OPTIMAL SIMULTANEOUS FLOW IN SINGLE PATH COMMUNICATION NETWORKS

A THESIS

Presented to

The Faculty of the Division of

Graduate Studies and Research

by

Robert M. Siegmann

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Information and Computer Science

Georgia Institute of Technology

January, 1971

OPTIMAL SIMULTANEOUS FLOW IN

SINGLE PATH COMMUNICATION NETWORKS

Approved:

Pranas Zunde, Chairman

Miroslav Valach

John J. Jarvis

Daniel C. Fielder

George W. Brown

Date approved by Chairman: Feb. 12, 1971

ACKNOWLEDGMENTS

Writing this dissertation was an intellectual as well as an educational experience. I am solely responsible for its contents, but I recognize that it is based on the efforts of many individuals. I gratefully acknowledge their support and dedicate this dissertation to them.

The person to whom I owe the deepest gratitude is my wife, Carol. She consistently supported and encouraged me through the entire process. Secondly, I wish to thank Dr. Vladimir Slamecka. Without his wisdom and backing this dissertation would not have been written.

My thesis advisors deserve a special word of appreciation. My principal advisor, Dr. Pranas Zunde, taught me much more than is evidenced in this document. Doctors Miroslav Valach and John J. Jarvis spent many hours with me in shaping and refining my thesis into its final form. Also Doctors Daniel C. Fielder, Alex Orden, William Goffman, and George W. Brown were kind enough to read and comment on my work.

I acknowledge the moral support of my friend John Gehl, the excellent typing of the drafts by Mrs. Lyn Jackson, and the final copy by Mrs. Betty Sims.

This dissertation was partially supported under Grant GN-655 from the National Science Foundation. The financial support of this organization is sincerely appreciated.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This dissertation documents a method for finding an optimum solution to a communication network design problem in which only one path is selected for message transmission between each pair of stations in a directed network. The method finds a minimum cost network flow configuration which satisfies all the message flow constraints while allowing all messages to flow simultaneously, i.e., at the same time. The results are based on techniques drawn from the optimization literature and concepts taken from graph and network theory. The solution method can be applied in part or in whole to various types of scientific, technical, or business information networks.

In order to use the solution method, the exact configuration of nodes and arcs in a strongly connected communication network must be known. Also required are the unit cost and maximum number of messages allowed on each arc, the message flow requirements for each pair of network nodes, and the maximum length of any communication path. The solution method will find a least cost solution (if one exists) having a single path flow between each pair of stations and satisfying all the message flow constraints simultaneously. The optimum solution is expressed in terms of the sequence of arcs which define each path and the total cost of the network flow.

The solution method is based on three algorithms. The first is a versatile path finding technique called the path algorithm which finds

the least cost restricted length path between all pairs of stations in the network. It produces the actual sequence of arcs along each of these paths and finds the maximum flow capacity for each path. The second and third algorithms integrate the path algorithm into a branch and bound technique to find a global solution to the network problem. All three algorithms are described, proved and illustrated.

Also included in the dissertation are: (1) A computer simulation comparison of the relative speed of the path algorithm and one of the fastest known shortest path algorithms; (2) necessary conditions for a network to be solved by the solution method and bounds on the cost of a network solution; (3) a complete example of the use of the solution method; (4) a summary of the dissertation results and a list of some follow-on areas of study; and (5) an extensive bibliography of related literature.

CHAPTER I

INTRODUCTION

Background and Objective

This dissertation documents a method of applying graph theoretical and optimization techniques to solve a real world problem encountered in the design of certain types of communication networks. The problem is a variation of the minimum cost, simultaneous flow network problem (hereafter referred to as "the network problem") for networks in which the message flow between each pair of stations is constrained to a single transmission path. The complete problem definition is delayed until Chapter III in order to allow time to develop the necessary terminology and concepts. Incorporated in the problem description is the requirement that no path in the network have a length greater than a pre-set maximum value. The solution method described herein will find the global optimum solution for any network configuration if such a solution exists.

The literature search which preceded our work on the simultaneous flow network problem revealed a marked tendency by its contributors to consider the problem from the point of view of linear programming (i.e., L.P.) [F28,F43].* Instead of attacking the problem from an L.P. point of view, our effort was purposely oriented toward introducing a new solution approach.

---

*Bibliographic references are denoted in this manner throughout the dissertation.

The motivation for selecting the network problem can be traced to several sources:

1. It is easily argued that mathematical tools for designing communication systems are considerably less developed than the tools available for analyzing such systems after they are operational.

2. According to current trends and predictions in the information and communication industry, increased interest will be demonstrated in the decade of the 1970's towards consolidating geographically dispersed components of information systems by interconnected communication networks.

3. The concepts related to the topic of graph theory offer a rich and promising future for use in modeling many diverse types of information systems.

Outline of the Paper

The contents of each chapter in the dissertation are briefly summarized in this section. The overall documentation strategy was to present the basic results by using a descriptive approach strengthened with theorems and proofs and illustrated with appropriate examples.

We discuss overall considerations about the dissertation in Chapter I. Also included in this introductory chapter are the definition of terms and conventions used throughout the paper. Chapter II is devoted to the description of an algorithm which finds optimum paths in a network. The algorithm is proved and illustrated in this chapter. In Chapter III the two-phased algorithmic solution (called "the network algorithm") to the minimum cost simultaneous flow problem is presented.

The proof that the network algorithm finds a global optimum is also contained in the chapter. Chapter IV contains theorems which establish necessary conditions and cost bounds for the network problem. The network algorithm is demonstrated in Chapter V where it is used to solve a communication flow problem for a large, decentralized company.

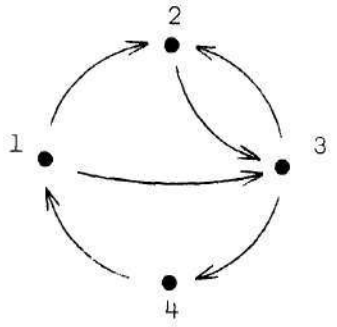
Concluding remarks about the dissertation results and fruitful areas for further research are presented in Chapter VI. Supporting results to certain topics in the dissertation are presented in the Appendix. An extensive bibliography is given in six sections which are representative of the many facets of this dissertation topic.

Terminology and Notation

There is a large number of definitions which have to be given in this paper in order to acquaint the reader with the required concepts. The policy here is to define a term when needed, beginning with a DEF in the left margin. Because many aspects of a network, such as its nodes, its arcs and its paths, are sets of well-defined elements, set notation is frequently used. It is assumed that the reader is familiar with set theory and the standard ways of describing sets.

DEF A *directed network*, G, is a pair of finite sets (N,A) where A is a subset of the Cartesian product set defined on N×N. An individual element of N is called a *node* and an individual element of A is called an *arc*. Multiple arcs between nodes are not permitted.

There is frequent use of a geometric (i.e., pictorial) representation of the relational network definition given above. For example, the directed network G can be defined in either of the following ways:



$G = (N,A)$

$N = \{1,2,3,4\}$

$A = \{(1,2),(1,3),(2,3),(3,2),(3,4),(4,1)\}$

The number of nodes in the node set N (i.e., its cardinality) is always referred to as "n" and the number of arcs in the arc set A is always referred to as "m." An element $\alpha \epsilon A$ can either be written as α or it can be written in terms of its defining nodes as $[x,y]'$ where the arc α is directed from node x to node y. Note the use of [ ]' to denote an arc.

Certain kinds of ordered sets play an important role in our work. The most basic ones are now defined.

DEF A *route* $[x_1,x_n]$ is an ordered set of arcs

$$[x_1,x_n] = \{[x_1,x_2]',[x_2,x_3]',\ldots,[x_{n-1},x_n]'\}$$

in which the second node of each arc in the set (except the last) is the first node of the next arc in the set. Note the use of square brackets *without* a prime to denote a route between a node-pair.

DEF A *cycle* is a route $[x_1,x_n]$ in which the first node in the route is the same as the last node, i.e., $x_1=x_n$.

DEF A simple *path* $\pi^*(x,y)$ is a route $[x,y]$ which does not contain a cycle.

The concept of "simple path" is of fundamental importance in this dissertation. A simple path between node x and node y is always denoted by $\pi^*(x,y)$, $\pi'(x,y)$, or $\bar{\pi}(x,y)$; whereas the collection of all the simple paths from x to y is written as $\pi(x,y)$. Sometimes the ordered pair (x,y) is dropped from $\pi(x,y)$ when referring to a simple path having unspecified end-points, e.g., π^* is a designation for a simple path. Frequently, the word "simple" is dropped from "simple path," since simple paths are our only concern in this dissertation.

A path could be defined in terms of the ordered set of nodes along the path. In order to distinguish this from the equivalent definition in terms of arcs, we define the following terms:

DEF The *node set* $NS(\pi^*)$ of a path π^* is the ordered set of nodes on the path π^*.

DEF The *arc set* $AS(\pi^*)$ of a path π^* is the ordered set of arcs on the path π^*.

We conclude this section on terminology with the following definitions:

DEF A *loop* is an arc $\alpha = [x,y]'$ in which $x=y$.

Loops are useful to include in some types of directed networks, but since they are not needed in our work, we omit them from all further consideration.

DEF The *length of a path* $\ell(\pi^*)$ is the number of arcs in the arc set $AS(\pi^*)$ or one less than the number of nodes in the node set $NS(\pi^*)$.

The concept of path length is extremely useful to us and is used throughout the dissertation.

DEF A *node-pair* (x,y) is an ordered pair of two distinct nodes x and y.

The term "node-pair" is used to refer to the general communication relationship from node x to node y. The set of all node-pairs NP is defined as:

$$NP = \{(i,j) \mid i,j \in N,\ i \neq j\}.$$

Communication Networks

In this dissertation the concern is with a particular kind of communication network in which information always flows in one direction on a single fixed path between each pair of stations. The concept of a directed graph is used to model this as follows: Each node in the graph represents a station and a directed arc represents the one-directional transmission line between two stations. Since the flow of messages in a network always has associated sending and receiving stations, the concept of a node-pair takes on an important significance.

DEF A *strongly connected network* is a directed network in which at least one path exists between each node-pair in the network.

DEF The *simultaneous flow* of messages in a network requires that messages be transmitted between all node-pairs concurrently. Any particular message has a unique origin node and destination node, and the network is designed to handle all required message flows simultaneously.

Although a general communication network does not have to be strongly connected and does not have to have simultaneous transmission of messages between its node-pairs, many types of communication systems require these properties. This dissertation is directed to and oriented around such networks.

This concept of a communication network includes many communication systems in which the configuration of nodes and arcs in the network is based on fixed parameters. This includes all networks having pre-established and fixed message routing between stations, many networks having directional message flows, and some non-electrical networks. Our concept of a communication network is, therefore, general enough to include a wide range of practical network configurations.

For example, the results can be applied in part or in whole to network design problems in the following proposed or operational communication networks:

—Management Information Systems (e.g., Lockheed's INTERLOC).

—Science Literature Networks (e.g., Hungary's TECHNOINFORM).

—Educational Networks (e.g., EDUNET and ERIC).

—News Media (e.g., UPI).

—Library Networks (e.g., NELINET).

—Military Communication Systems (e.g., AUTODIN).

—Military Command and Control Networks (e.g., WWMCCS).

—Medical Literature Networks (e.g., MEDLARS).

—Hospital Networks (e.g., THOMIS).

—Police Networks (e.g., NCIC).

—Business Networks (e.g., SABRE).

As mentioned earlier, the network design of the types of networks listed above has always suffered from a lack of quantitative design tools. Because of the stochastic nature of the operational use of such networks, probability theory, queuing theory and stochastic processes are frequently applied to the analysis of network operation. These are reasonably refined modeling tools and are extremely useful for network analysis. Unfortunately, quantitative tools for network design are virtually nonexistent. The network designer is generally forced to use simulation in order to test various design configurations. This indirect and sometimes time-consuming activity is frequently helpful, but it does not always produce optimum results. The lack of adequate tools for network design is the most pressing problem facing network designers.

Information Flow and Cost

An individual arc in a communication network represents a channel over which information is transmitted. In fact, since a single arc may be used by many different paths in a network, the total information flow on the arc depends on the number of paths using the arc and their

respective flows. Two of the most important characteristics of this flow of information are expressed in terms of amount and cost.

In order to solve the network problem, the amount of information (i.e., messages) transmitted between the stations in the network must be known. This message flow between different node-pairs must be given in some type of equivalent information unit. In this dissertation an information unit will be defined as a message having a standard length. Therefore, the amount of information flow on an arc, on a path, or between a node-pair is the number of equivalent standard messages transmitted per time period.

The cost which is associated with an arc is the unit cost for each message transmitted over the arc. To find the total cost per time period in an arc, the amount of information flow on an arc is multiplied by the unit cost of flow for that arc. The concept of cost is important in our work, because the overall cost of a network configuration is used as the variable which is minimized in solving network design problems.

Network Flow

Network flow theory was introduced in 1956 by Ford and Fulkerson when they published their max-flow, min-cut theorem [F15]. This well-known theorem states that the maximum flow from node x to node y in any network is equal to the value of the minimum cut over all cut sets separating x and y. This was the first definitive result in relating the dynamic concept of flow with the static concept of cut set. Unfortunately, this theorem does not hold for a network having simultaneous flows [F61].

In this section we define the terms which relate to the dynamic flow aspects of a network having a single transmission path between each node-pair.

DEF The *flow on an arc* α, $f(\alpha)$, is the amount of information transmitted over the arc α per unit of time.

DEF The *capacity of an arc* α, $b(\alpha)$, is the maximum amount of information which can be transmitted over arc α per unit of time.

DEF The *unit cost of an arc* α, $c(\alpha)$, is the unit cost of transmitting information which flows on arc α.

DEF The *capacity of a path* $\pi^*(x,y)$, $b\left(\pi^*(x,y)\right)$, between node-pair (x,y) is the maximum amount of information which can be transmitted over the path, and is equal to the smallest capacity of the path's component arcs. That is,

$$b\left(\pi^*(x,y)\right) \equiv b^*(x,y) \equiv b(\pi^*) = \min_{\alpha \in AS(\pi^*)} [b(\alpha)].$$

DEF The *flow on a path* $\pi^*(x,y)$, $f\left(\pi^*(x,y)\right)$, is the amount of information transmitted over the path per unit of time, i.e.,

$$f\left(\pi^*(x,y)\right) \equiv f^*(x,y) \equiv f(\pi^*).$$

DEF The *unit cost of a path* $\pi^*(x,y)$, $c\left(\pi^*(x,y)\right)$, is the unit cost of transmitting messages over the path and is the sum of the unit cost of each arc in the arc set of the path. That is:

$$c\left(\pi^*(x,y)\right) \equiv c^*(x,y) \equiv c(\pi^*) = \sum_{\alpha \epsilon AS(\pi^*)} c(\alpha).$$

DEF A *minimum cost path* $\bar{\pi}(x,y)$ for node-pair (x,y) is a path between (x,y) having a minimum cost. That is:

$$\bar{\pi}(x,y) = \left\{ \pi^*(x,y) \middle| c\left(\pi^*(x,y)\right) = \underset{\pi'(x,y)\epsilon\pi(x,y)}{\text{Min}} \left\{ c\left(\pi'(x,y)\right) \right\} \right\}.$$

A minimum cost path $\bar{\pi}(x,y)$ is defined in a relative way because a path may have to satisfy additional conditions (e.g., a length constraint) before it is considered as an element of $\pi(x,y)$, which is the collection of all the paths between x and y.

In networks in which flows exist simultaneously between all node-pairs, a particular arc may be used to carry messages between several node-pairs. In order to account for this, the following definitions are required:

DEF A *path flow component*, $f_\alpha\left(\pi^*(x,y)\right)$, is the amount of flow required on arc α by the flow on the path $\pi^*(x,y)$. Alternative symbols for a path flow component are:

$$f_\alpha\left(\pi^*(x,y)\right) \equiv f_\alpha{}^*(x,y) \equiv f_\alpha(\pi^*).$$

Now the earlier definition of arc flow can be expressed in terms of path flow components.

DEF The *flow on arc* α, $f(\alpha)$, is the sum of all its paths flow components. That is:

$$f(\alpha) = \sum_{(i,j)\epsilon NP} f_{\alpha}(i,j).$$

A node-pair represents a required communication connection (therefore, it can represent a path) over which information is transmitted. In setting up a design problem, we are given the amount of information which is required to be transmitted between each node-pair. This is called the node-pair flow requirement and is defined as:

DEF A *node-pair flow requirement*, $r(x,y)$, is the amount of information required to flow (over a single path) from node x to node y.

The following definitions complete the definitional base needed in this dissertation. These terms refer to collections of paths between all node-pairs in a network.

DEF A *network path set* $\{\pi^*(i,j)\}$ is a collection of n(n-1) paths in which one path $\pi^*(x,y)$ is selected for each node-pair (x,y) in the network. That is:

$$\{\pi^*(i,j)\} = \{\pi^*(x,y) \mid \pi^*(x,y)\epsilon\pi(x,y) \text{ for all } (x,y)\epsilon NP\}.$$

A minimum cost path set is represented as $\{\bar{\pi}(i,j)\}$.

The following terms are needed to describe the network algorithm of Chapter III. In this algorithm the required flow between two nodes (i.e., the flow requirement) is assigned to each node-pair in a manner which leads to the solution of the problem.

DEF A *simultaneous flow assignment* $\{f(\pi^*(i,j))\}$ is an allocation of the node-pair flow requirement $r(x,y)$ to each path $\pi^*(x,y)$ (and therefore to the arcs in each path) in a network path set. That is:

$$\{f(\pi^*(i,j))\} = \Big\langle f(\alpha) \,\Big|\, \alpha \in A,$$
$$f_\alpha(\pi^*(x,y)) = \begin{cases} r(x,y) \text{ if } \alpha \in \pi^*(x,y) \\ 0 \text{ otherwise} \end{cases} \Big\rangle,$$
$$f(\alpha) = \sum_{\pi^*(x,y) \in \{\pi^*(i,j)\}} f_\alpha(\pi^*(x,y)) \Big\rangle .$$

DEF A *feasible simultaneous flow assignment* is a simultaneous flow assignment to the node-pairs in a network path set such that:

$$f(\alpha) \leq b(\alpha) \text{ for all } \alpha \in A.$$

DEF The *cost of a simultaneous flow assignment*, COST, is defined in either of the following alternative ways:

a. $$\text{COST} = \sum_{(i,j) \in NP} f(\pi^*(i,j)) \cdot c(\pi^*(i,j)) \text{ or}$$

b. $$\text{COST} = \sum_{\alpha \in A} f(\alpha) \cdot c(\alpha) \text{ where } f(\alpha) = \sum_{(i,j) \in NP} f_\alpha(\pi^*(i,j))$$

for the simultaneous flow assignment $\{f(\pi^*(i,j))\}$.

The Single Path Approach

The path approach for investigating the structural aspects of graphs is not new. Kirchhoff [D35] showed in the 19th century that paths have close relationships with other graph theory concepts. The application of path-oriented approaches to network design problems, however, has been hampered by the fact that an enormous number of paths may exist even in moderate-sized networks. For example, a directed network having ten nodes and ninety arcs has almost one billion simple paths.*1 Fortunately, there are ways to find paths in networks without making an unreasonable number of computations.

One method is to associate a weight (i.e., cost) with each arc and to find the least-weighted paths. This drastically reduces the number of paths which have to be considered. For example, a maximum of 3,240 paths*2 have to be examined in the ten-node network mentioned above in order to find one least weighted path for each of the 90 node-pairs.

The theory and applications discussed in this dissertation are oriented towards finding the optimum single transmission path between each node-pair. Since there are 90 node-pairs in a ten-node network, only 90 optimum paths need be found. The optimum single path between a node-pair is one which optimizes the path weight. The ability to work only with optimum paths demonstrates the power of our solution method.

---

*See Chapter VII for the derivation of the following formulas:

1. $n(n-1)\sum_{x=0}^{n-2} x!\binom{n-2}{x}$

2. $n(n-1)^2(n-2)/2$

This method is presented in more detail by considering the concept of arc weight and path length.

Arc Weight and Least Weighted Paths

The term "weight" generically refers to any one of a number of parameters which can be associated with the arcs of a network. For example, time, distance, cost, capacity, reliability and value are possible arc weights. By assigning one type of weight (e.g., cost) to all the arcs in a network many different kinds of problems can be described and studied. In this dissertation arc cost is usually defined as the arc weight, for this is generally the most important optimization variable in a communication network.

DEF The *weight of arc* α, $w(\alpha)$, is an assigned positive or negative real number which numerically represents some variable associated with the arc.

DEF The *weight of a path* $\pi^*(x,y)$, $w(\pi^*(x,y))$, is a calculated positive or negative real number which is obtained as the sum of the weights in the arc set of path $\pi^*(x,y)$. That is:

$$w^*(x,y) \equiv w(\pi^*(x,y)) = \sum_{\alpha \in AS(\pi^*)} w(\alpha).$$

The term "shortest path" is frequently used in connection with path finding algorithms. However, it has a double meaning because it may refer to (1) a path of least length, or (2) a path of least weight. The term is avoided whenever possible in this dissertation since arc

length is never used as arc weight. Whenever the term is used, it is always used in the sense of (2). To find the least weighted path between two nodes x and y, all paths between x and y must be implicitly checked and then the path having the minimum or least weight can be selected. Since more than one path may have the least weight, multiple paths having the same least weight value are possible. Fortunately, the theorems on which our work is based allow us to select, arbitrarily, any least weighted path for each node-pair. This capability forms the bridge between the use of a shortest path optimization procedure and the solution of problems requiring single path transmission between all node-pairs.

Path Length

The length of a path is an important consideration in communication networks. The length of a path was defined as the number of arcs in the arc set of the path. Subscripting a path designation (e.g., $\bar{\pi}_p$) will be used as a means of displaying the length of a path. For example, $\bar{\pi}_4(x,y)$ is a least weighted path of length four between node-pair (x,y). The length subscript is also used for weights (e.g., $\bar{w}_4(x,y)$ is the weight of path $\bar{\pi}_4(x,y)$).

The length of a path in a communication system is frequently an important design consideration. For example, the different types of connecting or switching stations along a transmission path influence the overall operation of certain types of communication systems. Frequently the strength and/or reliability of the transmitted message signal can be described as a decreasing function of the number of such connecting

or switching stations along a path. A valid means of keeping the signal within acceptable limits is to restrict the length of all acceptable paths in the network. The ability to restrict path length is embedded in the network problem which is considered in this dissertation. The results obtained in this dissertation by constraining path length are new and extend the state of the art for the types of networks considered.

Consider, for example, a communications network connecting geographically dispersed users of scientific information. Assume that a computerized library or data bank of scientific abstracts exists at the center of the network. Formatted requests are transmitted from users to the computer, and fixed length responses are directed back to the users. This is the basic type of network considered in this dissertation, for messages can be represented in terms of a standard length and the maximum number of arcs along each path (i.e., its length) can be fixed in order to insure proper transmission.

CHAPTER II

THE PATH ALGORITHM

Introduction

A crucial step in finding the solution to the network problem discussed in the next chapter is the determination of a least cost path between each node-pair in the network. The solution to a number of practical problems dealing with networks is realized by finding the least weighted paths (sometimes called the shortest path) between the nodes in the network. For example, finding a communication system with the least number of relay stations, finding the fastest way out of a maze, and finding the shortest bus route in a city are all mathematically equivalent to the classical least weighted path problem.

In this chapter we develop an algorithm, called the path algorithm, to find least weighted paths between all node-pairs in a network. The algorithm is based on a forward dynamic programming approach [A01]. The algorithm is discussed in detail because it provides a systematic means of finding least weighted labeled paths which are constrained by path length. Any modification of another shorted path algorithm which achieves the same results can be used in place of the path algorithm in the solution algorithm of Chapter III.

In order to solve the network problem, a method must be created for finding only those least weighted paths which are less than or equal in path length (i.e., number of arcs in the path) to a pre-set integer

L. That is, the network problem requires a least weighted path $\bar{\pi}(x,y)$ such that $\ell\left(\pi(x,y)\right) \leq L$ for each node-pair $(x,y) \epsilon NP$.

Numerous shortest path algorithms have been documented in the literature. Two surveys (Pollack [E36] and Dreyfus [E10]) illustrate the many variations in the shortest path algorithms which exist. The "path algorithm" uses the well-known forward dynamic programming approach to find least weighted paths. The underlying recursion relation for the path algorithm is:

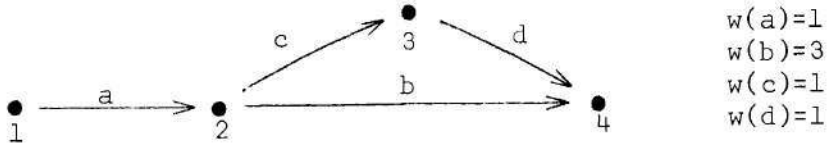
$$w_{p+1}(x,y) = \underset{\substack{k \epsilon N \\ k \neq x,y}}{\text{Min}} \left[w_p(x,k) + w_1(k,y) \right].$$

Our approach excludes loops from consideration in finding least weighted paths between node-pairs and therefore node k in the recursion relation cannot be node x or node y. Other approaches exist which permit node k to vary over all nodes in the network. The recursion relation states that if the least weighted path from node x to node y passes through node k^*, then the path from x to k^* must be the least weighted path between node-pair (x,k^*).

Equivalent approaches for finding least weighted, length constrained paths can be found in the literature. In general these approaches are based on matrix multiplication [E40] or related methods dealing with the variable adjacency matrix [D29].

It is informative to compare the path algorithm with the popular shortest path algorithms (e.g., Floyd [E13], Dantzig [E06], and Murchland [E27]). These algorithms find all the shortest paths in a network,

but are based on a computational method which precludes the effective manipulation of path length. This shortcoming of these algorithms can be illustrated by considering how the Floyd algorithm finds the shortest path between node-pair (1,4) in the following partial network:



Let the maximum path length L be equal to two. In the Floyd algorithm (see Figure 2.1) the least weighted path $\bar{\pi}(1,4)$ is found by using the weights of the paths $\bar{w}(1,2)$ and $\bar{w}(2,4)$. Since the least weighted path $\bar{\pi}(2,4) = \{c,d\}$, the least weighted path $\bar{\pi}(1,4)$ is $\bar{\pi}(1,4) = \{a,c,d\}$ which happens to be of length three. Unfortunately, the weight $\bar{w}(2,4)$ of the path $\bar{\pi}(2,4) = \{c,d\}$ overlays the weight $\bar{w}(2,4)$ of the path $\bar{\pi}(2,4) = \{b\}$ in the computations and is thereafter never computed again.

The underlying reason why the Floyd algorithm (and other algorithms mentioned) cannot easily handle the length-constrained problem is inherent in the computational method used by the algorithm. This method can be briefly explained as follows:

Method 1. For all $k,i,j=1,\ldots,n$

IF $\bar{w}(i,k) + \bar{w}(k,j) < \bar{w}(i,j)$,

THEN $\bar{w}(i,j) \leftarrow \bar{w}(i,k) + \bar{w}(k,j)$.

An algorithm based on this method finds a least weighted path $\bar{\pi}(x,y) = \{x=x_1,x_2,\ldots,x_p,\ldots,x_r=y\}$ between node-pair (x,y) by using the

least weighted paths $\bar{\pi}(x,x_p)$ and $\bar{\pi}(x_p,y)$ for any $x_p \epsilon \bar{\pi}(x,y)$. Now if we restrict p to be equal to r-1, then the least weighted path $\bar{\pi}(x,y)$ is found by using $\bar{\pi}_p(x,x_p)$ and $\bar{\pi}_1(x_p,y)$. That is:

$$\bar{w}_{p+1}(x,y) = \bar{w}_p(x,x_p) + \bar{w}_1(x_p,y).$$

This modification of method 1 is now stated as:

Method 2. For all i,k,j=1,...,n

$$\left.\begin{array}{l} \text{IF} \quad \bar{w}_p(i,k) + \bar{w}_1(k,j) < \bar{w}(i,j), \\ \text{THEN} \quad \bar{w}_{p+1}(i,j) \leftarrow \bar{w}_p(i,k) + \bar{w}_1(k,j) \end{array}\right\} \text{For all } p=1,\ldots,L.$$

An algorithm based on method 2 overcomes the length difficulty discussed in the previous example. Since it calculates $\bar{\pi}(1,4)$ one arc at a time, it terminates when $\bar{\pi}_2(1,4) = \{a,b\}$. The path $\bar{\pi}_3(1,4) = \{a,c,d\}$ found by method 1 has a smaller weight, but it violates the length restriction (i.e., L=2) and therefore is not found by method 2.

Comparison of Floyd and Path Algorithms

We show in this section that the path algorithm is much closer to the Floyd algorithm in terms of required storage space and speed than might be expected. Even though the Floyd algorithm does not find length-constrained least weighted paths, it does represent a popular method of

finding least weighted paths. It is one of the fastest known algorithms and serves as a reasonable standard for comparison purposes.

Figure 2.1 illustrates the similarity of the two algorithms in terms of their respective FORTRAN programs. The maximum number of executed instructions and the total number of storage locations required by these two programs are given in the following table:

Algorithm	Executed Instructions			Storage Locations
	Additions	Comparisons	Stores	
Floyd	n^3	$n^2(2n+1)$	$2n^3$	n^2
Path	$n^2(n-1)^2$	$n^2(n-1)(2n-1)$	$3n^2(n-1)^2$	n^3

From the above table it would appear that the computer program for the path algorithm requires n times more storage locations and approximately n times more executed instructions than the program for the Floyd algorithm; however, by making minor modifications to the path algorithm program, the two programs actually become much closer in terms of storage requirements and execution time.

With some additional instructions the three-dimensional array, W[P,I,K], used in the FORTRAN version of the path algorithm can be converted into several two-dimensional arrays, thereby only requiring approximately 2n storage locations more than the n^2 required by the Floyd algorithm.

```
      DO 40 I=1,N
      DO 40 P=1,L
      DO 40 K=1,N
      IF W[P,I,K]=INF 10, 40, 40
10    DO J=1,N-1
      IF W[1,K,J]=INF 20, 40, 40
20    W=W[P,I,K]+W[1,K,J]
      IF W=SW[I,J] 30, 40, 40
30    SW[I,J]=W
      W[P+1,I,J]=W
40    CONTINUE
```

Computer Program of Path Algorithm

```
      DO 40 I=1,N
      DO 40 J=1,N
      IF W[J,I]=INF 10, 40, 40
10    DO 40 K=1,N
      IF W[I,K]=INF 20, 40, 40
20    W=W[J,I]+W[I,K]
      IF W=W[J,K] 30, 40, 40
30    W[J,K]=W
40    CONTINUE
```

Computer Program of Floyd Algorithm

Figure 2.1. FORTRAN Program of Path and Floyd Algorithms

It appears from the FORTRAN programs in Figure 2.1 that the execution times of the two algorithms differ by a multiplicative factor of L, the value of the length constraining P loop. Therefore, as $L \to 1$, the execution times of the two algorithms become approximately equal. As $L \to n$, the path algorithm appears to execute n times the instructions required by the Floyd algorithm.

Because of the numerous computations required in the algorithms, a computer simulation program was written in order to compare the relative execution times of the two algorithms. An obvious efficiency was included in the path algorithm program. Instructions were added to terminate it if no least weighted paths were found in any iteration. The FORTRAN programs used in the simulation and a typical printout are given in the appendix and entitled "Computer Simulation Programs and Sample Printout."

The computer simulation program was written in FORTRAN and executed on the UNIVAC 1108. It produced the interesting and unexpected result that the path algorithm takes approximately twice the execution time required by the Floyd algorithm. This ratio of two tended to be independent of the number of nodes and the number of arcs in a network. In the simulation random strongly connected networks were created

—with arc weights ranging from 0 to 15,

—having the following number of nodes: 5, 10, 15, 20 and 30,

—and containing either n(n-1) arcs or approximately $\frac{n(n-1)}{2}$ arcs.

The results of the computer simulation are summarized on Figures 2.2 and 2.3. Each value which is plotted represents the average of the

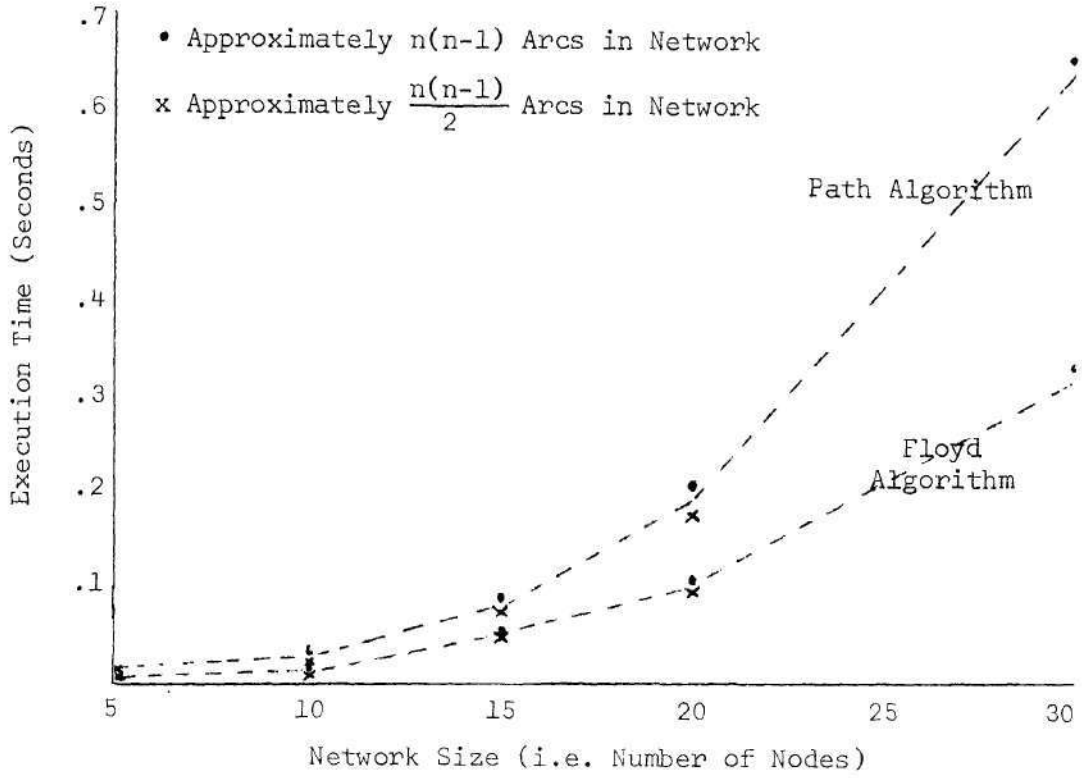


Figure 2.2. Execution Times for Path and Floyd Algorithms

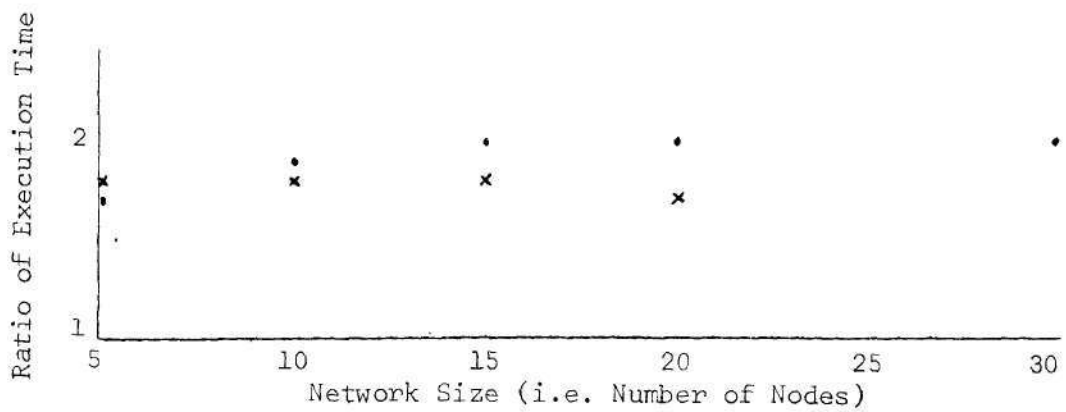


Figure 2.3. Ratio of Execution Times for Path/Floyd Algorithm

execution time of ten networks having the indicated number of nodes and arcs. Figure 2.2 gives the actual execution times for networks having 5 to 30 nodes. The upper and lower curves on Figure 2.2 represent, respectively, the execution times for the path and the Floyd algorithm. The two types of arc densities (i.e., n(n-1) arcs or $\frac{n(n-1)}{2}$ arcs) used are indicated by different marks (i.e., · and ×). Figure 2.3 is obtained directly from Figure 2.2 and gives the ratio of the execution time of the path to the Floyd algorithm. Note that this ratio is consistently less than or equal to two and that the path algorithm tends to be a little faster for networks having fewer arcs.

The computer simulation was not intended to be exhaustive and the results, though interesting, are only suggestive. We can conclude, however, that the two algorithms are reasonably close to one another in execution time for the types of networks considered in the simulation. The simulation work had to be terminated in order to continue the main topic of the dissertation. However, it uncovered a natural follow-on area for further research.

Additional Features of the Path Algorithm

The loop structure of the path algorithm allows it to find the least weighted path between a particular node-pair (x,y) in $\frac{1}{n}$th the time required to find paths for all node-pairs. This is accomplished by setting the I loop index to any node x ($1 \leq x \leq n$) and then executing the rest of the algorithm. Because of its loop structure, the Floyd algorithm cannot be manipulated in this manner. Therefore, combining the above comments with those discussed in the previous section, it is

strongly suggestive that the path algorithm is approximately $\frac{n}{2}$ times *faster* than the Floyd algorithm in finding the least weighted path for any particular node-pair (x,y) in a network.

In later chapters, the path algorithm will be used to find length constrained least weighted paths. A question naturally arises: How does the execution time of the path algorithm vary for different values of path length? We modified the computer simulation programs used in the previous section in order to determine an empirical answer to this question. The results are given in Figure 2.4. In this figure the cumulative increase in execution time is plotted against increasing values of path length for two 20-node networks. The first network contained 50 arcs whose longest least weighted path was of length six. The second network contained 380 arcs and had a longest path of length four. The figure illustrates the effect the path length restriction has on the execution time of a 20-node network having different numbers of arcs.

The Path Finding Problem and Solution

The path algorithm was created in order to solve a path finding problem which arises in communication networks. This problem is formally stated below and is tailored to the path requirements developed in the next chapter.

Given:

G1. A set N of n nodes where $1 \leq n < \infty$.

G2. A set A of m arcs where $n-1 \leq m \leq n(n-1)$.

G3. A weight $w(\alpha)$ for each $\alpha \epsilon A$ where $w(\alpha)$ is a real number.

G4. A maximum length L for any path where L is an integer between 2 and n-1.

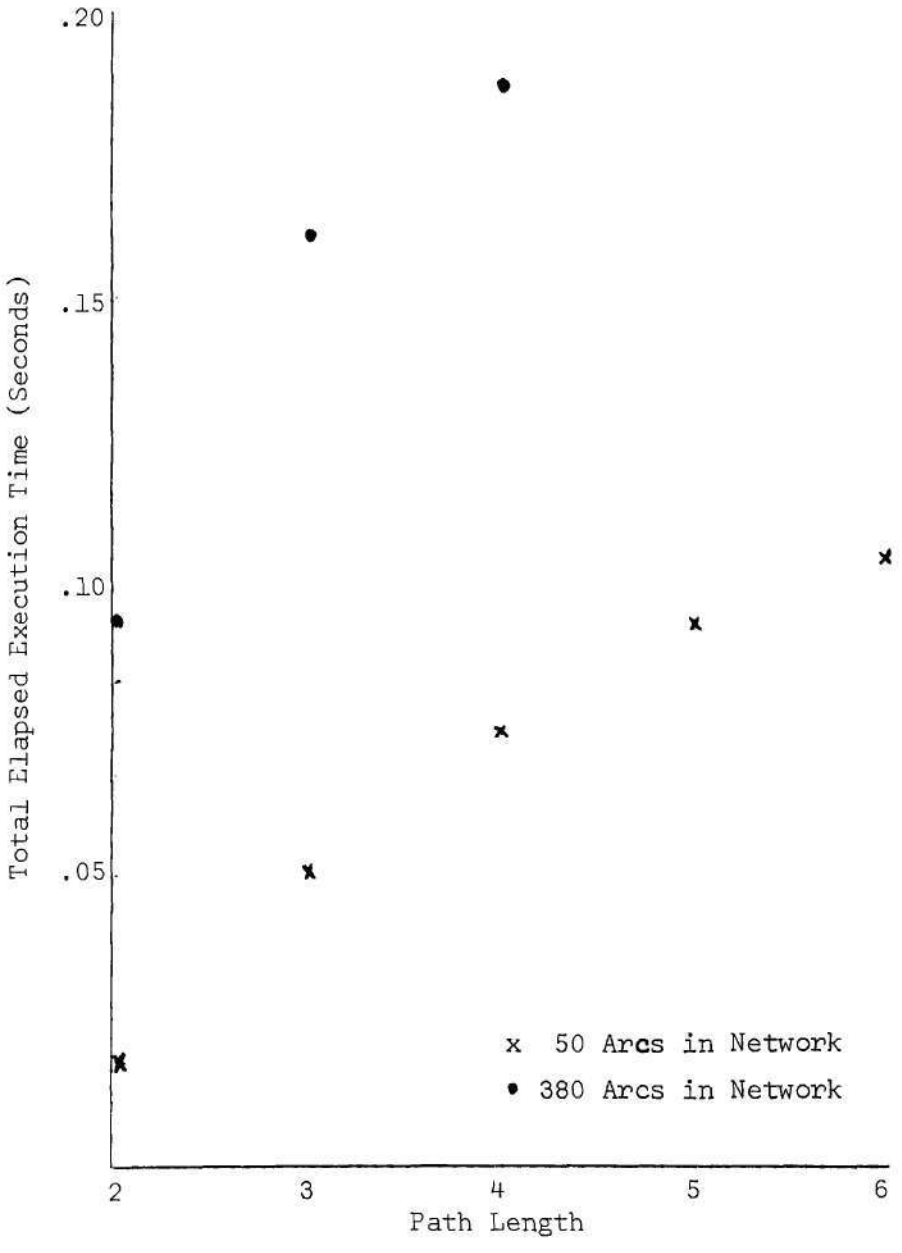


Figure 2.4. Execution Time for Length Constrained Least Weighted Paths in a 20-node Network

Find:

F1. The least weight $\bar{w}(i,j)$ for the path $\bar{\pi}(i,j)$,

(a) for all $(i,j)\epsilon NP$, or
(b) for a particular (x,y).

F2. The arc set $\{\bar{\pi}(i,j)\}$ for the least weighted path $\bar{\pi}(i,j)$,

(a) for all $(i,j)\epsilon NP$, or
(b) for a particular (x,y).

Subject to:

$\ell\left(\bar{\pi}(i,j)\right)\leq L$ for all $(i,j)\epsilon NP$.

The path algorithm, introduced in the previous section and explained in detail in the next section, solves the above constrained path finding problem. The FORTRAN version of the path algorithm, listed in Figure 2.1, was used to illustrate the overall method. Before discussing the path algorithm in detail, we comment on some aspects of the algorithm.

1. Condition F2 in the above statement of the problem requires that the actual sequence of arcs along each path be found. This is accomplished in the path algorithm by creating a matrix $[\bar{\eta}_x]$ of node labels for each node x. This matrix is created by the algorithm in order to store the least weighted backward link for the path $\bar{\pi}(x,y)$ in the network.

DEF The *backward link* $\bar{\eta}_x(y,z)$ in the matrix $[\bar{\eta}_x]$ is the node just prior to node z in the node set $NS\left(\bar{\pi}(x,y)\right)$.

Since it can be proved that the least weighted path from any node x to any other node y cannot contain a cycle, the single link $\bar{\eta}_x(y,z)$ is

sufficient to trace the entire path backwards from node y to node x. An additional algorithm, called the auxiliary algorithm (AA), is required to identify the actual arc sets from the matrix $[\bar{\eta}_x]$ for each node-pair. This algorithm is discussed later in the chapter.

2. Conditions (a) and (b) under F1 and F2 in the above problem statement specify that the algorithm is to be used to find either all paths or a single path in the network. Since the loop structure of the algorithm is set up to find all the least weighted paths emanating from a single node before advancing to another node, it can be used to find the least weighted path for any specific node-pair (x,y). This is achieved by restricting the value of i in the algorithm to the first node x in the desired node-pair (x,y).

3. Negative arc weights can be used in the algorithm. However, cycles cannot be stored in the matrix $[\bar{\eta}_x]$. Therefore, networks having cycles with negative weights cannot be handled by the current path algorithm. If no negative cycles are present in a network, the algorithm will automatically find simple paths. (See Lemma 2.3.)

4. Since the least weighted path between a node-pair is not necessarily unique, the algorithm is set up to find the *first* such path which it encounters. This means that a least weighted path having the shortest length is always selected, because the path algorithm grows paths by appending one arc at a time.

In the event the path algorithm were extended to solve other types of network problems, a number of variations is possible within the computational framework. Some of these are:

—Path weight calculated as a function of several variables.

—Calculation of paths having maximum weight.

—Calculation of Hamiltonian cycles.

—Retention of all least weighted paths for each node-pair.

The Path Algorithm

After the initialization step, the path algorithm is defined by the sequence of steps lettered below. See Figure 2.5 for the corresponding flow chart.

Initialization

—Set $\bar{w}_p(i,j) \leftarrow INF$ for all $(i,j) \epsilon NP$ and all $p=1,\ldots,L$.

—Set $\bar{w}(i,j) \leftarrow INF$ for all $(i,j) \epsilon NP$.

—For each arc $[x,y]' \epsilon A$, $\bar{n}_x(y,y) \leftarrow x$ and $\bar{w}_1(x,y) \leftarrow w[x,y]'$.

—To find least weighted paths for all node-pairs, set $n_1=1$ and $n_2=n$. To find the least weighted path for node-pair (x,y), set $n_1=x$ and $n_2=x$.

A. Set the i index equal to n_1.

B. Set the p index equal to 1.

C. Set the k index equal to 1.

D. If i=k, then go to 0; otherwise continue.

E. If there is a path of length p, $\pi_p(i,k)$, between node-pair (i,k), then continue to step F. If not, go to step 0.

F. Set the j index equal to 1.

G. If i=j or k=j, then go to Step M. If not, continue to step H.

H. If there is an arc, $w_1(k,j)$, between node-pair (k,j), then continue to step I. If not, go to M.

I. Calculate the weight $w^*(i,j)$ of the new path $\pi^*_{p+1}(i,j)$ as follows:

$$w^*(i,j) \leftarrow \bar{w}_p(i,k) + \bar{w}_1(k,j).$$

J. If the weight, $w^*(i,j)$, of the new path between (i,j) is less than the weight $\bar{w}(i,j)$ of all previously found paths for node-pair (i,j), then go to K. Otherwise, go to step M.

K. Store the backward link k and the new weight $\bar{w}(i,j)$ as follows:

1. $\bar{n}_i(j,s) \leftarrow \left\{ \begin{matrix} \bar{n}_i(k,s) \text{ for } s=1,\ldots,n;\ s\neq j \\ k \text{ for } s=j \end{matrix} \right\}$

2. $\bar{w}(i,j) \leftarrow \bar{w}_p(i,j) \leftarrow w^*(i,j)$.

M. If the j index has ranged over all nodes, then go to step O. If not, go to step N.

N. Increment the j index by one.

Go to step G.

O. If the k index has ranged over all nodes, go to step Q.

If not, go to step P.

P. Increment the k index by one.

Go to step D.

Q. If the p index has ranged over all nodes, $1 \leq p \leq L$, then continue to step S. If not, go to step R.

R. Increment the p index by one.

Go to step C.

S. If the i index has ranged over all nodes, $1 \leq n_1 \leq n_2 \leq n$, then go to step U. Otherwise go to step T.

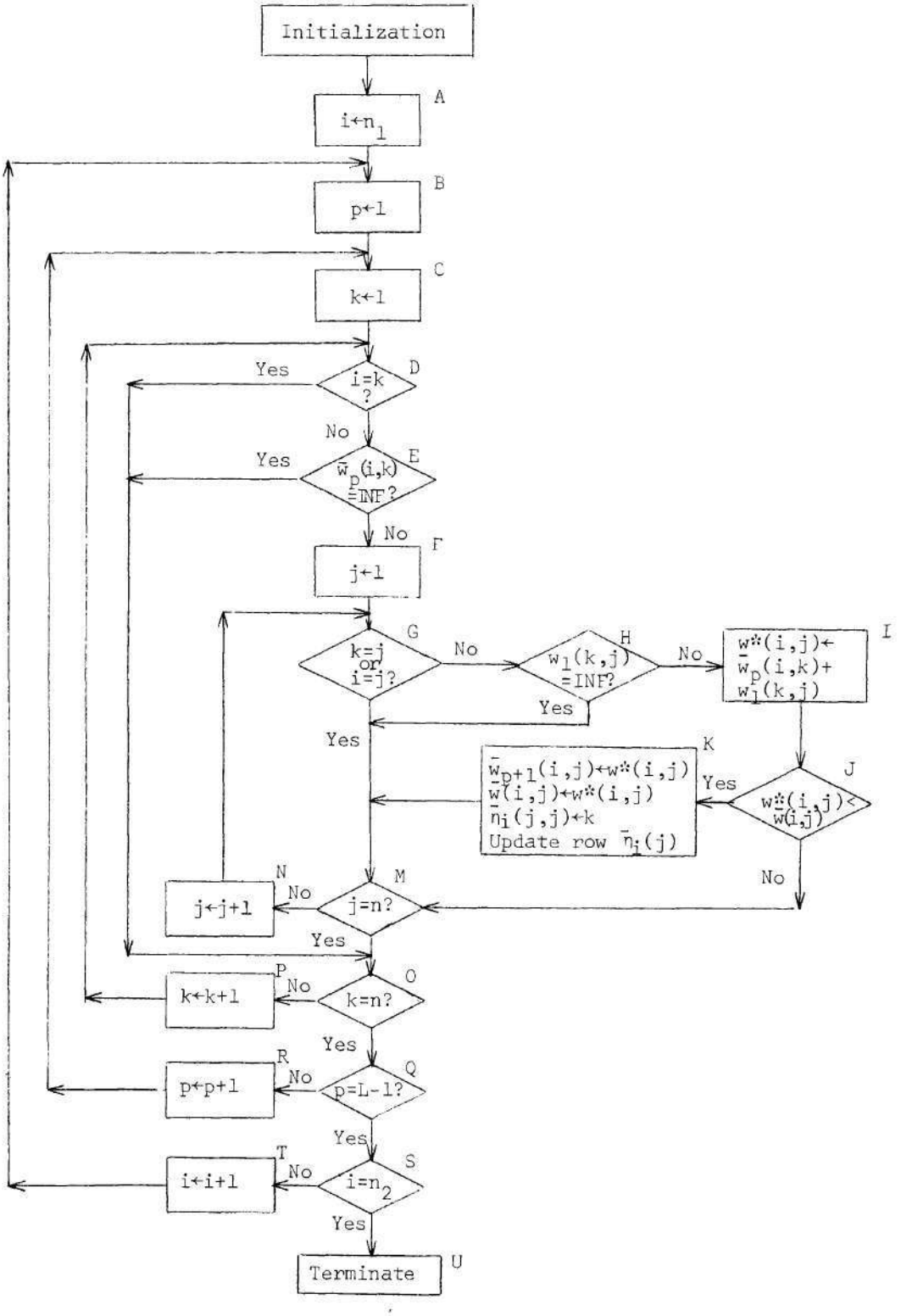


Figure 2.5. The Path Algorithm

T. Increment the i index by one.

Go to step B.

U. Terminate the algorithm.

Proof of the Path Algorithm

Lemma 2.1

Only the least weighted paths of length p are required to calculate the least weighted paths of length p+1 where path weight is calculated as the sum of the weights of the arcs along the path.

Proof. Assume that a path $\pi_p^*[x,y]$ is not a least weighted path of length p between node-pair (x,y). That is, there exists another path $\bar{\pi}_p[x,y]$ of length p such that $w(\pi_p^*) > w(\bar{\pi}_p)$.

Create the path $\pi_{p+1}^*(x,z)$ of length p+1 by concatenating the path $\pi_p^*(x,y)$ with the arc [y,z]'. Let the weight of arc [y,z]' be equal to w_1. Therefore, the weight of $\pi_{p+1}^*(x,z)$ is $w(\pi_{p+1}^*) = w(\pi_p^*) + w_1$. However, π_{p+1}^* is not the least weighted path of length p+1 between (x,z) because if path $\bar{\pi}_p$ is used instead of π_p^* to find a path $\bar{\pi}_{p+1}$ between x and z,

$$w(\pi_{p+1}^*) = w(\pi_p^*) + w_1 > w(\bar{\pi}_p) + w_1 = w(\bar{\pi}_{p+1}).$$

Therefore, a path of length p which is not a least weighted path cannot be used to calculate a path of length p+1 which is a least weighted path. □

Lemma 2.2

The least weighted paths of length p are used in the path algorithm to find the least weighted paths of length p+1 for each node-pair (i,j).

Proof. Step I in the algorithm finds the weight of all paths of length p+1 between node-pair (i,j) as follows:

$$\bar{w}_{p+1}(i,j) = \bar{w}_p(i\ k) + \bar{w}_1[k,j]'$$

for each node-pair (i,j) and all $k \epsilon N$, $i \neq k \neq j$. This calculation finds the weight of every path $\bar{\pi}_{p+1}(i,j)$ given the least weighted paths $\bar{\pi}_p(i,k)$ and the arcs $[k,j]'$. Steps I, J, and K in the algorithm insure that only least weighted paths of length p are used to find the least weighted paths of length p+1 for each node-pair (i,j). Specifically,

- —Step I uses only paths of length p to calculate the weights of paths of length p+1 for each (i,j).
- —Steps J and K (overall values of $k \epsilon N$, $i \neq k \neq j$) retain only the path of length p+1 between (i,j) which has least weight.

Thus the lemma is proved. □

Lemma 2.3

When p=n in the path algorithm, then *all* least weighted simple paths in a network on n nodes have been examined.

Proof. If p=n, then the p index in the algorithm runs from 1 to n-1 and all least weighted paths of length 1 to n-1 are found. The maximum length of a simple path is n-1 (for if this were not the case, at least one node would appear twice on the path and the path would not

be simple. Therefore, the algorithm examines all least weighted simple paths. □

Theorem 2.1

The path algorithm finds a path $\bar{\pi}(i,j)$ having least weight $w(\bar{\pi}(i,j))$ for each node-pair (i,j). That is:

$$w(\bar{\pi}(i,j)) = \underset{1\le p\le L}{\text{Min}} \{w(\bar{\pi}_p(i,j))\} \text{ for each } (i,j)\epsilon NP.$$

Proof. From Lemma 2.3, all least weighted simple paths are considered by the algorithm when p=n. If L<n, then the algorithm considers only those simple paths $\pi(i,j)\ni\ell(\pi(i,j))\le L$. This can be verified by examining the p index loop in the algorithm. For a given p, the least weighted paths of length p, $\bar{\pi}_p(i,j)$, are used to find the least weighted paths of length p+1, i.e., $\bar{\pi}_{p+1}(i,j)$. By stopping the algorithm at p=L-1, no paths $\bar{\pi}(i,j)\ni\ell(\bar{\pi}(i,j))>L$ can be created.

From Lemma 2.2, the algorithm finds, for each node-pair, the least weighted path of length p+1 from the least weighted paths of length p, i.e.,

$$w(\bar{\pi}_{p+1}(i,j)) = \underset{\substack{k\epsilon N\\ k\neq i,j}}{\text{Min}} \{w(\bar{\pi}_p(i,k)) + w(\pi_1(k,j))\}.$$

Now if the weight $w(\bar{\pi}_{p+1}(i,j))$ associated with path $\bar{\pi}_{p+1}(i,j)$ is less than the weight $\bar{w}(i,j)$ calculated to date for the node-pair (i,j), i.e.,

$$w(\bar{\pi}_{p+1}(i,j))<\bar{w}(i,j),$$

then the new path of length p+1 is the least weighted path found so far by the iterative procedure. It is, therefore, stored as the current least cost weight $\bar{w}(i,j)$ for node-pair (i,j). Since this process continues over all $p \leq L$,

$$\bar{w}\left(\bar{\pi}(i,j)\right) = \text{Min } \{\bar{w}\left(\bar{\pi}_1(i,j)\right), \bar{w}\left(\bar{\pi}_2(i,j)\right), \ldots, \bar{w}\left(\bar{\pi}_L(i,j)\right)\}.$$

$$= \underset{1 \leq p \leq L}{\text{Min}} \{\bar{w}\left(\bar{\pi}_p(i,j)\right)\}.$$

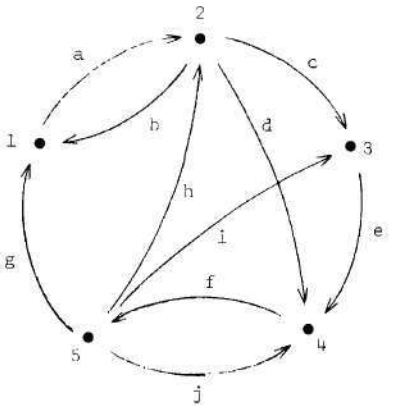
This method is sufficient to find the least weighted paths for all node-pairs (i,j) because of Lemma 2.1. □

Example

An example is now given to demonstrate how to use the path algorithm to solve an actual problem. Consider the five-node network shown on Figure 2.6 and assume that the arcs represent communication lines and the nodes represent stations in a communication network. Note that the network is strongly connected and therefore each station can send messages to all other stations. Associated with each communication line is a fixed cost which is assigned to the weight of the arc. We desire to use the path algorithm to find the least cost paths of any length (i.e., L=n) between all node-pairs in the network.

Figure 2.6 gives the incremental, as well as the final, solution to the above problem. This tabular layout is called a path table and is used to show the step-by-step results obtained from the path algorithm. The entries which are encircled in the path table represent paths that

Communication Network



Arc	Cost
a	2
b	3
c	2
d	4
e	1
f	2
g	-1
h	1
i	4
j	2

Path Table

(i,j)	$\bar{\pi}_1(i,j)$	$\bar{w}_1(i,j)$	$\bar{\pi}_2(i,j)$	$\bar{w}_2(i,j)$	$\bar{\pi}_3(i,j)$	$\bar{w}_3(i,j)$	$\bar{\pi}_4(i,j)$	$\bar{w}_4(i,j)$	$\bar{\pi}(i,j)$	$\bar{w}(i,j)$
(1,2)	a	2							{a}	2
(1.3)			ac	4			(adfi)	12	{a,c}	4
(1,4)			ad	6	ace	5			{a,c,e}	5
(1,5)					adf	8	acef	7	{a,c,e,f}	7
(2,1)	b	3			(dfg)	5	(cefg)	4	{b}	3
(2,3)	c	2			(dfi)	10			{c}	2
(2,4)	d	4	ce	3					{c,e}	3
(2,5)			df	6	cef	5			{c,e,f}	5
(3,1)					efg	2	(efhb)	7	{e,f,g}	2
(3,2)					efh	4	(efga)	4	{e,f,h}	4
(3,4)	e	1							{e}	1
(3,5)			ef	3					{e,f}	3
(4,1)			fg	1	(fhb)	6			{f,g}	1
(4,2)			fh	3	(fga)	3			{f,h}	3
(4,3)			fi	6	fhc	5			{f,h,c}	5
(4,5)	f	2							{f}	2
(5,1)	g	-1	(hb)	4					{g}	-1
(5,2)	h	1	(ga)	1					{h}	1
(5,3)	i	4	hc	3					{h,c}	3
(5,4)	j	2	(hd ie)	5 5	hce	4			{j}	2

Figure 2.6. Communication Network and Path Table Solution

are potential candidates for the least cost path (from step I of Figure 2.5), but have weights which are greater than or equal to a previously calculated path weight for the same node-pair (i.e., fail to qualify in step J).

Auxiliary Algorithm

The path algorithm gives as an output the matrices, $[\bar{w}]$ and $[\bar{n}_1]$. Now $[\bar{n}_i]$ contains all the links which are needed to reconstruct all the least weighted paths back to node i in the network. In this section an algorithm (called "the auxiliary algorithm") is presented which finds the network path set PS as well as the capacity of each path in the set. The network path set is referred to as $\{\bar{\pi}(i,j)\}$ and the path capacities as $b(\bar{\pi}(i,j))$ for each $(i,j) \epsilon NP$.

The auxiliary algorithm is presented in flow chart form on Figure 2.7. Because of the simplicity of the algorithm, a formal discussion and proof are omitted. Note that if no path is found by the path algorithm for node-pair (x,y), then $\{\bar{\pi}(x,y)\} = \phi$ and $b(\bar{\pi}(x,y)) = MAX$.

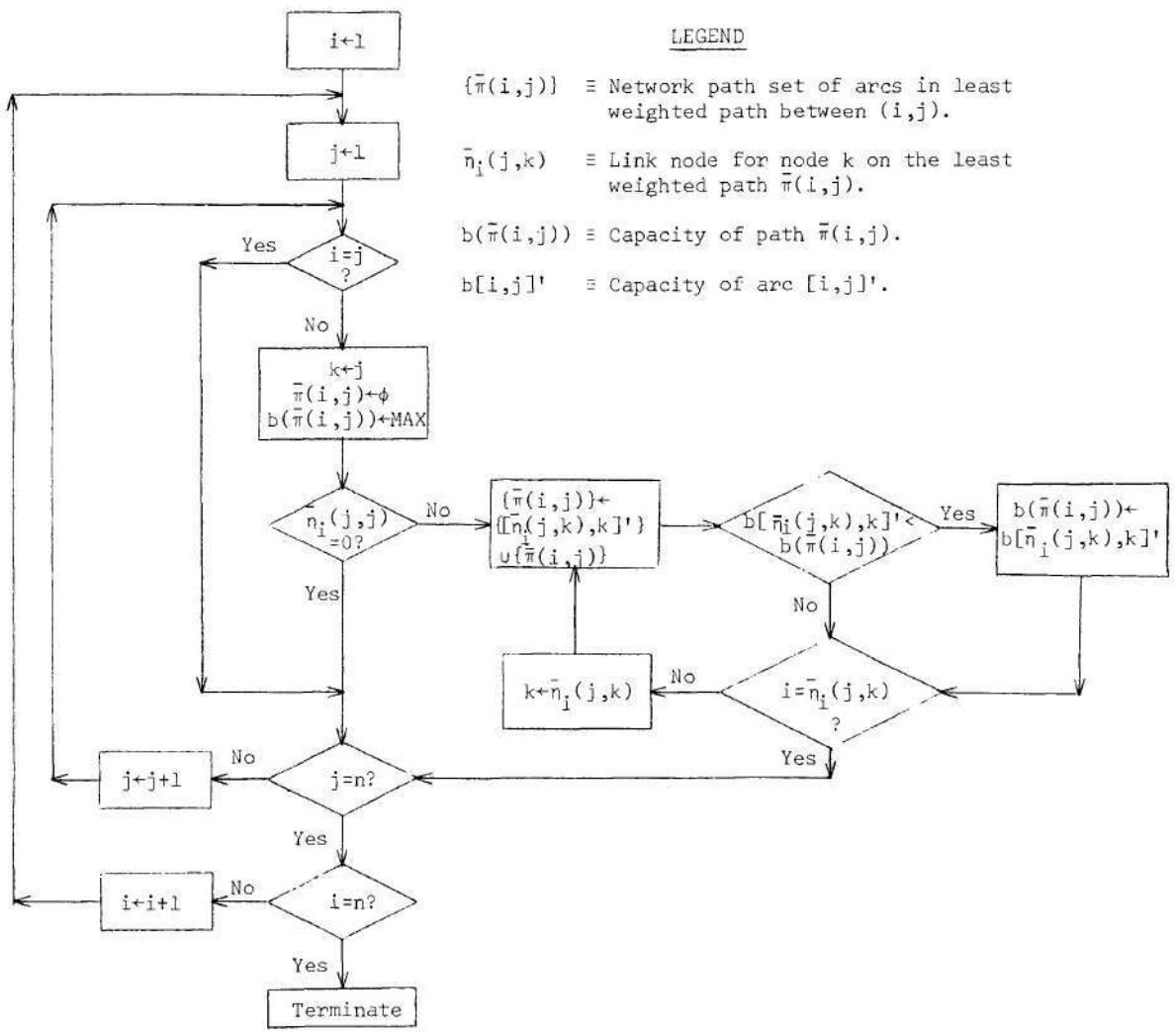


Figure 2.7. Auxiliary Algorithm

CHAPTER III

SOLUTION TO NETWORK PROBLEM

Introduction

In this chapter the simultaneous flow, minimum cost problem for single path networks is solved. The solution is expressed in the form of a two-phased solution algorithm which is called "the network algorithm" (NA). The network algorithm is described in a general way so that it can be applied to any network configuration which is adequately defined. If an optimum solution exists for any given network configuration, it can always be found by using the network algorithm. The algorithm is illustrated in Chapter V where it is used to solve a communication problem for a large, decentralized company.

The simultaneous flow problem has been studied by many people since its introduction in 1958 by Ford and Fulkerson [F13]. The most recent solutions use a linear programming approach to solve the undirected minimum cost formulation of the problem (e.g., Tang [F60], Gomory and Hu [F28], Tomlin [F64], and Hu [A13]). Slight variations are common in the expression of the problem as evidenced by the bipath solution of Tang and the handling of directed networks by Tomlin. A description of the variation of the problem which is considered in this dissertation now follows.

The Network Problem

The network problem solved in this chapter is formally stated below. The network G(N,A) under consideration must be strongly connected with a finite node set N containing n nodes ($1 \leq n$) and a finite arc set A containing m arcs $(n-1 \leq m \leq n(n-1))$.

Given:

1. The cost per unit flow $c(\alpha)$ for each $\alpha \in A$, where $c(\alpha) \geq 0$.
2. The maximum flow capacity $b(\alpha)$ for each $\alpha \in A$, where $b(\alpha) \geq 0$
3. The node-pair flow requirement $r(i,j)$ for each node-pair $(i,j) \in NP$ and $r(i,j) \geq 0$.
4. The maximum length (i.e., number of arcs) L for any simple path $\bar{\pi}(i,j)$ where $1 \leq L \leq n-1$.

Find:

A network path set $\{\bar{\pi}(i,j), \forall (i,j) \in NP\}$ (i.e., a solution) having a simultaneous flow assignment $\{f(\bar{\pi}(i,j)), \forall (i,j) \in NP\}$ which satisfies the following *network conditions:*

1. *Minimum Cost.* The overall cost of a network flow assignment is a minimum over all possible network path sets. That is:

$$\underline{\text{COST}} = \underset{\{\bar{\pi}(i,j)\}}{\text{Min}} \left\{ \sum_{(i,j) \in NP} f(\bar{\pi}(i,j)) \cdot c(\bar{\pi}(i,j)) \right\}$$

2. *Path Length.* The length $\ell(\bar{\pi}(i,j))$ of each path $\bar{\pi}(i,j)$ for all $(i,j) \in NP$ is less than or equal to L. That is: $\ell(\bar{\pi}(i,j)) \leq L$ for all $(i,j) \in NP$.

3. *Flow Requirements.* The flow assignment $f(\bar{\pi}(i,j))$ along the path $\bar{\pi}(i,j)$ for all $(i,j)\epsilon NP$ is greater than or equal to the flow requirement $r(i,j)$ for node-pair (i,j). That is: $f(\bar{\pi}(i,j))\geq r(i,j)$ for all $(i,j)\epsilon NP$.

4. *Arc Capacity.* The flow $f(\alpha)$ on arc α for all $\alpha\epsilon A$ is less than or equal to the arc capacity $b(\alpha)$. That is: $f(\alpha)\leq b(\alpha)$ for all $\alpha\epsilon A$.

Where:

COST $\equiv$ Minimum cost of a network solution for the network $G(N,A)$.

$\bar{\pi}(i,j)$ $\equiv$ The simple path between node-pair (i,j).

$AS(\bar{\pi}(i,j))$ $\equiv$ The arc set of the path $\bar{\pi}(i,j)$.

$f(\bar{\pi}(i,j))$ $\equiv$ The flow on path $\bar{\pi}(i,j)$.

$r(i,j)$ $\equiv$ The flow requirement for node-pair (i,j).

$$f_\alpha(i,j) \equiv \begin{cases} 0 & \text{if } \alpha \notin AS(\bar{\pi}(i,j)). \\ r(i,j) & \text{if } \alpha\epsilon AS(\bar{\pi}(i,j)). \end{cases}$$

$$f(\alpha) \equiv \sum_{(i,j)\epsilon NP} f_\alpha(i,j) \equiv \text{The total flow on arc } \alpha\epsilon A.$$

$b(\alpha)$ $\equiv$ The maximum flow capacity on arc $\alpha\epsilon A$.

$c(\alpha)$ $\equiv$ The cost per unit flow on arc $\alpha\epsilon A$.

$\ell(\bar{\pi}(i,j))$ $\equiv$ The length of the simple path $\bar{\pi}(i,j)$.

L $\equiv$ The maximum length of any simple path.

The Solution Approach

A solution expressed in terms of a network path set $\{\bar{\pi}(i,j), \forall(i,j)\epsilon NP\}$ is required to solve the network problem. The sequence of arcs for each node-pair in this set represents the path over which messages should be sent in order to relize the minimum overall cost of

message flow in the network. In order to simplify the notation, the term "solution" will be used in a general sense in this chapter. That is, any bona fide network path set can be considered a solution. In order to distinguish different solutions, the following notation is used to denote solution $K:\{\bar{\pi}(i,j),\forall(i,j)\epsilon NP\}_K$.

The network conditions 1 and 4 are referred to, respectively, as the optimality and feasibility conditions. An *acceptable solution* is defined as a solution which first satisfies conditions 2, 3 and 4, and is then selected as a least cost solution which satisfies condition 1. It is, therefore, properly referred to as an optimum feasible solution.

The path algorithm (PA), developed in Chapter II, is an integral part of the solution approach discussed in this chapter. It is used in both phases of the network algorithm to find a solution which satisfies network conditions 1 and 2. In the Phase I algorithm a solution is found which satisfies network conditions 1, 2 and 3. The Phase II algorithm finds a solution (if one exists) which satisfies all four conditions. That is, it finds an acceptable solution to the network problem.

The Network Algorithm: Phase I

See Figure 3.1 for a flow chart of the following algorithm. The inputs to Phase I are:

1. The set of arcs A in a strongly connected network SCN.
2. The unit cost $c(\alpha)$ for each $\alpha\epsilon A$.
3. The maximum flow capacity $b(\alpha)$ for each $\alpha\epsilon A$.
4. The maximum path length L.

The sequence of steps required in the Phase I algorithm now follows:

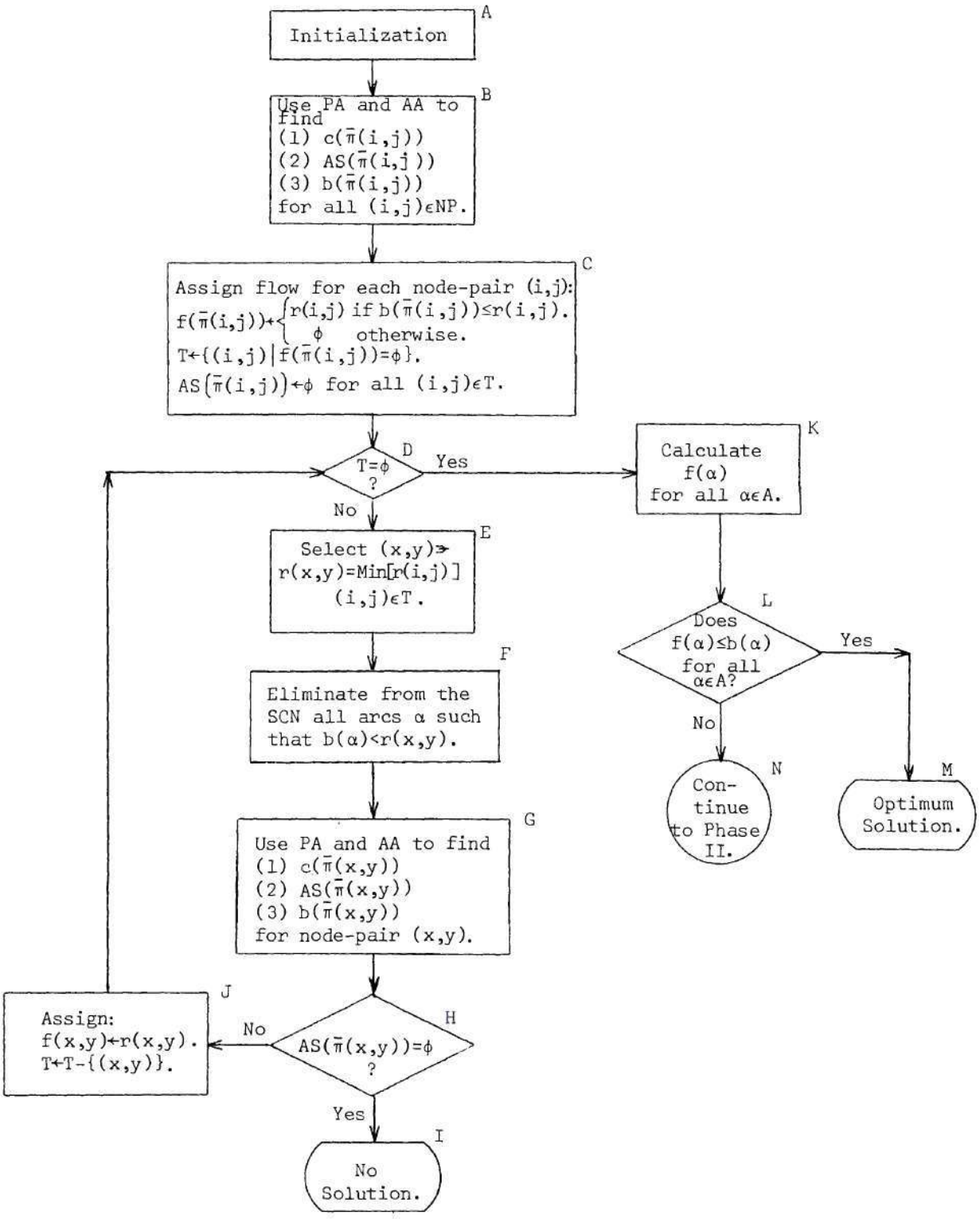


Figure 3.1. Phase I Algorithm

A. Initialize the set of node-pairs T; i.e., $T \leftarrow \phi$. (The node-pairs which are eventually placed in set T require individual handling to find paths with required capacity.)

B. Use the path algorithm (PA) and the auxiliary algorithm (AA) to find for each node-pair (i,j) in SCN such that $\ell(\bar{\pi}(i,j)) \leq L$, (1) the cost of the least cost path $c(\bar{\pi}(i,j))$; (2) the least cost arc set $AS(\bar{\pi}(i,j))$; (3) the path capacity $b(\bar{\pi}(i,j))$. (Since SCN is strongly connected, at least one path exists between each node-pair.)

C. Assign node-pair flow requirements to each path. If the calculated capacity $b(\bar{\pi}(x,y))$ of a path is greater than or equal to the flow requirements, then assign a flow between node-pair (x,y) equal to the flow requirements. If $b(\bar{\pi}(x,y))$ is less than r(x,y), then not enough flow can be assigned to that particular path to satisfy the flow requirements. Therefore, place in set T the node-pairs which do not currently have paths which can carry the required flow, i.e.,

$$T = \{(i,j) \mid r(i,j) > b(\bar{\pi}(i,j)), \forall (i,j) \in NP\}.$$

Set $AS(\bar{\pi}(i,j)) \leftarrow \phi$ for all $(i,j) \in T$, since the paths $\bar{\pi}(i,j)$ have not been found which can carry the required flow.

D. If set T is empty (i.e., flow has been assigned to all paths $\bar{\pi}(i,j)$), then go to step K. If T contains at least one element, then continue to step E.

E. Find a node-pair (x,y) in T which has a least value for its flow requirements. That is, select $(x,y) \ni r(x,y) = \underset{(i,j)\epsilon T}{\text{Min}} \{r(i,j)\}$.

F. Eliminate from the SCN for the remaining steps in Phase I all arcs α having their flow capacity $b(\alpha)$ less than the flow requirement $r(x,y)$. (These arcs can be eliminated because they cannot be used to carry the flow requirements for any node-pair remaining in set T.)

G. Use the path algorithm (PA) and the auxiliary algorithm (AA) to find for node-pair (x,y) in SCN such that $\ell\left(\bar{\pi}(x,y)\right) \leq L$, (1) the cost of the least cost path $c\left(\bar{\pi}(x,y)\right)$; (2) the least cost arc set $AS\left(\bar{\pi}(x,y)\right)$; (3) the path capacity $b\left(\bar{\pi}(x,y)\right)$.

H. If $AS\left(\bar{\pi}(x,y)\right) = \phi$, then the PA could not find a least weighted path of length L or less between node-pair (x,y). Therefore, go to step I which terminates the algorithm. If $AS\left(\bar{\pi}(x,y)\right) \neq \phi$, then the PA has found a least weighted path for node-pair (x,y). Continue to step J.

I. Exit from the Phase I algorithm. A path could not be found between node-pair (x,y) which has length L or less and/or can carry the flow requirement r(x,y).

J. Assign the node-pair flow requirement r(x,y) to the path flow $f\left(\bar{\pi}(x,y)\right)$ for node-pair (x,y).

Eliminate node-pair (x,y) from the set T.

Go to step D.

K. (At this point in the algorithm a network path set $\{\bar{\pi}(i,j), \forall(i,j)\epsilon NP\}$ has been found for the SCN where $\bar{\pi}(i,j)$ represents the least cost path between (i,j) which satisfies network conditions 1, 2 and 3.) Calculate the flow $f(\alpha)$ for each arc $\alpha\epsilon A$. Use the set of flows $\{f(\bar{\pi}(i,j)), \forall(i,j)\epsilon NP\}$ to calculate $f(\alpha)$ as follows:

$$f(\alpha) = \sum_{(i,j)\epsilon NP} f_{\alpha}(i,j) \quad \text{for all } \alpha\epsilon A,$$

where

$$f_{\alpha}(i,j) = \left\{ \begin{matrix} 0 & \text{if } \alpha \notin A(\bar{\pi}(i,j)) \\ r(i,j) & \text{if } \alpha\epsilon A(\bar{\pi}(i,j)) \end{matrix} \right\} \text{for all } (i,j)\epsilon NP.$$

L. If $f(\alpha) \leq b(\alpha)$ for all $\alpha\epsilon A$, then continue to step M. If not, go to step N.

M. The Phase I algorithm has found a solution to the network problem which satisfies the four network conditions. Output the following:

1. The network path set $\{\bar{\pi}(i,j), \forall(i,j)\epsilon NP\}$.
2. The flow $f(\alpha)$ for each $\alpha\epsilon A$.
3. The total network cost, COST, as calculated by:

$$\underline{\text{COST}} = \sum_{\alpha\epsilon A} f(\alpha) \cdot c(\alpha).$$

Terminate the Phase I algorithm.

N. At this point in the Phase I algorithm, a solution has been found which satisfies conditions 1, 2 and 3. Phase II is required in

order to satisfy condition 4.

Terminate the Phase I algorithm.

Proof of Phase I Algorithm

Lemma 3.1

If a least cost path set $\{\bar{\pi}(i,j), \forall (i,j) \epsilon NP\}$ is found by the Phase I algorithm such that the path flow $f(\bar{\pi}(i,j)) = r(i,j)$, then the network cost, COST = $\sum_{(i,j)\epsilon NP} f(\bar{\pi}(i,j)) \cdot c(\bar{\pi}(i,j))$, is a minimum.

Proof. Regardless of the network path set $\{\pi^*(i,j), \forall (i,j) \epsilon NP\}$ selected by the Phase I algorithm, the flow which is assigned to each path $\pi^*(i,j)$ is always the same; that is:

$$f(\pi^*(i,j)) = r(i,j).$$

This assignment is performed in step C or step J of the Phase I algorithm.

By Theorem 2.1, the path algorithm always finds the least cost path $\bar{\pi}(i,j)$ for each node-pair. Since each $f(\bar{\pi}(i,j))$ is multiplied by the unit cost $c(\bar{\pi}(i,j))$ of path $\bar{\pi}(i,j)$, a minimum value for each $c(\bar{\pi}(i,j))$ results in a minimum cost, COST, for the network because,

$$\sum \text{Min}\{f(\bar{\pi}(i,j)) \cdot c(\bar{\pi}(i,j))\} = \text{Min}\{\sum f(\bar{\pi}(i,j)) \cdot c(\bar{\pi}(i,j))\}.$$

Therefore, as long as a least cost path is found for each node-pair, the calculation of the total cost is a minimum. □

Lemma 3.2

Given an assignment of flow to the paths in a network path set $\{\pi'(i,j), \forall (i,j) \in NP\}$, then

$$\sum_{(i,j)\in NP} f\left(\pi'(i,j)\right) \cdot c\left(\pi'(i,j)\right) = \sum_{\alpha\in A} f(\alpha) \cdot c(\alpha),$$

where $f(\alpha) = \sum_{(i,j)\in NP} f_\alpha\left(\pi'(i,j)\right)$ for all $\alpha \in A$ and

$c(\alpha) = \sum_{(i,j)\in NP} c_\alpha\left(\pi'(i,j)\right)$ for all $\alpha \in A$.

Proof.

$$\sum_{(i,j)\in NP} f\left(\pi'(i,j)\right) \cdot c\left(\pi'(i,j)\right) = \sum_{(i,j)\in NP} \left[f\left(\pi'(i,j)\right) \cdot \sum_{\alpha\in A} c_\alpha\left(\pi'(i,j)\right)\right]$$

$$= \sum_{(i,j)\in NP} \sum_{\alpha\in A} \left[f_\alpha\left(\pi'(i,j)\right) \cdot c_\alpha\left(\pi'(i,j)\right)\right]$$

$$= \sum_{\alpha\in A} \sum_{(i,j)\in NP} \left[f_\alpha\left(\pi'(i,j)\right) \cdot c\left(\pi'(i,j)\right)\right]$$

$$= \sum_{\alpha\in A} \left[\sum_{(i,j)\in NP} f_\alpha\left(\pi'(i,j)\right)\right] \left[\sum_{(i,j)\in NP} c_\alpha\left(\pi'(i,j)\right)\right]$$

$$= \sum_{\alpha\in A} f(\alpha) \cdot c(\alpha). \quad \square$$

Theorem 3.1

The Phase I algorithm finds a network path set $\{\bar{\pi}(i,j), \forall (i,j) \in NP\}$ such that:

$$1.\quad \underline{\text{COST}} = \sum_{(i,j)\in NP} f\left(\bar{\pi}(i,j)\right) \cdot c\left(\bar{\pi}(i,j)\right) = \sum_{\alpha\in A} f(\alpha) \cdot c(\alpha)$$

where COST is the minimum total cost of any network path set which satisfies the following conditions:

2. $\ell\left(\bar{\pi}(i,j)\right) \leq L$ for all $(i,j)\in NP$.

3. $f\left(\bar{\pi}(i,j)\right) = r(i,j)$ for all $(i,j)\in NP$.

Proof. We begin the proof by establishing the fact that the Phase I algorithm finds a least cost path for each node-pair (x,y) which is constrained by conditions 2 and 3 above.

In step C in the Phase I algorithm a node-pair (x,y) is placed in set T if the capacity of the path $\bar{\pi}(x,y)$ is less than the required capacity r(x,y) for the node-pair. For each node-pair (x,y) in the network not placed in set T, Theorem 2.1 guarantees that a least cost path $\bar{\pi}(x,y)$ is found and that condition 2 holds for the path. Step C in the algorithm satisfies condition 3 as the required flow is assigned to the path.

Special handling is performed by the algorithm for those node-pairs in set T. New paths must be found between these node-pairs which have a capacity greater than or equal to their respective flow requirements. Since the capacity of a path is calculated as:

$$b[\bar{\pi}(x,y)] = \min_{\alpha\in A\left(\bar{\pi}(x,y)\right)} \{b(\alpha)\},$$

arcs α' having a capacity less than the required path capacity r(x,y) cannot be used to create a path between (x,y). If the arcs α' are

eliminated from consideration for each (x,y) and the path algorithm then used, Theorem 2.1 can be used to guarantee that a least cost path is found (if it exists) which satisfies condition 2. Step J in the algorithm satisfies condition 3. Since each $(x,y)\epsilon T$ is selected in turn such that:

$$r(x,y) = \underset{(i,j)\epsilon T}{\text{Min}} \{r(i,j)\},$$

the arc elimination at step F eliminates only those arcs α' which are less than r(x,y) and are, therefore, ineligible for use on any path between (x,y).

Since the Phase I algorithm finds the minimum cost path for each node-pair, we apply Lemma 3.1 to these separate results in order to validate network condition 1 for the entire network path set. Lemma 3.2 completes the proof by confirming that the two equations for computing total network cost are equivalent. □

Introduction to Phase II Algorithm

The Phase II algorithm uses a branch and bound philosophy to solve the network problem. The real power in a branch and bound approach lies in the way branches are constructed and bounds used in a particular algorithm to achieve the desired results. Fortunately, the overall Phase II optimization problem is consistent with and readily adapted to a branch and bound method of solution.

In the Phase II algorithm a solution tree, abbreviated ST, is constructed for the network problem being solved. Each node in the

solution tree represents a possible solution to the network problem. That is, each node represents a network path set for the given problem. Network conditions 1 and 4 play a significant role in the algorithm while the other two conditions are automatically satisfied for all solutions which are considered. Earlier we remarked that if a solution satisfies network condition 1 it is called an *optimum solution*; if a solution satisfies network condition 4 it is called a *feasible solution.* Thus an optimum feasible solution is desired for the network problem.

The Phase II algorithm initializes the solution tree by placing the infeasible solution found in Phase I at the root node. Phase II then generates new nodes in the tree until it either finds an optimum feasible solution or finds that no optimum solution exists.

There are four kinds of terminal nodes in ST. They are labeled and defined as follows:

NS - Nodes signifying that no solution is possible.

IS - Nodes representing solutions which are infeasible.

FS - Nodes representing non-optimum, feasible solutions (i.e., a solution which satisfies conditions 2, 3 and 4, but *not* 1).

OS - Nodes representing an optimum, feasible solution (only one of these is selected by the algorithm).

The cost of each solution node in ST, COST, is calculated and assigned to the node and is used by the algorithm as the bound for the node. A cost equal to MAX (i.e., a very large number) is assigned to NS nodes since no cost can be calculated for them.

The Phase II algorithm selects an IS solution node in ST and examines those arcs $\alpha \epsilon A$ of its infeasible solution for which $f(\alpha) > b(\alpha)$.

The algorithm then reassigns the excess flow in any arc which has a flow greater than its capacity to other arcs and thereby creates new solution nodes in ST. The creation of new nodes in ST from an IS node is called "branching" or "flow reassignment." The new nodes created by flow reassignment can be any of the four types of nodes already discussed (i.e., NS, IS, FS, OS). The process of determining the type of node created by flow reassignment is called "evaluation."

Lemma 3.4, given near the end of this chapter, guarantees that the cost (i.e., bound) associated with each node in ST along any path from the root node will be greater than or equal to the cost (i.e., bound) of the preceding or parent IS node. Therefore, no further branching is necessary for an NS node whose cost equals MAX or an IS or FS node whose cost is greater than the cost of another FS node. One of the FS nodes having least cost is selected as the optimum feasible solution OS to the original network problem.

The overall Phase II algorithm is organized in the following major sections (steps are discussed in the next section):

—Reassignment of flow for an IS node. Steps C-H.

—Evaluation of new nodes created by flow reassignment. Steps I-Q.

—Determination of whether a new FS node qualifies as the optimum solution to the network problem. Steps R-T.

Since the algorithm is based on a branch and bound philosophy, traditional branch and bound characteristics of the algorithm are now presented.

Branching Characteristic. Overflow arc(s) in IS node.

Bounding Characteristic. Cost of flow for solution node.

Branching Policy. Choose an infeasible solution having the smallest number of overflow arcs.

Optimum Feasible Solution. A solution which satisfies network conditions 1, 2, 3 and 4.

Convergence Criterion. At least one new arc will be eliminated from consideration from a node-pair for each new node in the solution tree.

Root Node. Infeasible solution from Phase I.

The Network Algorithm: Phase II

The corresponding flow chart of the steps in the Phase II algorithm is given in Figure 3.2. The sequence of steps now follows.

A. Initialize the Phase II algorithm.

1. Place the infeasible solution from Phase I into the root node of the solution tree ST and calculate the cost of the Phase I solution.

2. Create the set IS of infeasible solution nodes on which branching has not been performed; i.e., IS = {K|K is an IS node and the flow on solution K has not been reassigned}. (Note that the root node is the only element of IS at this time.)

3. Initialize the minimum cost; i.e., COST←MAX.

4. Create the history set H(K) for the root node R=K; i.e.,

$$H(K) = \{(a,i,j) \mid \forall a \epsilon A \text{ and } \forall (i,j) \epsilon NP \ni r(i,j) > b(a)\}.$$

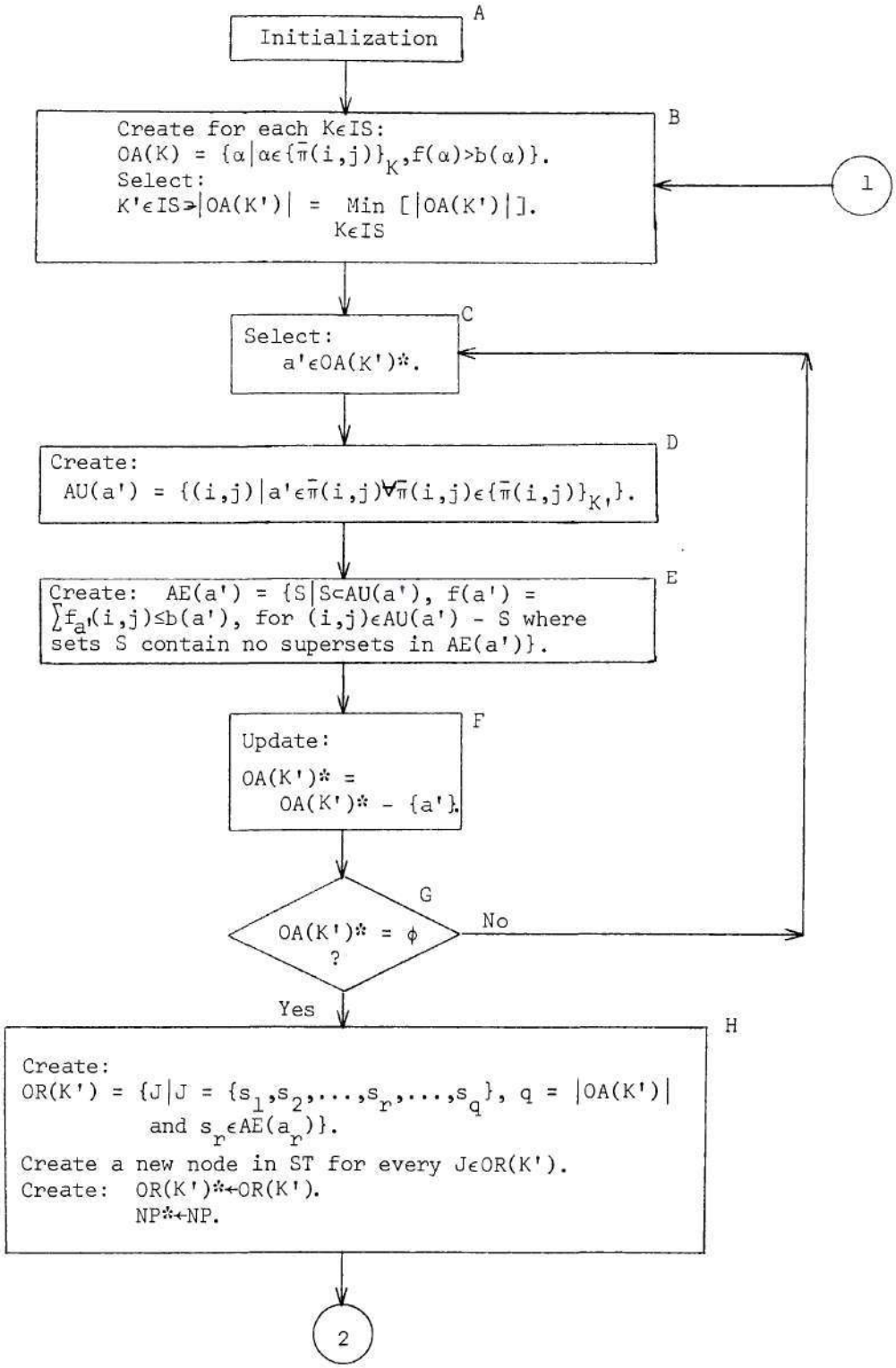


Figure 3.2. Phase II Algorithm

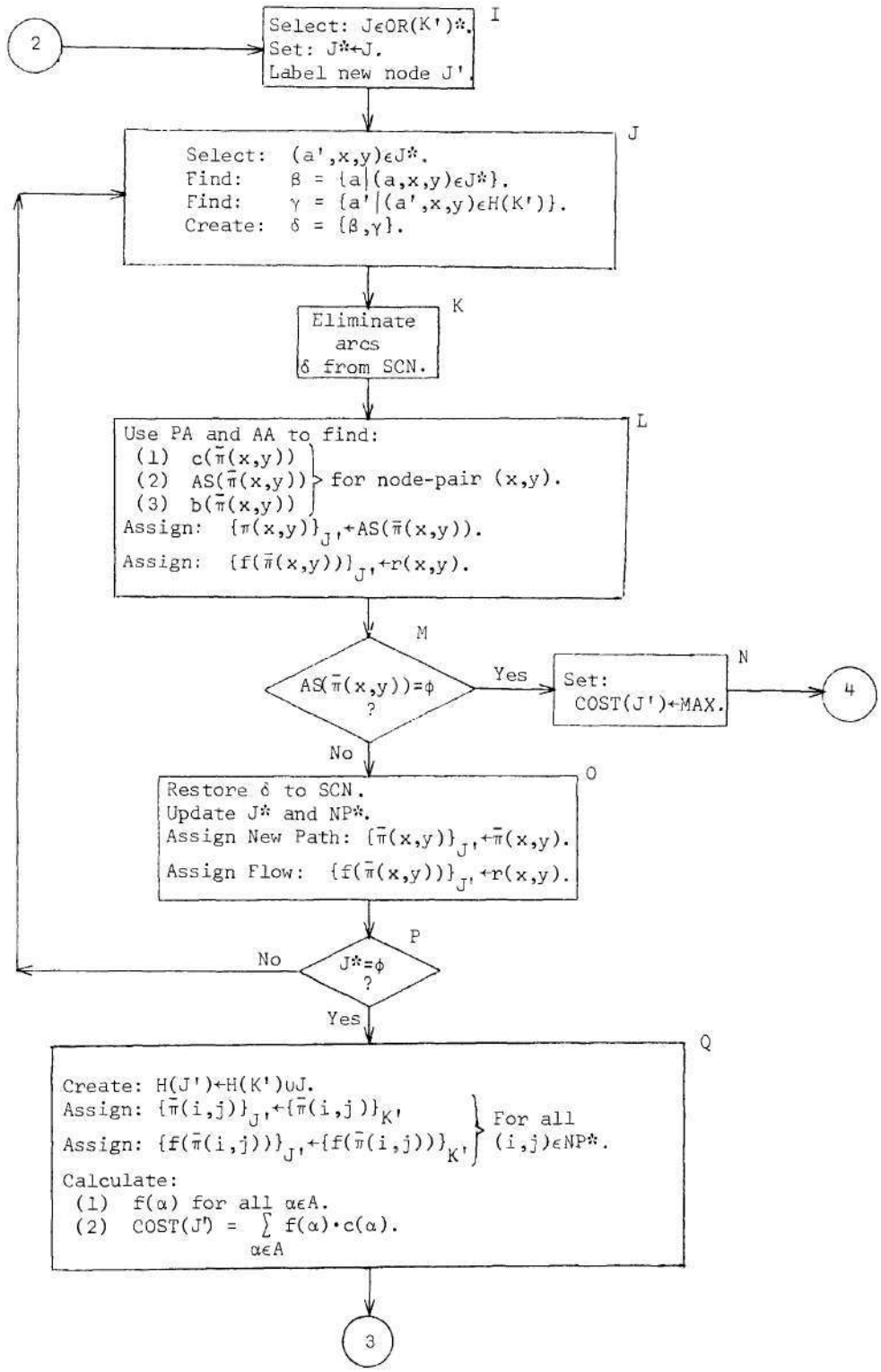


Figure 3.2. Phase II Algorithm (Continued)

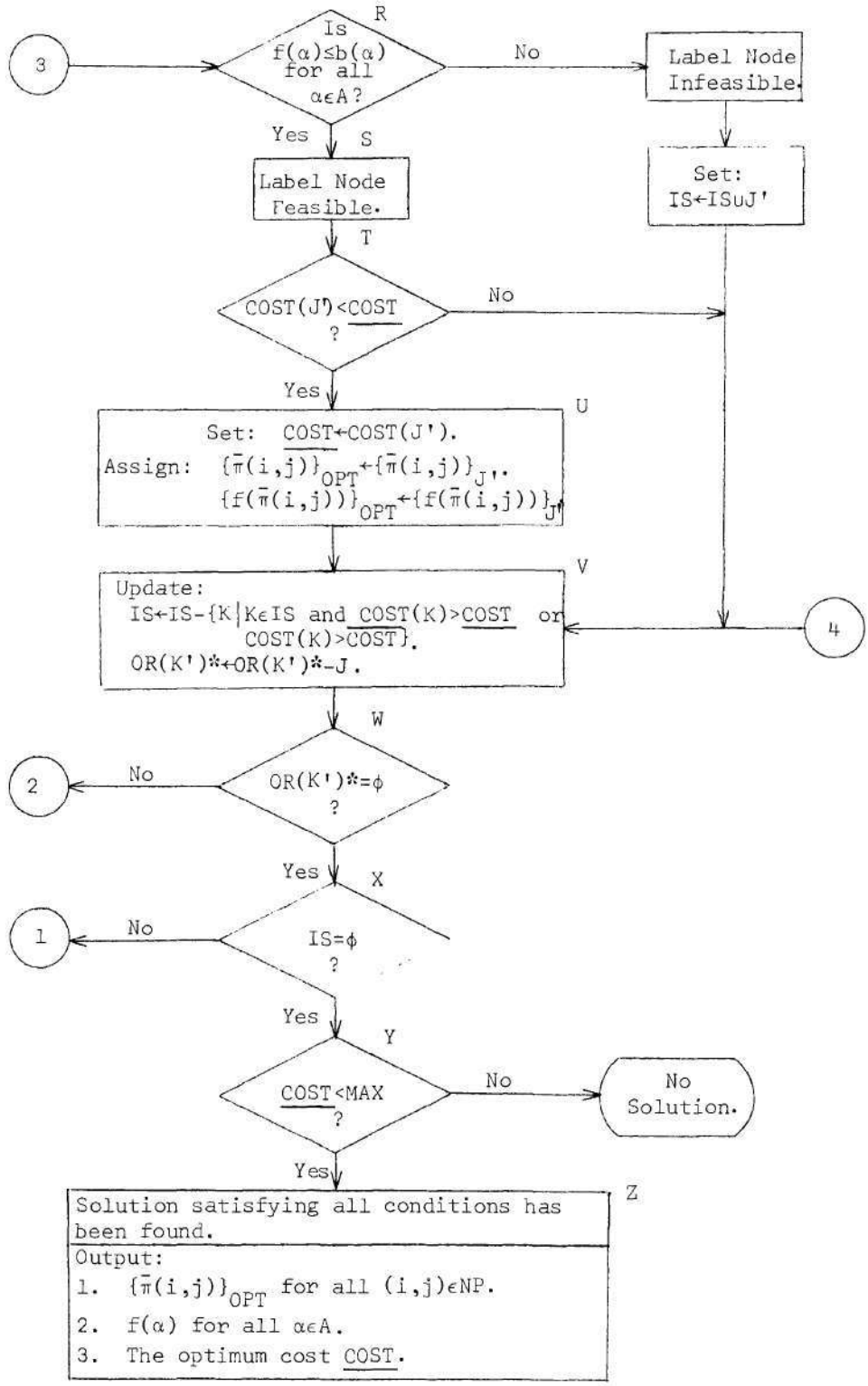


Figure 3.2. Phase II Algorithm (Continued)

The history set H(K) for node K is the set of triplets (a,i,j) in which arc a will not be used to construct a path between node-pair (i,j) for solution node K and for any solution created from node K.

5. Calculate the maximum cost of the Phase II solution. That is:

$$\overline{COST} = \sum_{\alpha \epsilon A} b(\alpha) \cdot c(\alpha).$$

B. Find the set of overflow arcs OA(K) for each infeasible solution node $K \epsilon IS$. That is: Create $OA(K) = \{\alpha | \alpha \epsilon \{\bar{\pi}(i,j)\}_K, f(\alpha) > b(\alpha)\}$, for each $K \epsilon IS$. Select a solution node $K' \epsilon IS$ having the minimum number of overflow arcs. That is: $|OA(K')| = \underset{k \epsilon IS}{Min}\{|OA(K)|\}$. Create a working set $OA(K')^* \leftarrow OA(K')$.

C. Select any overflow arc $a' \epsilon OA(K')^*$ in solution node K'.

D. Create the arc utilization set AU(a') of node-pairs (i,j) such that the overflow arc a' is on the path $\bar{\pi}(i,j)$ in the network solution $\{\bar{\pi}(i,j)\}_{K'}$. That is:

$$AU(a') = \{(i,j) | a' \epsilon \bar{\pi}(i,j) \text{ for all } \bar{\pi}(i,j) \epsilon \{\bar{\pi}(i,j)\}_{K'}\}.$$

E. Find a set AE(a') of subsets $S \subset AU(a')$ such that:

(1) If S is removed from AU(a'), then the flow f(a') for all node-pairs remaining in AU(a') is:

$$f(a') = \sum_{(i,j) \epsilon AU(a')-S} f_{a'}(i,j) \leq b(a').$$

(2) If $S_1 \subset AE(a')$ and $S_2 \subset AE(a')$ such that $S_1 \subset S_2$, then remove S_2 from AE(a'); i.e., all sets $S \subset AE(a')$ contain no supersets in AE(a').

(The significance of this step in the algorithm is discussed in Lemma 3.5. The set AE(a') contains summary information concerning the overflow characteristics of arc a'.)

F. Eliminate arc a' from the set OA(K'). That is, $OA(K')^* \leftarrow OA(K')^* - \{a'\}$.

G. If the set OA(K') is empty, continue to step H. If not, go to step C.

H. Redefine the set AE(a') as a set of triplets $\{(a',i,j) | (i,j) \in AE(a')\}$ for each $a' \in OA(K')$.

Create a set OR(K') in which each element $J \in OR(K')$ consists of one element S from each of the q sets AE(a'); i.e., OR(K') = $\{J | J = \{s_1, s_2, \ldots, s_r, \ldots, s_q\}$, $q = |OA(K')|$, and $s_r \in AE(a_r)\}$.

Create a new node in ST branching from node K' for every element $J \in OR(K')$.

(Each element J in OR(K') represents a solution having a minimum increase in cost over solution K' and eliminating the infeasibility on the arcs $a' \in OA(K')$. See Lemma 3.5 for an explanation.)

Create a working copy $OR(K')^*$ of set OR(K'); i.e., $OR(K')^* \leftarrow OR(K')$.

I. Select one of the sets $J \in OR(K')^*$ for evaluation.

Label the new node being evaluated J'.

Create a working copy J^* of set J; i.e., $J^* \leftarrow J$.

Create a set of all node-pairs NP^*; i.e., $NP^* \leftarrow NP$.

J. Select any element $(a',x,y) \epsilon J^*$.

Find those arcs β which *cannot* be used by a path between (x,y) in the reassignment of overflow for solution node J'. That is:

$$\beta = \{a | (a,x,y) \epsilon J^*\}.$$

Find those arcs γ which *cannot* be used by a path between (x,y) for the solution at node K'. That is:

$$\gamma = \{a | (a,x,y) \epsilon H(K')\}.$$

Create the set of all arcs δ which *cannot* be used by a path between (x,y) for solution node J.

$$\delta = \{\beta,\gamma\}.$$

K. Eliminate the arcs δ from the SCN.

(These arcs δ are restricted from being used on a path between node-pair (x,y).)

L. Use the path algorithm (PA) and the auxiliary algorithm (AA) to find the following for node-pair (x,y) in SCN where all path lengths are less than or equal to L.

(1) The least cost path $\bar{\pi}(x,y)$.

(2) The cost $c\left[\bar{\pi}(x,y)\right]$ of path $\bar{\pi}(x,y)$.

(3) The capacity $b(\bar{\pi}(x,y))$ of path $\bar{\pi}(x,y)$.

M. If $\bar{\pi}(x,y) = \phi$, then the path algorithm was unable to find a path. In this case go to step N. If the path set is not empty, then continue to step O.

N. Set the cost of node J' equal to MAX: i.e., COST(J')←MAX.
Go to step V.

O. Restore the eliminated arcs δ to the network.
Eliminate all references to the node-pair (x,y) in the set J^* and NP^*. That is: $J^* = J^* - \{(a,x,y) | a \epsilon A\}$ and $NP^* = NP^* - \{(x,y)\}$.
Assign the new path $\bar{\pi}(x,y)$ to the solution at node J'.
That is: $\{\bar{\pi}(x,y)\}_{J'} \leftarrow \bar{\pi}(x,y)$.
Assign the required flow r(x,y) to the path $\bar{\pi}(x,y)$.
That is: $\{f(\bar{\pi}(x,y))\}_{J'} \leftarrow r(x,y)$.

P. If $J^* = \phi$, go to step Q, for all flows have been reassigned for the new node J'. If not, then continue to step J. (When set J^* is empty, then a new solution node J' has been created in the solution tree.)

Q. Create a history set H(J') for node J' which contains the accumulation of arc eliminations. That is:

$$H(J') \leftarrow H(K') \cup J.$$

Assign the previous least cost network path sets $\{\bar{\pi}(i,j)\}_{K'}$, and simultaneous flow assignments $\{f(\bar{\pi}(i,j))\}_{K'}$, to solution node J' for

all node pairs $(i,j)\epsilon NP^*$.

Find the flow $f(\alpha)$ for the solution at node J'; i.e.,

$$f(\alpha) = \sum_{(i,j)\epsilon NP} f_\alpha(i,j) \quad \text{for all } \alpha\epsilon A.$$

Calculate the cost of the solution for node J':

$$\underline{\text{COST}}\ (J') = \sum_{\alpha\epsilon A} f(\alpha) \cdot c(\alpha)$$

R. If the solution at node J' is feasible (i.e., $f(\alpha)\leq b(\alpha)$ for all $\alpha\epsilon A$), then label the node "feasible" and go to step T. If not, label it "infeasible" and go to Step S.

S. Place the solution node J' into the set of infeasible nodes IS.
That is: IS = IS$\cup$J'.
Go to step V.

T. If the cost of solution node J' is minimum (i.e., COST (J')$<$<u>COST</u>), then go to step U. If not, go to step W.

U. Set the minimum cost, <u>COST</u>, equal to the cost of solution node J'; i.e., <u>COST</u>$\leftarrow$COST(J').
Store the solution paths $\{\bar{\pi}(i,j)\}_{J'}$ into $\{\bar{\pi}(i,j)\}_{OPT}$ as the optimum feasible solution.

V. Eliminate any node from the set IS having a cost greater than the optimum cost or greater than the maximum cost. That is:

$$IS = IS - \{K \mid K \epsilon IS \text{ and } COST(K) > \underline{COST} \text{ or } COST(K) > \overline{COST}\}.$$

Update the unevaluated node set: $OR(K')^{*} \leftarrow OR(K')^{*} - J$.

W. If all unevaluated nodes in the solution tree have been evaluated (i.e., $OR(K')^{*} = \phi$), then continue to step X. If not, go to step I.

X. If the flow in all infeasible nodes has been reassigned (i.e., IS = ϕ), then continue to step Y. If not, go to step B.

Y. If the optimum solution has been found (i.e., COST<MAX), then continue to step Z. If not, the problem has no solution and the algorithm terminates.

Z. The Phase II algorithm has found a solution which satisfies all the network conditions. Output the following:

1. The network path set $\{\bar{\pi}(i,j), \forall (i,j) \epsilon NP\}_{OPT}$.
2. The arc flow $f(\alpha)$ for all $\alpha \epsilon A$ which is computed from the network flows $\{f(\bar{\pi}(i,j)), \forall (i,j) \epsilon NP\}_{OPT}$.
3. The cost of the optimum solution, COST.

Terminate the Phase II algorithm.

Proof of the Phase II Algorithm

Lemma 3.3

For any two sets J_1, $J_2 \epsilon OR(K')$ where J_1 and J_2 are selected such that $J_1 \subset J_2$, the cost of the solution nodes J_1' and J_2' obtained, respectively, from sets J_1 and J_2 satisfy the following inequality:

$$COST\ (J_1') \leq COST\ (J_2').$$

Proof. At step J in the Phase II algorithm, the arc set β is obtained for each node-pair in J^*. Since $J_1 \subset J_2$, then either (1) $\beta_1 \subset \beta_2$ for some node-pair (x,y) or (2) an additional set β_2' will be created for a node-pair (x,y) occurring in J_2, but not in J_1.

In the first case at least one additional arc will be eliminated from the network in calculating a least cost path at step L in the algorithm for solution node J_2'. In the second case a new least cost path will be found for at least one additional node-pair in finding the solution for node J_2'. Therefore, in either case, at least one additional arc α' will be eliminated from consideration for some node-pair (x,y) in finding the solution for node J_2'.

Consider the paths between the node-pair (x,y) obtained above. Now the least cost path $c\left(\bar{\pi}(x,y)\right)$ for node-pair (x,y) in solution J_2' must be greater than or equal to $c\left(\bar{\pi}(x,y)\right)$ for solution J_1'. This is because an arc elimination restricts the selectivity of arcs which might be used on a path between (x,y). Since a new path must be found between the node pair, the cost of the path can either increase or stay the same. Since the total cost of any solution node J' can be calculated as:

$$\text{COST}(J') = \sum_{(i,j)\in NP} f\left(\bar{\pi}(i,j)\right) \cdot c\left(\bar{\pi}(i,j)\right)$$

for the network assignment $\{f\left(\bar{\pi}(i,j)\right)\}_{J'}$,

and the flow $f\left(\bar{\pi}(i,j)\right)$ between all node-pairs in solutions J_1' and J_2' is

the same (i.e., $f\{\bar{\pi}(i,j)\} = r(i,j)$), it follows that:

$$COST(J_1') \leq COST(J_2').$$

This proves the lemma. □

Lemma 3.4

The cost associated with a solution node J' in solution tree is equal to or greater than (i.e., monotonically nondecreasing) the cost of its immediate predecessor solution node K' (i.e., $COST(K') \leq COST(J')$ if solution node J' results from the branching at node K').

Proof. The solution which the Phase II algorithm finds for node J' is based primarily on the solution found for its immediate predecessor node K'. In step Q in the algorithm all least cost path sets not calculated in steps J through P for solution J' are assigned to it from the solution at node K'.

Now at least one new path must be calculated for node J' which was not in the solution at node K'. This can be seen by considering the history sets H(K') and H(J') for solution nodes K' and J', respectively. Now $H(K') \subset H(J')$. This is because the set H(J') contains additional elements (arcs and/or node-pairs) which were not in H(K'). If this is not the case, a contradiction occurs. The elements of the set J' were specifically chosen, because they were already being used by a path in the K' solution. If they were used in $\{\bar{\pi}(i,j)\}_{K'}$, then they could not be eliminated at an earlier solution node. Therefore,

$$H(K') \subset [H(K') \cup J] = H(J').$$

By Lemma 3.3, $COST(K') \leq COST(J')$ and the lemma is proved. □

Lemma 3.5

Consider an infeasible solution node $K' \in IS$ in the solution tree. The Phase II algorithm generates all possible solution nodes from node K' which:

1. Reduce the overflow in all the overflow arcs of solution K' (i.e., the algorithm makes the flow feasible in all infeasible arcs of solution K').

2. Produce a minimum increase in cost from the node K' solution.

Proof. Step E in the Phase II algorithm creates an arc elimination set AE(a'). Each $S \in AE(a')$ is a set of node-pairs (i,j) such that if arc a' is not used on the path $\bar{\pi}(x,y)$ for all $(i,j) \in S$, then the remaining flow on arc a' is feasible.

Each set $S \in AE(a')$ produces a solution J' having a minimum increase in cost over solution K'. To prove this, first consider (1) the sets $S_1, S_2 \in AE(a')$ such that $S_1 \subset S_2$. Let solution J_1' be generated from set S_1 and solution J_2' from set S_2. By Lemma 3.3, $COST(J_1') \leq COST(J_2')$. Therefore, the solution having a minimum increase in cost is generated from set S_1. Next consider (2) the sets $S_0, S_1 \in AE(a')$ such that $S_0 \subset S_1$. Now this case is not possible if S_1 is generated as in step E, part 2, of the Phase II algorithm. The smaller set S_0 would lead to a solution which still had an overflow in arc a'. Therefore, each element $S_1 \in AE(a')$ as obtained by the Phase II algorithm produces a feasible solution having a minimum increase in cost over solution K'.

Since each set $S_1 \epsilon AE(a')$ for each overflow arc $a' \epsilon OA(K')$ leads to a feasible solution at a minimum increase in cost, the collection of sets $J = \{S_1, S_2, \ldots, S_q\}$ also leads to feasible solutions having a minimum increase in cost. This is obvious because each element in OR(K') is created by using one set from AE(a') for each overflow arc a' at solution K'. Therefore, when the elements $J \epsilon OR(K')$ are used to generate new solutions, all overflow is eliminated in the infeasible arcs at a minimum increase in cost.

This proves the lemma. □

Lemma 3.6

The Phase II algorithm terminates in a finite number of steps.

Proof. Consider a strongly connected network (SCN) having a finite number of nodes and arcs. The Phase II algorithm creates a solution tree (ST) in order to find the optimum solution. Since the algorithm is based solely on creating and testing the solution nodes in ST, a proof that ST has a finite number of nodes is sufficient to prove the lemma.

Each node in ST is created in order to attempt to reallocate flow away from overflow arcs in an infeasible solution in such a way that a least cost feasible solution is found if one exists. Steps Q and J in the algorithm insure that once an arc has been eliminated from being used on a path between a node-pair for solution node K', it will continue to be eliminated for the same node-pair for all future solution nodes in ST emanating from node K'.

Now at least one new element is introduced into the history set H(J') which was not in H(K') where solution node J' is created by a branch from solution node K'. This was proved in Lemma 3.4. Also, when H(M') for some solution node M' contains all possible combinations of arcs and node-pairs as elements, the path algorithm will be unable to construct a path between any node-pair. This is because all arcs have been eliminated from the network. Therefore, the algorithm will not be able to create any new nodes in ST.

Collecting the facts already established:

—At least one new element is added to H(K') for each solution node K in ST.

—An element added to H(K') is also included in H(J') for any solution node J' in the solution tree having node K' on the path between node J' and the root node.

—When H(M') contains all combinations of arcs and node-pairs, then no additional nodes can be created from node M'.

Therefore, the number of solution nodes along any path from the root node is finite.

The number of paths from the root node in ST to any terminal node is also finite. This is true because only a finite number of new nodes is created for every infeasible node K' in ST. The finiteness of branching stems from the following facts:

—Only a finite number of arcs $a' \epsilon A$ can have overflow; i.e., OA(K') is always a finite set (step B in the algorithm).

—Only a finite number of ways exist to reassign the overflow in an arc a'; i.e., AE(a') is always finite (step E in the algorithm).

Since the solution tree has a finite number of new solution nodes emanating from each solution node and a finite number of paths exists in

ST from the root node to all terminal nodes, the solution tree has a finite number of nodes and the lemma is proved. □

Theorem 3.2

The Phase II algorithm finds a solution (if one exists) to the network problem which satisfies network conditions 1, 2, 3 and 4 in a finite number of steps.

Proof. The theorem is proved by considering each of the above conditions in turn. Lemmas 3.5 and 3.6 guarantee that the Phase II algorithm will generate feasible solutions (if any exist) having minimum increase in cost over the previous solution node in a finite number of steps. And network condition 2 is verified by Theorem 2.1 because all paths found in Phase I or Phase II of the algorithm are found by the path algorithm.

Network condition 3 is satisfied by the Phase II algorithm in a manner similar to the way it was satisfied in the Phase I algorithm. In step A of Phase II all arcs $a \in A$ which have a capacity less than or equal to the requirements, i.e., $b(a) < r(x,y)$ for each node-pair (x,y) in SCN, are placed in $H(R')$ for the root node R'. These arcs, for the particular node-pair (x,y), are eliminated from SCN at step K prior to using the path algorithm to find a least cost path $\bar{\pi}(x,y)$ between (x,y). Therefore, no path $\bar{\pi}(x,y)$ found by the path algorithm can have a capacity less than $r(x,y)$. Since $f\left(\bar{\pi}(x,y)\right)$ is set equal to $r(x,y)$ at step L and all flows for the root node satisfy condition 3 (Theorem 3.1), then network condition 3 is satisfied for all solutions found in Phase II.

Network condition 4 is guaranteed in Phase II by the decision at step R. In order for a solution to be labeled feasible, it must meet the test given at this step. That is: $f(\alpha) \leq b(\alpha)$ for all $\alpha \epsilon A$.

The proof that the Phase II algorithm satisfies network condition 1 must now be established. Theorem 3.1 guarantees that the Phase II algorithm starts with the root node having a least cost solution. In Lemma 3.4 it is established that the cost of each solution node in ST on any path from the root node is monotonically nondecreasing. In Lemma 3.5 it was proved that:

—All permissible least cost ways are found to reduce all the overflow for an infeasible solution node.

—Each newly created solution node J' represents a least cost increase in the solution cost over the previous solution node K'.

Therefore, the cost of any feasible solution node found in ST represents the least cost way of arriving at that solution along a unique path in ST from the root node. This means that a feasible solution having the minimum increase in cost over the root node cost is an optimum solution to the network problem. Steps T and U check each feasible solution node created in ST and retain one having the smallest cost as the optimum feasible solution.

Since each of the conditions stated in the theorem is satisfied by the Phase II algorithm, we have proven that the network algorithm solves the simultaneous flow network problem. □

Computational Considerations

To this point in the chapter the solution algorithm has been described and its validity proved. In this section some topics related to improving the efficiency of our branch and bound algorithm are discussed. Exact computational estimates of a branch and bound algorithm cannot be made because the creation of the solution tree completely depends on the specific problem being solved. Since programming and evaluating the algorithm were not part of the research, we can only advance general comments about the algorithm based on hand calculations and our knowledge of programming techniques. It has been noted by Agin in a study of branch and bound algorithms [E01] that, as computational experience is gained with a particular algorithm, simplifications are usually found which greatly reduce computation time. We have already found this to be true in our work with the solution algorithm and expect that its implementation on a computer will result in further steps toward computational efficiency.

There is an increasing use of branch and bound algorithms in the literature. In many cases it is the only known solution method for certain types of problems. It should be judged primarily on this basis, according to Agin, rather than on its computational efficiencies. It is interesting to note, however, that a branch and bound solution to the integer programming problem requires significantly less computer time than the conventional analytical method [E23]. This suggests that with adequate computational experience, branch and bound approaches may actually be the best way to obtain global optimum solutions to certain types of problems.

A list of computational considerations were compiled as the solution algorithm was developed. We now briefly discuss some of these considerations:

1. Algorithm Heuristics

The algorithm is set up to branch from a node in the solution having the least number of overflow arcs. This heuristic tends to reduce the number of new nodes created while effectively searching for a feasible solution. In the problems solved to date, this appears to be a reasonably good heuristic. On the other hand, using the node having the least number of elements in its history set is an appealing alternative method for the selection of nodes for branching.

Branch and bound algorithms tend to require a lot of storage space in a computer. In order to reduce the storage requirements of the algorithm, it is usually advantageous to find a feasible solution as fast as possible. Our branching heuristic accomplishes this by advancing down a branch (i.e., paths) in the tree, and selecting at each step the node having the least number of overflow arcs. This tends to quickly produce a feasible solution which can be used as a bound in eliminating non-optimum paths in the solution tree.

Two other types of branching strategies are possible for the solution algorithm:

1. Branch from lowest bound.
2. Branch from newest active bounding problem.

The first has the advantage that the total amount of computation is minimized while storage requirements may become large. The second

requires more time but uses a minimum amount of storage. The selection of a branching heuristic, in the end, must be based on empirical results.

2. Linked Lists

To simplify the description of the algorithm, several new sets were introduced for each node created in the solution tree. Each new set was created by adding elements to the related sets defined at the previous node. Two examples of these sets are (1) the history set for the node and (2) the new paths (i.e., set of arcs) found by the path algorithm for the node. It is possible to define a single linked list for each of these sets for the entire problem. Whenever a new node is created, a record of changes (i.e., additions) at that node is stored with a link back to the record associated with the previous (i.e., parent) node in the tree. This forms a chained record which points from any solution node back to the root node. This record contains the complete contents of the particular set for the solution node in question.

3. Short Cuts

The computation time of the solution algorithm can be significantly reduced by eliminating nodes in the solution tree which have a low probability of producing feasible solutions. Probably the easiest method of doing this is to restrict the number of branches created by a node to some reasonable number. (Of course, the nodes having the least number of overflow arcs would be selected first.) When paths are indiscriminately eliminated in the solution tree, a global solution cannot

be guaranteed, but a satisfactory feasible solution may be obtained. The framework of the solution tree is amenable to numerous shortcuts by pruning branches.

4. Equal Cost Paths

The path algorithm always selects the first least cost path which it finds between a node-pair. Other least cost paths of the same length may exist but are disregarded until a need for them arises in the dynamics of a particular problem solution. Then they are recomputed. It may prove advantageous to identify all least cost paths, carry them through the steps in the algorithm and check them first when an overflow occurs. This may result in an overall reduction in computational time, but it is hard to confirm this without extensive empirical evidence.

5. Optimum Root Node Selection

It is imperative to establish a good solution at the root node of a branch and bound solution tree. The entire method depends on the closeness of the root solution to the final optimum solution. The solution method in this dissertation was specifically designed around this consideration. In fact, the branch and bound algorithm (Phase II) is not even required if overflow does not occur in the minimum cost solution in Phase I.

CHAPTER IV

NECESSARY CONDITIONS AND COST BOUNDS

Introduction

The solution algorithm discussed in the previous chapter is designed to find an acceptable solution to the simultaneous flow network problem if such a solution exists. In order to preclude futile attempts to find a nonexistent solution, it is advantageous to establish a set of necessary conditions in this chapter which can be applied directly to a network configuration. If any necessary condition is not satisfied, then the solution algorithm should not be used, for an acceptable solution does not exist.

Also included in this chapter are two theorems which establish upper and lower cost bounds for the solution of any network and a discussion of bounds for path length. At the end of the chapter an algorithm is described which can be used to assist in calculating some of the conditions and bounds discussed in the chapter.

As stated in Chapter III, the type of network which is considered in this dissertation is described by the following network parameters:

—Arc costs $c(\alpha)$, $\alpha \epsilon A$.

—Arc capacities $b(\alpha)$, $\alpha \epsilon A$.

—Node-pair capacity requirements $r(i,j)$,$(i,j) \epsilon NP$.

—Maximum path length L.

Any network which has values assigned to the above parameters is called a *network definition* or a *network configuration.*

If all the values of a network configuration are known for a particular communication network, then the method of Chapter III can be applied to these values to solve the simultaneous flow problem. An acceptable solution to the problem is expressed in terms of a network path set, $\{\bar{\pi}(i,j), \forall (i,j) \epsilon NP\}$, which satisfies the following network conditions (see Chapter III for a detailed explanation):

1. $\underline{COST} = \text{Min}\{\sum_{\alpha \epsilon A} (\alpha) \cdot c(\alpha)\} = \text{Min}\{\sum_{(i,j) \epsilon NP} f(\bar{\pi}(i,j)) \cdot c(\bar{\pi}(i,j))\}.$

2. $\ell(\bar{\pi}(i,j)) \leq L.$

3. $f(\bar{\pi}(i,j)) = r(i,j).$

4. $\sum_{(i,j) \epsilon NP} f_{\alpha}(\bar{\pi}(i,j)) = f(\alpha) \leq b(\alpha).$

It is always desirable to develop necessary conditions which require a minimum expenditure of computation effort and which are powerful enough to detect network configurations that do not have acceptable solutions. Unfortunately, strong necessary conditions which are elegantly simple to apply are generally difficult to prove. In this chapter we state and prove three necessary conditions which cover a range from a gross test to a fine test of the defining parameters of a network. As the test conditions become more refined, the required computations to test the condition become more involved.

The research directed toward producing effective necessary *and* sufficient conditions produced fragmentary results and the results obtained were not significant enough to include herein. It is concluded

that due to the nature of the network problem, powerful sufficient conditions are extremely difficult to develop. The establishment of such conditions is a prime candidate for follow-on research.

The activity of establishing cost and path length bounds was approached from the vantage point of the earlier results and was not based on traditional greatest lower bound and least upper bound philosophies. The objective of the two sections in this chapter on bounds was to obtain reasonable ways of determining gross limits on the cost and length parameters of a communication network.

The only optimization criterion (i.e., network condition 1) for the network problem is the total cost of a network solution which satisfies all the network conditions. Since cost plays such a central role in our work, it was advisable to establish bounds on the overall network cost. These bounds can be found before the solution algorithm is actually applied to the problem. Two theorems are proved which can be used to find cost bounds for any network. These bounds can be found independently from the necessary conditions and do not necessarily guarantee the existence of a solution within the bounds.

Necessary Conditions

Three necessary conditions are now stated and proved which must be satisfied in order for a network description to have an acceptable solution. These conditions basically involve the relationship between the node-pair requirement matrix R (i.e., $r(i,j), \forall (i,j) \epsilon NP$) and the arc capacity matrix B (i.e., $b(i,j), \forall (i,j) \epsilon NP$). The weakest condition is presented first.

Theorem 4.1

The following conditions must hold for each row (condition (a)) and each column (condition (b)) of the node-pair requirement matrix R and the arc capacity matrix B for an acceptable solution to exist for the simultaneous flow problem:

(a) $\underset{j}{\text{Max}}\{r(x,j)\} \leq \underset{j}{\text{Max}}\{b[x,j]'\}$ for x fixed and $j \epsilon N$, $j \neq x$.

(b) $\underset{i}{\text{Max}}\{r(i,y)\} \leq \underset{i}{\text{Max}}\{b[i,y]'\}$ for y fixed and $i \epsilon N$, $i \neq y$.

Proof. Part (a) is proved first. Consider any node-pair (x,y) having a flow requirement r(x,y). For an acceptable solution to exist, it is necessary for a path $\pi^*(x,y)$ to exist between the two nodes such that

$$r(x,y) \leq b\left(\pi^*(x,y)\right). \tag{1}$$

This is required because a single path must carry all the flow between each node-pair. From the definition of path capacity we have

$$b\left(\pi^*(x,y)\right) = \underset{\alpha \epsilon \pi^*(x,y)}{\text{Min}} \{b(\alpha)\}. \tag{2}$$

Now the maximum required flow leaving node x and destined for any other node y must use one of the arcs $[x,j]'$ for $j \epsilon N$, $j \neq x$. The maximum flow requirement $\underset{j}{\text{Max}}\{r(x,j)\}$ cannot exceed the largest capacity of any arc leaving node x. Thus, $\underset{j}{\text{Max}}\{b[x,j]'\}$ can be used as an upper bound on the capacity as follows:

$$b\left(\pi^{*}(x,y)\right) = \min_{\alpha\in\pi^{*}(x,y)} \{b(\alpha)\} \leq \max_{j}\{b[x,j]'\}. \qquad (3)$$

Combining Equations (1) and (3), we obtain:

$$\max_{j}\{r(x,j)\} \leq \max_{j}\{b[x,j]'\} \qquad (4)$$

which proves part (a) of the theorem. The proof of part (b) follows a similar argument. □

Theorem 4.2

The following conditions must hold between the respective row sums (condition (a)) and column sums (condition (b)) of the node-pair requirement matrix R and the arc capacity matrix B in order for an acceptable solution to exist for the simultaneous flow problem.

(a) $\sum_{j} r(x,j) \leq \sum_{j} b[x,j]'$ for x fixed and all $j\in N$, $j\neq x$.

(b) $\sum_{i} r(i,y) \leq \sum_{i} b[i,y]'$ for x fixed and all $i\in N$, $i\neq y$.

Proof. We prove part (a) by first making the following preliminary calculations:

—The maximum flow between node x and all other nodes in the network can be found by adding the capacities of all arcs leaving node x, i.e.,

$$\sum_{j} b[x,j]' \quad \text{for all } j\in N,\ j\neq x.$$

—The total flow requirement between node x and all the other nodes in the network can be found by summing the flow requirements leaving node x; i.e.,

$$\sum_j r(x,j) \quad \text{for all } j\epsilon N,\ j\neq x.$$

Now in order to meet the total flow requirements to the other nodes in the network, the flow out of node x must first use the arcs $[x,j]'$ for all $j\epsilon N$, $j\neq x$. That is:

$$\sum_j r(x,j) \leq \sum_j f[x,j]'. \tag{1}$$

By network condition 4 we know that $f(\alpha)\leq b(\alpha)$ for all $\alpha\epsilon A$. Since this condition is required of every arc, it is required for those arcs leaving node x. Therefore:

$$\sum_j f[x,j]' \leq \sum_j b[x,j]'. \tag{2}$$

By combining (1) and (2) we get

$$\sum_j r(x,j) \leq \sum_j b[x,j]'$$

which proves part (a) of the theorem. The proof of part (b) follows in a similar fashion. □

Theorem 4.3

The following inequality is a necessary condition which must hold for each node-pair (x,y) in a network for an acceptable solution to exist for the simultaneous flow problem:

$$r(x,y) \leq \max_{\pi' \in \pi(x,y)} \{b[\pi'(x,y)]\}. \quad \square$$

Proof. From network condition 3, the flow along a path π^* between node-pair (x,y) must be equal to the requirements for that node-pair. That is:

$$r(x,y) = f[\pi^*(x,y)]. \tag{1}$$

Now the flow along any particular path $\pi^*(x,y)$ can never exceed the capacity of the path, i.e.,

$$f[\pi^*(x,y)] \leq b[\pi^*(x,y)]. \tag{2}$$

Now let $b^*[\pi(x,y)]$ be the largest capacity of any path $\pi'(x,y)$ which exists between (x,y). Since $b^*[\pi(x,y)]$ represents the upper bound on the capacity of any path between (x,y),

$$b[\pi^*(x,y)] \leq b^*[\pi(x,y)] = \max_{\pi' \in \pi(x,y)} \{b[\pi'(x,y)]\}. \tag{3}$$

By combining Equations (1), (2), and (3), we have the following desired result which proves the theorem.

$$r(x,y) \leq \max_{\pi' \in \pi(x,y)} \{b[\pi'(x,y)]\}. \quad \square$$

In order to apply the necessary condition given in Theorem 4.3, it is necessary to devise a means of finding the maximum capacity of any path between (x,y) (i.e., $\text{Max}\{b[\pi(x,y)]\}$). Fortunately, a simple algorithm can be created which finds this value for each node-pair in the network. We call the algorithm the path optimization algorithm (POA) and describe it later in this chapter. It is a simplified variation of the path algorithm and can be used to find various kinds of optimum paths in a network.

Starting with the B (maximum arc capacity) matrix of the network, we use the POA to find the $\bar{B}$ (maximum path capacity) matrix. Because the POA uses the same path finding philosophy as the path algorithm, the path length for any node-pair can be restricted as required in network condition 2. Thus, the POA is used to find the matrix $\bar{B}$ while guaranteeing that the length of all paths which are considered is less than or equal to L.

After $\bar{B}$ is found, every element $\bar{b}(x,y)$ must be compared with the corresponding element r(x,y) in R according to the statement of Theorem 4.3. If $r(x,y) \leq \bar{b}(x,y)$ for all node-pairs (x,y), then the necessary condition as stated in the theorem is satisfied. If $r(x,y) > \bar{b}(x,y)$ for some (x,y), then an acceptable solution does not exist for the particular network description being examined.

Cost Bounds

Network condition 1 is the criterion which is used by the network algorithm in finding an optimum solution to the network problem. In this section this condition is used along with other network conditions

to establish an upper and a lower cost bound for the total cost of transmitting the required flow over the network.

Theorem 4.4

An upper bound on the total cost (i.e., $\overline{COST}$) of an acceptable network solution is $\overline{COST} = \sum_{\alpha \in A} b(\alpha) \cdot c(\alpha)$.

Proof. According to network condition 1, the following formula can be used to calculate network cost:

$$\sum_{\alpha \in A} f(\alpha) \cdot c(\alpha). \tag{1}$$

Now the maximum value of this expression must give the maximum network cost; i.e.,

$$\overline{COST} = \text{Max}\{\sum_{\alpha \in A} f(\alpha) \cdot c(\alpha)\}. \tag{2}$$

Since the unit cost $c(\alpha)$ of each arc is a constant and the maximum of a sum is the sum of the maximums, then (2) becomes

$$\text{Max}\{\sum_{\alpha \in A} f(\alpha) \cdot c(\alpha)\} = \sum_{\alpha \in A} \text{Max}\{f(\alpha)\} \cdot c(\alpha). \tag{3}$$

By network condition 4, $b(\alpha)$ is a maximum bound for $f(\alpha)$ since,

$$f(\alpha) \leq b(\alpha) \quad \text{for all } \alpha \in A. \tag{4}$$

Combining Equations (2), (3), and (4), the desired result is obtained:

$$\overline{\text{COST}} = \sum_{\alpha \in A} b(\alpha) \cdot c(\alpha). \quad \square$$

Theorem 4.5

A lower bound on the total cost (i.e., COST) of any network is:

$$\underline{\text{COST}} = \sum_{(i,j) \in NP} r(i,j) \cdot \underline{c}(i,j)$$

where $\underline{c}(i,j)$ is the cost of the least cost path between node-pair (i,j).

Proof. To prove the theorem a network path set $\{\pi^*(i,j), \forall (i,j) \in NP\}$, whose total cost after being assigned the required flow yields the minimum value over all the other possible network path sets, must be found. That is:

$$\underline{\text{COST}} = \min_{\pi(i,j)} \left\{ \sum_{(i,j) \in NP} \left(\pi(i,j)\right) \cdot c\left(\pi(i,j)\right) \right\} =$$

$$\sum_{(i,j) \in NP} f\left(\pi^*(i,j)\right) \cdot c\left(\pi^*(i,j)\right). \qquad (1)$$

By network condition 3:

$$f\left(\pi^*(i,j)\right) = r(i,j) \quad \text{for all } (i,j) \in NP. \qquad (2)$$

Therefore, the flow assignments remain the same regardless of the network path set being considered. Substituting Equation (2) into Equation (1) and using the fact that the minimum of a sum is the sum of the minimums:

$$\min_{\pi(i,j)} \Big\{ \sum_{(i,j)\in NP} f\big(\pi(i,j)\big) \cdot c\big(\pi(i,j)\big) \Big\} =$$

$$\sum_{(i,j)\in NP} r(i,j) \cdot \min_{\pi(i,j)} \big\{ c\big(\pi(i,j)\big) \big\}. \qquad (3)$$

This means that the network path set which is required is the one which produces the least cost paths $\underline{c}(i,j)$ between each node-pair (i,j). Combining Equations (1), (2), and (3) we have the following desired result which proves the theorem:

$$\underline{\text{COST}} = \sum_{(i,j)\in NP} r(i,j) \cdot \underline{c}(i,j). \quad \square$$

In order to apply Theorem 4.5, a means of finding the minimum cost path for each node-pair (x,y) (i.e., the matrix $\underline{C}$) must be found. The POA is used for this purpose, thereby including the path length constraint in the process of obtaining the value of the least cost paths in the network.

After $\underline{C}$ is found, the results of Theorem 4.5 are used to calculate a lower cost bound for the network. The element by element multiplication of R and $\underline{C}$ must be performed and the products summed to arrive at the overall total. In the next section the use of the POA is explained in detail. However, we note here that if the total cost, $\underline{\text{COST}}$, is less than zero, no solution exists. A negative total cost means that a path of length $k(1\leq k\leq L)$ does not exist between some node-pair, and the algorithm has assigned a large negative cost to that node-pair.

Bounds on Path Length

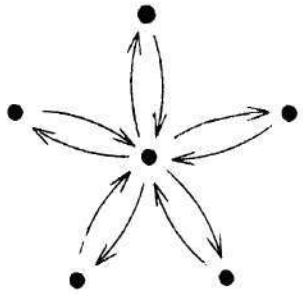
Establishing upper and lower bounds on the length of the least weighted paths in a network is an interesting problem which is indirectly related to our work. Since the path algorithm is set up to terminate calculations at a particular pre-set length and can be made to terminate at any iteration in which no new paths are found, there is no requirement to establish analytical bounds for path length. Finding bounds on path lengths in our networks is compounded by the fact that a least weighted path between two nodes may not be the path having the shortest length. In this section we discuss some basic considerations which relate to bounds on the length of least weighted paths in strongly connected networks.

Bratton [F06] was probably the first person to give serious attention to determining bounds on length, given only the number of nodes (n) and the number of arcs (m) in a network. He considered networks in which each arc has the same weight as all other arcs in the network. Even with a detailed investigation into the problem, his only general result is a conjecture which still remains unproven. This conjecture can be stated as:

$$\text{Min}\{L\} = \frac{2(n-1)}{m - n + 1} - 2.$$

This lower bound on path length is based on the well-known facts that (1) the number of topological cycles in an undirected network is equal to m-n+1 and (2) a strongly connected n-node network always exists

which has a minimum path length of 2 and is composed of 2(n-1) arcs. Networks which exemplify this statement all have the same star form as illustrated by the following example:

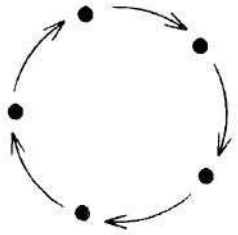


$n = 6$

$m = 10 = 2(n-1)$

$Min\{L\} = 2$

The lower bound on path length can be equal to n-1. This occurs when an n-node network having n arcs is strongly connected. The only way to obtain this type of network is to construct a Hamiltonian cycle through all n nodes in the network. For example:



$n = 5$

$m = 5$

$Min\{L\} = 4 = n-1$

The problem of establishing a minimum bound on path length, therefore, immediately reduces to the investigation of changes in path length which occur as the number of arcs in an n-node network varies from n to 2(n-1). Our preliminary research strongly suggests that a serious investigation into this problem will have to incorporate other structural aspects of the network (such as the degrees of nodes) in determining analytical bounds on path length.

An upper bound on path length is probably more difficult to establish than a lower-bound. Since only simple paths are being considered, no path length can be greater than n-1. A natural question arises: How can this trivial upper bound be reduced by knowing other types of structural information in a particular network? Our computer simulation results discussed in Chapter II unexpectedly shed some light on an answer to this question. In the computer runs on over 200 networks of various sizes, the maximum least weighted path lengths were always less than or equal to n/2 and were many times as low as n/4. These maximum length values would undoubtedly vary with the number of arcs in the networks and the range of arc weights which are permitted; however, they do suggest that long paths are quite rare and that an upper bound for path length of approximately n/2 may exist for most networks.

The Path Optimization Algorithm (POA)

In order to take full advantage of the theorems in this chapter, it was necessary to devise an effective way of computing the $\bar{B}$ and $\underline{C}$ matrices. We utilize the path construction philosophy of the path algorithm of Chapter II and appropriately modify it to create a general path-oriented optimization algorithm. Because the algorithm finds paths one arc at a time, it can also be used to satisfy network condition 2 by constraining the length of all paths. The POA, then, is the primary means by which this length constant is incorporated into the results of this chapter.

The flow chart of the POA is given in Figure 4.1. Note the similarity of this flow chart with the one in Figure 2.1. Since the arc sets are not needed for each path, the steps to find them have been dropped from the algorithm. However, the length constraint feature has been retained. Refer to Chapter II for a description of the individual steps in the algorithm.

The POA can be used to optimize any path weight calculation for which the following conditions hold:

—The path weights are calculated from the weights of their component arc weights.

—The path weight of a path of length p+1 is calculated from the path weights of length p and those of length one.

—Cycles do not interfere with the weight calculation.

Now the path algorithm in Chapter II produced a minimum cost path between each node-pair in which:

—The path costs were calculated as the *sum* of the component arc costs.

—The optimum cost path was selected as the path of length $k(1 \leq k \leq L)$ which had the minimum cost for each node-pair.

—Negative cycles were excluded from consideration.

Figure 4.2 gives the summary information about two uses of the POA. Column 1 of the figure represents the essential features of the POA which finds the minimum cost path for each node-pair in a network. Theorem 2.1 gives the proof that this algorithm accomplishes this task.

By modifying the POA at steps A, B, and C, it can also be used to find the value of the maximum capacity path between each node-pair. In column 2 of Figure 4.2, the modifications are shown which are necessary

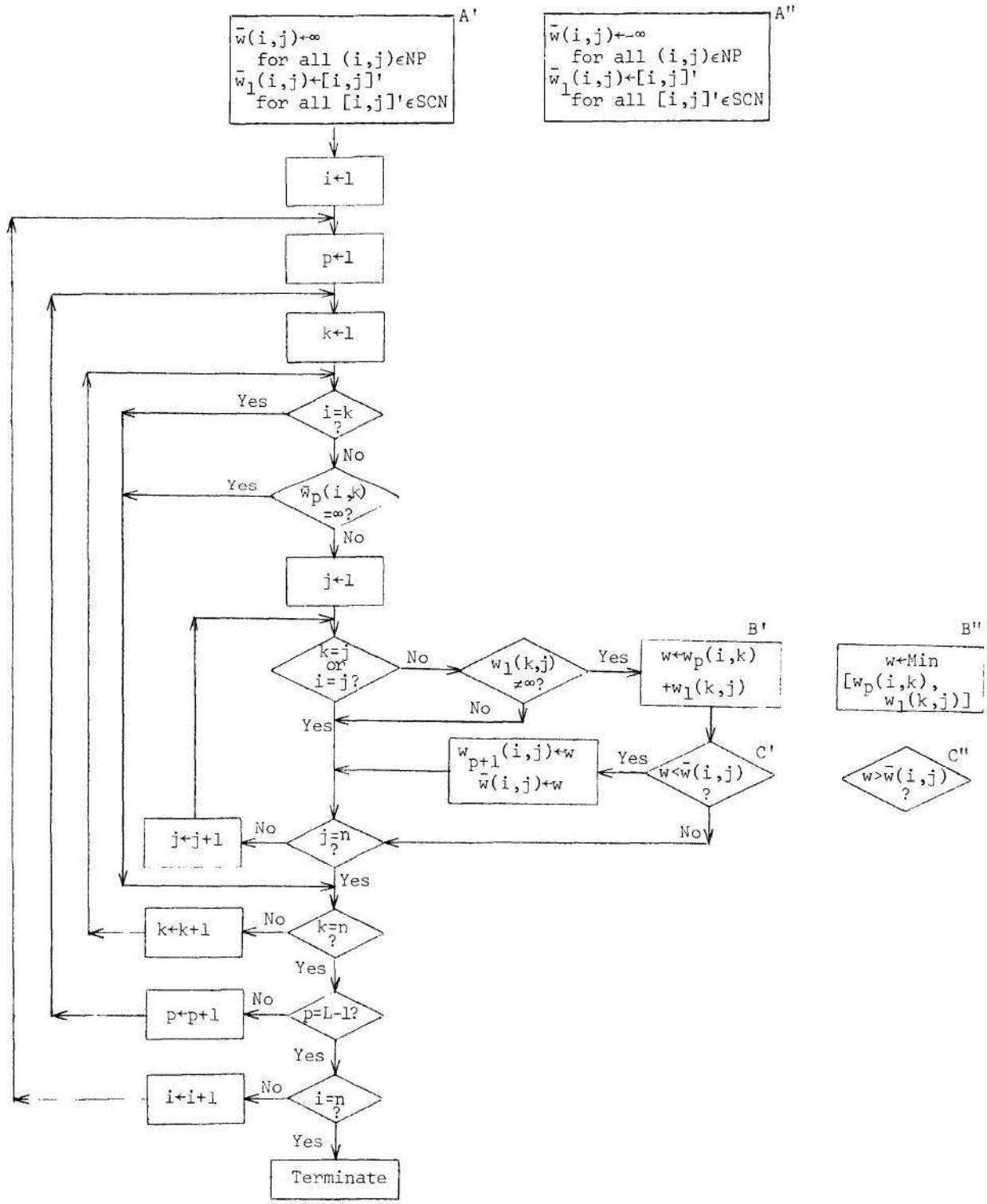


Figure 4.1. Path Optimization Algorithm (POA)

Type of Problem	Minimum Cost	Maximum Capacity
Initial Weight $w_1(i,j)$	Arc Cost $c[i,j]'$	Arc Capacity $b[i,j]'$
Initialization $\bar{w}(i,j)$ (Step A in Algorithm)	∞	$-\infty$
Path Weight Calculation (Step B)	$w \leftarrow w_p(i,k) + w_1(k,j)$	$w \leftarrow \text{Min}[w_p(i,k), w_1(k,j)]$
Optimum Path Weight Selection (Step C)	$\text{Min}[w, \bar{w}(i,j)]$	$\text{Max}[w, \bar{w}(i,j)]$
Optimum Weight $\bar{w}(i,j)$	Minimum Cost $\underline{c}(i,j)$	Maximum Capacity $\bar{b}(i,j)$

Figure 4.2. Two Variations of the POA

to calculate the maximum capacity for the node-pairs. The proof of this version of the algorithm follows the same argument used in the proof of Theorem 2.1 and is, therefore, not repeated here. The essential features of the maximum capacity calculations are:

—The path capacities are calculated by finding the minimum value of the component arc capacities.

—The maximum capacity path is selected as the path of length $k(1 \leq k \leq L)$ which has the maximum value for its capacity.

—Cycles do not interfere with capacity calculations.

The minimum distance (i.e., length) calculation between nodes in a network can also be obtained from the POA. Numerous methods are employed in the literature from matrix multiplication [A12] to Boolean operations [D59]. By assigning the weight of one to each arc which is present in the network and using the minimum cost version of the algorithm, the minimum path length can be found for each node-pair.

CHAPTER V

EXAMPLE OF SOLUTION METHOD

Problem Description

The XYZ Corporation, a large decentralized company having five corporate divisions spread throughout North and Central America, has a communication network problem. Over its 43 years of existence different types of communication patterns have developed between the widely separated division headquarters. The Company now owns a network of private communication lines among the divisions' headquarter offices. These lines were purchased over a period of 15 years with no particular objective in mind. The top management of the Company is quite concerned over the fact that these existing communication lines are not being used efficiently to transmit intracompany information from division to division. They have unequivocally stated that no plans for a company-wide management information system will be approved until the present network is operated in an optimal fashion.

The following facts are known about the existing communications system:

—There are ten communication lines in the system over which standard-length messages are sent.

—The unit cost of sending a message depends on the particular communication lines used to transmit the message.

—The cost associated with sending messages over a single communication line is the product of the unit cost of the communications line and the number of messages transmitted over the line.

—The number of messages sent over a line per minute can be no greater than the capacity of the line.

—The transmission error rate increases sharply for those messages relayed by more than two stations.

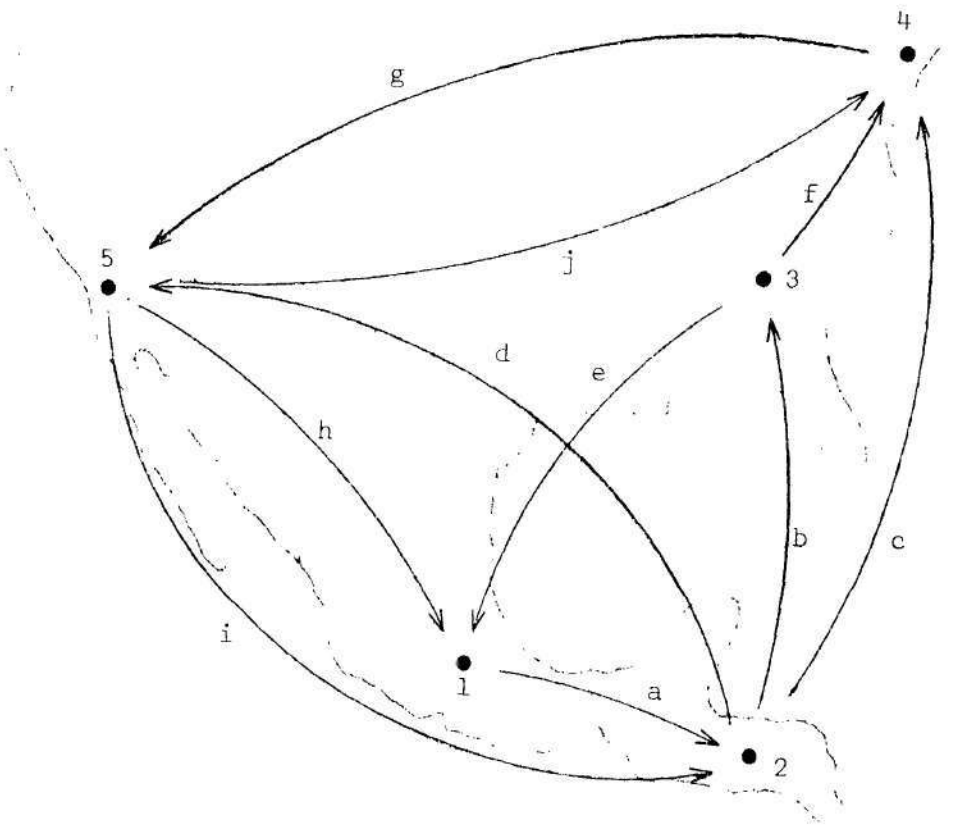
An analysis of the number of messages sent between all the divisions was recently made. It included all messages and not only the ones transmitted over the existing communication network. From this study the average number of messages required per day between each pair of divisions was determined. All the statistics concerning the configuration and use of the XYZ's corporate information network are presented in the network description on Figure 5.1.

The communication problem facing the XYZ Corporation can be simply stated as follows: What is the optimal routing of messages between divisions of the Company in order to operate the information system at minimal cost? A solution to this information network problem is now found by applying the algorithm described in Chapter III.

Before initiating Phase I the three necessary conditions from Chapter IV are checked and verified. The results are presented in Figure 5.2.

Phase I Solution

The solution to the network problem is obtained by first applying the Phase I algorithm which was given and explained in Chapter III. See Figure 3.1 for a summarized description of this algorithm. The steps discussed below refer to the corresponding steps used in the Phase I algorithm. Branching in the algorithm occasionally causes nonsequentially alphabetic labeling in the following description.



Interdivisional Message Requirements r(i,j)

→	1	2	3	4	5
1	-	3	2	1	3
2	6	-	3	2	3
3	2	1	-	7	4
4	4	3	2	-	3
5	2	1	2	5	-

Communication Line Capacities b(i,j)

→	1	2	3	4	5
1	-	20	0	0	0
2	0	-	10	5	10
3	10	0	-	20	0
4	0	0	0	-	20
5	20	20	0	10	-

Communication Line Unit Costs c(i,j)

→	1	2	3	4	5
1	-	2	-	-	-
2	-	-	4	3	6
3	2	-	-	1	-
4	-	-	-	-	4
5	3	6	-	3	-

Maximum Length of Any Communication Path is Three (i.e., L=3).

Figure 5.1. XYZ Corporation Communication Network

x	$\max_j[r(x,j)]$	$\max_j[b[x,j]']$
1	3	20
2	6	10
3	7	20
4	4	20
5	5	20

y	$\max_i[r(i,y)]$	$\max_i[b[i,y]']$
1	6	20
2	3	20
3	3	10
4	7	20
5	4	20

Theorem 4.1 Calculations

x	$\sum_j r(x,j)$	$\sum_j b(x,j)$
1	9	20
2	14	25
3	14	30
4	12	20
5	10	50

y	$\sum_i r(i,y)$	$\sum_i b(i,y)$
1	14	30
2	8	40
3	9	10
4	15	35
5	13	30

Theorem 4.2 Calculations

$$R = \begin{bmatrix} - & 3 & 2 & 1 & 3 \\ 6 & - & 3 & 2 & 3 \\ 2 & 1 & - & 7 & 4 \\ 4 & 3 & 2 & - & 3 \\ 2 & 1 & 2 & 5 & - \end{bmatrix} \qquad \bar{B} = \begin{bmatrix} - & 20 & 10 & 10 & 10 \\ 10 & - & 10 & 10 & 10 \\ 20 & 20 & - & 20 & 20 \\ 20 & 20 & 10 & - & 20 \\ 20 & 20 & 10 & 10 & - \end{bmatrix}$$

Theorem 4.3 Calculations

Figure 5.2. Necessary Condition Calculations

B. The path algorithm and auxiliary algorithm are used to produce the results listed in Figure 5.3. The path table contains only the least cost paths found at each iteration. Note that the path {g,h,a,b} between node-pair (4,3) having a minimum cost of 13 was not used because it exceeds the length requirement.

C. The last two columns on Figure 5.3 can be used to determine if any of the path capacities are less than their respective requirements. In this example $b(\bar{\pi}(i,j)) \geq r(i,j)$ for all $(i,j) \epsilon NP$; therefore the set T remains empty.

K. The flow on any particular arc is the sum of the flows on all paths using the arc. Figure 5.4 illustrates a systematic process of determining the arc flow. The results are summarized in the following table:

α	a	b	c	d	e	f	g	h	i	j
$c(\alpha)$	2	4	3	6	2	1	4	3	6	3
$b(\alpha)$	20	10	5	10	10	20	20	20	20	10
$f(\alpha)$	16	15	3	6	9	11	16	12	2	5
		↑								

L. A comparison of $f(\alpha)$ and $b(\alpha)$ for all $\alpha \epsilon A$ in the above table yields $f(b) > b(b)$. This infeasibility condition on arc b requires that the Phase II algorithm be initiated.

Node Pair	$\bar{\pi}_1$	$c(\bar{\pi}_1)$	$b(\bar{\pi}_1)$	$\bar{\pi}_2$	$c(\bar{\pi}_2)$	$b(\bar{\pi}_2)$	$\bar{\pi}_3$	$c(\bar{\pi}_3)$	$b(\bar{\pi}_3)$	$AS(\bar{\pi}(i,j))$	$c(\bar{\pi}(i,j))$	$b(\bar{\pi}(i,j))$	$r(i,j)$
(1,2)	a	2	20							{a}	2	20	3
(1,3)				ab	6	10				{a,b}	6	10	2
(1,4)				ac	5	5				{a,c}	5	5	1
(1,5)				ad	8	10				{a,d}	8	10	3
(2,1)				be	6	10				{b,e}	6	10	6
(2,3)	b	4	10							{b}	4	10	3
(2,4)	c	3	5							{c}	3	5	2
(2,5)	d	6	10							{d}	6	10	3
(3,1)	e	2	10							{e}	2	10	2
(3,2)				ea	4	10				{e,a}	4	10	1
(3,4)	f	1	20							{f}	1	20	7
(3,5)				fg	5	20				{f,g}	5	20	4
(4,1)				gh	7	20				{g,h}	7	20	4
(4,2)				gi	10	20	gha	9	20	{g,h,a}	9	20	3
(4,3)							gib	14	10	{g,i,b}	14	10	2
(4,5)	g	4	20							{g}	4	20	3
(5,1)	h	3	20							{h}	3	20	2
(5,2)	i	6	20	ha	5	20				{h,a}	5	20	1
(5,3)				ib	10	10	hab	9	10	{h,a,b}	9	10	2
(5,4)	j	3	10							{j}	3	10	5

Figure 5.3. Path Table for Phase I Solution

Node Pair	Node-Pair Req't	ARCS									
		a	b	c	d	e	f	g	h	i	j
(1,2)	3	3									
(1,3)	2	2	2								
(1,4)	1	1		1							
(1,5)	3	3			3						
(2,1)	6		6			6					
(2,3)	3		3								
(2,4)	2			2							
(2,5)	3				3						
(3,1)	2					2					
(3,2)	1	1				1					
(3,4)	7						7				
(3,5)	4						4	4			
(4,1)	4							4	4		
(4,2)	3	3						3	3		
(4,3)	2		2					2		2	
(4,5)	3							3			
(5,1)	2								2		
(5,2)	1	1							1		
(5,3)	2	2	2						2		
(5,4)	5										5
Total Arc Flow		16	15	3	6	9	11	16	12	2	5

Figure 5.4. Arc Flow Tabulation

Phase II Solution

The Phase II algorithm is now applied to the results obtained in Phase I. The solution steps are again labeled with the letters used in the Phase II flow chart given in Figure 3.2.

A. The algorithm begins by calculating the cost of the Phase I solution (i.e., the root node (R) of the solution tree).

$$COST(R) = \sum_{\alpha \epsilon A} f(\alpha) \cdot c(\alpha) = 293.$$

It also calculates the cost of the maximum cost. That is, $\overline{COST} = \sum_{\alpha \epsilon A} b(\alpha) \cdot c(\alpha) = 2640.$

B. Since the solution tree at this stage only contains the root node, select it for branching.

D. Consider the single overflow arc b in this step. From Figure 5.3 find the node-pair paths which contribute to the total flow in this arc and the amount contributed by each. A table containing this information now follows:

Elements in AU(b) i.e. (b,x,y)	(b,1,3)	(b,2,1)	(b,2,3)	(b,4,3)	(b,5,3)
Amount of Flow Contributed i.e. r(x,y)	2	6	3	2	2

E. The flow in arc b must be reduced so that its flow does not exceed its capacity. Since the capacity of b is ten, the flow in arc b can be reduced to within capacity by eliminating the paths between the

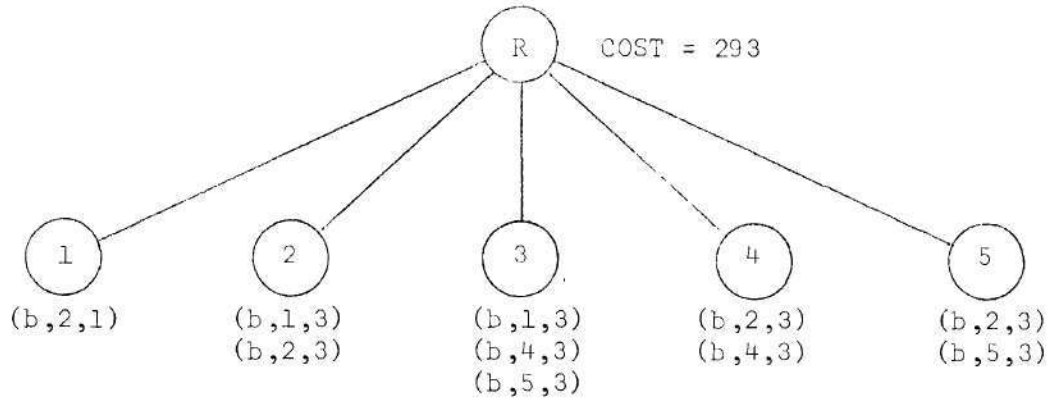
following sets of node-pairs:

$$AE(b) = \langle\{(b,2,1)\},\ \{(b,1,3),(b,2,3)\},\ \{(b,1,3),(b,4,3),(b,5,3)\}$$

$$\{(b,2,3),(b,4,3)\},\ \{(b,2,3),(b,5,3)\}\rangle .$$

Note that the elements in AE(b) are the smallest sets which eliminate the overflow in arc b.

H. Since only one arc had overflow, the set OR(R) is identical to the set AE(b). Create one branch from the root node in the solution tree for each of the elements in OR(R). This results in the following solution tree where the labels represent the way the overflow is reduced:



I. Select, arbitrarily, node 1 for evaluation.

J. Node-pair (2,1) is selected and the following arc sets are found for this node-pair:

$\beta = \{b\}$ $\gamma = \{a\}$ $\delta = \{b,a\}$

L. The arcs b and a are eliminated from the network and the path algorithm and auxiliary algorithm used to find the following least cost path between node-pair (2,1):

$\bar{\pi}(2,1)$	
$AS(\bar{\pi}(2,1))$	$c(\bar{\pi}(2,1))$
{d,h}	9

Q. Create the history set H(1) = {(a,2,1),(b,2,1)}, assign the required flow to $\bar{\pi}(2,1)$ and calculate the flow on all arcs for the solution at node 1.

α	a	b	c	d	e	f	g	h	i	j
$c(\alpha)$	2	4	3	6	2	1	4	3	6	3
$b(\alpha)$	20	10	5	10	10	20	20	20	20	10
$f(\alpha)$	16	9	3	12	3	11	16	18	2	5

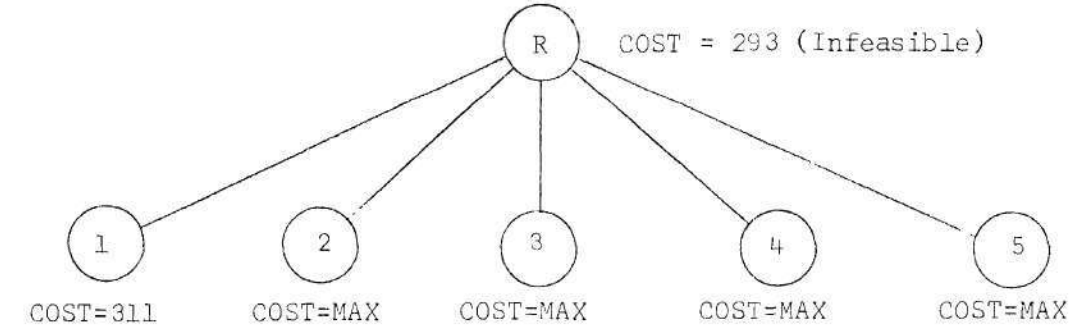
↑

The cost of the node 1 solution is COST(1) = 311.

R. An overflow is present on arc d at solution node 1. Therefore, the solution is infeasible and is placed in the set IS of infeasible nodes.

The remaining four branches from the root node result in solutions for which a cost cannot be obtained (i.e., COST = MAX). This is due to the fact that no alternative paths exist in the network to carry the flow between the respective node-pairs in the solution nodes 2, 3, 4, and 5 when arc b is dropped from consideration.

The results at this point in the solution are summarized in the following solution tree diagram:



X. The infeasible node set IS contains one infeasible solution node (i.e., node 1). Therefore, the arc overflows in this solution must be examined next.

B. Select solution node 1 for branching.

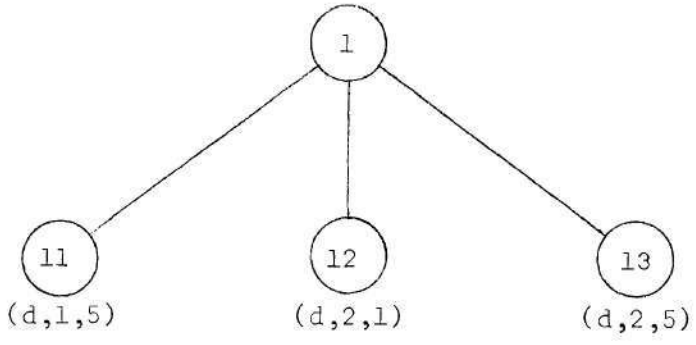
D. Consider the single overflow arc d in this step. Find the node-pair paths which contribute to the total flow in this arc and the amount of flow contributed by each. That is:

Elements in AU(d) i.e. (d,x,y)	(d,1,5)	(d,2,1)	(d,2,5)
Amount of Flow Contributed i.e. r(x,y)	3	6	3

E. The flow in arc d must be reduced to within capacity. Since the capacity of d is ten, the smallest sets which eliminate the overflow in arc d are:

$$AE(d) = \{\{(d,1,5)\}, \{(d,2,1)\}, \{(d,2,5)\}\}.$$

H. Since only one arc had overflow, the set OR(1) is the same as the set AE(d). Create one branch from node 1 in the solution tree for each of the elements in OR(1). This yields the following branches out of node 1.



I. Select solution node 11 for evaluation.

J. Node-pair (1,5) is selected and the following sets are found:

$$\beta = \{d\} \qquad \gamma = \phi \qquad \delta = \{d\}$$

L. Arc d is eliminated from the network and the path and auxiliary algorithms used to find the following least cost path between node-pair (1,5):

$\bar{\pi}(1,5)$	
$AS(\bar{\pi}(1,5))$	$c(\bar{\pi}(1,5))$
{a,c,g}	9

Q. Create the history set H(11) = {(a,2,1),(b,2,1),(d,1,5)}, assign the required flow to $\bar{\pi}(1,5)$ and calculate the flow on all arcs for the solution at node 11.

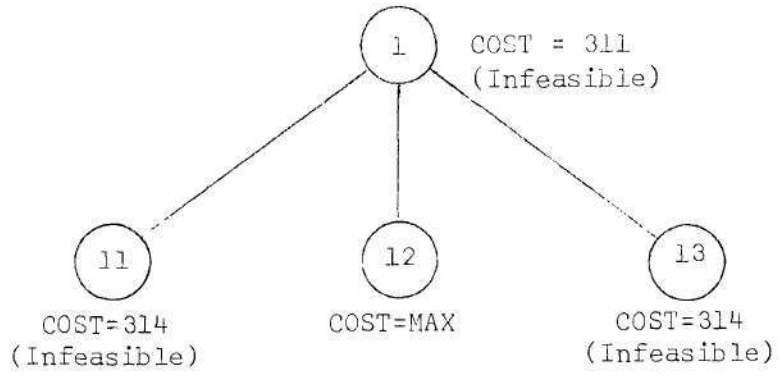
α	a	b	c	d	e	f	g	h	i	j
c(α)	2	4	3	6	2	1	4	3	6	3
b(α)	20	10	5	10	10	20	20	20	20	10
f(α)	16	9	6	9	3	11	19	18	2	5

↑

The cost of the node 11 solution is COST(11) = 314.

R. An overflow is present on arc c in the above solution. Therefore, the solution is infeasible and is placed in the IS set of infeasible nodes for further branching.

Steps I-R are repeated for solution nodes 12 and 13 and the following results obtained:



X. The set IS now contains two infeasible solution nodes (i.e., node 11 and node 13).

B. Select solution node 11 for branching.

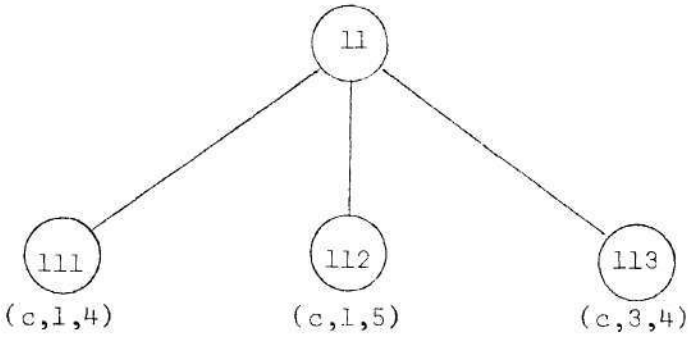
D. Consider the single overflow arc c in this step. Find the node-pair paths which contribute to the total flow in this arc:

Elements in AU(c) i.e. (c,x,y)	(c,1,4)	(c,1,5)	(c,3,4)
Amount of Flow Contributed i.e. r(x,y)	1	3	2

E. To reduce the flow in arc c to within its capacity of 5, we create the sets:

$$AE(c) = \langle\{(c,1,4)\}, \{(c,1,5)\}, \{(c,3,4)\}\rangle.$$

H. The set OR(11) is the same as set AE(c) since only one arc has overflow. Create the following branches from solution node 11:



I. Select solution node 111 for evaluation.

J. Node-pair (1,4) is selected and the following sets found:

$$\beta = \{c\} \qquad \gamma = \phi \qquad \delta = \{c\}$$

L. Arc c is eliminated from the network and the path and auxiliary algorithm used to find the following least cost path between node-pair (1,4):

$\bar{\pi}(1,4)$	
$AS(\bar{\pi}(1,4))$	$c(\bar{\pi}(1,4))$
{a,b,f}	7

Q. Create the history set H(111) = {(a,2,1), (b,2,1), (d,1,5), (c,1,4)}, assign the required flow to $\bar{\pi}(1,4)$ and calculate the flow on all arcs for the solution at node 111.

α	a	b	c	d	e	f	g	h	i	j
c(α)	2	4	3	6	2	1	4	3	6	3
b(α)	20	10	5	10	10	20	20	20	20	10
f(α)	16	10	5	9	3	12	19	18	2	5

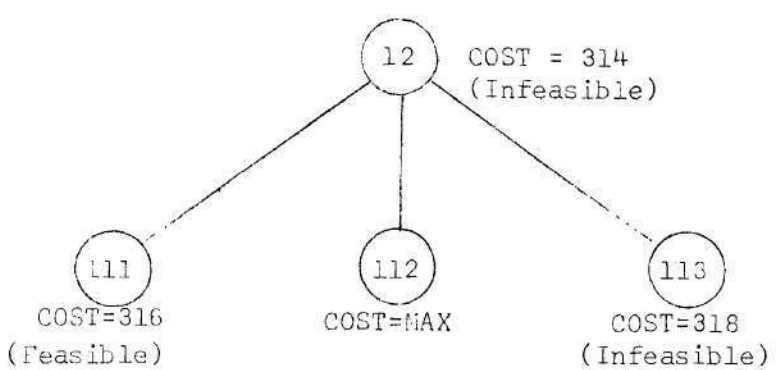
The cost of the solution at node 111 is COST(111) = 316.

R. No overflow is found on the arcs in the node 111 solution, therefore the solution is feasible.

U. The feasible solution at node 111 is a candidate for the optimum feasible solution to the network problem.

No infeasible solution nodes can be eliminated at this time, for the only one present in the tree is node 13 which has a cost equal to 314. However, the minimum solution cost, COST, is set equal to 316 and future infeasible nodes having a greater cost can be dropped from consideration.

Steps I-R are repeated for solution nodes 112 and 113 with the following results:



X. The set IS now contains one solution (node 13). (Node 113 is not placed in IS since COST(113)>COST.)

The step by step illustration of the Phase II algorithm terminates at this point. Node 13 requires branching and it is necessary to return to step B to continue the solution process until all infeasible nodes are handled. Since the basic use of the algorithm has been illustrated, we omit the details and summarize the results.

Final Results

Figure 5.5 illustrates the entire solution tree for the communication network problem being solved. This chart was obtained by completing all the steps required by the Phase II algorithm. Solution nodes 111 and 131 represent optimal, feasible solutions to the original problem. It is coincidental that both of these solution nodes give the identical cost and the identical network path set to the problem. The complete solution in terms of the optimal assignment of arcs to the paths between all node-pairs is given in Figure 5.6.

Network conditions 2, 3, and 4 can be verified from the two tables in Figure 5.6. Network condition 2 is satisfied because no path in Table A has a length greater than 3. A comparison of columns 2 and 3 of Table A shows that all the path capacities are greater than the flow requirements which are assigned to them. This means that condition 3 is satisfied. Finally in Table B, the flows in the arcs are always less than or equal to the capacities of the arcs, thus validating condition 4.

With this optimum solution, the XYZ Corporation can set up the required transmission routes for its existing communication system. Hopefully, the management of the Corporation will continue to use the network algorithm in the design of the networks for its future management information system.

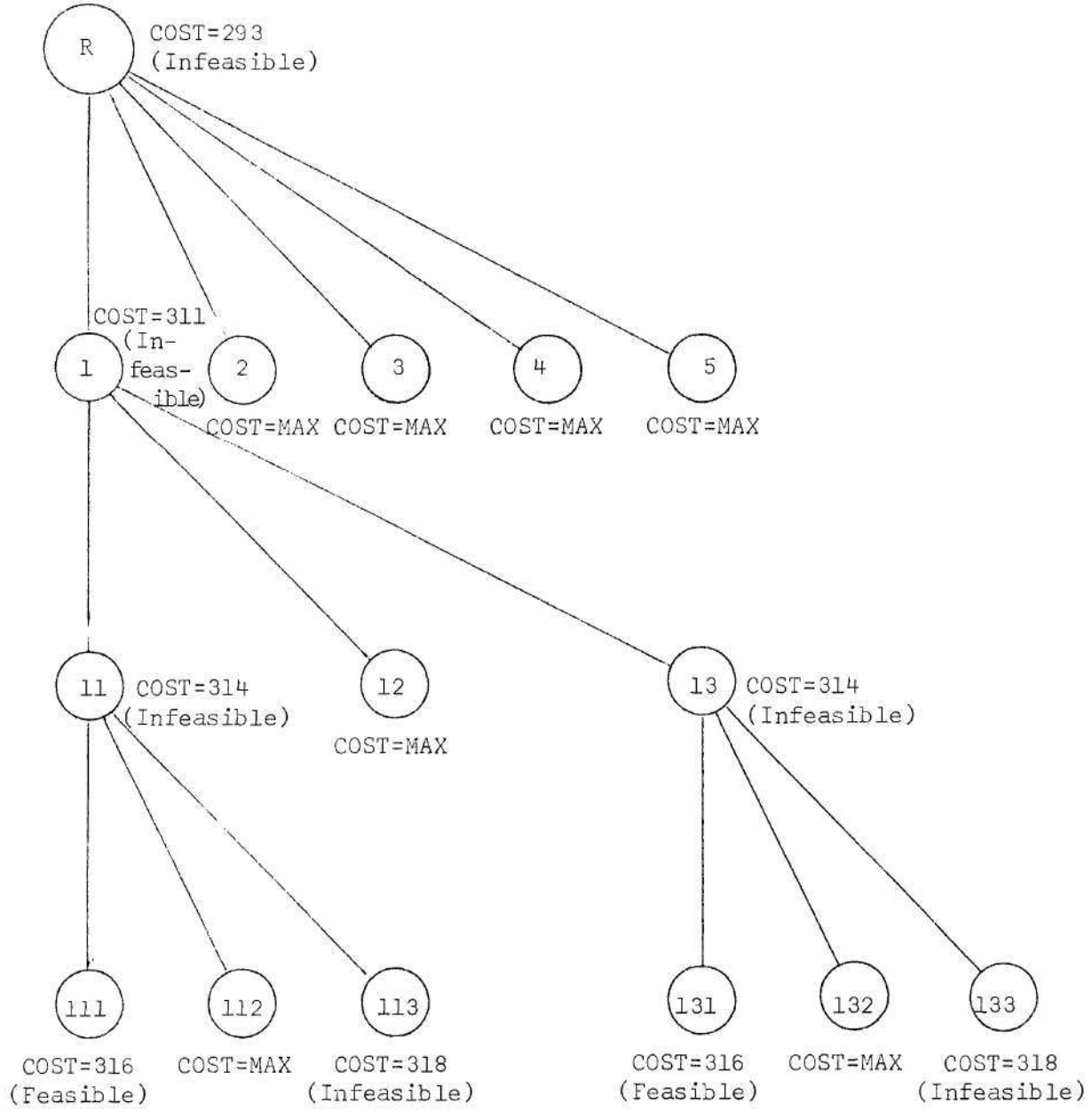


Figure 5.5. Solution Tree

		Solution Paths				
Node Pair	Node-Pair Req'ts	Capacity	Cost	Arc Set		
(1,2)	3	20	2	a		
(1,3)	2	10	6	a	b	
(1,4)	1	10	7	a	b	f
(1,5)	3	5	9	a	c	g
(2,1)	6	10	9	d	n	
(2,3)	3	10	4	b		
(2,4)	2	5	3	c		
(2,5)	3	10	6	d		
(3,1)	2	10	2	e		
(3,2)	1	10	4	e	a	
(3,4)	7	20	1	f		
(3,5)	4	20	5	f	g	
(4,1)	4	20	7	g	h	
(4,2)	3	20	9	g	h	a
(4,3)	2	10	13	g	i	b
(4,5)	3	20	4	g		
(5,1)	2	20	3	h		
(5,2)	1	20	5	h	a	
(5,3)	2	10	9	h	a	b
(5,4)	5	10	3	j		

Table A

	Solution Arcs		
Arcs	Cost	Capacity	Flow
a	2	20	16
b	4	10	10
c	3	5	5
d	6	10	9
e	2	10	3
f	1	20	12
g	4	20	19
h	3	20	18
i	6	20	2
j	3	10	5

Table B

Figure 5.6. Minimum Cost Solution to Communication Problem

CHAPTER VI

SUMMARY AND CONCLUSIONS

Summary of Results and Conclusion

This dissertation documents a solution to a variation of the simultaneous flow, minimum cost network flow problem. The solution approach is based on the concept of least cost paths, and it uses a branch and bound search technique to find the minimum cost solution. The solution algorithm represents a new solution approach to problems dealing with the optimal assignment of simultaneous flows in communication networks.

Our first result was the creation of a versatile optimal path-finding algorithm. The algorithm was described, proved, and illustrated in Chapter II. By using an incremental path creation concept (i.e., creating paths one arc at a time), the path algorithm was designed to find least weighted paths whose lengths are less than or equal to a pre-set integer L. Because of the generalized use of arc weight in the path algorithm, it has a wide applicability in solving many types of network optimization problems.

The main result of the dissertation was the solution of the network problem. The complete solution algorithm was presented in Chapter III and illustrated in Chapter V. The two phases of the algorithm were discussed separately, but they are used in sequence to find a minimum cost solution, if such a solution exists. The path algorithm is used as

an integral part of the solution algorithm in order to find the actual least cost values and arc sets for paths between all node-pairs.

Additional results were obtained in Chapter IV in the form of necessary conditions and solution cost bounds for any strongly connected network which is adequately defined. The theorems proved in Chapter IV are used to detect network configurations which (1) are not solvable (i.e., do not satisfy a necessary condition) and (2) have a solution cost (occurring within the cost bounds) which is unacceptable.

Our solution approach to the network problem was intentionally oriented around graph theory topics. Almost all previous results reported in the literature were obtained within a linear programming framework. Therefore, we have based our work on the conviction that the problem can also be solved by using concepts based exclusively on elementary graph theory. The positive results discussed above force us to conclude that this conviction was validated. The extent of the validation depends on more conclusive results obtained from the operational use of the solution algorithm.

Further Areas of Study

Although our main objective was to demonstrate the existence of a solution to the network problem, a number of interesting problems were uncovered within the framework of the dissertation results. Now several topics can be identified for refinement and further research.

1. Necessary and Sufficient Conditions. More analytical results on this topic would enhance the use of the solution algorithm.

Additional ways are needed to detect the existence of a solution before applying the solution algorithm to a problem.

2. Shortcuts in the Algorithms. The algorithms discussed in this dissertation represent the general steps required to obtain a solution. A detailed analysis of each algorithm should lead to simplifications and shortcuts which would save computational time or storage locations. Also, a complete study of alternative heuristics would lead to a more effective branch and bound technique.

3. Programming and Implementation. The solution algorithm should be programmed and operational statistics gathered from various types of network configurations. The results from this work would be extremely useful in developing and extending all parts of the solution algorithm.

4. The Path Algorithm. The path algorithm was studied using analytical as well as simulation tools. Our work should be extended in order to complete its development for numerous problems where path length is a constraining variable.

5. Science Information Networks: Networks which handle scientific information have certain properties which make them amenable to the results obtained in this dissertation. For example, formatted and standard sized message transmissions are reasonably common in communicating science information. Continued research in specific applications of the dissertation results is a natural area for further research.

APPENDIX

Enumeration Theorems

In this appendix, two path enumeration theorems are proved. The theorems are used in Chapter I to find the number of different types of paths in directed networks.

Theorem 7.1

The number of simple paths between all node-pairs in a complete, loopless, directed network is:

$$n(n-1) \sum_{x=0}^{n-2} x! \binom{n-2}{x}.$$

Proof. Let (x,y) be one of the n(n-1) node-pairs in a complete loopless directed network. The desired result is obtained by considering the simple paths of increasing length between (x,y).

The arc [x,y]' is the only path of length 1 between (x,y). Since there are n-2 other nodes (other than x or y) in the network, there are n-2 paths of length 2 which exist between (x,y). Now consider paths of length 3. Let the two intermediate nodes between (x,y) be w and z. There are n-2 ways to choose w and afterwards n-3 ways to choose z. There are two nodes to select from a set of n-2 nodes and the order of selection is important. Therefore, the number of paths of length 3 is the permutation of n-2 nodes taken 2 at a time. That is: (n-2)(n-3).

Paths of increasing length follow the same pattern as the case of paths of length 2 with the following overall result:

$\sum_{x=0}^{n-2} x! \left\{ {n-2 \atop x} \right\}$ is the number of directed paths of length 1 to n-1 between node-pair (x,y) (i.e., x=0 to n-2). Since this result holds for each of the n(n-1) node-pairs, the theorem is proved. []

Theorem 7.2

The maximum number of simple paths between all node-pairs as found *by the path algorithm* for a complete, loopless directed network is:

$$n(n-1)^2 (n-2)/2.$$

Proof. Let (x,y) be one of the n(n-1) node-pairs in a complete, loopless network. The desired result is obtained by using the looping framework of the path algorithm and considering the simple paths of increasing length between (x,y).

The arc [x,y]' is the only path of length one between (x,y). However, this arc is not found by the algorithm because it is part of the definition of the network. Now paths of length two are created by appending two paths of length one. Since there are n-2 other nodes (i.e., other than x and y) in the network and there exist arcs between all distinct nodes, there are n-2 simple paths of length 2 between node-pair (x,y).

Beginning with paths of length 3, the results begin to differ with Theorem 7.1. The reason for this is based on the way the path algorithm finds paths of next higher length. The algorithm uses the

least weighted paths of length p and paths of length one (i.e., arcs) to create paths of length p+1. The algorithm selects *one* path $\bar{\pi}_p(x,z)$ of length p containing p+1 nodes for each node-pair and then appends the remaining (n-p) arcs one at a time to this path. This creates (n-p) possible new paths of length p+1 between node-pair (x,y). One of these is selected as the path having least weight and the process is repeated for n-2 iterations. At the final iteration only one simple path exists between (x,y).

Since the above process exists for all node-pairs, the number of calculations are:

$$n(n-1)[(n-2)+(n-3)+\ldots+1] = n(n-1)^2(n-2)/2. \quad \square$$

Computer Simulation Programs and Sample Printout

In order to obtain some of the results discussed in Chapter II a computer simulation program was written in FORTRAN and executed on the UNIVAC 1108 at the Georgia Tech Computer Center. The basic version of the Floyd and path algorithms actually used in the simulation program is given in Figure 7.1. Over 200 random networks were generated and executed by the algorithms. One of the 20-node networks is listed in matrix form in Figure 7.2 along with the output produced by the simulation program. Note that since no weights were assigned to the arcs in the main diagonal of the random input matrix in Figure 7.2 (i.e., $w(i,i)=+\infty$), non-zero weights were found for cycles and are shown in the diagonal of the shortest path matrix.

```
C          PATH ALGORITHM
  474 CALL TIME(DUM,ICLK)
      DO 60 I = 1,N
      ISWT = 0
      DO 56 IP = 1,N
      IF(ISWT) 60,48,60
   48 ISWT = 1
      DO 56 K = 1,N
      IF(INP(IP,I,K).EQ.INF) GO TO 56
      DO 55 J =1,N
      IF(INP(1,K,J).EQ.INF) GO TO 55
      IW = INP(IP,I,K) + INP(1,K,J)
      IF(IW.GE.IPTH(I,J)) GO TO 55
      IPTH(I,J) = IW
      INP(IP+1,I,J) = IW
      IPLN(I) = IP +1
      ISWT = 0
   55 CONTINUE
   56 CONTINUE
   60 CONTINUE
      CALL TIME(DUM,JCLK)
```

```
      CALL TIME(DUM,ICLK)
C       FLOYD ALGORITHM
      DO 51 I =1,N
      DO 51 J=1,N
      IF(IN(J,I).EQ.INF) GO TO 51
      DO 50 K=1,N
      IF(IN(I,K).EQ.INF) GO TO 50
      IW = IN(J,I) + IN(I,K)
      IF(IW.GE.IN(J,K)) GO TO 50
      IN   (J,K) = IW
   50 CONTINUE
   51 CONTINUE
      CALL TIME(DUM,JCLK)
```

Figure 7.1. FORTRAN Programs Used in Computer Simulation

```
RANDOM INPUT MATRIX 20X20    1
  ***   0   0   0   0   0   0   0   0   0   0  12   5   0   0   0   0   0   0   0
    0 ***   0   0   0   0   0   0   0   2   5  11   0   0   0   0   0   0   0   4
    0   0 ***   0   0   0   0   0   6  14   0   0   0   0   0   0   0   0   5   1
    0   0   0 ***   0   0   0   5  10   0   0   0   0   0   0   0   2   4   1   0
    0   0   0   0 ***   2   5   3   0   0   0   0   0   0   0   0   3   0   0   0
    0   0   0   0   0 ***   0   0   0   0   0   0   0   0   0   1   0   0   0   0
    0   0   0   1   6   6 ***   0   0   0   0   0   0   0   6   6   0   0   0   0
    0   1   7   7  11   0   0 ***   0   0   0   0   0  15   2   0   0   0   0   0
    0  14   8   0   0   0   0   0 ***   0   0   0   4  10   0   0   0   0   0   0
    2   0   0   0   0   0   0   0   0 ***   1   3  13   0   0   0   0   0   0   0
    7   0   0   0   0   0   0   0   0   0 ***   0   0   0   0   0   0   0   0  14
    0   0   0   0   0   0   0   0   0   1   2 ***   0   0   0   0   0   0   4   2
    0   0   0   0   0   0   0   0   2   7   0   0 ***   0   0   0   1   5   6   0
    0   0   0   0   0   0   0   5  14   0   0   0   0 ***   0   0  14   6   0   0
    0   0   0   0   0   1   5   5   0   0   0   0   0   0 ***   1   0   0   0   0
    0   0   0   0   7   1  10   0   0   0   0   0   0   0   0 ***   0   0   0   0
    0   0   2   7   6   0   0   0   0   0   0   0   0   0   8   8 ***   0   0   0
    2  10   2   0   0   0   0   0   0   0   0   0   0   0  10   0   0   0 ***   0   0
    9   0   0   0   0   0   0   0   0   0   0   0   0   4   1   0   0   0   0 ***   0
    0   0   0   0   0   0   0   0   0   0   0   0   5   0   0   0   0   0   0   0 ***

FLOYD ALGORITHM
     TIME OF EXECUTION    .0500 SECS
     SHORTEST PATH MATRIX
  12  16   8  13  12  14  17  15   7  12  13  12   5  12  14  14   6  10  11   9
   4  16  12  17  16  18  21  15  11   2   3   5   9  10  17  18  10  14   9   4
   9  12  12  17  16  14  18  11   6   7   8   6   9   6  13  14  10  12   5   1
   6   6   4   9   8   8  12   5   7   8   9  10   5   2   7   8   2   4   1   5
   8   4   5   6   9   2   5   3  11   6   7   9  11   8   5   3   3  10   7   6
  16  12  13  12   8   2  11  11  19  14  15  17  17  14  13   1  11  16  13  14
   7   7   5   1   6   6  11   6   8   9  10  11   6   3   6   6   3   5   2   6
   5   1   7   7  10   3   7   7  12   3   4   6  10   9   2   3   9  11   8   5
  11  14   7  12  11  13  16  14   6  11  12  13   4  10  13  13   5   9  10   8
   2  14  10  15  14  16  19  13   9   4   1   3   7   8  15  16   8  12   7   5
   7  23  15  20  19  21  24  22  14  19  20  19  12  19  21  21  13  17  18  14
   3  11  11  16  15  13  17  10  10   1   2   4   8   5  12  13   9  11   4   2
   7  11   3   8   7   9  12  10   2   7   8   9   6   7   9   9   1   5   6   4
   8   6   8  12  15   8  12   5  14   8   9  11  13  14   7   8  14   6  13   9
  10   6  10   6   8   1   5   5  13   8   9  11  11   8   7   1   8  10   7  10
  15  11  12  11   7   1  10  10  18  13  14  16  16  13  12   2  10  15  12  13
  11  10   2   7   6   8  11   9   8   9  10   8  11   8   8   8   9  11   7   3
   2  10   2  15  14  16  19  13   8   9  10   8   7   8  15  16   8  12   7   3
   9   7   7  12  11   9  13   6   6   9  10  12   4   1   8   9   5   7  10   8
   8  16  16  21  20  18  22  15  15   6   7   5  13  10  17  18  14  16   9   7

PATH ALGORITHM
     TIME OF EXECUTION    .1050 SECS
     PATH LENGTH  6  6  5  5  5  6  6  5  5  5  6  6  5  5  6  6  4  4  4  5
     SHORTEST PATH MATRIX
  12  16   8  13  12  14  17  15   7  12  13  12   5  12  14  14   6  10  11   9
   4  16  12  17  16  18  21  15  11   2   3   5   9  10  17  18  10  14   9   4
   9  12  12  17  16  14  18  11   6   7   8   6   9   6  13  14  10  12   5   1
   6   6   4   9   8   8  12   5   7   8   9  10   5   2   7   8   2   4   1   5
   8   4   5   6   9   2   5   3  11   6   7   9  11   8   5   3   3  10   7   6
  16  12  13  12   8   2  11  11  19  14  15  17  17  14  13   1  11  16  13  14
   7   7   5   1   6   6  11   6   8   9  10  11   6   3   6   6   3   5   2   6
   5   1   7   7  10   3   7   7  12   3   4   6  10   9   2   3   9  11   8   5
  11  14   7  12  11  13  16  14   6  11  12  13   4  10  13  13   5   9  10   8
   2  14  10  15  14  16  19  13   9   4   1   3   7   8  15  16   8  12   7   5
   7  23  15  20  19  21  24  22  14  19  20  19  12  19  21  21  13  17  18  14
   3  11  11  16  15  13  17  10  10   1   2   4   8   5  12  13   9  11   4   2
   7  11   3   8   7   9  12  10   2   7   8   9   6   7   9   9   1   5   6   4
   8   6   8  12  15   8  12   5  14   8   9  11  13  14   7   8  14   6  13   9
  10   6  10   6   8   1   5   5  13   8   9  11  11   8   7   1   8  10   7  10
  15  11  12  11   7   1  10  10  18  13  14  16  16  13  12   2  10  15  12  13
  11  10   2   7   6   8  11   9   8   9  10   8  11   8   8   8   9  11   7   3
   2  10   2  15  14  16  19  13   8   9  10   8   7   8  15  16   8  12   7   3
   9   7   7  12  11   9  13   6   6   9  10  12   4   1   8   9   5   7  10   8
   8  16  16  21  20  18  22  15  15   6   7   5  13  10  17  18  14  16   9   7
```

Figure 7.2. Sample Simulation Program Printout

BIBLIOGRAPHY

A significant amount of the literature on graph theory and information networks was reviewed in order to support the research documented in this dissertation. Much of the material was not related, or was only tangentially related, to our areas of interest. The citations to those topics which were related are included in our Bibliography. For ease of reference, we have organized the citations into the following subject categories:

A. Graph and Network Theory Books

B. Communication and Information Networks

C. Structure and Connectivity of Graphs

D. Related Graph Theory Topics

E. Path-Finding and Optimization Algorithms

F. Flow and Synthesis Considerations in Networks

A. Graph and Network Theory Books

A01. Bellman, R., *Dynamic Programming*. Princeton: Princeton Univ. Press, 1957.

A02. Berge, C. *Theory of Graphs and Its Applications*. New York: Wiley, 1962. (Translation.)

A03. Busacker, R. G., and Saaty, T. L. *Finite Graphs and Networks*. New York: McGraw-Hill, 1965.

A04. Erdös, P., and Katona, G. (eds.). *Theory of Graphs*. Proceedings of the Colloquium held at Tihany, Hungary, Sept. 1966. New York: Academic Press, 1968.

A05. Fiedler, Miroslav (ed.). *Theory of Graphs and Its Applications*. Proceedings of the Symposium held in Smolenice in June 1963. New York: Academic Press, 1964.

A06. Flament, C. *Applications of Graph Theory to Group Structure*. Englewood Cliffs, N. J.: Prentice-Hall, 1963.

A07. Ford, L. R., and Fulkerson, D. R. *Flows in Networks*. Princeton: Princeton University Press, 1962.

A08. Frank, H., and Frisch, I. T. *Communication and Transmission Networks*. (To be published.)

A09. Harary, F. (ed.). *A Seminar on Graph Theory*. New York: Holt, Rinehart & Winston, 1967.

A10. Harary, F. *Graph Theory*. New York: Addison Wesley, 1969.

A11. Harary, F. (ed.). *Proof Techniques in Graph Theory*. New York: Academic Press, 1969.

A12. Harary, F., Norman, R. Z., and Cartwright, D. *Structural Models: An Introduction to the Theory of Directed Graphs*. New York: Wiley, 1965.

A13. Hu, T. C. *Integer Programming and Network Flows*. New York: Addison Wesley, 1970.

A14. Kaufmann, A. *Graphs, Dynamic Programming and Finite Games*. New York: Academic Press, 1967.

A15. Kim, W. H., and Chien, R. T. W. *Topological Analysis and Synthesis of Communication Networks*. New York: Columbia University Press, 1962.

A16. König, D. *Theorie der endlichen und unendlichen Graphen.* Akademische Verlags Gesellschaft 14.B.H. 1936. Also by Chelsea Publishing Co., 1950.

A17. Ore, Oystein. *Theory of Graphs.* American Math. Soc., 1962.

A18. Ore, Oystein. *Graphs and Their Uses.* New York: Random House, 1963.

A19. Ponstein, J. *Matrices in Graph and Network Theory.* Royal Van Gorcum Ltd., 1966.

A20. Seshu, S., and Reed, M. B. *Linear Graphs and Electrical Networks.* New York: Addison-Wesley, 1961.

A21. *Theory of Graphs.* Proceedings of the International Symposium, Rome, 1966. New York: Gordon and Breach (Paris: Dumod), 1967.

A22. Tutte, W. T. *Connectivity in Graphs.* Math. Expositions No. 15. Toronto: University of Toronto Press, 1966.

B. Communication and Information Networks

B01. Bagrunovskii, K. A. "The Statement of a Problem in the Analysis of a Network Diagram." (In Russian.) *Vychisl. Sistemy* 11 (1964): 71-93.

B02. Becker, J. "Information Network Prospects in the U. S." *Library Trends* 17 (Jan. 1969): 306-317.

B03. Becker, J., and Olsen, W. C. "Information Networks." Cuadra, C. A. (ed.), *Annual Review of Information Science and Technology*, Vol. 3, 1968. pp. 289-327.

B04. Block, G., and Judd, D. R. "Computer Networks." *Science Journal* (Sept. 1967): 19-40.

B05. Bollobas, B. "A Problem of the Theory of Communication Networks." *Acta Math. Acad. Sci. Hungar.* 19 (1968): 75-80.

B06. Brown, G. W., Miller, J. G., and Keenan, T. A. (eds.). *Edunet: Report on the Summer Study on Information Networks.* New York: Wiley, 1967.

B07. Chartrand, G. "A Graph-Theoretic Approach to a Communications Problem." *J. SIAM Appl. Math.* 14 (1966): 778-781.

B08. Cherry, F. Colin. "Generalized Concepts of Networks." *Proc. Symposium on Information Networks*, Polytechnic Inst. of Brooklyn, 1954. pp. 175-184.

B09. Fox, Jerome (ed.). *Proc. Symposium on Information Networks*, April 1954. New York: Polytechnic Inst. of Brooklyn, 1954.

B10. Hakimi, S. L. *Application of Graph Theory to the Synthesis of Networks*. Research Report, Northwestern University, Evanston, Ill., October 25, 1965.

B11. Iri, M. "On the Basic Theory of General Information Networks and Its Applications." Doctoral Thesis submitted to the Division of Research in Math. and Physical Sciences, Graduate School of the University of Tokyo, March 1960.

B12. Iri, M. "Theory of General Information Networks: An Algebraic and Topological Foundation to the Theory of Information Handling Systems." *Proc. Sympos. Math. Theory of Automata* (New York, 1962), Polytechnic Inst. of Brooklyn. pp. 415-435.

B13. Kalaba, R. E. "On Some Communication Network Problems." *Proc. Symp. Applied Math.* Vol. X. Amer. Math. Soc., 1960.

B14. Kessler, M. M. *Technical Information Flow Patterns*. Lincoln Labs., Lexington, Mass., MIT, 1961. Also in WJCC, 1961.

B15. MacKenzie, Kenneth D. "Decomposition of Communication Networks." *J. Math. Psychology* 4 (1967): 162-174.

B16. Nance, R. E. "An Analytic Model of a Library Network." *J. ASIS* 21 (Jan.-Feb. 1970): 58-66.

B17. Overhage, C. F. G. "Information Networks." Cuadra, C. A. (ed.), *Annual Review of Information Science and Technology*, Vol. 4, 1969. pp. 339-377.

B18. Ponstein, J. "Matrix Description of Networks." *J.SIAM* 9 (1961): 233-268.

B19. Roginskiy, V. N. "Designing the Structure of Communications Networks." *Engineering Cybernetics* (May-June 1963): 148-152.

B20. Samuelson, Kjell. "Systems Design Concepts for Automated International Networks." *Proc. ASIS Annual Meeting*, 1969. Vol. 6. p. 431.

B21. Scantlebury, R. A., et al. "The Design of a Message Switching Centre for a Digital Communications Network." *Proc. Fourth Congress of the Intl. Fed. for Info. Proc.*, Edinburgh, 5-10 August 1968.

B22. Shaw, M. E. "Random vs. Systematic Distribution of Information in Communication Nets." *J. Personnel* 25 (1956): 59-69.

B23. Shimbel, A. "Applications of Matrix Algebra to Communication Nets." *Bull. Math. Biophysics* 13 (1951): 165-178.

B24. Sunaga, T., and Iri, M. "Theory of Communication and Transportation Networks." *RAAG Memoirs* 2 (1952): 444-468.

B25. Swanson, R. "Information System Network--Let's Profit From What We Know." Office of Aero. Res., USAF, June 1966.

B26. Tolchan, A. Y. "A Method for Optimizing the Structure of a Communications Net." (In Russian.) *Prob. Per. Inf.* 15 (1963): 42-60.

B27. Wall, Eugene. "Possibilities of Articulation of Information Systems Into a Network." *ADI* 19 (April 1968): 181-187.

B28. Weinberg, A. M., et al. *Science, Government and Information.* The President's Science Advisory Committee. The White House, Washington, D. C., Jan. 10, 1963.

B29. Yaged, B., Jr. "Basic Planning for Future Communications Networks." *1968 IFAC Symposium: Optimal Systems Planning*, June 20-22, 1968. pp. 36-53.

C. Structure and Connectivity of Graphs

C01. Akers, S. B., Jr. "A Modification of Lee's Path Connection Algorithm." *IEEE Trans. EC* 16 (Feb. 1967): 97-98.

C02. Beineke, L. W., and Harary, F. "The Connectivity Function of a Graph." *Mathematika* 14 (1967): 197-202.

C03. Bellert, S. "Topological Considerations and Synthesis of Linear Networks by Means of the Method of Structural Numbers." *Arch. Elektrotechniki* 12 (1963): 473-500.

C04. Bollobas, B. "On Graphs With at Most Three Independent Paths Connecting Any Two Vertices." *Studia Sci. Math. Hungar.* 1 (1966): 137-140.

C05. Bollobas, B. "Graphs of a Given Diameter." In: Erdos (ed.), *Theory of Graphs*. Budapest, 1968. pp. 29-36.

C06. Cartwright, D., and Gleason, T. C. "The Number of Paths and Cycles in a Graph." *Psychometrika* 31 (June 1966): 179-199.

C07. Cartwright, D., and Harary, F. "The Number of Lines in a Digraph of Each Connectedness Category." *SIAM Rev.* 3 (1961): 309-314.

C08. Chartrand, G., and Harary, F. "Graphs With Prescribed Connectivities." Erdos (ed.), *Theory of Graphs*, Budapest, Akademiai Kiado, 1968. pp. 61-63.

C09. Chen, Y. C. "The Connectedness of Directed Graphs and Applications." Ph.D. Dissertation, Department of Electrical Engineering, Columbia University, New York, 1966.

C10. Chen, Y. C., and Wing, O. "Connectivity of Directed Graphs." *Proc. Second Allerton Conference on Circuit and Systems Theory*, Sept. 1964. pp. 530-543.

C11. Chen, Y. C., and Wing, O. "Some Properties of Cycle-Free Directed Graphs and the Identification of the Longest Path." *J. Franklin Inst.* 281 (1966): 293-301.

C12. Cummius, R. L. "Hamiltonian Circuits in Tree Graphs." Ph.D. Dissertation, Digital Computer Lab., Univ. of Illinois, 1962.

C13. Dirac, G. A. "Connectedness and Structure in Graphs." *Rend. Circ. Mat. Palermo* 9 (1961): 114-124.

C14. Dirac, G. A. "Connectivity Theorems for Graphs." *Qtrly J. Math. Oxford*, Ser. (2), 3 (1952): 171-174.

C15. Dirac, G. A. "Some Results Concerning the Structure of Graphs." *Can. Math. Bull.* 6 (1963): 183-210.

C16. Edmonds, J. "Existence of k-Edge Connected Ordinary Graphs With Prescribed Degrees." *J. Res. Nat. Bur. Stand. Sect. B* 68 (1964): 73-74.

C17. Gallai, T. "On Directed Paths and Circuits." Erdös, P., and Katona, G. (eds.), *Theory of Graphs*. Budapest, 1968.

C18. Goldman, A. J. "Realizing the Distance Matrix of a Graph." *J. Res. Nat. Bur. Stand. Sect. B70B* (1966): 153-154.

C19. Griswold, R. G. "Connectivity and Pseudo Trees in Directed Graphs." Ph.D. Dissertation, Rensselaer Poly. Inst., 1964.

C20. Guillemin, E. A. "How to Grow Your Own Trees From Given Cut-Set or Tie-Set Matrices." *IRE Trans. CT 6* (May 1959): 110-126.

C21. Hakimi, S. L. "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph." *Op. Res.* 12 (1964): 450-459.

C22. Hakimi, S. L., and Yau, S. S. "Distance Matrix of a Graph and Its Realizability." *Qtrly App. Math.* 22 (1964): 305-317.

C23. Harary, F. "The Maximum Connectivity of a Graph." *Proc. Nat. Acad. Sci. USA* 48 (1962): 1142-1146.

C24. Harary, F., and Ross, I. C. "A Procedure for Clique Detection Using the Group Matrix." *Sociometry* 20 (1957): 205-215.

C25. Harary, F., and Ross, I. C. "Identification of the Liaison Persons of an Organization Among the Structure Matrix." *Mgmt. Sci.* (April-July 1955): 251-258.

C26. Harary, F., and Trauth, C. A., Jr. "Connectedness of Products of Two Directed Graphs." *J. SIAM* 14 (1966): 250-254.

C27. Hobbs, A. M., and Grossman, J. W. "Thickness and Connectivity in Graphs." *J. Res. Nat. Bur. Stand.*, Sect. B (to appear).

C28. Jensen, H., and LaPatra, J. W. "Topologically Distinct Subsets in a Complex Network." *Proc. Third Allerton Conf.*, 1965. pp. 868-878.

C29. Johnson, M., Dulmage, A. L., and Mendelsohn. "Connectivity and Reducibility Graphs." *Can. J. Math.* 14 (1962): 529-539.

C30. Kahn, A. B. "Topological Sorting of Large Networks." *Comm. ACM* 5 (1962): 558-562.

C31. Kleitman, D. J. "Methods for Investigating the Connectivity of Large Graphs." *IEEE Trans. CT* 16 (May 1969): 232-233.

C32. Luce, R. D. "Networks Satisfying Minimality Conditions." *Amer. J. Math.* 75 (1953): 825-835.

C33. Mayeda, W. "Properties of Classes of Paths." Report R-212, Coordinated Science Lab., University of Illinois, Urbana, Ill., May 1964.

C34. Milic, M. M. "A Graph Theory Approach to the Analysis of Multiterminal Element Networks." *Proc. Fourth Allerton Conf.*, 1966. pp. 218-223.

C35. Minty, G. J. "A Simple Algorithm for Listing All Trees of a Graph." *IEEE Trans. CT* 12 (March 1965): 120.

C36. Moon, J. W. "On Cliques in Graphs." *Israel J. Math.* 3 (March 1965): 23-28.

C37. Murty, U. S. R., and Vijayan, K. "On Accessibility in Graphs." *Sankhya, Ser. A,* 26 (1965): 270-302.

C38. Norman, R. L. "A Matrix Method for Location of Cycles of a Directed Graph." *Am. Inst. Chem. Eng. J.* 11 (1965): 450-452.

C39. Ore, O. "Studies on Directed Graphs, I." *Annals of Math.* 63 (1956): 383-406.

C40. Ore, O. "Studies on Directed Graphs, II." *Annals of Math.* 64 (1956): 142-158.

C41. Pennington, A. J. "Fundamental Connection Matrices." *Proc. First Allerton Conf.,* 1963. pp. 579-588.

C42. Pereira, J. M. S. "Some Results on the Tree Realization of a Distance Matrix." *Symposium on the Theory of Graphs,* Rome, 1966. pp. 383-388.

C43. Piekarski, Marian. "Listing of All Possible Trees of a Linear Graph." *IEEE Trans. CT* 12 (March 1965): 124-125.

C44. Ponstein, J. "Self-Avoiding Paths and the Adjacency Matrix of a Graph." *J. SIAM* 14 (1966): 600-609.

C45. Prihar, Z. "Topological Properties of Telecommunication Networks." *Proc. IRE* 44 (1956): 927-933.

C46. Ramamoorthy, C. V. "Analysis of Graphs by Connectivity Considerations." *JACM* 13 (1966): 211-222.

C47. Ramamoorthy, C. V. "Connectivity Considerations of Graphs Representing Discrete Sequential Systems." *IEEE Trans. EC* 14 (Oct. 1965): 724-727.

C48. Sabidussi, G. "Graphs with Given Group and Given Graph-Theoretic Properties." *Can. J. Math.* 9 (1957): 515-525.

C49. Sachs, H. "Construction of Non-Hamiltonian Planar Regular Graphs of Degrees 3, 4 and 5 With Highest Possible Connectivity." *International Symposium on Theory of Graphs,* Rome, 1966. pp. 373-382.

C50. Shimbel, A. "Structural Parameters of Communication Networks." *Bull. Math. Biophysics* 15 (1953): 501-507.

C51. Trauth, C. A. "On the Connectedness of Directed Graphs Under Binary Operations." Ph.D. Dissertation, Dept. of Math., Univ. of Mich., 1963.

C52. Tutte, W. T. "The Thickness of a Graph." *Nederl. Akad. Wetensch. Proc.* (1963). pp. 567-577.

C53. Unger, S. H. "An Algorithm for Finding the Reachability Matrix of a Directed Linear Graph." *IEEE Trans. CT* 16 (Feb. 1969): 130-132.

C54. Vizing, V. G. "On the Number of Edges in a Graph With Given Radius." (In Russian.) *Dokl. Akad. Nauk. SSSR* 173 (1967): 1245-1246.

C55. Watkins, M. F. "A Lower Bound for the Number of Vertices of a Graph." *Amer. Math. Monthly* 74 (1967): 297.

C56. Whitney, H. "Congruent Graphs and the Connectivity of Graphs." *Amer. J. Math.* 54 (1932): 150-168.

C57. Wing, O., and Kim, W. H. "The Path Matrix and Its Realizability." *IEEE Trans. CT* 6 (1959): 267-272.

C58. Wing, O., and Kim, W. H. "The Path Matrix and Switching Functions." *J. Franklin Inst.* 268 (1959): 251-269.

D. Related Graph Theory Topics

D01. Ahlborn, T. J. "On Directed Graphs and Related Topological Spaces." M.A. Thesis, Kent State University, August 1964.

D02. Ash, R. B., and Kim, W. H. "On the Realizability of a Circuit Matrix." *IRE Trans. CT* 6 (1959): 219-233.

D03. Auslander, L., and Trent, H. M. "On the Realization of a Linear Graph Given Its Algebraic Specification." *J. Acoustical Soc. of America* 33 (Sept. 1961): 1183-1192.

D04. Baraukin, E. W. "Precedence Matrices." Univ. of Chicago Management Sciences Research Report, Research Project No. 26, December 1953.

D05. Bedrosian, S. D. "Evaluation of Network Determinants Via the Key Subgraph." *Proc. First Allerton Conf.*, 1963. pp. 589-602.

D06. Bedrosian, S. D. "Minimizing Vulnerability in a Communications Network." *Proc. Second Allerton Conf.*, 1964. pp. 70-80.

D07. Bhargava, T. N., and Ahlborn, T. J. "On Topological Spaces Associated With Digraphs." *Acta. Math. Acad. Sci. Hungar.* 19 (1968): 47-52.

D08. Boesch, F. T., and Frisch, I. T. "On the Smallest Disconnecting Set in a Graph." *IEEE Trans. CT* 15 (Sept. 1968): 286-288.

D09. Boesch, F. T., and Thomas, R. E. "Optimal Damage Resistant Communications Networks." *Proc. 1968 IEEE International Conf. on Communications.* pp. 688-693.

D10. Bratton, D. *Efficient Communication Networks.* Cowles Commission Discussion Paper in Econ. No. 2119, Feb. 23, 1955.

D11. Cederbaum, I. "Applications of Matrix Algebra to Network Theory." *IRE Trans. CT* 6 (May 1959): 127-137.

D12. Chen, Wai-Kai. "Generation of Trees and k-Trees." *Proc. Third Allerton Conf.*, Oct. 1965. pp. 889-899.

D13. Denel, D. R., and Gill, A. "Some Decision Problems Associated With Weighted Directed Graphs." *J. SIAM Appl. Math.* 14 (Sept. 1966): 970-979.

D14. Dirac, G. A. "Short Proof of Menger's Graph Theorem." *Mathematika* 13 (1966): 42-44.

D15. Dunn, W. R., and Chan, S. P. "Topological Formulation of Network Functions Without Generation of k-Trees." *Proc. Sixth Allerton Conf.*, 1968. pp. 822-831.

D16. Erdös, P., and Gallai, T. "Graphs with Prescribed Degrees of Vertices." (In Hungarian.) *Mat. Lapok* 11 (1960): 264-274.

D17. Erdös, P., and Stone, A. "On the Structure of Linear Graphs." *Bull. Amer. Math. Soc.* 52 (1946): 1087-1091.

D18. Foulkes, J. P. "Directed Graphs and Assembly Schedules." *Proc. Symp. App. Math.* Vol. 10, American Math. Soc. (1960). pp. 281-289.

D19. Frank, H. "Analysis of Network by Vertex Reduction." *Proc. Second Allerton Conf.*, 1964. pp. 485-507.

D20. Gould, R. "Graphs and Vector Spaces." *J. Math. and Phys.* 37 (1958): 193-214.

D21. Hakimi, S. L. "On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph, I." *J. SIAM Appl. Math.* 10 (Sept. 1962): 496-506.

D22. Hakimi, S. L. "On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph, II, Uniqueness." *J. SIAM Appl. Math.* 11 (March 1963): 135-147.

D23. Hakimi, S. L. "On Realizability of a Set of Trees." *IEEE Trans. CT* 8 (March 1961): 11-18.

D24. Hakimi, S. L. "On the Degrees of the Vertices of a Directed Graph." *J. Franklin Inst.* 279 (April 1965): 290-308.

D25. Hakimi, S. L. "On the Trees of a Graph and Their Generation." *J. Franklin Inst.* 272 (Nov. 1961): 347-359.

D26. Hakimi, S. L. "Recent Progress and New Problems in Applied Graph Theory." *IEEE Region Six Conference Record*, 1966.

D27. Hakimi, S. L., and Green, D. G. "Generation and Realization of Trees and k-Trees." *IEEE Trans. CT* 11 (June 1964): 247-255.

D28. Harary, F. "On the Line-Group of Two Terminal Series-Parallel Networks." *J. Math. Phys.* 38 (1959): 112-118.

D29. Harary, F. "The Determinant of the Adjacency Matrix of a Graph." *SIAM Review* 4 (1962): 202-210.

D30. Harary, F. "Unsolved Problems in the Theory of Graphs." *Publ. Math. Inst. Hung. Acad. Sci.*, Ser. A, 5 (1960): 63-95.

D31. Harary, F., and Plamer, E. M. "Enumeration of Locally Restricted Digraphs." *Can. J. Math.* 18 (1966): 853-860.

D32. Harary, F., and Tutte, W. T. "The Number of Plane Trees With a Given Partition." *Mathematika* 11 (1964): 99-101.

D33. Harary, F., and Wilcox, G. "Boolean Operations on Graphs." *Math. Scand.* 20 (1967): 41-51.

D34. Hatcher, T. R. "The Vertex Matrix and the Cut Set Schedule as Special Cases of a More General Matrix." *IEEE Trans. CT* 5 (1958): 369-370.

D35. Kirchhoff, G. "Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird." *Ann. Phys. Chem.* 72 (1847): 497-508.

D36. Lasser, D. J. "Topological Ordering of a List of Randomly Numbered Elements of a Network." *Comm. of the ACM* 4 (1961): 167-168.

D37. Mayeda, W. "Reducing Computation Time in the Analysis of Networks by Digital Computer." *IEEE Trans. CT* 6 (March 1959): 136-137.

D38. McAndrew, M. H. "The Polynomial of a Directed Graph." *Proc. Amer. Math. Soc.* 16 (1965): 303-309.

D39. Mayeda, W., et al. "Generation of Complete Trees." *IEEE Trans. CT* 15 (June 1968): 101-105.

D40. Meetham, A. R. "Algorithm to Assist in Finding the Complete Subgraph of a Given Graph." *Nature*, July 2, 1966. p. 105.

D41. Myers, B. R., and Auth, L. V., Jr. "The Number and Listing of All Trees in an Arbitrary Graph." *Proc. Third Allerton Conf.*, October 1965. pp. 906-912.

D42. Opler, A. "A Brief Survey of Topological Representations." *Proc. ADI* (1964): 499-502.

D43. O'Neill, P. V., and Slepian, P. "The Number of Trees in a Network." *Proc. Third Allerton Conf.*, October 1965. pp. 900-905.

D44. Ore, O. "Sex in Graphs." *Proc. Amer. Math. Soc.* 7 (1960): 533-539.

D45. Pape, Uwe. "The Transformation and Analysis of Networks by Means of Computer Algorithms." *International J. of Computer Math.* 3 (1968): 75-110.

D46. Parthasarathy, K. R. "Enumeration of Ordinary Graphs With Given Partition." *Can. J. Math.* 20 (1968): 40-47.

D47. Pelikan, J. "Valency Conditions for the Existence of Certain Subgraphs." Erdös (ed.), *Theory of Graphs*, 1968. pp. 251-258.

D48. Ponstein, J. "A Generalization of the Incidence Matrices of a Graph." *International Symposium on Theory of Graphs*, Rome, 1966. pp. 315-332.

D49. Ramamoorthy, C. V. "Generating Functions of Abstract Graphs With Systems Applications." Doctoral Thesis, Harvard Univ., May 1964.

D50. Reed, M. B., and Seshu, S. "On Topology and Network Theory." *Proc. Univ. of Ill. Symp. on Circuit Analysis* (1955). pp. 2-1 to 2-16.

D51. Rescigno, A., and Segre, G. "On Some Topological Properties of the Systems of Compartments." *Bull. Math. Biophysics* 26 (1964): 31-38.

D52. Resh, J. A. "On Networks and Bi-Complete Graphs." Univ. of Illinois, Urbana, Coordinate Science Lab., Report July 1963.

D53. Ross, I. C., and Harary, F. "The Square of a Tree." *Bell System Tech. Journal* 39 (1960): 641-647.

D54. Sabidussi, G. "The Centrality Index of a Graph." *International Symposium on Theory of Graphs*, Rome, 1966.

D55. Shein, N. P., and Frisch, I. T. "Vertex Weighted Trees With Fewest Relay Vertices." *J. SIAM Appl. Math.* 17 (Sept. 1969): 897-903.

D56. Trent, H. M. "Note on the Enumeration and Listing of All Possible Trees in a Connected Linear Graph." *Proc. National Acad. Sci. U. S.* (October 1954). pp. 1004-1007.

D57. Tutte, W. T. "An Algorithm for Determining Whether a Given Binary Matroid is Graphic." *Proc. Amer. Math. Soc.* 11 (1960): 905-917.

D58. Unger, S. H. "GIT--A Heuristic Program for Testing Parts of Directed Line Graphs for Isomorphism." *Comm. ACM* 7 (Jan. 1964): 27.

D59. Warshall, S. "A Theorem on Boolean Matrices." *J. ACM* 9 (1962): 11-12.

D60. Yau, S. S. "A Generalization of the Cut Sets for Application to Communication Nets." Ph.D. Thesis, Department of Electrical Engineering, Univ. of Ill., Urbana, Ill., June 1961.

D61. Zaretskiy, K. A. "Construction of a Tree by the Collection of Distances Between Terminal Vertices." *Uspekhi Matemat. Nauk.* 20 (1965): 90-92.

E. Path-Finding and Optimization Algorithms

E01. Agin, N. "Optimal Seeking with Branch and Bound." *Mgmt Sci.* 13 (Dec. 1966): B176-B185.

E02. Burt, J. M. "All Shortest Distances in Large Serial Networks." Western Mgmt. Sci. Inst., UCLA, July 1970.

E03. Bellman, R. and Kalaba, R. "On the kth Best Policies." *J. SIAM Appl. Math.* 8 (1960): 582-588.

E04. Clarke, S., Krikorian, A. and Rausen, J. "Computing the N Best Loopless Paths in a Network," *J. SIAM Appl. Math.* 11 (1963): 1096-1102.

E05. Danielson, G. H. "On Finding the Simple Paths and Circuits in a Graph." *IEEE Trans. CT* 15 (Sept. 1968): 294-295.

E06. Dantzig, G. B. "All Shortest Routes in a Graph." Operations Research Technical Report 66-3, Stanford University, Nov. 1966.

E07. Dantzig, G. B. "On the Shortest Route Through a Network." Rand Corporation Report P-134, Santa Monica, California, 1959.

E08. Deo, N., and Hakimi, S. L. "Shortest Generalized Hamiltonian Path." *Proc. Third Allerton Conf.* (October 1965), pp. 879-887.

E09. Dijkstra, E. "A Note on Two Problems in Connection with Graphs." *Numer. Math.* 1 (1959): 269-271.

E10. Dreyfus, S. E. "An Appraisal of Some Shortest-Path Algorithms." *Operations Research* 17 (May-June 1969): 395-412.

E11. Farbey, B. A., Land, A. H., and Murchland, J. P. "The Cascade Algorithm for Finding All Shortest Distances in a Directed Graph." *Mgmt. Sci.* 14 (Sept. 1967): 19-28.

E12. Fielder, D. C. "Step by Step Formation of Matrices from Objects in Sequence." *IEEE Trans. Education* 12 (March 1969): 66-69.

E13. Floyd, R. W. "Algorithm 97, Shortest Path." *Comm. ACM* 5 (1962): 345.

E14. Hoffman, W., and Pavley, R. "A Method for the Solution of the Nth Best Path Problem." *J. ACM* 6 (1959): 506-514.

E15. Hu, T. C. "A Decomposition Algorithm for Shortest Paths in a Network." *Operations Research* 16 (1968): 91-102.

E16. Hu, T. C. *Multi-Terminal Shortest Paths.* Operations Research Center, Univ. of Calif. at Berkeley, ORC 65-11 (1965).

E17. Hu, T. C. "Revised Algorithms for Shortest Paths." *J. SIAM Appl. Math.* 15 (1967): 207-218.

E18. Hu, T. C., and Torres, W. T. "Shortcut in the Decomposition Algorithm for Shortest Paths in a Network." *IBM J. of Research & Development* (July 1969): 387-390.

E19. Jarvis, John J. "Optimal Attach and Defense of a Command and Control Communications Network." Ph.D. Dissertation, Johns Hopkins University, Baltimore, Md., 1968.

E20. Kamae, Takahiko. "A Systematic Method for Finding All Directed Circuits and Enumerating All Directed Paths." *IEEE Trans. CT* 14 (June 1967): 166-171.

E21. Klee, V. "String Algorithm for Shortest Paths in Directed Networks." *Operations Research* 12 (1964): 428-432.

E22. Kruskal, J. B., Jr. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem." *Proc. Amer. Math. Soc.* 7 (1956): 48-50.

E23. Lawler, E. L., and Wood, D. E. "Branch and Bound Methods: A Survey." *Operations Research* 14 (1966): 699-719.

E24. Michle, W. "Link-Length Minimization in Networks." *Operations Research* 6 (1958): 232-243.

E25. Mills, G. "A Decomposition Algorithm for the Shortest-Route Problem." *Operations Research* 14 (1966): 279-291.

E26. Moore, E. F. "Shortest Path Through a Maze." *Proc. International Symposium on Switching Circuits*, Harvard Univ. (1959): 285-292.

E27. Murchland, J. D. "A New Method for Finding All Elementary Paths in a Complete Directed Graph." Report LSE-TNT-22, London School of Economics, October 1965.

E28. Murchland, J. D. "An Inductive Matrix Method for Finding All Shortest Paths in a Directed Graph." Report LSE-TNT-25, London School of Economics, March 29, 1966.

E29. Parikh, S. C., and Frisch, I. T. "Finding the Most Reliable Routes in a Communications System." *IEEE Trans. CS* 11 (December 1963): 402-407.

E30. Parthasarathy, K. R. "Enumeration of Paths in Digraphs." *Psychometrika* 29 (1964): 153-165.

E31. Paul, A. J. "Generation of Directed Trees, 2-Trees and Paths Without Duplication." Ph.D. Dissertation, University of Illinois, Urbana, Illinois, 1965.

E32. Peart, R. M., Randolph, P. H., and Bartlett, T. E. "The Shortest Route Problem." *Operations Research* 8 (1960): 866-868.

E33. Peteanu, V. "An Algebra of the Optimum Path in Networks." *Mathematica (Cluj)* 9 (1967): 335-342.

E34. Pollack, M. "Solution of the kth Best Route Through a Network." *J. Math. Anal. and Appl.* 3 (1961): 547-559.

E35. Pollack, M. "The kth Best Route Through a Network." *Operations Research* 9 (1961): 578-580.

E36. Pollack, M., and Wiehenson, W. "Solutions of Shortest-Route Problem--A Review." *Operations Research* 8 (1960): 224-230.

E37. Ponstein, J. "Self Avoiding Paths and the Adjacency Matrix of a Graph." *J. SIAM Appl. Math.* 14 (1966): 600-609.

E38. Prim, R. C. "Shortest Connection Networks and Some Generalizations." *Bell Systems Tech.* 36 (November 1957): 1389-1401.

E39. Raimond, J. F. "Minimaximal Paths in Disjunctive Graphs by Direct Search." *IBM J. Res. and Dev.* (July 1969): 391-399.

E40. Saigal, R. "A Constrained Shortest Path Problem." *Operations Research* 16 (1968): 205-209.

E41. Sakarovitch, M. "The K Shortest Routes and K Shortest Chains in a Graph." Univ. of Calif. at Berkeley, ORC-66-32, Oct. 1966.

E42. Taga, Y., and Suzuki, T. "On the Estimation of Average Length of Chains in a Communicative Pattern." *Amer. Inst. Stat. Math.* 9 (1958): 149-156.

E43. Thorelli, L. E. "An Algorithm for Computing All Paths in a Graph." BIT 6 (1966): 347-349.

F. Flow and Synthesis Considerations in Networks

F01. Ali, A. A. "On the Analysis of Weighted Communications Networks." *IEEE Trans. CT* 16 (May 1969): 223-225.

F02. Ali, A. A. "Synthesis of Communication Nets." Ph.D. Thesis, University of London, 1965.

F03. Ash, R. B., and Kim, W. H. "On the Synthesis of Information Networks." Delivered at the Conference of the Union Radio-Scientifique Internationale, Pennsylvania State Univ., October 1958.

F04. Arinal, J. C. "Maximal Biflow on Undirected Network." *IBM J. of Res. and Dev.* 13 (July 1969): 373-379.

F05. Boldyreff, A. "Determination of the Maximal Steady State Flow of Traffic Through a Railroad Network." *Operations Research* 3 (1955): 443-446.

F06. Bratton, D. *Efficient Communication Networks.* Cowles Commission Discussion Paper in Econ., No. 2119, Feb. 23, 1955.

F07. Cauer, W. *Synthesis of Linear Communication Networks.* New York: McGraw-Hill, 1958. Translated by G. E. Knausenberger and J. W. Warfield.

F08. Chien, R. T. "A Method for Computing Maximum Flows Through a Communications Network." *Proceedings of the Sixth Nat. Symp. on Communication Systems* (1960). pp. 282-285.

F09. Chien, R. T. "Synthesis of a Communications Network." *IBM J. of Res. and Dev.* 4 (1960): 311-320.

F10. Chien, R. T., Gomory, R. E., Hu, T. C. "Communication Networks." Part VII of "Progress in Circuit Theory, 1960-1963," edited by Louis Weinberg. *IEEE Trans. CT* 11 (March 1964): 2-29.

F11. Deo, N., and Hakimi, S. L. "Minimum Cost Increase of the Terminal Capacities of a Communications Network." *IEEE Trans. Comm. Tech.* 14 (Feb. 1966): 63-64.

F12. Elias, P., Feinstein, A., and Shannon, C. E. "A Note on the Maximum Flow Through a Network." *IEEE Trans. Info. Theory* 2 (1956): 117-119.

F13. Ford, L. R., Jr., and Fulkerson, D. R. "A Suggested Computation for Maximal Multi-Commodity Network Flows." *Management Science* 5 (Oct. 1958): 97-101.

F14. Ford, L. R., and Fulkerson, D. R. "Constructing Maximal Dynamic Flows from Static Flows." *Operations Research* 6 (1958): 419-433.

F15. Ford, L. R., and Fulkerson, D. R. "Maximal Flow Through a Network." *Can. J. Math.* 8 (1956): 399-404.

F16. Foster, R. M. "Topological and Algebraic Considerations in Network Synthesis." *Proc. Symp. on Modern Network Synthesis,* Polytechnic Inst. of Brooklyn, 1952. pp. 8-18.

F17. Frank, H. "Dynamic Communications Networks." *IEEE Trans. Comm. Tech.* 15 (Apr. 1967): 156-163.

F18. Frank, H. "Vulnerability of Communications Networks." *IEEE Trans. Comm. Tech.* 15 (Dec. 1967): 778-789.

F19. Frank, H. "Dynamic Communication Networks with Capacity Constraints." *IEEE Trans. Comm. Tech.* 17 (August 1969): 432-437.

F20. Frank, H., and Hakimi, S. L. "On the Optimum Synthesis of Statistical Communication Networks Pseudo Parametric Techniques." *J. Franklin Inst.* 284 (December 1967): 407-416.

F21. Frank, H., and Hakimi, S. L. "Parametric Analysis of Statistical Communication Networks." *Quarterly of Applied Mathematics* 26 (July 1968): 249-263.

F22. Frisch, I. T., and Sen, D. K. "Algorithms to Realize Directed Communication Nets." *IEEE Trans. CT* 14 (Dec. 1967): 370-379.

F23. Frisch, I. T., and Kim, W. H. "Realization of Communication Networks with Maximum Information Flow." *Proc. Seventh National Symposium on Communications Systems,* 1961. pp. 254-261.

F24. Frisch, I. T., and Shein, N. P. "Necessary and Sufficient Conditions for Realizability of Vertex-Weighted Communications Networks." *IEEE International Conference on Communications,* 1968. pp. 682-687.

F25. Fulkerson, D. R., "A Network Flow Feasibility Theorem and Combinatorial Applications." *Can. J. Math.* 11 (1959): 440-451.

F26. Gale, D. "A Theorem on Flows in Networks." *Pacific J. Math.* 7 (1957): 1073-1082.

F27. Gale, D. "Transient Flows in Networks." *Mich. Math. J.* 6 (1958): 59-63.

F28. Gomory, R. E., and Hu, T. C. "An Application of Generalized Linear Programming to Network Flows." *J. SIAM Appl. Math.* 10 (1962): 260-283.

F29. Gomory, R. E., and Hu, T. C. "Multi-Commodity Network Flows." IBM Research Report RC-865, January 16, 1963.

F30. Gomory, R. E., and Hu, T. C. "Multi-Terminal Network Flows." *J. SIAM Appl. Math.* 9 (December 1961): 551-570.

F31. Gomory, R. E. "Synthesis of a Communications Network." *J. SIAM Appl. Math.* 12 (June 1964): 348-369.

F32. Gould, R. "The Application of Graph Theory to the Synthesis of Contract Networks." *Proc. of International Symposium on the Theory of Switching.* Annals of the Comp. Lab. of Harvard Univ., No. 29, 1957.

F33. Hakimi, S. L. "Analysis and Design of Communications Networks With Memory." *J. Franklin Inst.* 287 (1969): 1-17.

F34. Hakimi, S. L. "An Algorithm for the Construction of the Least Vulnerable Comm. Network or the Graph With the Maximum Connectivity." *IEEE Trans. CT* 16 (May 1969): 229-230.

F35. Hakimi, S. L. "Simultaneous Flows Through a Communications Network." *IEEE Trans. CT 9* (June 1962): 169-175.

F36. Hobbs, E. W., and MacWilliams, F. J. "Topological Network Analysis as a Computer Program." *IRE Trans. CT* 6 (March 1959): 135.

F37. Hu, T. C. "Multicommodity Network Flows." *Operations Research* 11 (1963): 344-360.

F38. Hu, T. C. "On the Feasibility of Simultaneous Flows in a Network." *Operations Research* 12 (1964): 359-360.

F39. Hu, T. C. "Recent Advances in Network Flows." *J. SIAM Appl. Math.* 10 (1968): 354-359.

F40. Iri, Masao. "Algebraic and Topological Foundations of the Analysis and Synthesis of Oriented Switching Circuits." *RAAG Memoirs* 2 (1958): 469-518.

F41. Jewell, W. S. "A Primal-Dual Multi-Commodity Flow Algorithm." Operations Research Center Report ORC 66-24, Univ. of California at Berkeley, September 1966.

F42. Jewell, W. S. "Multi-Commodity Network Solutions." Operations Research Center Report ORC 66-23, University of California at Berkeley, September 1966.

F43. Kalaba, R., and Juncosa, M. "Optimal Design and Utilization of Communication Networks." *Management Science* 3 (1956): 33-34.

F44. Kim, W. H., and Chien, R. T. *Topological Analysis and Synthesis of Communications Networks.* New York: Columbia University Press, 1962.

F45. Kleinrock, L. "Some Results on the Design of Communications Nets." *IEEE International Conference of Communications* 1968. pp. 699-705.

F46. Luce, Macy, Christie and Hay. "Information Flow in Task Oriented Groups." MIT Res. Lab. Electronics Tech. Report No. 264, 1953.

F47. Mayeda, W. "Maximum Flow Through a Communications Network." Interim Report No. 13, Univ. of Ill., DA-11-022-ORD-1983 (1960).

F48. Mayeda, W. "Maximum Flow Under Controlled Edge Flows." *IEEE International Conference on Communications* 1968. pp. 694-698.

F49. Mayeda, W. "Synthesis of Switching Functions by Linear Graph Theory." *IBM J. Res. and Dev.* 4 (1960): 320-328.

F50. Mayeda, W. "Terminal and Branch Capacity Matrices of a Communication Net." *IRE Trans. CT* 7 (1960): 260-269.

F51. Okada, S. "Algebraic and Topological Foundations of Network Synthesis." *Proc. Symp. on Modern Network Synthesis.* Polytechnic Inst. of Brooklyn, 1955. pp. 283-322.

F52. Onaga, K. "Optimum Flows in General Communications Networks." *J. Franklin Inst.* 283 (April 1967): 308-327.

F53. Parker, S. R., and Lohse, H. J. "A Direct Procedure for the Synthesis of Network Graphs From a Given Fundamental Loop or Cut-Set Matrix." *IEEE Trans. CT* 16 (May 1969): 221.

F54. Resh, J. A. "On the Synthesis of Oriented Communication Nets." *IEEE Trans. CT* 12 (December 1965): 540-546.

F55. Roboeker, J. T. *Concerning Multi-Commodity Flows.* Rand Corporation Report RM-1799, 1956.

F56. Saigal, R. *Multicommodity Flows in Directed Networks.* Operations Research Center Report ORC 67-38, Univ. of Calif. at Berkeley, September 1967.

F57. Sakarovitch, M. "The Multicommodity Maximum Flow Problem." Operations Research Center ORC 66-25, Univ. of Calif., Sept. 1966.

F58. Sen, D. K., and Frisch, I. T. "Synthesis of Oriented Communication Nets." *Proc. IEEE Symp. on Signal Transmission and Processing* (Columbia Univ., New York, 1965). pp. 90-101.

F59. Tang, D. T. "Bi-Path Networks and Multi-Commodity Flows." *IEEE Trans. CT* 11 (Dec. 1964): 468-474.

F60. Tang, D. T. "Communication Networks With Simultaneous Flow Requirements." *IEEE Trans. CT* 9 (1962): 176-182.

F61. Tang, D. T. "On Flows in Communications Network." *Proc. Second Allerton Conf.*, Sept. 1964. pp. 83-97.

F62. Tang, D. T. "Optimal Trees for Simultaneous Flow Requirements." *Proc. Nat. Elec. Conf.* 19 (Oct. 1963): 28-32.

F63. Tang, D. T. and Chien, R. T. "Analysis and Synthesis Techniques of Oriented Communication Nets." *IEEE Trans. CT* 7 (1961): 39-43.

F64. Tomlin, J. A. "Minimum Cost Multicommodity Network Flows." *Operations Research* 14 (1966): 45-51.

F65. Tuller, W. G., and Cheatham, T. P., Jr. "Communication Theory and Network Synthesis." *Symp. on Information Networks*, Polytechnic Inst. of Brooklyn, April 1954.

F66. Tuy, H. "Some Theorems on Network Flows" in *Theory of Graphs*. Proceedings of the Colloquium held at Tihany, Hungary, Sept. 1966. pp. 173-184.

F67. Wing, O. "Minimal Realization of a Communications Network Under Uniform Cost." IBM Res. Rpt. ES 0026, August 1960.

F68. Wing, O., and Chien, R. T. "Optimal Synthesis of a Communications Net." *IEEE Trans. CT* 7 (1961): 44-49.

VITA

Robert Martin Siegmann was born on June 2, 1936, in Charleston, South Carolina. He received the degree of B.S. (Mathematics) from the College of Charleston in 1958, the degree of M.S. (Mathematics) from the University of South Carolina in 1960, and the degree of M. S. (Information Science) from the Georgia Institute of Technology in 1968.

He was employed by the International Business Machines Corporation from 1960 to 1966 and has been on educational leave from IBM since 1966. He has held part-time positions teaching at the University of South Carolina, American University and Georgia Tech. He is currently employed as a Research Associate at Georgia Tech.