

GO TO 920
910 AD=DY(I,L)+DX(K,L)+DY(K,J)
IF(DY(I,J).LE.AD) GO TO 915
DX(I,J)=AD;CX(I,J)=CY(I,L);CX(J,I)=CY(J,K)
IF(I.EQ.L) CX(I,J)=K
IF(J.EQ.K) CX(J,I)=L
GO TO 920
915 CX(I,J)=CY(I,J); CX(J,I)=CY(J,I); DX(I,J)=DY(I,J)
920 CONTINUE
DO 930 I=1,N1 ;I1=I+1
DO 930 J=I1,N
930 FY(I,J)=0
DO 980 I=1,N1 ;I1=I+1
DO 980 J=I1,N
IF(R(I,J).EQ.0) GO TO 980
NX=R(I,J); NY=I
940 LO=CX(NY,J)
IF(NY.GT.LO) GO TO 950
FY(NY,LO)=FY(NY,LO)+NX
GO TO 960
950 FY(LO,NY)=Y(LO,NY)+NX
960 IF(LO.EQ.J) GO TO 980
NY=LO
GO TO 940
980 DY(I,J)=DX(I,J)
DO 985 I=1,N
DO 985 J=1,N
CY(I,J)=CX(I,J)
985 IF (I.LT.J) DY(J,I)=DY(I,J)
RETURN
END
```

THE COMPUTATIONAL RESULT OF A NUMBER OF PROBLEMS IS LISTED INTHE TABLE OF NEXT PAGE.

TABLE OF COMPUTATIONAL RESULTS FOR THE GLOBAL SEARCH PROCEDURE

Number of nodes.	Number of arcs.	Number of pairs of requirements points.	Number of problems using different cost functions.[1]	Permissible error as % of C_o.[2]	Average number of iterations (i.e., subproblems).	Average computation time per problem in CRU's.[3]
6	12	10	4	1.0%	22	5.0
6	12	10	4	2.0%	19	4.1
15	36	25	4	1.0%	70	100
15	36	25	4	3.0%	55	80
22	80	100	3	1.0%	95	340
22	80	100	3	3.0%	75	280
35	100	280	1	2%	156	1520
35	100	280	2	4%	130	1300

[1] By using different values of the parameter Z in the cost function in the program.

[2] C_o is the guessed value of the minimum cost based on the cost data.

[3] A CRU is approximately 1 second of computing time.

APPENDIX C

The solution procedures considered for this problem always use continuous functions and continuous variables. However, except for the solution method, this assumption of continuity of the cost function is not needed. The values of the cost function at integral points are enough data to get the total cost. If the requirements (r_{ij}) are integers, then the following observation shows that all the extreme points (one of which is found at each iteration of our procedures) of the convex polyhedron are also integral.

The problem is:

$$\text{Minimize} \quad Z = \sum_{m=1}^{M} f_m(y_m) \ ,$$

subject to $x_{ij}^k \geqq r_{ij} \ \forall$ (i j pairs) and $x_{ij}^k \geqq 0$, where $\sum_{\text{all } i,j \ni i>j} A_{ij} X_{ij} = Y$.

This is equivalent to:

$$\text{Minimize} \quad Z(X) = \sum_{m=1}^{M} f_m\left(\sum_{i,j \ni i>j} a_{ij}^m X_{ij} \right) ,$$

subject to $DX \geqq R$, $x_{ij}^k \geqq 0$, where a_{ij}^m is the mth column of the matrix A_{ij} and D is a $\left(\sum_{i,j} P_{ij}\right) \times n$ matrix (n = the number of source-sink pairs) with P_{ij} 1's in each row and $R = \{r_{ij}\}$, an n-vector. Now,

$$D = \begin{bmatrix} 1111 & & & \\ & 111 & & \\ & & 11111 & \\ & & & \ddots \end{bmatrix}$$

is a totally unimodular matrix.[1]

The theorem of Hoffman and Kruskal [Page 125, Ref. H-3] proves that if the vector R is integral then all the extreme points of the polyhedron defined by the constraints are integral.

[1]Definition: A matrix is said to be totally unimodular if and only if every subdeterminant of D is equal to +1 , -1 , or 0 .

APPENDIX D

The property of concave nondecreasing functions established below is used in Chapter 3.

<u>Theorem</u>:

Let $f(y)$ be: (i) a concave function in $y \in [0,a]$, (ii) $f(y) \geqq 0$ (i.e., it is bounded below) and (iii) nondecreasing in y , then

(a) $f(y)$ is continuous (except perhaps at 0) . If it is not nondecreasing, then it is not necessarily continuous at point a .

(b) Both the left hand and the right hand derivatives D^-f and D^+f exist at all points (except at 0 where D^-f is not defined and D^+f may not be finite; and at a where D^+f is not defined).

(c) The following inequalities are true:

(i) $D^-f(y_1) \geqq D^-f(y_2) \quad \forall \quad y_2 > y_1$

(ii) $D^+f(y_1) \geqq D^+f(y_2) \quad \forall \quad y_2 > y_1$

(iii) $D^-f(y) \geqq D^+f(y) \quad \forall \quad y \in (0,a)$.

<u>Proof</u>:

This proof has been done by Hardy, Polya and Littlewood [Page 91, Ref. H-2] for any convex function defined in the open interval (H,K) which is bounded above in some interval i interior to (H,K) .[1] If

[1]The inequalities (i), (ii) and (iii) are the reverse of the inequalities in the theorem of Hardy, Polya and Littlewood because they are for convex functions.

$f(y)$ is a concave function, $-f(y)$ is convex. Condition (ii) guarantees a lower bound for the concave function which is equivalent to the upper bound of the convex function $-f(y)$. Here $(H,K) = (0,a)$. So, the proof is exactly similar. Condition (iii) guarantees that at a there cannot be a discontinuity because then the value of f would have to decrease.

Thus, this theorem is valid for a piecewise linear concave function (a continuous version of the function as shown in Figure 1.5).

REFERENCES

C-1 Cabot, A. Victor and Richard L. Francis, "Solving Some Non-Convex Quadratic Minimization Problem by Ranking the Extreme Points," J. ORSA, Vol. 18, No. 1, pp. 82-86, (January-February 1970).

C-2 Chien, R. T., "Synthesis of a Communication Net," IBM J., Vol. 4, pp. 311-320, (1960).

C-3 Cottle, R. W. and W. C. Mylander, "Ritter's Cutting Plane Method for Nonconvex Quadratic Programming," INTEGER AND NONLINEAR PROGRAMMING, (J. Abadie, ed.), North Holland, Amsterdam, (1970).

D-1 Degroot, Morris H., OPTIMAL STATISTICAL DECISIONS, McGraw-Hill Book Company, pp. 198-201, (1970).

D-2 Dreyfus, S. E., "An Appraisal of Some Shortest Path Algorithms," J. ORSA, Vol. 17, No. 3, (May-June 1969).

E-1 Edmonds, Jack and Richard M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," ORC 70-24, Operations Research Center, University of California, Berkeley, (1970).

E-2 Edwards, Ward, Harold Lindman and Leonard J. Savage, "Bayesian Statistical Inference for Psychological Research," READINGS IN MATHEMATICAL PSYCHOLOGY, Vol. II by Luce, Bush and Galanter, John Wiley & Sons, pp. 519-568, (1964).

F-1 Falk, John E. and Richard M. Soland, "An Algorithm for Separable Nonconvex Programming Problems," Management Science (Theory), Vol. 14, No. 9, (May 1968).

F-2 Floyd, R. W., "Algorithm 97: Shortest Path," Communication of the Association for Computing Machinery, Vol. 5, p. 345, (1962).

F-3 Ford, L. R. and D. R. Fulkerson, "A Suggested Computation for Maximal Multicommodity Network Flows," Management Science, Vol. 5, pp. 97-101, (1958).

F-4 Ford, L. R. and D. R. Fulkerson, FLOWS IN NETWORKS, Princeton University Press, Princeton, New Jersey, (1962).

G-1 Goldstein, M. and B. Rothfarb, "The One Terminal Telpak Problem," J. ORSA, Vol. 19, No. 1, pp. 156-169, (January-February 1971).

G-2 Gomory, R. E. and T. C. Hu, "Synthesis of Communication Networks," J. SIAM, Vol. 12, No. 2, pp. 348-369, (June 1964).

G-3 Gomory, R. E. and T. C. Hu, "An Application of Generalized Linear Programming to Network Flows," J. SIAM, Vol. 10, No. 2, pp. 260-283, (June 1962).

G-4 Gomory, R. E. and T. C. Hu, "Multi Terminal Network Flows," J. SIAM, Vol. 9, No. 4, pp. 551-570, (December 1961).

G-5 Grinold, Richard C., "A Multicommodity Max-Flow Algorithm," J. ORSA, Vol. 16, No. 6, pp. 1234-1238, (November-December 1968).

H-1 Hadley, G., LINEAR PROGRAMMING, Addison-Wesley, Reading, Massachusetts, (1969).

H-2 Hardy, G. H., J. E. Littlewood and G. Polya, INEQUALITIES, Cambridge University Press, (1964).

H-3 Hu, T. C., INTEGER PROGRAMMING AND NETWORK FLOWS, Addison-Wesley, Reading, Massachusetts, (1969).

H-4 Hu, T. C., "The Developments of Network Flow and Related Areas in Programming," invited survey lecture delivered at the Seventh International Mathematical Programming Symposium at the Hague, The Netherlands, (September 16, 1970).

J-1 Jarvis, John J., "On the Equivalence Between the Node-Arc and Arc-Chain Formulations for the Multicommodity Maximal Flow Problem," Naval Research Logistics Quarterly, Vol. 16, pp. 525-529, (December 1969).

J-2 Jeffreys, Harold, THEORY OF PROBABILITY, Third Edition, Oxford at the Clarendon Press, (1961).

K-1 Kleitman, D., K. Steiglitz and P. Weiner, "The Design of Minimum Cost Survival Network," Technical Memorandum TM-105, National Resource Analysis Center, Systems Evaluation Division, (February 1969).

K-2 Konno, Hiroshi, "Bilinear Programming," Part I: Algorithm for Solving Bilinear Program, Technical Report No. 71-9; Part II: Application of Bilinear Programming, Technical Report No. 71-10, Operations Research House, Stanford University, (August 1971).

K-3 Kornai, J. and T. Liptak, "Two Level Planning," Econometrica, Vol. 33, pp. 141-169, (1965).

L-1 Lawler, E. L. and D. E. Wood, "Branch and Bound Methods: A Survey," J. ORSA, Vol. 14, pp. 697-719, (1966).

L-2 Lin, Shen, "Computer Solutions of the Travelling Salesman Problem," The Bell System Technical Journal, pp. 2245-2269, (December 1965).

M-1 Mayeda, W., "Terminal and Branch Capacity Matrices of Communication Net," I.R.E. Transaction on Circuit Theory, Vol. 7, pp. 251-269, (1970).

M-2 Morgan, Bruce W., AN INTRODUCTION TO BAYESIAN STATISTICAL DECISION PROCESSES, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, p. 46.

M-3 Murchland, J. D., "The Effect of Increasing or Decreasing the Length of a Single Arc on all Shortest Distances in a Graph," LBS-TNT-26, (September 4, 1967).

M-4 Murty, Katta, "Solving the Fixed Charge Problem by Ranking the Extreme Points," J. ORSA, Vol. 16, No. 2, pp. 268-279, (March-April 1968).

N-1 North, D. W. and W. F. Rousseau, "A User's Manual for TREE," Stanford Research Institute Project Report 8104-6, (June 1971).

R-1 Raiffa, Howard, DECISION ANALYSIS, Addison-Wesley, Reading, Massachusetts, pp. 279-284, (1968).

R-2 Reiter, Stanley and Gordon Sherman, "Discrete Optimizing," J. SIAM, Vol. 13, No. 3, pp. 864-889, (September 1965).

R-3 Ritter, K., "Problem für Nichtkonkave, Quadratische Funktionen," Doctoral Dissertation, Albert Ludwig University, Freiburg, (1964); "Stationary Points of Quadratic Maximum-Problems," Z. Wahrscheinlichkeitstheorie verw., Geb. 4, pp. 149-158, (1965); "A Method for Solving Maximum Problem with a Nonconcave Quadratic Objective Function," Z. Wahrscheinlichkeitstheorie verw., Geb. 4, pp. 340-351, (1966).

R-4 Rothschild, B. and A. Whinston, "On Two Commodity Network Flows," J. ORSA, Vol. 14, No. 3, pp. 377-388, (May-June 1966).

S-1 Sanderson, Robert D., "Enumerative Algorithms for Solving a Class of Network Synthesis Problems," ORC 70-11, Operations Research Center, University of California, Berkeley, (April 1970).

T-1 Tomlin, J. A., "Minimum-Cost Multicommocity Network Flows," J. ORSA, Vol. 14, pp. 45-51, (February 1966).

T-2 Tui, Hoang, "Concave Programming Under Linear Constraints," Soviet Mathematics, pp. 1437-1440, (July-December 1964).

V-1 Veinott, A. F., Jr., "Minimum Concave Cost Solution of Leontiff Substitution Models of Multi-Facility Inventory Systems," Technical Report No. 12, Operations Research House, Stanford University, (October 15, 1967).

V-2 Veinott, A. F., Jr., "Extreme Points of Leontief Substitution Systems," LINEAR ALGEBRA AND ITS APPLICATIONS-1, pp. 181-194, (1968).

W-1 Walkup, D., "On Branch and Bound Method for Separable Concave Program," Boeing Scientific Research Laboratory Research Manuscript.

W-2 Watling, G. C. and J. H. Weber, "TELSYN--A Computer Program for Designing Private Switched Communication Networks," (Technical Report not for publication), Bell Telephone Laboratories, Inc., Holmdel, New Jersey.

Y-1 Yaged, Bernard, Jr., "Minimum Cost Routing for Static Network Models," Networks, Vol. 1, No. 2, pp. 139-172.

Y-2 Yen, J. Y., "Finding the k Shortest Loopless Paths in a Network," Management Science, Vol. 17, No. 11, pp. 712-716, (July 1971).

Z-1 Zangwill, W. I., "Minimum Concave Cost Flows in Certain Networks," Management Science (Theory), Vol. 14, No. 7, pp. 429-450, (March 1968).

Z-2 Zangwill, W. I., NONLINEAR PROGRAMMING--A UNIFIED APPROACH, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, (1969).

Z-3 Zwart, Philip B., "Nonlinear Programming: Global Use of the Lagrangian," Journal of Optimization Theory and Applications, Vol. 6, No. 2, pp. 150-160, (1970).