

7/25/68

MIN-MAX PATH FLOW
IN DIRECTED NETWORKS

A THESIS

Presented to

The Faculty of the Division of Graduate
Studies and Research

by

Robert Glenn Hinkle

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

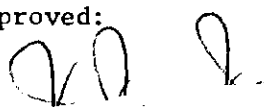
in the School of Industrial and Systems Engineering

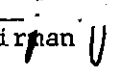

Georgia Institute of Technology

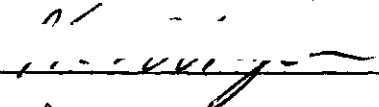
June, 1971

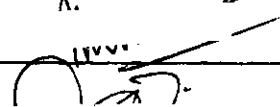
MIN-MAX PATH FLOW
IN DIRECTED NETWORKS


Approved:



Chairman  







Date approved by Chairman: May 31, 1971

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to Dr. J. J. Jarvis who served as my advisor throughout this research and as chairman of my reading committee. He provided the guidance, criticism, encouragement and friendship which was necessary for the successful completion of this work.

I am also grateful to the other members of the reading committee, Dr. D. C. Fielder, Dr. C. M. Shetty and Dr. V. E. Unger for their helpful suggestions. I am especially grateful to Dr. Shetty for several critical suggestions which have improved this dissertation. Dr. D. R. Fulkerson read a preliminary draft and offered some valuable suggestions.

I could never have reached this point in my education without the help of many professors in my graduate studies. I would especially like to acknowledge the guidance and encouragement given me by Dr. L. A. Johnson and Dr. H. E. Smalley.

The idea for this research arose from an investigation of the applicability of network flow theory to a specific strategic transportation problem. This investigation was being conducted by the author and Dr. Jarvis for the Naval Weapons Laboratory (NWL), Dahlgren, Virginia. My studies were financed through the Full-Time Advanced Study Program of the Naval Weapons Laboratory, which has continued to support me throughout this research. And for this support I express my appreciation to NWL management, and particularly to my supervisor, Mr. O. F. Braxton, and to Mr. Ralph Niemann, Head of the Warfare Analysis Department at the time I entered the Full-Time Advanced Study Program.

I wish to thank Mrs. Dorothy Elam and Mrs. Janice Sullivan for the excellent job they did on the typing while meeting very tight deadlines. I also thank Mr. John Winder for the excellent drawings and Mr. F. L. Jones for helping to arrange the typing and printing of this dissertation. I appreciate the extra effort put forth by Mrs. Dorothy Elam in helping to eliminate errors in the typing and format.

My deepest debt of gratitude must be to my wife, Doris, and my two children, Kimberly and Kevin. They have sacrificed much while I have been engrossed in this effort.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
LIST OF TABLES	vi
LIST OF ILLUSTRATIONS	vii
Chapter	
I. INTRODUCTION	1
Statement of the Problem	
Example	
Graphs and Networks	
Network Flow Problems	
Objectives	
Literature Survey	
Results	
II. CHARACTERIZATION OF THE MIN-MAX PATH FLOW PROBLEM	23
Formulation of the Min-Max Path Flow Problem	
Comparison With Other Problems	
Fractional Solutions	
Approaches to Algorithm Development	
III. A PRIMAL ALGORITHM FOR SOLVING THE MIN-MAX PATH FLOW PROBLEM	38
Reducing Flow on the Longest Path	
The Constrained Shortest Path Problem	
Example	
The Path Removal Algorithm	
Convergence	
IV. AN EXPANDED NETWORK FORMULATION OF THE MIN-MAX PATH FLOW PROBLEM	84
A Dynamically Expanded Network	
The Relationship Between $G(N,A)$ and $D_L(G)$	
Solution of the Bundle-Constrained Maximum Flow Problem	
The Expanded Network Algorithm Convergence	

	Page
Improving Computational Efficiency Algorithm for Generating Dynamically Expanded Network	
V. CONCLUSIONS AND RECOMMENDATIONS	139
Other Applications Extensions and Future Research Results and Conclusions	
APPENDIX	157
BIBLIOGRAPHY	160
VITA	163

LIST OF TABLES

Table		Page
1.	Path Enumeration	25
2.	Minimum Cost Solution	27
3.	Min-Max Path Solution	27
4.	Min-Max Path Flow Solution	31
5.	Constrained Shortest Path Labels	80
6.	Simplex Tableau for Bundle-Constrained Maximum Flow Problem	110
7.	Column Seven of B^{-1}	147
8.	Simplex Basis	149
9.	Basis Inverse	150

LIST OF ILLUSTRATIONS

Figure	Page
1. Transportation Network	2
2. Graph of an Assignment Problem	6
3. Network Model of an Assignment Problem	9
4. Example Network	26
5. Network With Fractional Min-Max Flows	31
6. Network With Covered Flows	34
7. Minimum Cost Flow	73
8. Min-Max Path Flow	77
9. Network With Arc Lengths and Costs	79
10. Six-Period Expanded Network	85
11. $D_1(G)$; 11-Period Expansion of $G(N,A)$	86
12. Network With Bundle Constraints	94
13. Arc Identification for Example Network	107
14. Reduced Network	109
15. Network With Modified Cost	112
16. Network With all Shortest Path Labels	120
17. Network With Labels up to Period 11	121
18. Static Network	137
19. Six-Period Expansion $G(N,A)$	137
20. Computation of B^{-1} on the Network	145
21. Network With Four Basic Paths	148

SUMMARY

The objective of this dissertation is to report the results of research on the min-max path flow problem. The min-max path flow problem is a variation of the well known maximal flow problem. The variation results from the addition of a secondary objective function which is to minimize, over all flows which produce the required total flow, the length of the longest path which carries flow. The secondary objective function associates with a flow, the length of the longest path from source to sink in the network which carries flow.

The research had two specific objectives. The first was to characterize the min-max path flow problem in terms compatible with other network flow problems by characterizing its solution and by showing its relationship to other network flow problems. The second objective was to develop a solution algorithm for the min-max path flow problem.

The results of this research are summarized as follows: The min-max path flow problem is shown to be a generalization of the bottleneck assignment problem and the time-minimizing transportation problem. Both of these problems have all all-integer extreme points. Unlike these problems and unlike other single-commodity maximum flow problems it is shown that the min-max path flow problem does not in general have all all-integer extreme points.

A comparison is made between the min-max path problem and other network flow problems. The structure of the min-max path problem is

closer to that of multicommodity networks than to single commodity networks.

The most natural formulation of the problem results in a mixed-integer programming problem. The problem is to optimize a min-max objective function over all maximum path flows on the network.

Two algorithms were developed to solve this problem. The first algorithm solves the integer programming problem by solving a sequence of linear programming problems. Successive problems differ only in the coefficients in the objective function. Thus, throughout the computations, the algorithm searches over the set of extreme points of the same convex polytope. A procedure for moving from one extreme point to another is developed which consists of solving a constrained shortest path problem on the network. An algorithm for solving the constrained shortest path is presented. It is shown that the amount of computation required to solve the constrained shortest path problem is of the same order of magnitude as that required to determine the shortest path from s to all nodes in a network. That is, one solution of the constrained shortest path problem requires on the order of n^3 additions and comparisons, where n is the number of nodes in the network. The algorithm utilizes a maximum flow algorithm to find an initial feasible solution to the problem. The flow is decomposed into path flows and the longest paths are identified. A constrained shortest path problem is then solved to identify a non-basic path which would, upon introduction into the basis, tend to reduce the net flow on the set of longest paths currently in the basis.

The second algorithm makes use of an expanded version of the network in which only paths of length less than an arbitrary length are represented. The min-max path flow problem is shown to be equivalent to a bundle constrained maximum flow on the expanded network for some value of L . The length of the min-max path is then L .

This bundle constrained maximum flow problem is solved by the use of a decomposition procedure in which there are no more, and quite likely fewer, master constraints than arcs in the original network. The single subproblem is a minimum cost maximum flow problem.

The first algorithm is referred to as the path removal algorithm. It has possible application in other network flow problems. It can be used, for example, to solve a multiterminal maximal flow problem which has one or two source-sink pairs which are inadmissible.

The procedure in the expanded network algorithm for implicitly representing all paths of length L or less by the expanded network may be useful in other network problems which restrict the solution to paths of length L or less.

The expanded network algorithm has application to maximal dynamic flows with total arc capacities in addition to the normal arc flow rates.

CHAPTER I

INTRODUCTION

The objective of this dissertation is to report the results of research on the min-max path flow problem. The min-max path flow problem is a variation of the well known maximal flow problem.

Stated briefly, the problem is to determine the required flow in a capacited network for which the maximum cost of any unit of flow from the source to the sink is minimized.

Before presenting the mathematical concepts of networks and network flows on which this research builds, we will attempt to give an intuitive motivation of the min-max path flow problem by way of example.

Example

A physical interpretation of the min-max path flow concept is provided by the example transportation problem depicted in Figure 1. Suppose there are trucks located at city s , the source node, and it is desired to transport the trucks to city t , the sink node, via cities x , y , and z , called transshipment nodes. The arcs between the nodes (cities) represent transportation links that are available and the arc numbers represent, respectively, the arc capacity (trucks/unit time) and the time required to transverse the arc. There are, in this transportation network, several routes that a given truck can take. For example, it could go from s to y , and from y to t ; or it

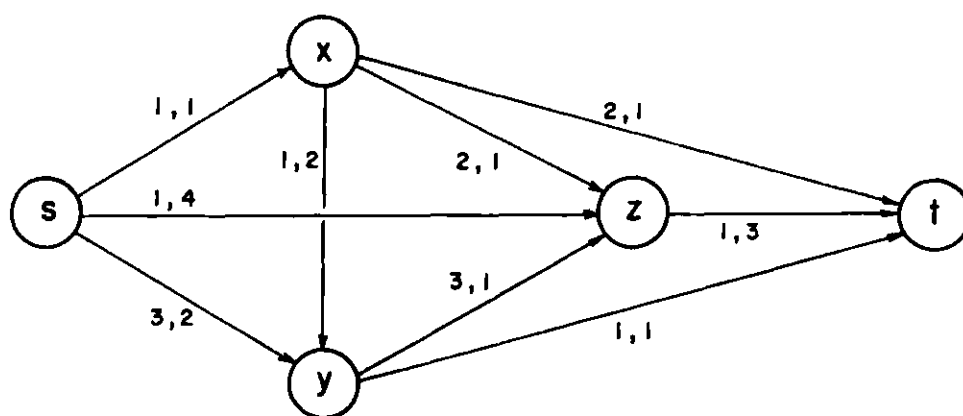


Figure 1. Transportation Network

could go from s to x , from x to y , from y to z , and z to t . The problem then, is that of selecting the set of routes to be used in transporting the trucks from s to t so that the required number of trucks arriving at t per unit time (flow) is achieved and the amount of time it takes all trucks to get from s to t is minimized. Henceforth, this problem will be referred to as the min-max path flow problem. We do not lose any generality by assuming we want the maximum flow in the network since we can always place an availability restriction at the source.

After presenting the basic concepts in network flow theory and formulating some of the important models, we will be in a better position to discuss the specific objectives and the scope of this research. Generally speaking, our objective has been to investigate the structure of the min-max path flow problem, characterize it in terms compatible with other results in network flow theory, and develop a solution algorithm.

Graphs and Networks

It is convenient to begin a discussion of network flow theory by first discussing the more general basic concepts of graph theory. There are several good references for more complete discussions of these topics. See, for example, Busacker and Saaty (5) and Berge (2) for a discussion of graph theory and its applications. The classic text in network flow theory is that of Ford and Fulkerson (12). The books by Hu (23) and by Berge and Gouila-Houri (3) also provide extensive treatment of graphs and flow networks. Elmaghraby (10) and (11) provides a good discussion of the applications of networks to management science. We follow the notation of Ford and Fulkerson.

Definition

A graph is a set of elements N and a relation A , where

$$A \subseteq N \times N .$$

Symbolically, we will use the notation (N,A) to represent the graph.

In other words, a graph is a set of objects and a relationship between the objects. A network is called a finite network if N and A are finite sets.

Notation

Nodes will be represented by letters x and y or by an index such as i or j . The source node will sometimes be denoted by s and the sink node by t .

Arcs will be identified by their end-points, (x,y) or by an index i or j . Whether an index i refers to an arc or a node will be clear from the context. Arc (x,y) is said to be incident to nodes x and y . The initial node of arc (x,y) is x and the terminal node is y .

As an example of a situation which can be model by a graph let us consider the personnel assignment problem. Suppose we have m men and m jobs and we wish to assign each man to exactly one job and no more than one man to any job. We can let the men and the jobs be nodes. If we let $\{p_1, \dots, p_m\}$ represent the set of men and $\{r_1, \dots, r_m\}$ be the set of jobs, then the set of nodes N is given by

$$N = \{p_1, \dots, p_m, r_1, \dots, r_m\} ,$$

and the set of arcs (the relation) is given by

$$A = \{(p_i, r_k) \mid \text{man } p_i \text{ can be assigned to job } r_k\}$$

The "assignment" problem can be solved by use of this model. A solution is obtained by selecting a subset $A_1 \subseteq A$ which satisfies the condition that A_1 is a one-to-one mapping of N_1 onto N_2 , where

$$N_1 = \{p_1, \dots, p_m\}$$

$$N_2 = \{r_1, \dots, r_m\} .$$

This model is obviously not complete since we have specified no criteria for selecting one from among the many possible solutions to this problem. Figure 2 is a graphical model of a specific assignment problem. The arcs represent possible assignments; the heavy lines represent a solution.

Definition

A flow network is a graph (N,A) in which the arcs represent the possibility of flow of some commodity between the two nodes which are the end-points of the arc. The notation $G(N,A)$ designates the flow network which consists of the graph (N,A) ; arc numbers which represent the capacity of the arc; any special requirements for origin or termination of flow; and any cost associated with flow on specific arcs.

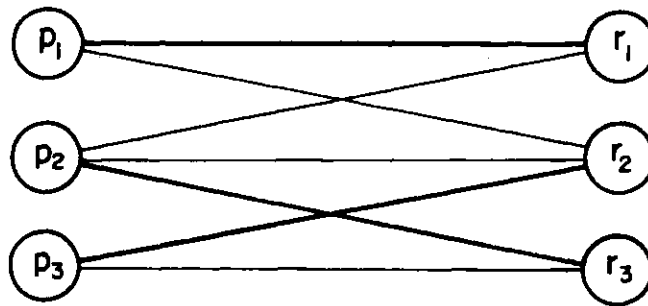


Figure 2. Graph of an Assignment Problem

Notation

We shall use $a(x,y)$ or $a(i)$ to denote the length of, or the unit cost of flow on arc (x,y) or arc i , respectively.

The expression $b(x,y)$ or $b(i)$ will likewise represent the capacity of arc (x,y) or arc i .

The variable $f(x,y)$ or $f(i)$ will be the amount of flow assigned to arc (x,y) or i . $\underline{F} = \underline{F}(A)$ will denote the vector of flows on the arcs of A .

Definition

A flow network is called a directed network if the arcs are orientated and net flow is allowed only in the specified direction.

If the arcs of a network are not directed then the network is called an undirected network. Throughout this dissertation we will be dealing only with directed networks.

Generally speaking, in a flow network, certain nodes will be sources of the commodity in question and other nodes will be termination points or sinks. Nodes which are neither sources nor sinks will be called transshipment nodes. Any assignment of flow in the network must respect the conservation of flow requirement at transshipment nodes. This condition states that the sum of flow into the node must equal the flow out. Without loss of generality, we can assume that the network has a single source and a single sink (12).

The assignment problem modeled as a graph in Figure 2 can also be modeled as a flow network. A flow of one unit on arc (p_i, r_k) indicates that the i^{th} man is assigned to job r_k . The capacity of each arc is one unit. If we can assign an efficiency index for each

man to each job, then we can let that index be a cost associated with the arc joining the corresponding nodes. Our objective then in selecting a complete assignment of men to jobs may be to maximize the total efficiency of the assignment.

We can expand the network to incorporate the constraint that each man can be assigned to only one job and no more than one man can be assigned to a single job. This is done by adding a fictitious source node s and arcs (s, p_i) for $i = 1, \dots, m$. Each arc (s, p_i) has a capacity of one and a cost of zero. Add also, a fictitious sink t and arcs (r_k, t) , $k = 1, \dots, m$. Each arc is given a capacity of one unit and a cost of zero. This network is depicted in Figure 3.

We can thus solve the assignment problem by finding a flow through the network of Figure 3 from s to t . We can interpret the objective of maximizing the total efficiency of all assignments by letting the arc cost $a(x, y)$ be one minus the efficiency of the corresponding assignment and finding the flow for which the sum of unit costs multiplied by arc flows is minimum.

Paths, Chains and Cycles

A chain between two nodes x and x_N in a graph (N, A) is a sequence of connected arcs and the incident nodes, $x_1, (x_1, x_2), x_2, \dots, x_{N-1}, (x_{N-1}, x_N), x_N$. If the arcs are all directed toward the higher index node, then the chain is called a path. If $x_1 \equiv x_N$, then the path (chain) is called a directed cycle (cycle). A path (chain) may contain a sub-path (chain) which is a directed-cycle (cycle). And if the path (chain) contains no cycles, then it is called a simple

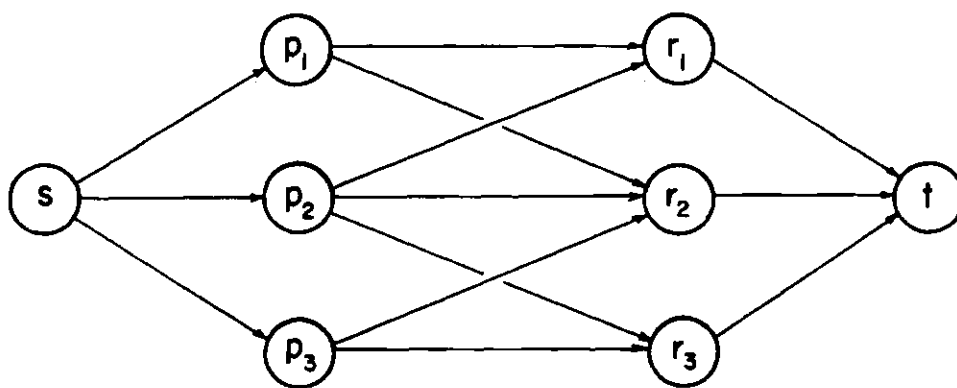


Figure 3. Network Model of an Assignment Problem

path (chain). A negative directed cycle is one, the sum of whose arc costs is negative.

In this dissertation we shall always use the term path but shall always be referring to simple paths from the source node s to the sink node t .

A network flow given as flows on arcs can be expressed as flows on paths from s to t . For a proof of this, see (11). In the network of Figure 3 each path is of the form:

$$s, (s, p_1), p_1, (p_1, r_k), r_k, (r_k, t), t .$$

A flow on the network can be expressed as a flow on paths of this type. The maximum flow assignable is the capacity of the smallest arc in the path. Thus the maximum flow which can be assigned any path in the given network is one. A flow of one indicates that man p_1 is assigned to job r_k .

Notation

If $G(N,A)$ is a directed network, then $\bar{C}(N,A)$ or \bar{C} will denote the set of simple paths in $G(N,A)$ from s to t . The individual paths will be denoted by an index C_j indicating the j^{th} path in the set \bar{C} ; or they will be identified by their arc-path incidence vector \underline{P}_j .^{*}

^{*}Throughout this dissertation, if indexed letters indicate variables or constants, such as P_{ij} , $a(i)$, or $f(y)$, then \underline{P}_j , \underline{a} , or \underline{F} will be used to denote the vector of variables or constants.

$$P_{ij} = \begin{cases} 1 & \text{if arc } i \text{ is in path } C_j \\ 0 & \text{otherwise} \end{cases}$$

We will use $h(C_j)$ or $h(P_j)$ to denote flow on path C_j . \underline{H} will be the vector of flows on all paths of $G(N,A)$.

The length of a path (or cost) is the sum of the lengths of all arcs in the path. Let C_i denote a path and C_i^A denote the set of arcs in path C_i , then if $\ell(C_i)$ denotes the length of path C_i ,

$$\ell(C_i) = \sum_{i \in C_i^A} a(i) = \underline{a}' \underline{P}_i$$

The length of a path in Figure 3 is defined as one minus the efficiency of the assignment represented in the path. If we refer to this as the inefficiency index of a particular man for a particular job, then a possible criterion for selecting an assignment of men to jobs is to select the assignment for which the most inefficient specific assignment is minimized. The problem, with this objective is the min-max path flow problem. This special case of the min-max path problem is called the bottleneck assignment problem; it has been solved by Gross (20). The simplicity of this special case of the min-max path problem is due to the fact that the graph is essentially a bipartite graph (if the fictitious source and sink nodes are dropped).*

*A directed bipartite graph is one in which the nodes can be divided into two disjoint sets, one with arcs only leaving and one with arcs only entering.

Network Flow Problems

In this section we discuss three network flow problems which are related to the min-max path flow problem. The problems are the single commodity maximal flow problems, the general minimal cost maximum flow problem, and the multicommodity flow problem.

Some additional notation is required.

Notation

Let $G(N,A)$ be a network and let F be a function defined on A .

Then

$$f(N,y) = \sum_{x \in N} f(x,y)$$

$$f(x,n) = \sum_{y \in N} f(x,y)$$

$$A(x) = \{y \mid y \in N, (x,y) \in A\}$$

$$B(y) = \{x \mid x \in N, (x,y) \in A\}$$

For any arc $i \in A$, we let $I(i)$ denote the initial node of arc i and $T(i)$ denote the terminal node of arc i .

Definition

If $G(N,A)$ is a directed network with single source s and single sink t , then $\bar{D} \subseteq A$ is a disconnecting set in $G(N,A)$ with respect to s and t , if there exists no path from s to t in $G(N,A - \bar{D})$, where $A - \bar{D}$ is the set of elements in A but not in \bar{D} .

Definition

A cut-set $D \subseteq A$, with respect to s and t , is a minimal disconnecting set of $G(N,A)$. That is, it is a disconnecting set, no proper subset of which is a disconnecting set.

Definition

A minimum cut-set of $G(N,A)$ is a cut-set, the sum of whose arc capacities is a minimum over all cut-sets.

Definition

An L -disconnecting set, \bar{D}^L , is a subset of A whose removal from A disconnects s from t along paths of length L or less. Minimum disconnecting sets and minimum L -disconnecting sets are analogous to minimum cut-sets.

A key theorem in single-commodity network theory is that the maximum flow is equal to the capacity of the minimum cut-set.

Network Flow

Let $G(N,A)$ be a network with source s and sink t . A node-arc flow on $G(N,A)$ is a nonnegative real-valued function defined on A which satisfies the condition:

$$f(x,N) - f(N,x) = \begin{cases} v, & x = s \\ 0, & x \neq s, t \\ v, & x = t \end{cases}$$

$$f(x,y) \leq b(x,y), (x,y) \in A$$

The variable v is the net amount of flow passing through the network.

An arc-path flow on $G(N,A)$ is a nonnegative real-valued function defined on \bar{C} which satisfies the conditions:

$$\sum_j h(C_j) P_{1,j} \leq b(i), i \in A$$

There are two flow problems in network flow theory which appear to be closely related to the min-max path flow problem. These two problems, the maximum flow problem and the general minimal cost flow problem, will be formulated and then a formulation of the min-max path flow problem will be presented.

If $G(N,A)$ is a network with source s and sink t , the maximum flow problem can be written as:

$$\begin{aligned} &\text{maximize: } v \\ &\text{subject to: } f(x,N) - f(N,x) = \begin{cases} v, & x = s \\ 0, & x \neq s, t \\ v, & x = t \end{cases} \quad (1) \end{aligned}$$

$$0 \leq f(x,y) \leq b(x,y), (x,y) \in A$$

The maximum value of $v = v_{\max}$ is of course unique, but there may be more than one flow function f defined on A which yields this value. If there is associated with each arc, a cost, $a(x,y)$, per unit of flow on the arc, then it may be desirable to determine that flow function for which the total flow value v is maximum and the total cost is a minimum.

Let $v = v_{\max}$ be the value of the maximum flow. Find F which will

$$\text{minimize: } \sum_{(x,y) \in A} a(x,y)f(x,y) \quad (2)$$

$$\text{subject to: } f(x,N) - f(N,x) = \begin{cases} v_{n \ a \ x}, & \text{if } x = s \\ 0, & \text{if } x \neq s, t \\ -v_{m \ a \ x}^m, & \text{if } x = t \end{cases}$$

$$0 \leq f(x,y) \leq b(x,y), \quad (x,y) \in A.$$

The solution to this problem, then is the minimum cost maximum flow for the network. This problem is known in the literature as the general minimal cost flow problem.

There is another class of network flow problems with which it will be of value to compare the min-max path flow problem. That problem is the multicommodity network problem.

The multicommodity network is characterized by pairs of nodes (s_i, t_i) , such that flows originating at s_i must terminate at t_i , $i = 1, \dots, n$. Let $I_n = \{1, 2, \dots, n\}$.

If we let $f(i : x, y)$ denote the flow of commodity i on arc (x, y) , then a multicommodity flow function on $G(N, A)$ is a real-valued function defined on $I_n \times A$ which satisfies:

$$f(i : x, y) \geq 0, \quad i \in I_n, (x, y) \in A \quad (3)$$

$$\sum_{I_n} f(i : x, y) \leq b(x, y), \quad (x, y) \in A$$

$$f(i : x, N) - f(i : N, x) = \begin{cases} v_{i, \max}, & \text{if } x = s_i \\ 0, & \text{if } x \neq s_i, t_i \\ -v_{i, \max}, & \text{if } x = t_i, i \in I_n \end{cases}$$

Objectives

The objective of this research was to investigate and characterize as completely as possible the min-max path problem and to develop a computationally feasible algorithm to solve this problem. Since the min-max path flow problem will be formulated as a mixed-integer programming problem, it would be of interest to investigate the development of a solution algorithm based on the theory and methods of integer programming. Specifically, it might be useful to consider the approaches to solving the fixed charge problem or other branch and bound approaches. In this research, however, we are restricting our attention to a consideration of the problem as a network flow problem. The ideal result would be to develop a formulation of the problem that would allow the solution to be performed on the network as in the case with the maximum flow and minimum cost maximum flow problem. An intermediate level of efficiency would be obtained if the problem could be formulated as a linear programming problem which would require a simplex-based solution procedure.

This research is limited to a consideration of single-commodity directed networks. The networks are assumed to be finite, and the arc lengths and capacities are assumed to be nonnegative integers. This

could be relaxed to allow rational capacities and arc lengths; and negative arc lengths could be allowed if we require that there be no negative directed cycles.

Literature Survey

While the literature on network flow theory is quite extensive, no specific mention of the min-max path problem has been found. As mentioned earlier, Gross (20) has developed an algorithm for solving the problem in the special case that the network is bipartite and all arc capacities are unity.

Hammer (22) has developed a procedure for solving the analogous problem for a standard transportation situation. The mathematical formulation of this problem is:

$$\underset{F}{\text{minimize:}} \quad t_0$$

$$\text{subject to:} \quad \sum_{i=1}^N f_{i,j} = a_i, \quad i = 1, \dots, m$$

$$\sum_{i=1}^N f_{i,j} = b_j, \quad j = 1, \dots, n$$

$$t_0 \geq t_{i,j}, \quad \forall t_{i,j} \text{ such that } f_{i,j} > 0$$

$$f_{i,j} \geq 0.$$

a_i represents the availability at warehouse i , b_j is the demand at destination j , and $f_{i,j}$ is the amount of the resource that is to be

shipped from warehouse i to destination j . The constant t_{ij} denotes the time required to ship any amount from i to j .

The solution computations for both of these problems are carried out on a modified form of the standard assignment and transportation tableau, respectively.

No way of extending the solution procedure for these two problems to solve the min-max path flow problem is apparent.

Motivation and Relevance of This Research

In addition to the two problems mentioned earlier, the need for a solution to this problem arises in several network flow problems, especially those which represent scheduling problems. However, the original problem which motivated this proposed research will be described first and other potential applications will then be discussed.

Strategic Transportation Problem

Suppose there are located at M sources, various amounts of a product. This cargo must be moved to intermediate loading points, loaded on carriers and then moved to some third point. The source might be inland marine bases, the cargo might be troops and equipment, the loading points could be ports, the carriers could be ships, and the sink could be some overseas port or beach. The problem, then, is to assign the cargo to loading points, assign the carriers to the loading points and schedule them through the loading points to on-load their cargo, and allow them to proceed to the objective area. The criterion for selecting an assignment and schedule is the minimum overall completion time.

The loading time of the carriers is taken as the unit of time (all have the same loading time). Port capacities are specified as the number of carriers that can be loaded per unit of time. It is assumed that all carriers have the same cargo capacity and the unit of cargo volume is taken as this capacity. It is arbitrarily assumed here that there are sufficient carriers to meet the demand for cargo capacity and the total amount of cargo at the bases must all be assigned to some port. Each ship will be assigned to only one port.

At the time of an operation, the ships will proceed to their assigned ports and move into a loading berth in assigned sequence. Once each ship gains access to a loading berth it will take on cargo that has arrived at the port. Obviously the critical portion of this operation is the sequence in which the ships assigned to a given port are allowed to on-load cargo. Hence the decision variables in this problem are the ports to which the ships are assigned and the sequence in which the ships load at the port. The uncontrollable variables whose values will determine the best values of the decision variables are the time required for each ship to reach each of the ports, the time required for each ship to reach the objective area from each of the ports, the capacity of each port, and the time required to move the cargo from each base to each port and the amount of cargo at each base.

In most cases, the availability of cargo at the ports is not the problem. If we disregard that part of the problem, then the remaining part is a min-max path flow problem.

This problem arises quite frequently in strategic transportation problems.

A Distribution System for Perishable Goods

The min-max path problem has application in certain transportation problems such as the one discussed here.

Suppose a distributor of perishable goods has a set of main distribution centers, a set of regional distribution centers which may also serve as retail outlets, and a set of local retail outlets. The distribution system can be represented by a network in which the distribution centers and retail outlets are nodes and the transportation links between the points are represented by arcs. The objective is to satisfy all demands while minimizing the delivery time of last unit delivered.

We provide one final example to show the wide-ranging field of problems to which this work applies.

A Communications Network

Consider a communication network such as is described in (25). The network consists of a set of source nodes and a set of sink nodes. The communication link between each pair of nodes has a certain capacity in terms of the number of simultaneous messages that can be transmitted over the link represented by the arc between the nodes.

If, as considered by Jarvis (25), the reliability of a message transmission decreases as the number of links over which the message travels increases, then it would be of interest to determine the routing of messages, that is the flow, so that the longest communication chain is minimized. In other words, in a situation in which the network would be saturated, that is maximum flow is desired, how should the message

be routed so that the maximum flow is achieved and the reliability of the least reliable message is maximized, i.e., the number of arcs in the longest path is a minimum.

Another problem would be to let the arc cost represent the time required for a message to transverse the arc and then determine the flow such that the time required for the last message to reach its destination is minimized.

Results

In Chapter II a network based mathematical formulation of the min-max path flow problem is given. The structure of the problem is examined in comparison with the maximum flow and minimum cost maximum flow problems and multicommodity flow problem.

The most important result, which helps set the computational structure in perspective, is the fact that the min-max path problem may not always have an all-integer optimal solution. In this regard, the problem is in the same class as the multicommodity flow problems as contrasted with the single commodity minimum cost maximum flow problem.

In Chapters III and IV two distinct approaches to the problem are formulated. In Chapter III a primal-type algorithm is developed. The algorithm solves the maximum flow problem and then uses a constrained shortest path problem to generate paths which tend to drive the longest path from the basis. A maximum flow algorithm is used to obtain the initial basic feasible solution. The solution proceeds from there in a revised simplex mode on the arc-path formulation of the problem. The dynamic programming algorithm used to solve the constrained shortest path is as efficient as algorithms for computing

shortest paths between all pairs of nodes of a network.

In Chapter IV a dual type algorithm is presented. The algorithm solves the min-max path problem by solving a sequence of bundle constrained maximum flow problems on a sequence of modified networks until the maximum flow equals the required flow on the original network. The definition of bundle-constrained maximum flow problems is delayed until Chapter IV.

CHAPTER II

CHARACTERIZATION OF THE MIN-MAX PATH FLOW PROBLEM

In Chapter I we discussed the maximum flow problem and related network flow problems.

In the maximal flow problem, there may be many flow functions which will produce the optimal flow into the sink. Any one of these is an acceptable solution. In the general minimal cost flow problem, the objective is to select from among the flow functions which achieve the maximal total flow, the one which has the minimum total cost. In the min-max path problem which will now be formulated, the objective is to select from among those flows which produce the maximum flow, that function for which the most expensive path with positive flow has minimum cost (length).

After deriving the mathematical formulation of the min-max path flow problem we investigate the structure of the problem and compare it with other network flow problems. This provides some insight into the general level of computational efficiency we can expect to achieve in an algorithm for solving the min-max path problem as compared with other network flow algorithms.

Several approaches to develop an algorithm were investigated before it was decided to develop completely two algorithms, one a primal type algorithm and the other a dual type algorithm. These

approaches are discussed briefly in this chapter. We conclude the chapter by briefly describing the two algorithms that are developed in Chapters III and IV.

Formulation of the Min-Max Path Problem

Let $P = (P_{ij})$ be the arc-path incidence matrix for the network $G(N, A)$

$$P_{ij} = \begin{cases} 1 & \text{if arc } i \text{ is in path } C_j \\ 0 & \text{otherwise,} \end{cases}$$

The problem then is to determine the flow \underline{H} which will

minimize: L^0

subject to: $\sum_j P_{ij} h(C_j) \leq b(i), \quad i \in A$

$$h(C_j) \geq 0, \quad C_j \in \overline{C}$$

$$\sum_j h(C_j) \geq v_{max}$$

$$L^0 \geq a' \underline{p}_j \delta_j$$

$$M\delta_j \geq h(C_j)$$

$$\delta_j = 0, 1.$$

The objective is to be minimized over all \underline{H} which yield the maximum flow through the network. M is selected greater than v_{max} .

Comparison with Other Problems

In a previous section, three network problems were discussed, the maximum flow problem, the general minimal cost flow problem and the min-max path problem.

Figure 4 is an example network. Table 1 is an enumeration of all paths in the network from the source to the sink. For purposes of comparison we give the minimum cost maximal flow in arc-path form in Table 2 and the min-max path flow solution in Table 3.

The maximum flow in the network is two units. The minimum cost solution has a total cost of 18 and the longest path in this solution is path C_2 whose length is 12. Table 3 gives the min-max path solution.

Table 1, Path Enumeration

Identification	Sequence of Nodes	Length
C_1	s, 1, 4, 5, t	6
C_2	s, 3, 6, t	12
C_3	s, 1, 2, 5, t	9
C_4	s, 3, 4, 5, 6, t	11
C_{5y}	s, 1, 2, 5, 6, t	11
C_6	s, 1, 4, 5, 6, t	8
C_7	s, 3, 4, 5, t	9

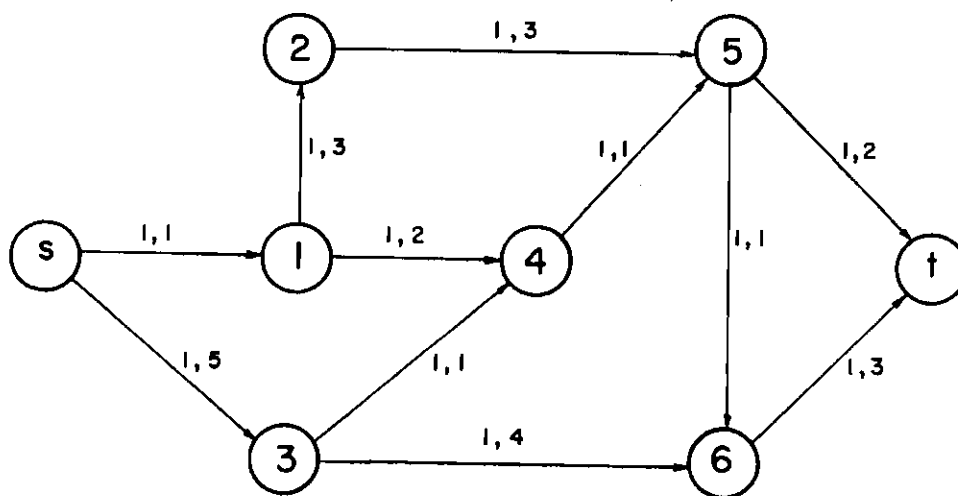


Figure 4. Example Network

Table 2. Minimum Cost Solution

Path	Flow	Length
C_1	1	6
C_2	1	12

Table 3. Min-Max Path Solution

Path	Flow	Length
C_3	1	9
C_4	1	11

The min-max path solution (with flow of two units) has a cost of 20, but the longest path is path C_4 of length 11.

Fractional Solutions

A natural question arises for any network flow problem concerning the existence of an integer optimal solution. The answer to this question gives us a clue as to the level of efficiency we might expect to be able to achieve in a solution algorithm. For if, as is the case with the maximum flow (1.1) and the minimum cost maximum flow (1.2) problems, all basic solutions are all-integer, then one should be able to devise a solution scheme for progressing through the extreme points of the convex solution space by using only additions and subtractions. This fact is what allows the solution of these two problems to be carried out very efficiently on the network.

The characteristic of problems (1.1) and (1.2), of having all-integer extreme points depends of course on the constraint matrix. These matrices have the unimodular property.

Definition

An $m \times n$ matrix A is said to be unimodular if the determinant of every $m \times m$ sub-matrix of A has value ± 1 , or 0.

Remark 1

A convex set S , defined by

$$S = \{ \underline{x} \mid A \underline{x} = \underline{b} \}$$

where \underline{b} is an all-integer vector, has all integer extreme points if A

is unimodular. A proof of this is given by Hu(23).

A simple example is sufficient to show the following remark (see for example, the minimum cost flow example in Chapter III of (12)).

Remark 2

The network arc-path incidence matrix is not unimodular.

Multicommodity flow problems do not in general have all-integer optimal solutions. Much research has been conducted on the structure of multicommodity flow problems. However, to date all computational algorithms use a combination of graph theoretic and algebraic methods. The algebraic portions are of the revised simplex, decomposition type, with graph theoretic methods used to generate non-basic columns which are candidates to enter the basis.

Hu (23) provides the most complete discussion of multicommodity flow problems. Jewell (28), Sakarovitch (41) and Saigal (39) report research on solution procedures they have developed. To date, the only multicommodity case for which a combinatorial solution procedure has been developed is the two-commodity algorithm of Rothschild and Whinston (38).

Tomlin (44) formulates the minimum-cost multicommodity flow problem in node-arc form and in arc-path form. Jarvis (26) shows that the computations involved in the two cases are identical. Thus, it appears that the inherent structure of the multicommodity flow problem forces one to deal with flow on paths rather than flow on arcs.

In the early stages of this research a conjecture was made concerning the existence of an all-integer solution to the min-max path flow problem. Later, an analogous conjecture was found in Sakarovitch's

paper dealing with the all integer solutions to multicommodity flow problems (41).

Conjecture

Let v_{\max} be the maximum flow on $G(N,A)$ and let $\bar{C}_0 \subseteq \bar{C}$ be a subset of the paths in $G(N,A)$ from s to t . Then if there exists a flow function H defined on \bar{C}_0 such that

$$v = \sum_{C_i \in \bar{C}_0} h(C_i) = v_{\max} ,$$

then there exists an all-integer flow function H_0 satisfying the same condition.*

This conjecture has intuitive appeal since any flow which satisfies this condition is an alternate optimal solution to the maximum flow problem, which has all all-integer extreme points. The problem which arises however, is that we have introduced additional zero-one variables and additional constraints which destroy the unimodular property of the original constraint set.

An example from Sakarovitch's paper is a counter-example to this conjecture. It can also be modified to show that the min-max path flow problem may not have an all-integer solution.

Figure 5 is the example network. The optimal solution is given in Table 4; all arc capacities and arc lengths are given on the network. We shall let $L^*(G,N)$ or just L^* denote the length of the longest path

*Sakarovich expresses the corresponding concept for multicommodity flows as "gapless" multicommodity networks.

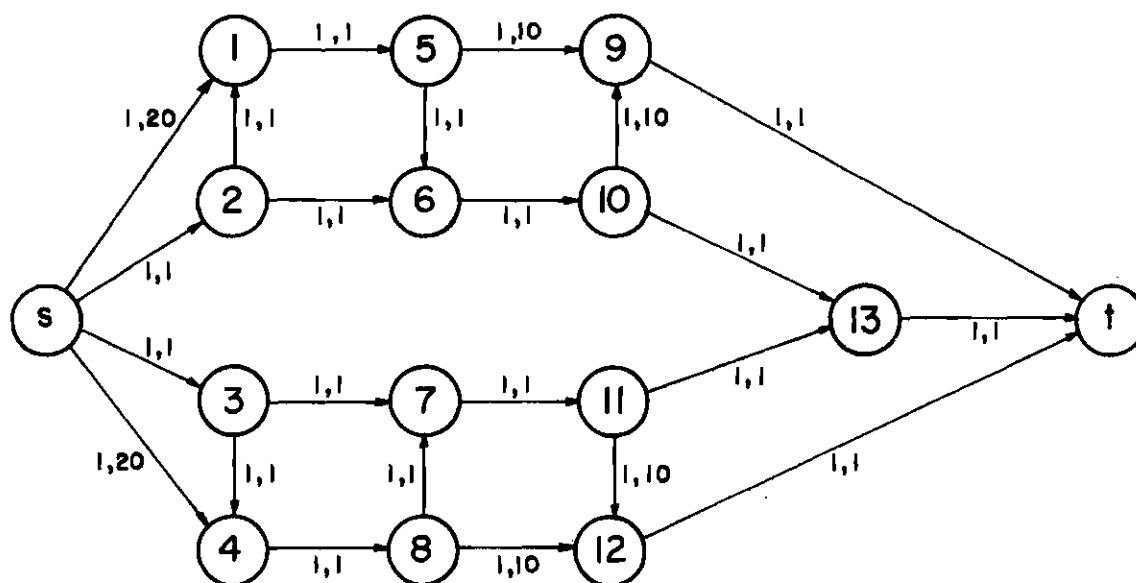


Figure 5. Network With Fractional Min-Max Flow

Table 4. Min-Max Path Flow Solution

Path (Nodes)	Length	Flow
s,1,5,6,10,13,t	25	$\frac{1}{2}$
s,2,1,5,9,t	14	$\frac{1}{2}$
s,2,1,6,10,9,t	14	$\frac{1}{2}$
s,4,8,7,11,13,t	25	$\frac{1}{2}$
s,3,4,8,12,t	14	$\frac{1}{2}$
s,3,7,11,12,t	14	$\frac{1}{2}$

in the optimal solution to the min-max path flow problem. For the given network, $L^* = 25$. By enumerating all paths in the network we could see that there is no all-integer solution to the maximum flow problem on paths of length 25 or less.

Sakarovitch shows that a sufficient condition for a multi-commodity network to be gapless and all-integer is that it be completely planar.* With respect to the min-max path flow problem we can make the following remark.

Remark 3

For the min-max path flow on a directed network to be all-integer it is neither necessary nor sufficient for the network to be planar.

The example of Figure 5 is a planar network when arc (s,t) is included but it does not have an all-integer solution, on the other hand, both the bottleneck assignment network and the time-minimizing transportation networks are non-planar. Both always have all-integer solutions.

Contained in the min-max path flow problem is the problem of finding the min-max path decomposition of a given flow on a network $G(N,A)$.

D. R. Fulkerson suggested in a private communication (16) a necessary condition for the min-max path decomposition to have longest path less than or equal to L . Let F be a node-arc flow defined on $G(N,A)$. Let $A_1 \subseteq A$ be defined: $A_1 = \{(x,y) \mid f(x,y) > 0, (x,y) \in A\}$.

*A planar network is a network, including an arc from the source to the sink, that can be drawn on a plane in such a way that no two edges intersect except at their endpoints. A multicommodity network is completely planar if it is planar when all source-sink pairs are connected including the super source and super sink.

Let \overline{C}^L be the set of all paths in $G(N,A)$ from s to t of length L or less.

Definition:

The set of paths \overline{C}^L is said to cover A_1 if $(x,y) \in A$ implies that there exists a path $C_j \in \overline{C}^L$ which contains (x,y) .

Fulkerson's suggestion was that a necessary condition for the min-max path decomposition of F to have maximum path length less than or equal to L is that \overline{C}^L cover A_1 .

The following example shows that this is not a sufficient condition. In Figure 6 all arcs have a flow of one unit.

The numbers adjacent to the arcs are arc lengths. Inspection shows that all flows can be covered by paths of length six. However, there is no path decomposition utilizing only paths of length six or less.

Approaches to Algorithm Development

Geoffrion (19) presents a thorough discussion of the solution of large scale linear programming problems. He classifies efforts in problem solving or more generally, algorithm development into one of two groups, search strategies or problem manipulation. Search strategies of course, are procedures for moving from one solution to a better one; problem manipulation is essentially attempts to formulate the problem in the most advantageous way for solution efficiency. Generally speaking, one considers both the proper problem formulation and the most efficient search strategies when developing a solution algorithm.

We must consider the selection of search strategies and the problem formulation simultaneously since it is possible to formulate the problem

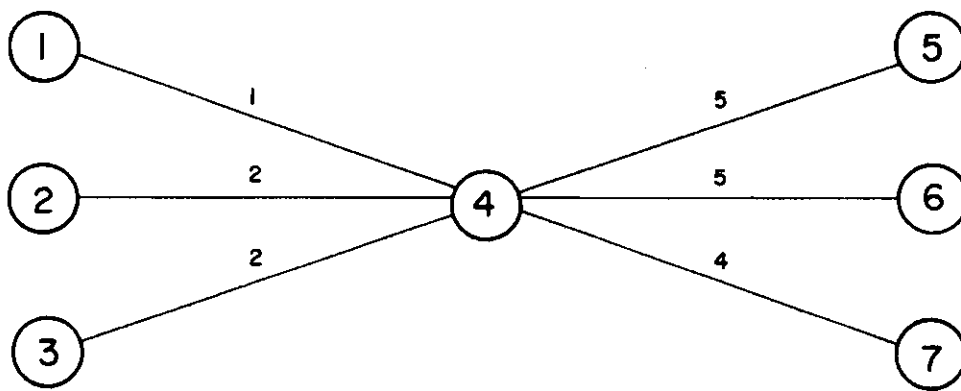


Figure 6. Network With Covered Flows

in the most natural way only to find out that the best search strategy for that formulation is very inefficient. Such is the case for the min-max path flow problem. The most natural formulation is the integer-linear programming formulation; however, solution algorithms for such problems are inefficient as compared to those for linear programming problems. Thus, we will mention two formulations of the problem which will be investigated from the network theory viewpoint.

Speaking in terms of the solution set over which we will be searching, we can consider two broad approaches to the formulation of the min-max path flow problem. The first will be a dual type formulation, while the second will be a primal type formulation.

Dual Approach

Begin with $L = 1$ and find the maximal flow in $G(N,A)$ on paths of length L or less. Increment L by one unit and continue until the maximum flow on paths of length L or less is equal to the maximum flow in the network.

Primal Approach

Find the maximum flow on $G(N,A)$ in arc-path form. Search over the set of maximum flows for the flow function whose longest path is minimized.

The first thing that comes to mind in regard to the dual approach is to generate all paths of length L or less for some specified value of $L > 1$, and solve the obvious linear programming problem to maximize the flow on these paths. There are two drawbacks to this approach. First, the problem of generating all paths of length L or less is inefficient when only a few of them will be in the solution. Second, a great deal

of computer storage will be required. External storage will be required and this storage must be accessed very frequently in the simplex procedure. The time required for this access will dominate the actual computation time.

In Chapter IV a formulation is presented along with a solution algorithm in which all paths of length L or less are represented implicitly in a modified network while paths of length greater than L are not in the network. While external storage will likely be required for certain information, it will be the type which is required at most L^* times and quite likely much less often.

During the course of this research, various primal approaches were considered. The first one was based on the idea of attaching a penalty to the arcs in the current longest path in an effort to force it out of the solution. The minimum cost maximum flow problem is solved. After decomposing the flow into arc-path form it is possible to attach penalties to increase the cost of certain arcs so that the current solution is no longer the minimum cost solution. It was felt that possibly this would ultimately force the longest path out of the solution.

An example problem was found for which the optimal min-max path solution could not be found in this way. After later discovering that the min-max path flow problem did not have all-integer optimal solutions, the existence of such an example was proved. Since the above approach attempts to solve the problem by solving a sequence of minimum cost flow problems and such problems always have all-integer solutions, it follows

that there exists no such sequence, for problems that do not have all-integer min-max path solutions.

In Chapter III we present an algorithm for removing flow on the current longest path if it is possible to do so while maintaining the maximum flow.

CHAPTER III

A PRIMAL ALGORITHM FOR SOLVING THE
MIN-MAX PATH FLOW PROBLEM

In this chapter we present a primal algorithm for solving the min-max path problem. The problem was initially presented as having a primary and a secondary objective function; the primary objective being to maximize the flow and the secondary objective being to minimize the length of the longest path assigned positive flow.

Ford and Fulkerson's algorithm can be used to determine the optimal value of the flow, v_{\max} . With this knowledge, the primary objective function can be written as a constraint on the problem. Thus we have the problem

minimize: L

subject to: $L \geq \delta_j \underline{A}' \underline{P}_j \quad j = 1, \dots, n$

$$\sum_{j=1}^n x_j \geq v_{\max}$$

$$\underline{P}\underline{X} \leq \underline{b}$$

$$\underline{X} \geq 0 ,$$

where

$$\delta_j = \begin{cases} 1, & \text{if } x_j > 0 \\ 0, & \text{otherwise,} \end{cases}$$

$\underline{X} = [x_1, x_2, \dots, x_n]'$, and x_i denotes the flow on path i . Thus n is the cardinality of the set of all paths in $G(N, A)$ from the source to the sink. P is the arc-path incidence matrix and \underline{b} is the vector of arc capacities.

Suppose we have a feasible solution. Then it follows from the nature of the objective function and the first n constraints that an improved solution can only be obtained by reducing to zero the flow x_j on all paths \underline{P}_j for which the corresponding constraint of this set is binding. Note that the general constraint of this set is

$$L \geq \ell_j = \underline{a}'\underline{P}_j,$$

if path \underline{P}_j is assigned positive flow. Thus, the j^{th} constraint of this set will be binding if the j^{th} path has length L and has x_j positive.

The algorithm presented in this chapter is based on the concept of finding non-basic paths to introduce into the basis which will tend to reduce the flow on those paths corresponding to these binding constraints while at the same time not introducing flow on longer paths.

Reducing Flow on the Longest Path

Consider the min-max path problem. A feasible solution can be obtained by solving the maximum flow problem on the network. Let us

consider the maximum flow problem formulated in arc-path form.

$$\text{Maximize: } v = x_1 + x_2 + \cdots + x_n \quad (1)$$

$$\text{Subj to: } \underline{P}\underline{X} \leq \underline{b}$$

$$\underline{X} \geq 0.$$

Let us suppose that we have an optimal solution to this problem and the basis is denoted by B^* . Suppose v_{max} denotes the value of the flow.

Let $I_B = \{j \mid \underline{P}_j \in \{B\}\}$. If j is greater than n , the total number of paths in $G(N,A)$, then \underline{P}_j is a slack path, or slack vector.

Let

$$I_B = \underset{\substack{j \in I_B \\ x_j > 0}}{\text{maximum}} \{a' \underline{P}_j\}$$

Then I_B is the length of the longest path which carries positive flow in the current solution.

We show in the appendix that we can obtain an initial feasible basis which will not contain any paths \underline{P}_j , such that $x_j = 0$. At subsequent iterations we will maintain a basis which contains no paths of

*The solution to this problem can be obtained by the Ford-Fulkerson algorithm. This algorithm produces a node-arc flow. However, a path decomposition procedure can be used to obtain an arc-path equivalent flow. An algorithm is given in the appendix.

length greater than L_B , except those which cannot be in an adjacent solution at a positive level. Two basic solutions are said to be adjacent if they differ by precisely one vector. Thus, if a path \underline{P}_j of length greater than L_B is in the basis at the zero level and one pivot could not increase it above zero, then it is left in the bases. Otherwise it must be removed before we continue. Later we shall introduce the term pure basis and show how it implements the above requirement.

The min-max path flow problem can be thought of then in terms of minimizing L_B .

LEMMA 1

The value of the optimal min-max path flow on a given network $G(N,A)$ is equal to L_B if and only if B is an optimal basic solution to (1) and there does not exist an optimal set of basic paths B' such that,

$$L_{B'} < L_B$$

Proof. This follows from the definition of L_B and the definition of the value of the min-max path flow on $G(N,A)$.

We require some additional notation before we formulate a problem which will allow us to determine whether such a basis B' exists or not.

Notation

$$I_L = \{j \in I_B \mid \underline{a}'\underline{P}_j < L_B\}$$

$$I_{L'} = \{j \in I_B \mid \underline{a}'\underline{P}_j = L_B\}$$

$$J_L = \{j \mid j \notin L_B \text{ and } \underline{a}'\underline{p}_j < L_B\}$$

$$J_L' = \{j \mid j \notin L_B \text{ and } \underline{a}'\underline{p}_j \geq L_B\}.$$

It follows from Lemma 1, that to obtain a solution better than the current solution to the min-max path problem it is necessary to be able to find an optimal basis B' which does not contain paths of the set J_L' or for which the flow on such paths is zero, while at the same time no path longer than $L_B - 1$ belongs to the basis B' .

Remark 1

To determine whether there exists a solution to the maximum flow problem with:

$$L_B' \leq L_B - 1,$$

it is only necessary to seek a solution to the following problem:

$$\text{Minimize: } \sum_{i=1}^n \delta_i x_i = Z \quad (2)$$

$$\text{Subject to: } \underline{P}\underline{X} \leq \underline{b}$$

$$\sum_{i=1}^n x_i = v_{\max}$$

$$\underline{X} \geq 0.$$

The coefficient δ_i is defined as follows, with $R > 0$ to be specified later:

$$\delta_i = \begin{cases} R, & \text{if } i \in J_L, \\ 1, & \text{if } i \in I_L, \\ 0, & \text{otherwise} \end{cases}$$

Remark 2

If the optimal solution to (2) has $Z^* = 0$, then this solution has a min-max path value of

$$L_B, \leq L_B - 1,$$

and hence is a better solution to the min-max path flow problem. If Z^* is positive, then the previous solution was optimal and the min-max path flow has the length of its longest path L_B .

The optimal solution to (1) is of course a feasible solution to (2). Finding the optimal solution to (2) will either drive the current longest paths out of the basis or reduce their flow to zero, or will reveal that they cannot be driven from the basis.

We can reformulate problem (2) as follows:

$$\text{minimize: } \sum_{i=1}^n \delta_i x_i - p \sum_{i=1}^n x_i = z' \quad (2')$$

$$\text{subject to: } P\underline{X} \leq \underline{b}$$

$$\underline{X} \geq 0,$$

with $p > 0$.

Since we are interested in determining whether (2) has an optimal solution with $z^* = 0$, the following lemma provides a sufficient statement of equivalence between problems (2) and (2').

Lemma 2

The optimal solution to (2) has $z^* = 0$ if and only if the optimal solution to (2') has $z'^* = -pv_{\max}$.

Proof. Suppose (2') has a solution \underline{x}^* with $z^* = 0$. This implies that

$$\sum_{i=1}^n \delta_i x_i^* = 0$$

and

$$\sum x_i^* = v_{\max}.$$

This solution is a feasible solution to (2') with

$$z' = -pv_{\max}$$

$$\Rightarrow z'^* \leq -pv_{\max}$$

But since $\sum_{i=1}^n \delta_i x_i$ is bounded below by zero, and $-p \sum_{i=1}^n x_i$ is bounded below by $-pv_{\max}$,

$$z'^* \geq -pv_{\max}$$

$$\Rightarrow z'^* = -pv_{\max}$$

It follows that \underline{X}^* is an optimal solution to (2').

Since the dimension of the solution space is greater for problem (2) than (2'), this may not be a basic solution but from the theory of the simplex method (18), a basic optimal solution exists.

Suppose (2') has an optimal solution with $z'^* = -pv_{\max}$, then that solution must have

$$\sum_{i=1}^n x_i = v_{\max}$$

and

$$\sum_{i=1}^n \delta_i x_i^* = 0 .$$

This follows from the fact that

$$\sum \delta_i x_i \geq 0$$

and

$$-p \sum x_i \geq -pv_{\max} .$$

Thus \underline{X}^* is an optimal solution to (2).

The objective function can be rewritten as

$$z = \sum (\delta_i - p)x_i .$$

After substituting the appropriate expressions for δ_i we have:

$$\begin{aligned} Z = & \sum_{k \in J_L} (R - p)x_k + \sum_{k \in I_L} (1 - p)x_k + \\ & \sum_{k \in I_L} (-p)x_k + \sum_{k \in J_L} (-p)x_k . \end{aligned}$$

If we order the basic vectors with those of length L_0 first, those of length less than L_0 next, and the slack paths last, then $\underline{C}_B^{(2')}$ has the form

$$\underline{C}_B^{(2')} = [\{(1 - p)\} : \{(-p)\} :: \{0\}] .$$

Since (1) and (2') have the same constraint set, and we are dealing with the same set of basic columns, B , in each problem, B^{-1} is the same for both (1) and (2').

If we let π denote the vector of dual variables for the current basic feasible solution to (1), and let ρ denote the vector of dual variables of problem (2') with respect to this same basic feasible solution to (2'), then

$$\pi = (1, 1, \dots, 1, 0, 0, \dots, 0)B^{-1}$$

$$= \pi = \sum_{I_L} B_k^{-1} + \sum_{I_L} B_k^{-1} ,$$

$$\rho = \sum_{I_L'} (1 - p) B_k^{-1} + \sum_{I_L} (-p) B_k^{-1}$$

$$\rho = \sum_{I_L'} B_k^{-1} - p \left(\sum_{I_L'} B_k^{-1} + \sum_{I_L} B_k^{-1} \right)$$

$$\rho = \sum_{I_L'} B_k^{-1} - p\pi .$$

We now state and prove a theorem which provides a set of necessary and sufficient conditions for a vector (path) \underline{P}_j to be a candidate to enter the basis for problem (2').

Theorem 1

A path \underline{P}_j^* is a candidate to enter the basis of problem (1) (in the sense that its relative cost coefficient with respect to problem (2') is less than zero), if and only if it satisfies the following three conditions:

$$\sum_{I_L'} B_k^{-1} \underline{P}_{j*} > 0 \quad (3)$$

$$\pi \underline{P}_{j*} \leq 1 \quad (4)$$

$$\underline{a}' \underline{P}_{j*} \leq L_B - 1 . \quad (5)$$

Proof. We first show that if conditions (3) - (5) are satisfied by a vector \underline{P}_{j*} , then it is a candidate to enter the basis in problem (2').

If

$$\underline{a}'\underline{P}_j* \leq L_8 - 1 \quad ,$$

then $C_{j*}^{(2')}$ = -p. From (4), we see that

$$\pi\underline{P}_j* - 1 \leq 0 \quad ,$$

and for $p > 0$,

$$p(\pi\underline{P}_j* - 1) \leq 0$$

$$\Rightarrow -p\pi\underline{P}_j* \geq -p \quad .$$

From (3), we have

$$\sum_{k \in I_L'} B_k^{-1} \underline{P}_j* > 0$$

Thus,

$$Z_{j*}^{(2')} = \rho \underline{P}_j* = \sum_{k \in I_L'} B_k^{-1} \underline{P}_j* - p\pi\underline{P}_j* > -p = C_{j*}^{(2')} .$$

It follows then that

$$C_{j*}^{(2')} - Z_{j*}^{(2')} < 0 \quad .$$

Thus \underline{P}_j* is a candidate to enter the basis of $(2')$.

We now show that if \underline{P}_{j*} is a candidate to enter the basis of (2'), then it satisfies conditions (3) - (5).

Gass (18) shows that in solving a linear programming problem by the simplex method it is sufficient to consider only vectors \underline{P}_j such that $C_j - Z_j < 0$. We want to show then that a vector \underline{P}_{j*} which satisfies

$$C_{j*}^{(z')} - Z_{j*}^{(z')} < 0 ,$$

satisfies conditions (3) - (5).

Consider

$$C_{j*}^{(z')} - Z_{j*}^{(z')} = \delta'_{j*} - p + (p\pi - \sum_{k \in I_L} B_k^{-1}) \underline{P}_{j*} ,$$

where

$$\delta'_{j*} = \begin{cases} R, & \text{if } j* \in J_L' \\ 1, & \text{if } j* \in I_L' \\ 0, & \text{otherwise} \end{cases} .$$

Collecting terms we have,

$$C_{j*}^{(z')} - Z_{j*}^{(z')} = p(\pi \underline{P}_{j*} - 1) - \left(\sum_{k \in I_L} B_k^{-1} \right) \underline{P}_{j*} + \delta'_{j*} .$$

Since we have an optimal solution to (1), we can assume that for paths of length $L_0 - 1$ or less,

$$C_j^{(z')} - Z_j^{(z')} = 1 - \pi \underline{P}_j \leq 0 .$$

If at some stage this does not hold then we can introduce the corresponding vectors into the basis. If the solution is optimal and such a vector exists, it must come into the basis at the zero level, otherwise the pivot operation would result in an increase in the value of the objective function. This would contradict the assumption of optimality. It will be shown that paths of length $L_B - 1$ or more are not candidates to enter the basis in any event.

If

$$1 - \pi \underline{P}_j \leq 0 ,$$

then

$$\pi \underline{P}_j - 1 \geq 0 ,$$

and

$$p(\pi \underline{P}_j - 1) \geq 0 .$$

Since δ' is also non-negative, we conclude

$$C_j^{(z')} - Z_j^{(z')} < 0$$

$$\Rightarrow p(\pi \underline{P}_j * - 1) - \left(\sum_{k \in I_L} B_k^{-1} \right) \underline{P}_j * + \delta_j' < 0$$

$$\Rightarrow \left(\sum_{k \in I_L} B_k^{-1} \right) \underline{P}_j * - 1 \geq 0 .$$

Hence condition (3) is satisfied.

From (2') we see that

$$C_j^{(2')} = \delta_j' - p = \begin{cases} R - p, & \text{if } \underline{a}' \underline{P}_j * \geq L_B \\ -p, & \text{otherwise} \end{cases}$$

If

$$\underline{a}' \underline{P}_j * \geq L_B ,$$

then

$$C_j^{(2')} - Z_j^{(2')} = R - p - \left(\sum_{k \in I_L} B_k^{-1} \right) \underline{P}_j * + p \pi \underline{P}_j * .$$

We shall show later in this chapter that we can insure, for sufficiently large (finite) p , that

$$-\left(\sum_{k \in I_L} B_k^{-1} \right) \underline{P}_j * + p \pi \underline{P}_j * > 0 , \text{ for all } j .$$

Hence, if we select R , such that

$$R - p \geq \left(\sum_{k \in I_L'} B_k^{-1} \right) \underline{P}_j^* - p \pi \underline{P}_j^* = \eta$$

then

$$C_j^{(z')} - Z_j^{(z')} \geq 0$$

If we choose $R = p$, then

$$R - p = 0 > \eta$$

since

$$\eta < 0.$$

It follows that no path of length greater than or equal to L_B is a candidate to enter the basis for problem (2'). Thus, in searching for a path to enter the basis, we can restrict our search to paths of length $L_B - 1$ or less. This is condition (5).

Again consider

$$C_j^{(z')} - Z_j^{(z')} = p(\pi \underline{P}_j^* - 1) - \left[\left(\sum_{k \in I_L'} B_k^{-1} \right) \underline{P}_j^* - \delta_j' \right]$$

Since we have only to consider non-basic paths of length $L_B - 1$ or less, we have $\delta_j' = 0$. Thus we have

$$C_j^{(2')} - Z_j^{(2')} = p(\pi \underline{P}_j - 1) - \left(\sum_{k \in I_L} B_k^{-1} \right) \underline{P}_j.$$

For a given \underline{P}_j , the second term on the right is fixed. By Lemma 2, p can be chosen arbitrarily large. It follows that $p(\pi \underline{P}_j - 1)$ dominates the expression if it is not zero. Hence p can be chosen large enough so that

$$\pi \underline{P}_j > 1$$

$$\Rightarrow p(\pi \underline{P}_j - 1) > \left(\sum_{I_L} B_k^{-1} \right) \underline{P}_j$$

We conclude that p can be selected large enough that no vector \underline{P}_j , such that

$$\pi \underline{P}_j > 1,$$

will have a negative relative cost coefficient and it is not necessary to bring such vectors into the basis. Thus, condition (4) holds.

Column Generation

We now consider the problem of finding vectors \underline{P}_j which satisfy conditions (3) - (5). If

$$\underline{a}' \underline{P}_j < L_B,$$

then

$$C_j^{(2')} - Z_j^{(2')} = -p + (p\pi - \sum_{I_L} B_k^{-1}) \underline{P}_j .$$

The path \underline{P}_j is a candidate to enter the basis then if

$$-p + (p\pi - \sum_{k \in I_L} B_k^{-1}) \underline{P}_j < 0 .$$

The problem of finding a vector to enter the basis can be expressed as the discrete programming problem:

$$\text{minimize: } (p\pi - \sum_{I_L} B_k^{-1}) \underline{X}_j \quad (6)$$

$$\text{subject to: } \underline{a}' \underline{X}_j \leq L_0 - 1$$

\underline{X}_j is a simple path in $G(N,A)$ from s to t .

If \underline{X}_{j*} is the optimal solution to this problem and

$$-p + (p\pi - \sum_{k \in I_L} B_k^{-1}) \underline{X}_{j*} < 0 ,$$

then bringing $\underline{P}_{j*} = \underline{X}_{j*}$ into the basis in problem (2') will tend to improve the value of the objective function. In other words, bringing \underline{P}_{j*} into the basis of problem (1) will tend to reduce the net flow on the set of longest paths in the basis. We have proved the following theorem which provides a necessary and sufficient condition for the existence of such a path.

Theorem 2

A path \underline{P}_j exists which satisfies the conditions (3), (4), and (5) if and only if problem (6) has a solution \underline{X}_{j*} for which

$$-p + (p\pi - \sum_{I_L'} B_k^{-1}) \underline{X}_{j*} < 0.$$

The conditions (3), (4) and (5) can be seen intuitively by considering the current solution to problem (1). Since this solution is assumed to be optimal to (1), we have

$$C_{j*}^{(1)} - Z_{j*}^{(1)} \leq 0, \quad \forall j.$$

Continuing to let I_L' denote the set of paths in the basis whose lengths are greater than or equal to L_B , we see that if we find a non-basic path \underline{P}_{j*} such that

$$\sum_{\ell \in I_L'} \hat{P}_{\ell j*} > 0,$$

where

$$\hat{\underline{P}}_{j*} = B^{-1} \underline{P}_{j*}$$

then, bringing \underline{P}_{j*} into the basis will tend to reduce the net flow on the set of longest paths.

But since

$$\hat{P}_{\ell j} = B_{k+1}^{-1} P_{j*} ,$$

we can write

$$\sum_{\ell \in I_L} B_{\ell}^{-1} P_{j*} > 0 ,$$

which is condition (3).

In reducing the net flow on $\{P_{\ell}\}$, $\ell \in I_L$, we do not want to reduce the total flow in the network since our primary objective is to maximize the flow. If P_{j*} is a non-basic vector, bringing it into the basis will not decrease the value of the objective function v below v_{\max} if its relative cost coefficient is not negative, that is

$$C_{j*}^{(1)} - Z_{j*}^{(1)} \geq 0 .$$

As indicated previously,

$$C_{j*}^{(1)} = 1$$

and

$$Z_{j*}^{(1)} = \pi P_{j*} .$$

Thus, this requirement can be expressed as

$$1 - \pi_{\underline{P}_j}^* \geq 0 ,$$

or,

$$\pi_{\underline{P}_j}^* \leq 1 .$$

This is condition (4).

Finally, we do not want to replace \underline{P}_ℓ with a path as long or longer than $\underline{a}'\underline{P}_\ell = L_B$, so that we have condition (5):

$$\underline{a}'\underline{P}_j^* \leq L_B - 1$$

As a result of Remark 1 and Theorems 1 and 2, we can outline a procedure for solving the min-max path flow problem. The two main components of the algorithm are a maximum flow algorithm used to obtain an initial feasible solution and a constrained shortest path algorithm to find non-basic paths to introduce into the basis which will improve the value of the objective function.

In broad terms the algorithm presented here takes the following form.

1. Solve the maximum flow problem on $G(N,A)$.
2. Determine an arc-path decomposition of the maximum flow.
3. Identify, from the set of basic paths with positive flow assigned, that subset of paths whose lengths are equal to that of the longest basic path with positive flow assigned.

4. Search for a non-basic path, shorter than the current longest chain, which, when introduced into the basis will tend to result in a net reduction of the flow assigned to the set of paths identified in step (3), while not introducing any flow on paths longer than the current longest path.

5. Repeat steps (3) and (4) until no such path as described in step (4) exist. The current solution then is optimal.

Step (1) can be carried out using Ford and Fulkerson's maximum flow algorithm. The path decomposition procedure is straight-forward and the one used here is given in Appendix A. It is shown that the set of paths obtained by the path decomposition algorithm, augmented by appropriate slack paths is a basis for problem (1).

Step (4) of the algorithm will make use of B^{-1} . Following the identification of B in Step (2), B^{-1} can be computed by one of the known methods for computing the inverse of a sparse matrix (44). From then on, B^{-1} is updated in the usual revised simplex method. In Chapter V, we discuss another approach for obtaining the information needed from B^{-1} without actually computing the entire matrix.

Based on the results of step (2), step (3) is obvious. Step (3) will be modified later to simplify step (4). This leaves step (4) which requires considerable elaboration. As indicated previously, step (4) can be carried out by finding the path in $G(N,A)$ to

$$\text{minimize: } (p\pi - \sum_{I_k} B_k^{-1}) \underline{X}_j \quad (6')$$

subject to: $\underline{a}'\underline{X}_j \leq L_B - 1$

\underline{X}_j is a path from s to t .

Thus, the problem is that of finding the shortest path with respect to the arc numbers

$$(p\pi - \sum_{k \in I_L} B_k^{-1}),$$

which is no longer than $L_B - 1$ with respect to arc numbers \underline{a} .

This problem is referred to as the constrained shortest path problem. Efficient shortest path and constrained shortest path algorithms require that arc numbers must be positive or at least there be no negative directed cycles.

We can assume $\pi \geq 0$, for if $\pi_i < 0$ for some i , the appropriate slack variable can be introduced in the basis.

Since p is arbitrarily large, pick p such that:

$$p > \frac{\sum_{k \in I_L} B_{kj}^{-1}}{\pi_j}, \quad \text{for all } \pi_j > 0,$$

from which it follows that

$$(p\pi - \sum_{k \in I_L} B_{kj}^{-1}) \geq 0, \quad \text{if } \pi_j > 0.$$

Thus, a problem could arise only if there exists a directed cycle in G such that $\pi_j = 0$ on all arcs of the cycle, while:

$$\sum_{k \in I_L} B_{kj}^{-1} > 0$$

on some arc of the cycle. This can be avoided by adopting the following procedure:

If the simplex multiplier π_j , corresponding to arc j is, $\pi_j = 0$, and $\sum_{k \in I_L} B_{kj}^{-1} > 0$, introduce the slack path S_i corresponding to a_i , into basis.

Lemma 4 and Theorem 3 which will shortly be stated and proved will establish the fact that this convention will eliminate the possibility of $G(N,A)$ containing negative directed cycles with respect to the arc lengths of problem (6). In order to simplify the statements of Lemma 4 and Theorem 3, we adopt the following definition: an optimal basic set of paths, \hat{B} , for problem (1) is said to be a pure basis if the following three conditions hold:

- (a) $\pi_i \geq 0$
- (b) $\pi_i = 0 \Rightarrow S_i \in \{\hat{B}_i\}$
- (c) $\underline{P}_r \in \{\hat{B}_i\} \Rightarrow a' P_r \leq L_B,$

where $\{\hat{B}_i\}$ denotes the set of columns of \hat{B} . Condition (a) holds as a result of adjustments suggested previously. We can show that if, at some

stage, condition (c) does not hold for some path, then we can replace that path in the basis by a slack path. Consider a path \underline{P}_r and suppose it is the r^{TH} basic vector. Let \underline{P}_j be any non-basic path. Bringing \underline{P}_j into the basis will tend to increase the flow on \underline{P}_r only if

$$B_r^{-1} \underline{P}_j < 0 .$$

But

$$B_r^{-1} \underline{P}_r > 0 ,$$

and since

$$P_{ij} \geq 0, \text{ for all } i,$$

some component of B_r^{-1} , say B_{rt}^{-1} must be positive. That is

$$B_{rt}^{-1} > 0 .$$

It follows then that

$$B_r^{-1} S_t = B_{rt}^{-1} > 0 ,$$

and S_t can be brought into the basis to replace \underline{P}_r . Since \underline{P}_r is in the basis at the zero level, the new solution will remain optimal.

We can verify the possibility of always satisfying condition (b) by proving Lemma 3.

Lemma 3

Let B represent an optimal basis for (1) and L_B denote the length of the longest path in $\{B_j\}$ which has positive flow assigned. Let I_L denote the index set of longest paths in $\{B_j\}$ and π the vector of simplex multipliers corresponding to the current basis. If

$$\sum_{k \in I_L} B_{ki}^{-1} > 0 ,$$

and

$$\pi_i = 0 ,$$

then the slack vector S_i can be introduced into the basis and the resulting basis will be an optimal feasible basis for (1) also.

Proof. It is well known from the theory of the simplex method that given an optimal basis B to a linear programming problem, an alternate optimal solution can be generated by bringing into the basis any non-basic vector whose reduced price is zero.

If π is the vector of simplex multipliers corresponding to the current optimal basis, then

$$Z_{s_1} - C_{s_1} = \pi S_1 - 0 = \pi_1 = 0$$

or

$$Z_{s_1} - C_{s_1} = 0$$

Thus S_1 can be brought into the basis without changing the value of the objective function. Thus the resulting basis containing S_1 is an optimal basis.

The fact that the solution remains basic, that is, the new set of vectors produced by bringing S_1 into the basis is independent, can be argued as follows. Since

$$\sum_{k \in I_L} B_{ki}^{-1} > 0,$$

$$\Rightarrow B_{\underline{k}i}^{-1} > 0$$

for some specific $\underline{k} \in I_L$. It follows that

$$B_{\underline{k}}^{-1} S_1 > 0.$$

If we apply the standard simplex rule for change of basis, the \underline{k}^{TH} basic variable will be a blocking variable. If no other variable reaches zero before x_k reaches zero, then S_1 replaces \underline{P}_k in the basis. In any event, some current basic variable is driven to zero for a finite

value of x_{s_1} . Gass (18) proves that if there exists a blocking variable in the basis with respect to S_1 and if x_{s_1} is increased until any one of the blocking variables becomes zero, then the resulting set of vectors is an independent set of vectors and hence correspond to an extreme point of the convex solution set.

We conclude then that S_1 can be brought into the basis and the result is an alternate optimal basic feasible solution to (1).

Q.E.D.

Throughout the remainder of this chapter, when discussing a basic optimal solution to (1) we will assume that the basis satisfies conditions (a), (b), and (c) just discussed and is hence a pure basis. The solution algorithm to be presented in this chapter will include necessary procedures to ensure that a current basis is transformed to a pure basis before it is used in the subsequent computations.

We now return to a formal statement and proof of Lemma 4 and Theorem 3.

Lemma 4

Let B be an optimal pure basis for problem (1), and let π be the corresponding vector of simplex multipliers. For a given arc $i \in A$ and any row B_k^{-1} of B^{-1} ,

$$\sum_{k \in I_L} B_k^{-1} > 0 \Rightarrow \pi_i > 0.$$

Proof. To prove this theorem, we prove the contrapositive of it, that is,

$$\pi_i = 0 \Rightarrow \sum_{k \in I_L'} B_{ki}^{-1} = 0 \quad \text{for all } k \neq i.$$

If $\pi_i = 0$, then S_i , the slack path i , is the basic vector with respect to arc . The i^{th} column of B is e_i , where, as usual, e_i is the column vector all of whose elements are zero except the i^{th} element and it is one. We can express the inverse of B as:

$$B^{-1} = (\text{Adjoint } B) / |B|$$

$$B^{-1} = (C_{ki})' / |B| ,$$

C_{ik} being the cofactor of element B_{ik} . The value of C_{ik} is determined by

$$C_{ik} = (-1)^{i+k} |M_{ik}|$$

where M_{ik} is the $(m-1)$ by $(m-1)$ submatrix of B obtained by deleting row i and column k .

It follows that

$$B_{ki}^{-1} = (C_{ik}) / |B|$$

For $k \neq i$,

$$|M_{ik}| = 0 ,$$

since M_{ik} is obtained by deleting row i and column k from B and row i is the only row which has a non-zero element in column i . As a result, for all $k \neq i$,

$$B_{ki}^{-1} = 0 ,$$

and

$$B_{ii}^{-1} = 1 .$$

However, since the basic path associated with arc i is the slack path S_i ,

$$i \notin I_L ,$$

hence

$$\sum_{k \in I_L} B_{ki}^{-1} = 0 .$$

Q.E.D.

To transform a given basis B into a pure basis we carry out the following steps.

- a. If $\pi_i < 0$, introduce S_i into the basis.
- b. If $P_r \in \{B_j\}$, $\underline{a}'P_r > I_B$, then replace \underline{P}_r by a slack path.

c. Begin at π_1 and scan π for $\pi_i = 0$. If $\sum_{k \in I_L} B_{ki}^{-1} > 0$, introduce S_i into the basis. Return to π_1 and begin the search again. Continue until π_m is reached and no changes have been made on the current pass. The new basis is a pure basis.

If we choose p such that

$$p > \sum_{i \in I_L} B_{ki}^{-1} / \pi_1 ,$$

for all i such that $\pi_i > 0$, then

$$p\pi_1 - \sum_{i \in I_L} B_{ki}^{-1} \geq 0, \text{ for all } i .$$

The value of p can be selected so that the lengths of all arcs, with respect to the arc numbers

$$p\pi_1 - \sum_{i \in I_L} B_{ki}^{-1}$$

are nonnegative.

We can now state the following theorem:

Theorem 3

Let $G'(N,A)$ be a network with arc numbers defined by

$$d(i) = p\pi_1 - \sum_{i \in I_L} B_{ki}^{-1} ,$$

where B is a pure optimal basis for the maximum flow problem defined on $G(N,A)$, and I_L is the index set of a subset of the real paths in the basis.

Then there exists a real number p_0 such that for

$$p \geq p_0 ,$$

there are no negative directed cycles in $G'(N,A)$.

Proof. Let

$$p_0 = \sum_{\substack{i \in I_L, \\ \pi_i > 0}} \{ \sum B_{k_i}^{-1} / \pi_i \} .$$

As indicated previously, then $d(i)$ as defined in this theorem, is non-negative for all arcs of the network and hence no negative directed cycles can exist.

Q.E.D.

The Constrained Shortest Path Problem

In this section we present a dynamic programming algorithm to solve the constrained shortest path problem (6). As indicated, solving this problem will generate a candidate path to enter the basis of problem (2) or alternately, problem (1).

Theorems 2 and 3 of the last section will be used to show that if we maintain a pure basis to (1) at all times, the algorithm to be presented here will converge in a finite number of iterations.

The dynamic programming algorithm presented here is a modification of Saigal's algorithm for finding the shortest path through a network which passes through a specified number of nodes or no more than

a specified number of arcs (40). Jokschi also presents a similar algorithm for solving the constrained shortest path problem (30). Both algorithms are themselves based on Ford and Fulkerson's combinatorial algorithm for determining the shortest path in a network (12).

Conceptually, the dynamic programming formulation of the problem can be viewed as follows. The stages of the problem correspond to discrete integral values of ℓ , the remaining length available with respect to the arc numbers \underline{a} . In other words, for a given node j , at stage ℓ , the remaining partial path to t must be of length ℓ or less. The states of the system are the possible nodes in the network. The optimality question at each stage ℓ is: if a unit of flow is currently at node j and the remaining path length available is ℓ , then what is the best node to move to next in order to minimize the cost of the path from j to t with respect to the arc numbers \underline{d} .

The calculations are carried out on the network and take the form of recursive labeling on each node of the network. At the ℓ^{th} iteration of the labeling we calculate for each node j , the cheapest path of length ℓ or less from j to t , given that the current label on all other nodes j' represents the cheapest path from j' to t whose length is $\ell - a(j, j')$ or less. The labeling begins at node t . At a given iteration we attempt to create a new label for every node. After every node has been considered we increase ℓ to $\ell + 1$ and begin at node t again. The labeling procedure terminates after iteration $\ell = L$. If s has a label, then the labels can be traced back to t to obtain the path with minimum cost and hence minimum relative cost coefficient with respect to the

current basis for problem (2). We proceed in the normal simplex fashion depending upon whether this relative cost coefficient indicates optimality or not.

If s was not labeled, the current solution is optimal.

The labeling procedure proceeds as follows:

1. Set $\ell = \hat{L} = \text{minimum}_{(i,t) \in A} \{a(i,t)\}$.
2. Label all nodes, $(-, \infty, \infty)$. The first element, $m(j)$ is the node from which the current node, j , was labeled, the second element, $g(j)$, is the distance associated with the current partial path from the node j to t and the third element, $h(j)$, is the cost associated with the current partial path from node j to t .
3. Select an unscanned node j and let

$$\Gamma(j) = \{j' \in N \mid (j, j') \in A \text{ and } g(j') + a(j, j') = \ell\}$$

4. Compute $\delta = \text{minimum}_{\Gamma(j)} \{h(j') + d(j, j'), \infty\}$

Let $m'(j)$ be any node j' for which the minimum occurs if $\delta < \infty$. Otherwise set $m'(j) = 0$.

5. If $m'(j) > 0$, and $\delta < h(j)$, then set

$$m(j) = m'(j)$$

$$g(j) = \ell$$

$$h(j) = \delta.$$

6. Mark this node as scanned for iteration ℓ and go to the next unscanned node. When all nodes have been scanned, if $\ell < L$, increase ℓ to $\ell + 1$, mark all nodes unscanned and begin at node t again. If $\ell = L$, terminate.

7. If there is a subset of arcs of zero length, then after all nodes have been scanned on a given iteration, repeatedly attempt to find improved labels on the nodes of this subset until a complete iteration through this subset generates no new labels. Return to step (3) with $\ell = \ell + 1$.

In solving the constrained shortest path problem we must make up to $L - \hat{L}$ complete iterations through the seven steps. It is not necessary to completely scan one node before going to the next. Thus, we simply move through a list of arcs, checking each to determine whether it allows a better label on its initial node. Each time we check an arc, it requires two additions and comparisons. The total computations required then is on the order of $2(L - \hat{L})m$, where m is the number of arcs in the network.

If there is an arc between each pair of nodes, then $m = n(n - 1)$, where n is the number of nodes. If L is the same order of magnitude as n , then the number of additions and comparisons is of the same order of magnitude as n^3 . This compares with the computations required to determine the shortest path from s to all nodes of a network, $n(n - 1)(n - 2) \approx n^3$.

We conclude then that the constrained shortest path procedure for generating paths to enter the basis in the path removal algorithm

is as efficient as column generating procedures for the multicommodity flow problems.

Other modifications can also be made to further reduce the computations required. For example, if we let $\hat{\alpha}_j$ denote the shortest path from s to t with respect to \underline{a} , then there is no need to generate labels on node j at any iteration beyond ℓ if

$$\ell = \alpha_j = L - \hat{\alpha}_j .$$

No completion of such a partial path would satisfy the constraint.

Recall also that we are solving the constrained shortest path problem to generate a path which satisfies conditions (3), (4), and (5). The computations are carried out in such a way that (4) and (5) are always satisfied. We can terminate the labeling procedure anytime s is labeled such that

$$\sum_{I_L} B_k^{-1} p_{j*} > 0 .$$

Example

We shall use the network of Figure 4 to illustrate the calculations for solving the constrained shortest path problem. The network is given in Figure 7. The circled number adjacent to each arc is the flow on that arc with respect to maximum flow on the network. All arc capacities are one and the arc lengths are given on each arc.

The minimum cost flow solution is given by one unit of flow on $C_1 = \{1, 4, 8, 10\}$ and one unit of flow on $C_2 = \{2, 6, 11\}$. We

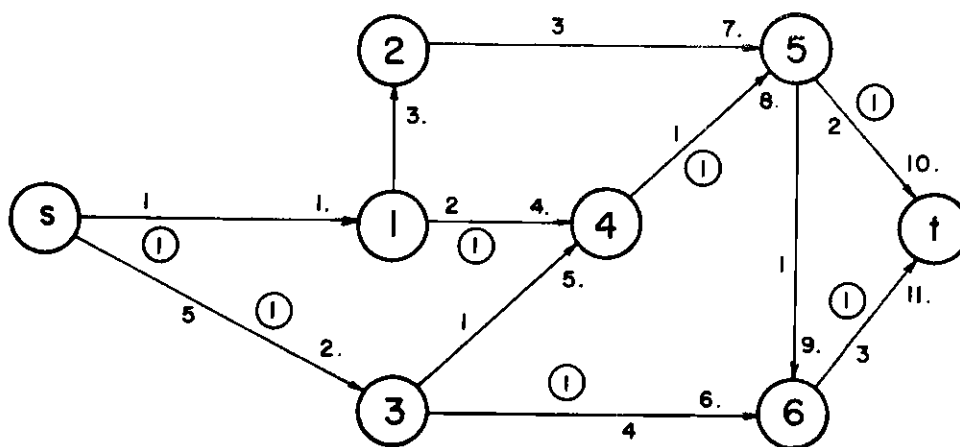


Figure 7. Minimum Cost Flow

consider C_1 and C_2 to be basic with respect to arcs one and two, respectively. The remainder of the basis consists of the appropriate slack paths.

After two iterations we still have $x_1 = h(C_1) = 1$, and $x_2 = h(C_2) = 1$, but we also have $C_3 = \{1, 3, 7, 10\}$ and $C_7 = \{2, 5, 8, 10\}$ in the basis with $x_3 = x_7 = 0$.

At this point we have:

$$L_B = \ell(C_2) = 12 ,$$

$$L_B - 1 = 11 ,$$

and

$$B^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The last row of B^{-1}

$$\pi = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

Since the second basic path is the longest, we have

$$B_2^{-1} = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0)$$

If we let $p = 4$, we have

$$p\pi - B_2^{-1} = (3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)$$

Thus we want to find \underline{p}_j to

$$\text{minimize: } (3, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0)\hat{X}_j$$

$$\text{Subject to: } (1, 5, 3, 2, 1, 4, 3, 1, 1, 2, 3)\hat{X}_j \leq 11$$

$$\hat{X}_j \text{ is a path from } s \text{ to } t.$$

The optimal solution to this constrained shortest path problem is

$$C_4 = \{2, 5, 8, 9, 11\}.$$

If we bring this path into the basis we obtain a solution:

$$x_4 = x_3 = 1$$

$$x_1 = x_7 = 0.$$

The remaining basic variables are slack paths.

In Figure 8 the arc flows corresponding to the current solution are shown by circled numbers. All arc capacities are one unit and the arc lengths are shown. The number with a decimal point following it is the arc identification number. For this solution

$$L_b = \ell(C_4) = 11$$

$$L_b - 1 = 10 ,$$

and

$$B^{-1} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The simplex multipliers are

$$\pi = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

The longest path with flow in the basis is the second path in the basis, C_4 . Its length is 11. Thus

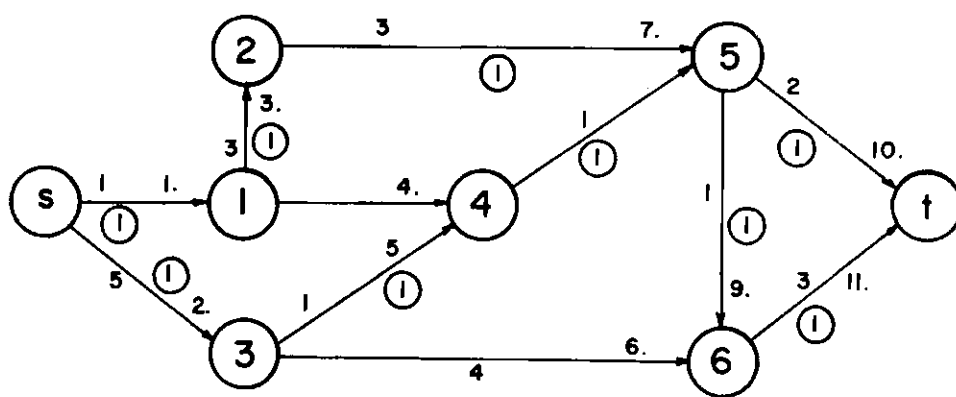


Figure 8. Min-Max Path Flow

$$L_B - 1 = 10 ,$$

and if we let $p = 4$, we have

$$\underline{d} = 4\pi - B_2^{-1} = (3, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0).$$

Figure 9 is the network with the capacities deleted. The arc numbers are the lengths \underline{a} and costs \underline{d} . The number adjacent to each node j is α_j . The labels are given in Table 5. For each value of ℓ only new labels are given.

The optimal solution to the corresponding constrained shortest path problem can be read from the table: $C_6 = \{1, 4, 8, 9, 11\}$. After pivoting we have

$$x_4 = x_3 = 1$$

$$x_6 = x_7 = 0$$

The longest path in the basis remains C_4 . We have

$$L_B - 1 = 10$$

$$B_2^{-1} = (1, 2, 0, 0, 0, 0, 0, -1, 0, -1, 0)$$

$$\pi = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

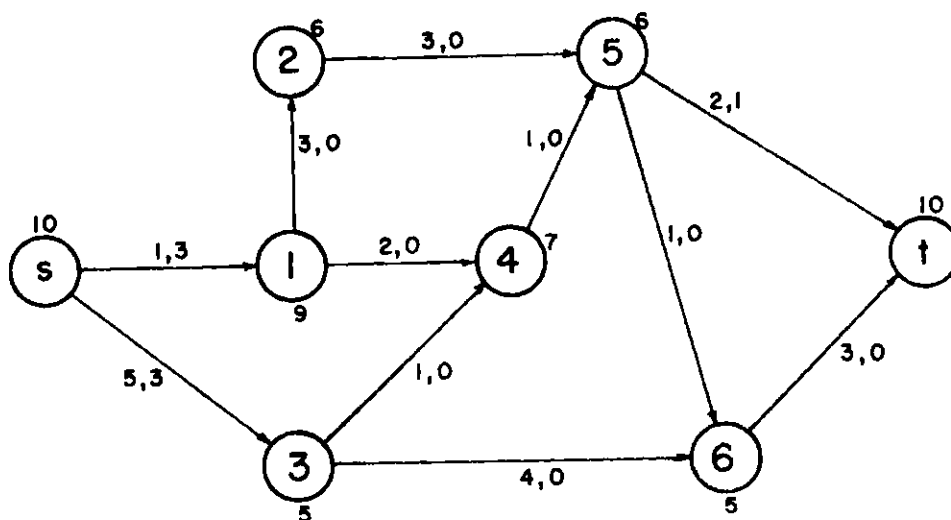


Figure 9. Network With Arc Lengths and Costs

Table 5. Constrained Shortest Path Labels

$g(j) = \ell$	Node	$h(j)$	$m(j)$
2	5	1	t
3	4 6	1 0	5 t
4	3 5	1 0	4 6
5	1 2 3 4	1 1 0 0	4 5 4 5
6	S	5	1
7	1 2	0 0	4 5
8	S	4	1

$$4\pi - B_2^{-1} = (3, 2, 0, 0, 0, 0, 0, 1, 0, 1, 0)$$

With these arc costs, the constrained shortest path is C_3 a path already in the basis, with $B_2^{-1}c_3 = 0$. Thus, we conclude that the current solution to the problem is optimal.

The Path Removal Algorithm

We shall now present a step-by-step statement of the algorithm suggested in this chapter for solving the min-max path flow problem.

1. Solve the maximum flow problem on $G(N,A)$. Let F^* be an optimal solution and let v_{max} be maximum flow.
2. Decompose F to an arc-path flow. Let B be a corresponding basic set of paths. Let π be the simplex multipliers.
3. Let B^0 be the corresponding "pure basis".
4. Determine L_{B^0} , the maximum length path in $\{B^0\}$.
5. Determine $I_L' = \{i \mid \underline{a}' B_i^0 = L_{B^0}\}$.
6. Let p be large positive number and compute

$$\underline{d} = p\pi - \sum_{k \in I_L'} B_k^{-1}.$$

7. Determine \underline{P}_{j*} , the solution to the constrained shortest path problem.
8. Check $\pi \underline{P}_{j*} = 0$, if not increase p and return to (6).
9. Check $\sum_{k \in I_L'} B_k^{-1} > 0$, if not, terminate. The current solution is optimal. Otherwise go to (10).
10. Introduce \underline{P}_{j*} into the basis. Return to Step (3).

Convergence

The optimal solution to the min-max path problem has a positive value of L_B . The iterations through Steps (3) - (10) begin with a finite value of L_B . Each time L_B is reduced, it is reduced by an integer value. Thus, in order to prove convergence, we must establish that there can be only a finite number of consecutive iterations through (3) - (10) without reducing L_B , and, that the dynamic programming algorithm used in Step (7) converges. We consider the former question first.

Solving problem (2) by the simplex algorithm will consist of one or more iterations of Steps (3) - (10) in which flow is reduced and finally driven off the set of longest paths. We can adopt the lexicographic pivot selection rule of Dantzig (7) in order to insure the convergence of the simplex on that problem. Thus, we can insure that problem (2) will be solved with $Z = 0$ if such a solution exists in a finite number of iterations and it follows that at such an event the value of L_B has been reduced.

The convergence of the dynamic programming algorithm follows from the fact that there are no negative directed cycles with respect to either \underline{d} or \underline{a} . At each iteration, ℓ is increased by one unit and hence it will reach its upper bound of $L_B - 1$ in a finite number of steps. We need not concern ourselves with labeling repeatedly around a cycle with zero length with respect to \underline{a} , since no paths including such a cycle would be cheaper than the residual simple path. Hence the labels would not be generated under the labeling procedure defined.

We conclude then that the algorithm converges in a finite number of steps to the min-max path, maximum flow on network $G(N,A)$.

CHAPTER IV

AN EXPANDED NETWORK FORMULATION OF THE MIN-MAX PATH FLOW PROBLEM

In this chapter the min-max path flow problem is interpreted as a maximum flow problem on an expanded version of the original network. The problem differs from the standard maximum flow problem in that it has imposed on it bundle constraints. Bundle constraints are inequalities which restrict the sum of flows on subsets of arcs of A . At this time, no totally graph theoretic methods have been developed to solve the maximum flow problem with bundle constraints except for some special planar graphs (3).

The algorithm developed in this chapter makes use of the decomposition concept of Dantzig and Wolfe (8). The master constraints number no more than the number of arcs in the original network $G(N,A)$, and there will be a single sub-problem of the shortest path or minimum cost maximum flow type.*

A Dynamically Expanded Network

If we interpret the length of arcs in the network $G(N,A)$ as the time required to transverse the arc, then we can define an expanded

*An alternative treatment would be to consider the bundle constraints as generalized upper bounds and make use of Dantzig and Van Slykes method for treating upper bounds on sums of variables. Using this approach one loses the network structure after the initial iteration however, and thus that procedure would be less attractive than the procedure presented here.

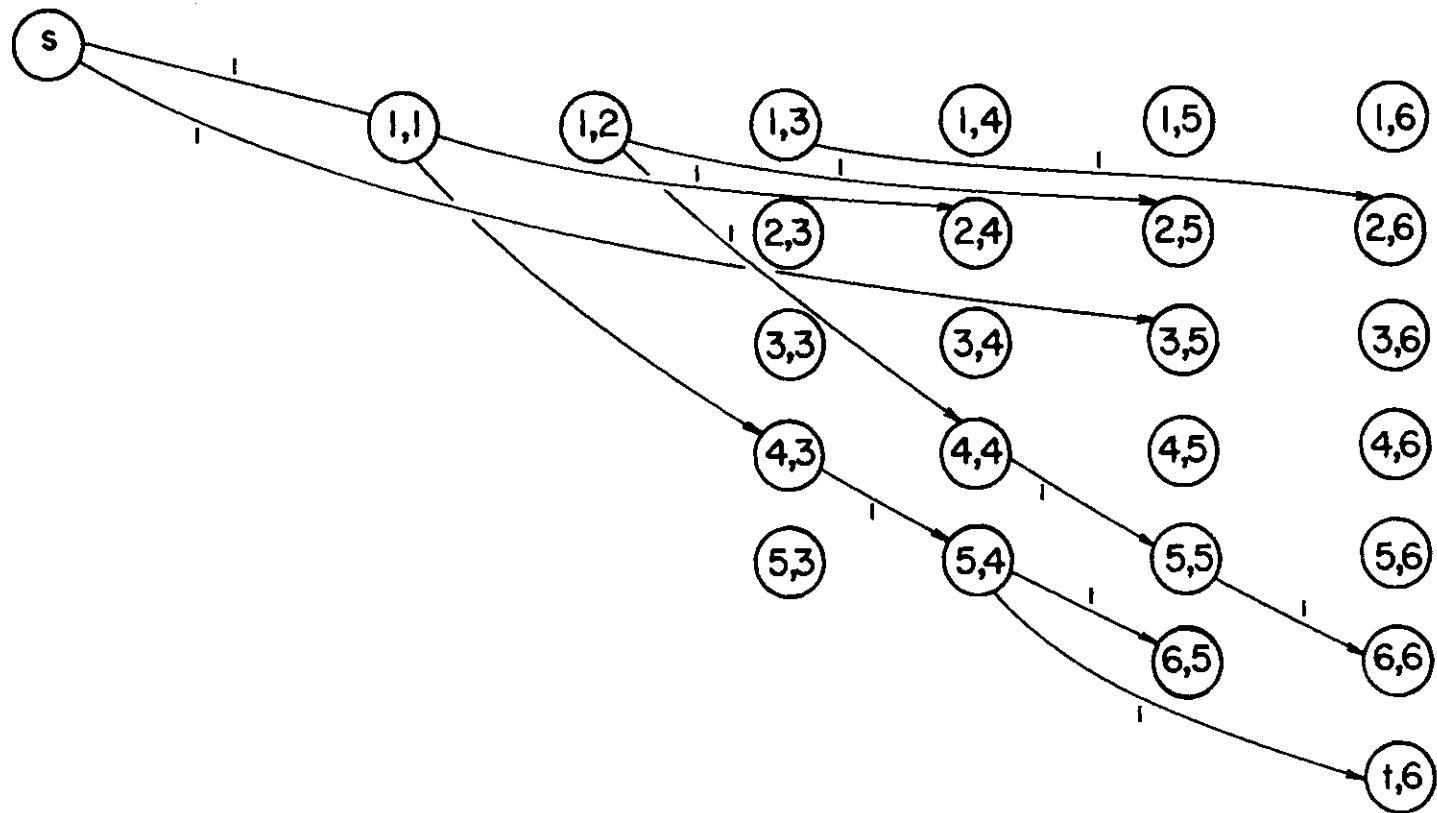


Figure 10. Six-Period Expanded Network

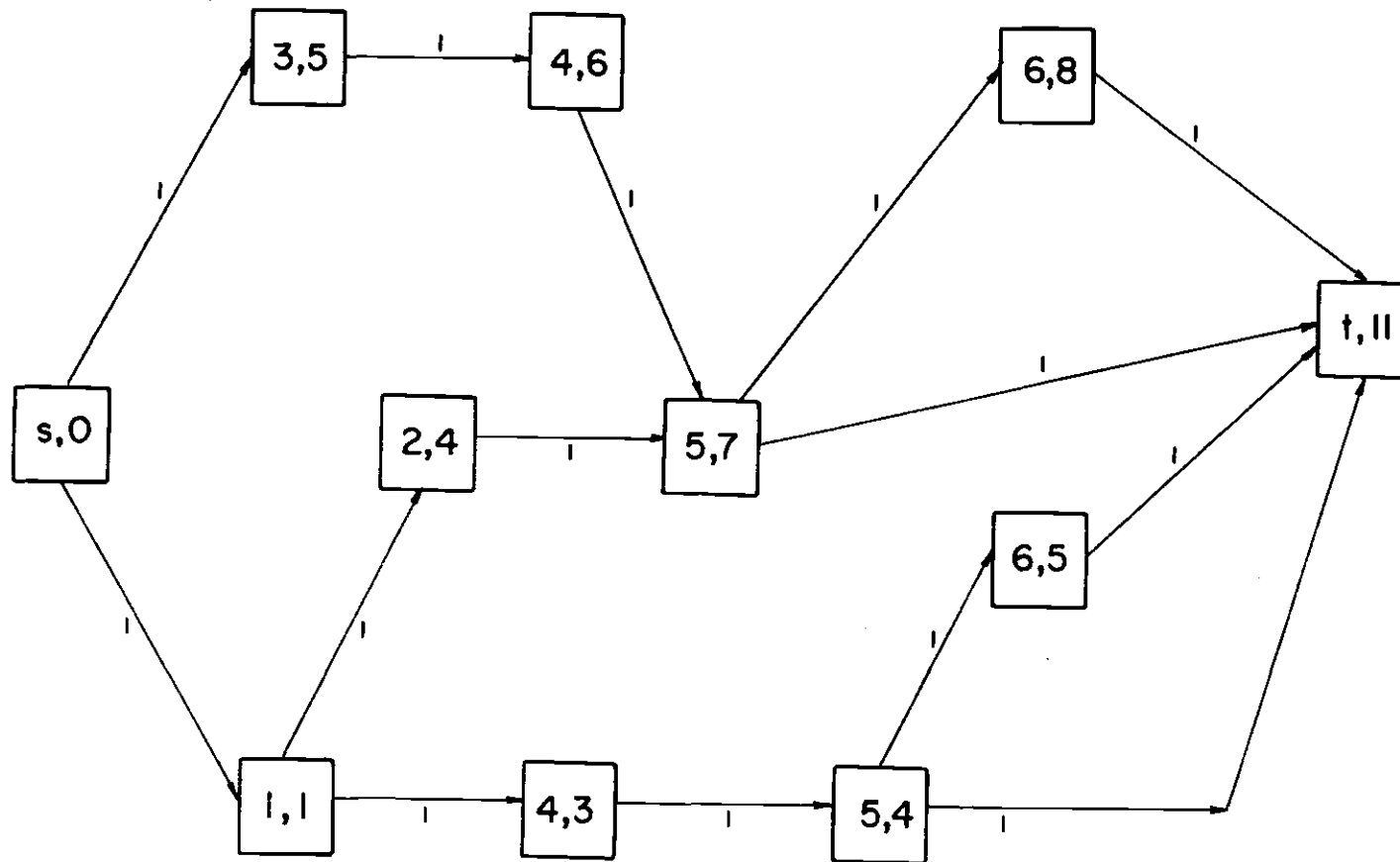


Figure 11. $D_{11}(G)$; 11-Period Expansion of $G(N,A)$

version of $G(N,A)$ in which all paths along which a unit of flow could reach t by time period L are represented but paths which have a transversal time greater than L are not included. We shall refer to this as the L -period expanded version of $G(N,A)$ and denote it $D_L(G)$.

The L -period expanded version $D_L(G)$ of $G(N,A)$ can be constructed from $G(N,A)$ as follows. The source s of $G(N,A)$ is the unique source s_L of $D_L(G)$. For node $x \in N$, $x \neq s$, there will be a sequence of nodes x_τ , $\tau = 0, 1, \dots, L$ in N_L . If $(x,y) \in A$ and $a(x,y)$ denotes the length of (time to transverse) arc (x,y) , then there will be in A_L a sequence of arcs $(x_\tau, y_{\tau-a(x,y)})$, $\tau = 0, 1, \dots, L-a(x,y)$. Each arc in the sequence will have capacity $b(x,y)$. We complete the network by adding a super sink \hat{t}_L and arcs (t_τ, \hat{t}_L) , $\tau = 0, 1, \dots, L$, with infinite capacity. All arcs in A_L have zero cost.

The network just described can be drastically reduced since many of the nodes and arcs generated are disconnected from either s_L or \hat{t}_L . A procedure for generating $D_L(G)$ from $G(N,A)$ and $D_{L-1}(G)$ is given at the end of this chapter. This procedure generates only nodes and arcs which are connected to both s_L and \hat{t}_L . Further reduction procedures are also discussed at that time.

Figure 10 is part of the 6-period expanded version of the network of Figure 4. Figure 11 is the reduced 11-period expanded version of the same network. We use (x,τ) on the network to represent node (x_τ) .

Comparison of $D_L(G)$ with the Full Dynamic Version of $G(N,A)$

The network $D_L(G)$ is an abridged version of Ford and Fulkerson's dynamically expanded network corresponding to $G(N,A)$. In the dynamic network flow problem the arc capacity is not an actual total capacity

for flow on the given arc but is instead a capacity per unit of time. Thus, flow can enter an arc in an amount equal to the arc capacity at time t and an equal amount can enter at time $t+1$. Therefore, each node is replicated for each period of time considered. A given path through the network may appear several times in the expanded version and it may be assigned flow in each case. On the other hand, the arcs in our problem have total flow capacities. Thus, it serves no purpose to allow the repetition of a given path in $D_L(G)$.

The replicates of a given node of $G(N,A)$ which appear in $D_L(G)$ represent the discrete points in time at which flow, leaving the source at time zero, could arrive at that node. Also, a given arc of $G(N,A)$ may appear more than once, or not at all, depending upon whether it can be used to transport flow, which can be made available at its initial node at a particular point in time, to its terminal node at a time which will enable it to ultimately reach the sink on or before time L , the maximum allowable time. In other words, a given arc will appear in $D_L(G)$ if it appears in a path of length L or less in $G(N,A)$.

The Relationship Between $G(N,A)$ and $D_L(G)$

In $D_L(G)$ there is a subset of arcs $(x_\tau, y_{\tau+a(x,y)})$, $\tau = 0, 1, \dots, L-a(x,y)$, corresponding to arc $(x,y) \in A$. We refer to this set of arcs in $D_L(G)$ as a bundle of arcs and denote it as $A_L(x,y)$.

Theorem 1

Suppose we have a feasible flow F^L on $D_L(G)$,

$$F^L = [f^L(x_\tau, y_{\tau+a(x,y)})], (x,y) \in A, \tau = 0, 1, \dots, L-a(x,y),$$

then the function on A defined by;

$$f(x,y) = \sum_{A_L(x,y)} F^L(x_\tau, y_{\tau+a(x,y)}), \quad (x,y) \in A,$$

is a flow function defined on A . That is, the network flow conservation equations are satisfied by:

$$F = [f(x,y)].$$

Proof. Since F^L is a feasible flow on $D_L(G)$,

$$f^L(x_\tau, N^L) - f^L(N^L, x_\tau) = \begin{cases} v^L, & x_\tau = s_L \\ 0, & x_\tau \neq s_L, \hat{t}_L \\ -v^L, & x_\tau = \hat{t}_L \end{cases}$$

For $x = s$:

$$f(s, N) = \sum_{y \in N} f^L(s_L, y_{a(x,y)}) = v^L$$

For $x = t$:

$$-f(N, t) = - \sum_{y \in N} \sum_{\tau=0}^{L-a(x,y)} f^L(y_\tau, x_{\tau+a(x,y)}) = - \sum_{\tau=0}^L f^L(t_\tau, \hat{t}_L),$$

but

$$- \sum_{\tau=0}^L f^L(t_\tau, \hat{t}_L) = - f^L(N, \hat{t}_L) = - v^L.$$

Thus,

$$-f(N, t) = -v^1$$

For $x \in N$, $x \neq s$, t :

$$f(x, N) - f(N, x) = \sum_N \sum_{\tau=0}^{L-a(x, y)} f^1(x_\tau, y_{\tau+a(x, y)}) - \sum_N \sum_{\tau=a(y, x)}^L f^1(y_{\tau-a(y, x)}, x_\tau)$$

If we define $f^1(x_\tau, y_{\tau'}) = 0$, for $\tau < 0$ or $\tau' > L$, then we can write:

$$f(x, N) - f(N, x) = \sum_N \sum_{\tau=0}^L f^1(x_\tau, y_{\tau+a(x, y)}) - \sum_N \sum_{\tau=0}^L f^1(y_{\tau-a(y, x)}, x_\tau)$$

We can now bring the second summation out and we have:

$$f(x, N) - f(N, x) = \sum_{\tau=0}^L \sum_N f^1(x_\tau, y_{\tau+a(x, y)}) - \sum_N \sum_{\tau=0}^L f^1(y_{\tau-a(y, x)}, x_\tau)$$

Each term inside the brackets is identically zero since $x_\tau \neq s_L, \hat{t}_L$.

Hence we have

$$f(x, N) - f(N, x) = \begin{cases} v^1, & \text{if } x = s \\ 0, & \text{if } x \neq s, t \\ -v^1, & \text{if } x = t, \end{cases}$$

which is a flow function defined on $G(N, A)$.

Q.E.D.

Theorem 2

If F^L is a flow defined on $D_L(G)$ and F is the corresponding flow on $G(N,A)$ as defined in theorem 1, then F is a feasible flow on $G(N,A)$ if and only if

$$f(x,y) = \sum_{A_L(x,y)} f^L(x_\tau, y_{\tau+a(x,y)}) \leq b(x,y), \quad (x,y) \in A.$$

Constraints of this type imposed on the flow on network $D_L(G)$ are called bundle constraints.

Proof. By theorem 1, the flow F corresponding to F^L is a flow on $G(N,A)$. If the bundle constraints are satisfied, then the individual arc capacities on $G(N,A)$ are satisfied. If:

$$f^L(x_\tau, y_{\tau+a(x,y)}) \geq 0,$$

then certainly

$$f(x,y) \geq 0.$$

Q.E.D.

At the end of this chapter we present an algorithm which generates the reduced version of $D_L(G)$ and prove, based on the construction of $D_L(G)$, that we can solve the min-max path flow problem by finding the minimum value of L for which

$$v_{max}^L \geq v_{max},$$

where v_{\max}^L denotes the value of the maximum bundle-constrained flow on $D_L(G)$ and v_{\max} denotes the maximum flow on $G(N,A)$.

We turn our attention now to the bundle-constrained maximum flow on $D_L(G)$.

Bundle Constrained Maximum Flow

The problem of finding a node-arc flow on $D_L(G)$ which corresponds to a maximal feasible flow on $G(N,A)$ can be stated as follows:

Find arc flows $f^L(x_\tau, y_{\tau+a(x,y)})$ to:

$$\text{maximize: } v^L \quad (1)$$

$$\text{subject to: } f^L(x_\tau, N^L) - f^L(N^L, x_\tau) = \begin{cases} v^L, & \text{if } x_\tau = s_L \\ 0, & \text{if } x_\tau \neq s_L, \hat{t}_L \\ -v^L, & \text{if } x_\tau = \hat{t}_L \end{cases} \quad (2)$$

$$\sum_{A_L(x,y)} f^L(x_\tau, y_{\tau+a(x,y)}) \leq b(x,y), \quad (x,y) \in A \quad (3)$$

$$f^L(x_\tau, y_{\tau+a(x,y)}) \geq 0, \quad (x_\tau, y_{\tau+a(x,y)}) \in A^L \quad (4)$$

Equations (3) are the bundle constraints. If the optimal solution to (1) - (4) has value

$$v_{\max}^L < v_{\max},$$

then there exists no feasible flow on $D_L(G)$ whose corresponding flow on $G(N,A)$ is a maximal feasible flow. In this case, we must expand the

network to $L+1$ time periods. That is, we generate $D_{L+1}(G)$ and resolve (1) - (4). If

$$v_{max}^L \geq v_{max}$$

and

$$v_{max}^{L-1} < v_{max},$$

then any path decomposition of the resulting flow is a min-max path flow on $G(N,A)$ and the min-max path length is L .

Iri (24) discusses the network with bundle constraints. He credits Kobayashi (34) with the development of the dual technique whereby the network flow problem with one bundle constraint can be reduced to a linear network flow problem in which the computations are carried out on the network. However he has the following to say about the general linear network flow problem with bundle constraints:

The network problems with bundle constraints cannot be solved in a purely graphical procedure, unlike ordinary network flow problems, but they require in general something like a simplex method for the general linear programming problems.

Example

It is helpful at this time to consider an example problem.

After further reductions, the network of Figure 11 is shown in Figure 12 with bundle arcs indicated by slash marks.* For convenience,

*Techniques for further reduction of the network will be discussed later.

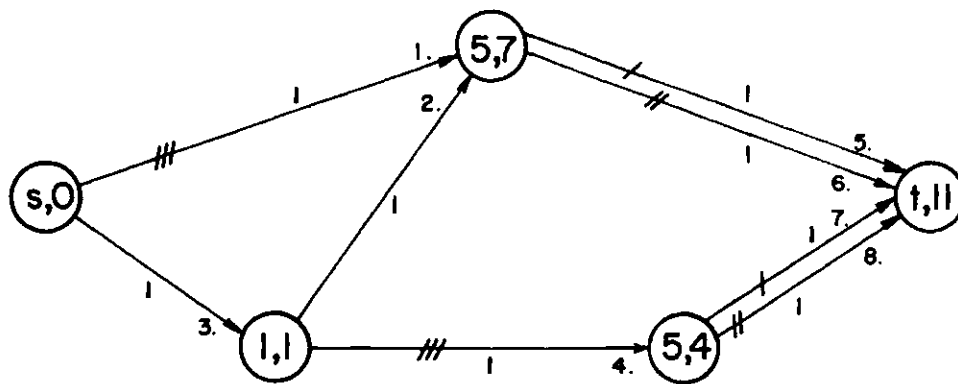


Figure 12. Network With Bundle Constraints

we use a single number to identify arcs. They are given in Figure 12 adjacent to each arc followed by a decimal.

The mathematical statement of the problem is:

$$\text{maximize: } v^{11}$$

subject to:

$$-v^{11} + f(1) + f(3) = 0 \quad (5)$$

$$-f(1) - f(2) + f(5) + f(6) = 0 \quad (6)$$

$$f(2) - f(3) + f(4) = 0$$

$$-f(4) + f(7) + f(8) = 0$$

$$v^{11} - f(5) - f(6) - f(7) - f(8) = 0$$

$$f(1) + f(4) \leq 1 \quad (7)$$

$$f(5) + f(7) \leq 1$$

$$f(6) + f(8) \leq 1$$

$$f(2) \leq 1$$

$$f(3) \leq 1$$

$$f(i) \geq 0, \quad i = 1, \dots, 8 \quad (8)$$

Solution of the Bundle Constrained Maximum Flow

The general form of the bundle constrained maximum flow problem is:

$$\text{maximize: } v^1 \quad (9)$$

$$\text{subject to: } B \underline{X} \leq \underline{b}_1 \quad (10)$$

$$A_1 \underline{X} - A_2 v^L = \underline{0} \quad (11)$$

$$\underline{X} \geq \underline{0}.$$

B is the bundle matrix corresponding to constraints (7) of the example problem, A_1 is the node-arc incidence matrix of $D_1(G)$, and A_2 is the vector $(-1, 0, 0, \dots, 0, 1)$. The dimension of this vector is equal to the number of nodes in the network. The maximum dimensions of B, depending upon the amount of possible reduction, are $m \times m$, where m is the number of arcs in $G(N, A)$. This includes bundles containing only one arc, i.e., upper bounds on arc flows are included where necessary. Those arcs which belong to bundles of more than one arc do not require separate constraints for individual arc flows since the bundle capacity is identical to the individual arc capacities.

Equations (11) are the conservation of flow equations, thus, feasible solutions to these equations can be generated by the use of one of the efficient maximum flow algorithms. It would be desirable to retain this structure on (11). To do so we can make use of the Dantzig-Wolfe decomposition concept (8). We shall do so by treating (10) as the master constraints and (11) as a single subproblem. Before continuing with this we will digress briefly to discuss the decomposition procedure. Our discussion follows that of Gass (18).

Decomposition

Consider the problem

$$\text{maximize: } \underline{C}'\underline{X} \quad (12)$$

$$\text{subject to: } B \underline{X} + I \underline{W} = \underline{b}_1 \quad (13)$$

$$A \underline{X} = \underline{b}_2 \quad (14)$$

$$\underline{X}, \underline{S} \geq 0 \quad (15)$$

The solution space to (14) is a bounded convex set. Thus, any point in this set can be expressed as a convex linear combination of the extreme points of this set. If we let $\underline{X}_1, \underline{X}_2, \dots, \underline{X}_n$ be the extreme points of the set S , where

$$S = \{\underline{X} \mid A\underline{X} = \underline{b}_2, \underline{X} \geq 0\}$$

then for any element \underline{X} in S , there exist nonnegative real numbers, $\lambda_1, \dots, \lambda_n$, such that

$$\underline{X} = \lambda_1 \underline{X}_1 + \lambda_2 \underline{X}_2 + \dots + \lambda_n \underline{X}_n. \quad (16)$$

The values $\lambda_1, \dots, \lambda_n$ satisfy the convexity condition:

$$\lambda_1 + \lambda_2 + \dots + \lambda_n = 1.$$

Any solution to problem (12)-(15) must, of course, satisfy (14) and hence must belong to S . As a result, if we let \underline{X}_ℓ denote the general

solution to (13)-(15), then we can express this solution in terms of the multipliers λ_1 and the extreme points of S as given by (16).

Substituting (16) into (13) we get the equations:

$$B(\lambda_1 \underline{X}_1 + \lambda_2 \underline{X}_2 + \dots + \lambda_n \underline{X}_n) + IW = b_1.$$

or

$$B\underline{X}_1 \lambda_1 + B\underline{X}_2 \lambda_2 + \dots + B\underline{X}_n \lambda_n + IW = b_1.$$

Also, substituting in (12), we have

$$\underline{C}'(\lambda_1 \underline{X}_1 + \lambda_2 \underline{X}_2 + \dots + \lambda_n \underline{X}_n).$$

or

$$\underline{C}'\underline{X}_1 \lambda_1 + \underline{C}'\underline{X}_2 \lambda_2 + \dots + \underline{C}'\underline{X}_n \lambda_n.$$

If we assume then that the extreme points of S are known, then the problem (12)-(15) can be expressed as that of finding positive real numbers λ_1 to:

$$\text{maximize: } (\underline{C}'\underline{X}_1)\lambda_1 + (\underline{C}'\underline{X}_2)\lambda_2 + \dots + (\underline{C}'\underline{X}_n)\lambda_n \quad (17)$$

$$\text{subject to: } (B\underline{X}_1)\lambda_1 + (B\underline{X}_2)\lambda_2 + \dots + (B\underline{X}_n)\lambda_n + IW = b_1 \quad (18)$$

$$\lambda_1 + \lambda_2 + \dots + \lambda_n = 1. \quad (19)$$

We of course do not require an enumeration of all the extreme points of S . We will generate the activity vectors as follows. Suppose we have a basic feasible solution to (17)-(19). Let (π, γ) be the corresponding simplex multipliers, where π is a vector corresponding to (18) and γ is a scalar corresponding to the single convexity constraint (19).

Suppose now that \underline{X}_j is an extreme point of S which is not represented in the current basic feasible solution to (17)-(19). The corresponding activity vector is

$$\underline{P}_j = [\underline{C}'\underline{X}_j, \underline{B}\underline{X}_j, 1] . \quad (20)$$

The vector \underline{P}_j is a candidate to enter the basis in (17)-(19) if

$$\pi \underline{B}\underline{X}_j + \gamma - \underline{C}'\underline{X}_j < 0 . \quad (21)$$

If there are no extreme points \underline{X}_j of S for which (21) holds, then the current solution is optimal. Thus, we can determine whether the current solution to (17)-(19) is optimal and if not find a candidate to enter the basis by finding the extreme point of S which maximizes

$$\pi \underline{B}\underline{X}_j - \underline{C}'\underline{X}_j .$$

Suppose \underline{X}_j^* is such a vector, then if

$$\pi \underline{B}\underline{X}_j - \underline{C}'\underline{X}_j + \gamma \geq 0 ,$$

the current solution is optimal and if not, the vector \underline{P}_j^* , defined by (20), is a candidate to enter the basis. The vector \underline{X}_j^* can be obtained by solving

$$\text{minimize: } (\pi B - \underline{C}') \underline{X}_j \quad (22)$$

$$\text{subject to: } A \underline{X}_j = \underline{b}_2 \quad (23)$$

$$\underline{X}_j \geq 0. \quad (24)$$

Throughout this discussion we have assumed that S is bounded. Only a modification is required if this is not the case. In our problem, any solution feasible to the master constraints (18) is feasible to the subproblem (23). Thus, (19) is always satisfied if (18) is satisfied. Hence we can drop the convexity constraint. We rewrite the problem as

$$\text{maximize: } (\underline{C}' \underline{X}_1) \lambda_1 + (\underline{C}' \underline{X}_2) \lambda_2 + \dots + (\underline{C}' \underline{X}_n) \lambda_n \quad (25)$$

$$\text{subject to: } (B \underline{X}_1) \lambda_1 + \dots + (B \underline{X}_n) \lambda_n + \underline{IW} = \underline{b}_1$$

$$\underline{\lambda} \geq 0,$$

where \underline{X}_i , $i = 1, \dots, n$, are the extreme point solutions to S .

$$S = \{ \underline{X} \mid A \underline{X} = \underline{b}_2, \underline{X} \geq 0 \}.$$

An initial basic feasible solution to (25) is given by:

$$\underline{IW} = \underline{b}_1.$$

Since the convexity constraint has been dropped, the dual variable γ is no longer in the problem. Thus, the dual variables are given by π and we wish to determine whether the optimal solution to (22)-(24) is negative. Let us assume that the matrix A of (14) is written

$$A = [-A_2 : A_1],$$

where A_1 and A_2 are the matrices of (11). Also, let

$$\overline{B} = [\underline{0} : B],$$

$$\overline{X}' = [v : x_1 \dots x_m]$$

$$\underline{c}' = [1 : 0 \ 0 \ 0 \dots 0].$$

Then

$$\pi \overline{B} - \underline{c}' = \pi B - 1,$$

and solving (22)-(24) can be done by searching for a path whose length with respect to arc lengths πB is less than one. If the shortest path has length greater than or equal to one, then the current solution to

(25) is optimal. If not, as indicated before, the shortest path identified is a candidate to enter the basis. Since (22)-(24) is unbounded, this path can be assigned any amount of flow consistent with (25).

The arc lengths, πB , can be assumed to be nonnegative since B has all nonnegative elements and if $\pi_i < 0$, for some i , the corresponding slack variable can be introduced into the basis and π_i becomes zero.

An alternative way of formulating this bundle-constrained maximum flow problem is to introduce all individual arc capacity constraints into the subproblem and delete all bundle-constraints from the master problem which involve only one arc. A solution to the master problem is now not necessarily feasible to the subproblem. Thus, it is necessary to include the convexity constraint.

Let \underline{b}_2 be the vector of individual arc capacities of $D_i(G)$. The problem can be written

$$\text{maximize: } v^L$$

$$\text{subject to: } B \underline{X} \leq \underline{b}_1$$

$$A_1 \underline{X} - A_2 v^L = \underline{0}$$

$$\underline{0} \leq \underline{X} \leq \underline{b}_2$$

The master problem is the same as before as given by equations (25) except that it has fewer constraints and it includes the convexity constraint. A vector \underline{X}_j is a candidate to enter the basis in the master

problem if

$$\pi B (\underline{X}_j; v^L)' - \underline{C}'(\underline{X}_j; v^L)' + \gamma < 0.$$

We are again searching for the vector \underline{X}_j which will

$$\text{minimize: } (\pi B - \underline{C}')(X; v^L)'$$

$$\text{subject to: } A_1 \underline{X} - A_2 v^L = \underline{0}$$

$$\underline{0} \leq \underline{X} \leq \underline{b}_2$$

Since $\underline{C}' = (\underline{0}':1)$, and $B_{i, m+1} = 0$, $i = 1, 2, \dots, m_1$, we can rewrite the problem as:

$$\text{minimize: } \pi B \underline{X} - v^L \tag{26}$$

$$\text{subject to: } A_1 \underline{X} - A_2 v^L = \underline{0}$$

$$\underline{0} \leq \underline{X} \leq \underline{b}_2 .$$

If \underline{X}_{j*} is the optimal solution and

$$\pi B \underline{X}_{j*} - v^L + \gamma < 0,$$

then the current solution to the master problem is optimal and hence we have the maximum bundle constrained flow on $D_L(G)$. If

$$\pi B \underline{X}_j^* - v^L + \gamma < 0$$

then \underline{X}_j^* is brought into the basis and we continue. The subproblem (26) is a minimum cost flow problem and it can be solved by Busacker and Gowen's algorithm or any of the algorithms available for this problem (5).

Due to the relation between the bundle constraints and the arc capacity constraints, the following modification is made to the standard decomposition method. The algorithm is outlined here.

1. Solve the maximum flow problem on $D_L(G)$. This is what would be done the first time the subproblem is solved in the decomposition method.
2. Select the first bundle constraint which is not presently in the master problem and check to see if it is satisfied by the current solution. If so, go to the next bundle constraint not currently in the master problem. If not, go to step 3.
3. Introduce this constraint into the master problem and introduce an artificial variable for this constraint into the basis. Go to step 4.
4. Find a feasible solution to the master problem using the phase I procedure. Return to step 2.
5. Terminate the procedure when all constraints not in the master problem have been checked without a violation.

The Expanded Network Algorithm

We now present an outline of the expanded network algorithm after which some additional comments will be considered.

1. Find v_{max} on $G(N,A)$.
2. Expand $D_L(G)$ until the sum of the capacities on all arcs leading into (t,L) equals or is greater than v_{max} .
3. Solve the bundle constrained maximum flow problem on $D_L(G)$.
4. If $v_{max}^L \geq v_{max}$, terminate, otherwise
5. Set $L = L+1$, expand $D_L(G)$ and return to step 3. Repeat until $v_{max}^L = v_{max}$.

The labels which are recorded when $D_L(G)$ is generated will be used also in generating $D_{L+1}(G)$. Thus, it is not necessary to begin step (5) anew each iteration. Also, the optimal solution on $D_L(G)$ is a feasible solution to $D_{L+1}(G)$. Hence it can be used to provide a good initial solution to $D_{L+1}(G)$.

Solution to the Example Problem

Suppose we know that the maximum flow on our example problem (see Figure 4) is

$$v_{max} = 2,$$

and suppose we know that

$$v_{max}^{10} = 1.$$

We want to determine v_{sax}^{11} . Figure 11 is the 11-period expanded version of the original network. The arc capacities and bundle constraints are given in Figure 12. In Figure 13 the arcs are numbered to simplify the problem matrix. Table 6 is the simplex tableau corresponding to equations (5)-(7) with appropriate slack variables added to (7), and with all individual arc capacities stated separately from the bundle constraints. We begin by ignoring the bundle constraints (the first three constraints in Table 6 following the objective row). Thus, we are just finding the maximum flow from s to $(t, 11)$. A solution is $\bar{X}_1 = (2, 1, 0, 1, 1, 1, 0, 1, 0)$.

The first value is the flow v and the remaining values are the arc flows. We check this solution against the constraints in the master problem.

The first constraint is violated since

$$\bar{x}_2 + \bar{x}_6 = 2 > 1.$$

Hence the master problem will now be

$$\text{maximize: } v - x_{s1}$$

$$\text{subject to: } 2\lambda_1 + x_{s1} - x_{s1} = 1$$

$$\lambda_1 = 1,$$

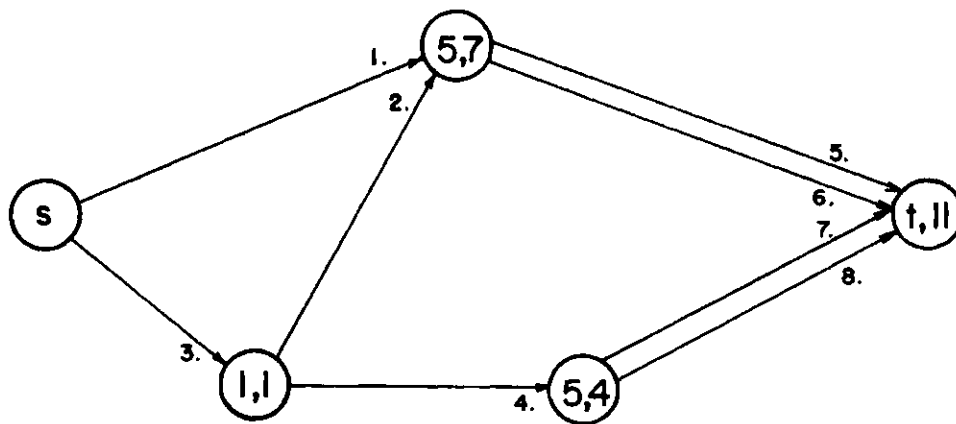


Figure 13. Arc Identification for Example Network

where X_{a1} is an artificial variable, given a cost of -1 to drive it from the basis. The current basic solution is

$$x_{a1} = 1$$

$$\lambda_1 = 1.$$

The new set of simplex multipliers is given by

$$[\pi; \gamma] = [-1:2] \begin{bmatrix} -1 & 2 \\ 0 & 1 \end{bmatrix} = [1:0]$$

$$\pi = [1], \quad \gamma = 0.$$

The subproblem is

$$\begin{aligned} \text{maximize: } & v^L - x_1 && -x_4 \\ & -v^L + x_1 && +x_3 && = 0 \\ & & -x_1 - x_2 && +x_6 + x_8 && = 0 \\ & & & x_2 - x_3 + x_4 && = 0 \\ & & & & -x_4 && +x_7 + x_8 && = 0 \\ & & & & & v^L && -x_6 - x_8 - x_7 - x_8 && = 0 \\ & 0 \leq x_i \leq 1, & i=1, \dots, 8 \\ & v^L \geq 0. \end{aligned}$$

This is a minimum cost maximum flow problem. The network with arc costs and capacities is given in Figure 14.

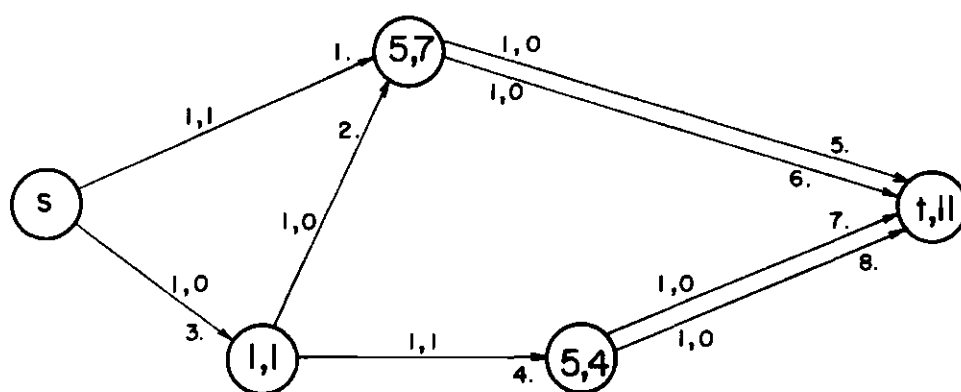


Figure 14. Reduced Network

Table 6. Simplex Tableau for the Bundle-Constrained Maximum Flow Problem

<u>b</u>	v	1	2	3	4	5	6	7	8	<u>S</u> ₁	<u>-S</u> ₂	-Z
0	1	0	0	0	0	0	0	0	0	<u>0</u>	1	1
1 1 1	0 0 0	1 0 0	0 0 0	0 0 0	1 0 0	0 1 0	0 0 1	0 1 0	0 0 1	I	<u>0</u>	<u>0</u>
<u>0</u>	-1 0 0 0 1	1 -1 1	1 -1 1	1 -1 -1	1 1 -1	1 1 -1	1 1 -1	1 1 -1	1 1 -1	<u>0</u>	<u>0</u>	<u>0</u>
1 1 1 1 1 1 1 1 1		1	1	1	1	1	1	1	1	<u>0</u>	I	<u>0</u>

The optimal solution is:

$$\begin{bmatrix} v^1 \\ \underline{x}_1 \end{bmatrix}^* = [1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0]^T,$$

with value 1.

Since $\gamma = 0$.

$$C_2 - Z_2 = v^1 - \pi B \underline{x}_2 - \gamma = 1 > 0.$$

Hence, we wish to bring λ_1^* into the basis of the master problem.

$$\underline{P}_2 = [C' \underline{x}_2 : B \underline{x}_2 : 1]$$

$$\underline{P}_2 = [1 : 0 : 1]$$

We use the revised simplex notation.

	X^B	\hat{B}^{-1}	\bar{P}_2	P_2
$-v^1$	$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$
$\leftarrow x_{a1}$				
λ_1				

$-v^1$	$\begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} 1 & \frac{1}{2} & -1 \\ 0 & -\frac{1}{2} & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix}$
λ_2		
λ_1		

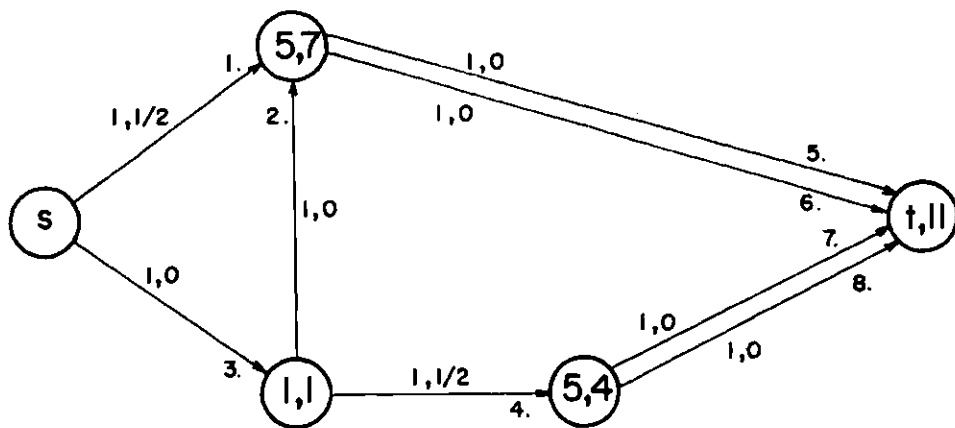


Figure 15. Network With Modified Costs

$$\pi B = [0 \ \frac{1}{2} \ 0 \ 0 \ \frac{1}{2} \ 0 \ 0 \ 0 \ 0].$$

The subproblem is now the minimum cost flow problem on the network of Figure 15.

The optimal solution is:

$$\begin{bmatrix} v_3^L \\ \underline{x}_3 \end{bmatrix} = [2 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0]',$$

with value 1 .

$$C_3 - Z_3 = v_3^L - \pi B \underline{x}_3 - \gamma = 1 + 1 = 3\frac{1}{2} > 0 ,$$

Thus, this vector is a candidate to enter the basis .

$$P_3 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{array}{l} -v^L \\ +\lambda_2 \\ \lambda_1 \end{array} \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & -1 \\ 0 & -\frac{1}{2} & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{array}{c} \bar{P}_3 \\ \begin{bmatrix} 3\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \end{array} \begin{array}{c} P_3 \\ \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \end{array}$$

$$\begin{array}{l} -v^L \\ \lambda_3 \\ \lambda_1 \end{array} \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & -4 \\ 0 & -1 & 2 \\ 0 & 1 & -1 \end{bmatrix}$$

Since $v_{\max}^L = v_{\max}$, we have an optimal basic feasible solution to the current problem (with one bundle constraint).

We go back now and check the current solution against each bundle constraint. The bundle constraints are the first three constraints given in Table 6. Since $\lambda_3 = 1$, the solution is

$$v^L = 2$$

$$x_1 = x_2 = x_3 = x_5 = x_6 = 1$$

$$x_4 = x_7 = x_8 = 0.$$

Substituting in the bundle constraints we have:

$$x_1 + x_4 = 1 \leq 1$$

$$x_5 + x_7 = 1 \leq 1$$

$$x_6 + x_8 = 1 \leq 1$$

All bundle constraints are satisfied and the current solution is optimal on $D_{11}(G)$. Since $v_{\max}^{11} = 2 = v_{\max}$, this solution is optimal min-max path flow on $G(N,A)$, the network of Figure 4.

Convergence

Since L^* is finite and L is increased by one unit each time steps (3)-(5) are performed, the speed of convergence depends upon the two sub-algorithms. The generation of $D_L(G)$ is certainly a finite process since there is a finite number of arcs between any pair of nodes. The convergence of the decomposition procedure of step (3) is the only real question.

Since step (3) makes use of the decomposition method to solve the linear programming problem, i.e., the constrained shortest path problem, we must consider the convergence of the master problem and the convergence of the subproblem.

The subproblem is a shortest path problem with all nonnegative arc lengths, hence it terminates in a finite number of iterations. (See Ford and Fulkerson's combinatorial algorithm (12)).

We can use the lexicographical rule for selecting the vector to leave the basis in case of ties in a given iteration of the revised simplex method of solving the master problem. Dantzig proves that this rule will insure the finite convergence of the revised simplex method (7). As is normally the case for linear programming algorithms, no provisions are incorporated to prevent cycling. However, if it turns out that the structure of these problems causes cycling to occur, we can incorporate the lexicographic choice rule.

Improving Computational Efficiency

There are three areas in which improvements may be made to the computational efficiency of the algorithm. First, since we must begin at some specific value of L and increase the value of L until a solution is obtained, we can reduce the number of iterations of the algorithm required by beginning with L as close to L^* , the optimal value of L , as possible. Secondly, the computational efficiency of the decomposition algorithm for solving the bundle constrained maximum flow on D_L will be improved by reducing the number of nodes and arcs in D_L to the minimum necessary (as was done for the example problem). Thirdly, provisions can be incorporated to allow recognition of the optimal solution as soon

as it is obtained. We shall now elaborate on these three approaches in the order they were presented.

The Initial Value of L . We mentioned earlier that in expanding D_L , it should be expanded until the sum of the capacities of all arcs leading into (t, L) equals or exceeds v_{max} . We can also check to ensure that the capacity of all arcs leading out of s equals v_{max} . We can determine an upper bound by finding the longest path in a path decomposition of the maximum flow on $G(N, A)$. We can find generally tighter bounds on L^* by solving the minimum cost maximum flow problem. If Z is the cost and L_u is the longest path in any path decomposition of the flow, then

$$L_\ell = \langle Z/v_{max} \rangle \leq L^* \leq L_u,$$

where $\langle x \rangle$ denotes the smallest integer greater than or equal to x .

The following remark and lemma also provide other constraints on the smallest value of L for which it is possible that $v_{max}^L = v_{max}$.

Remark 1

If we disregard the bundle constraints and find the maximum flow on $D_L(G)$ to be less than v_{max} , then $L^* > L$.

Lemma 1

If we disregard the bundle constraints and any minimum cut-set of $D_L(G)$ contains more than one arc belonging to the same bundle, then $L^* > L$, if $v_{max}^L \leq v_{max}$.

Proof. If two or more arcs of the same bundle belong to any minimum cut-set, then any maximum flow must have these arcs saturated.

Since the bundle has the same capacity as each arc in the bundle there is no way the bundle constraint can be satisfied by a maximal flow.

Q.E.D.

These two tests have not been incorporated into the algorithm so that nothing is known about their effect on the solution time.

Searching on L. The solution procedure presented begins at the largest known lower bound on L. It might be more efficient, however, to use a direct search method on L. This is possible since we know that:

$$v_{max}^L \geq v_{max} \Rightarrow L^* \leq L$$

and

$$v_{max}^L < v_{max} \Rightarrow L^* > L.$$

Network Reductions. Step (4) of the network expansion algorithm outlined some reduction techniques to reduce the size of the network. Lemma 1 allows some additional reduction. If a given arc of $D_L(G)$ is in a minimum cut-set but no other arcs of its bundle are in any minimum cut-set then we must solve the bundle-constrained maximum flow on $D_L(G)$. Before doing so, however, we can delete all arcs that are in a common bundle with any arc in any minimum cut-set if the maximum flow (without regard to the bundle constraints) is equal to v_{max} .

Another way the network $D_L(G)$ can be reduced is to delete any node having only one arc entering it and only one arc leaving it. These two arcs are replaced by a single arc between the initial node of the

former and the terminal node of the latter. This arc will have capacity equal to the minimum capacity of the two original arcs. The arc will belong to both bundles of the original arcs. This will be helpful only to eliminate arcs not in bundles.

Termination Procedure. Due to the likelihood of degeneracy in the bundle constrained maximum flow problem it is quite possible, for a given value of L , several iterations will be required after v_{\max}^L is obtained before it is recognized as the optimal solution. Thus, after each iteration of the master problem v^L should be compared to v_{\max} . If they are equal, then the current solution is optimal to the min-max path problem, so that no further improvement on v^L is required. Thus we terminate the algorithm at this point without having to verify that the current solution is optimal to $D_L(G)$.

Algorithm for Generating Dynamically Expanded Network

Let $G(N,A)$ be a static network with arc costs or lengths given by $a(x,y)$. Let L be an arbitrary positive integer.

The following algorithm will generate the corresponding L -period dynamic representation $D_L(G)$ of the network $G(N,A)$.

Step 1. Solve the shortest path problem from the source s to all nodes of N . Identify all alternate labels with respect to the labeling method of Ford and Fulkerson (12). The labels used here shall be of the form $[-,-,-]$. The first element designates the node from which this label originates, the second denotes the index of this label on the current node, and the last element denotes the length of the path from s to the current node along the path of labels (nodes) indicated. Thus, a label on any node can be traced back to the origin node, s . The label is associated

with one or more partial paths from the origin to that node, and the value of the fourth element in the label is the length of that partial path. Henceforth, we shall refer to partial paths of a specified length to node y instead of labels on node y . The fact that a given label is associated with one or more paths of the same length causes no confusion. Figure 16 is an example network with all optimal labels for shortest paths from the source s to all nodes of A .

In the context of this labeling procedure, the path (6,1) will refer to the partial path(s) associated with the first label on node 6. That path consists of arcs $\{1,4,7,9\}$ and has length five as indicated by the last element of the label.

Step 2. Begin with the first node and attempt to generate a new partial path to this node the length of which exceeds the length of current longest path to this node by exactly δ units. δ is initially set at zero. When a new label (partial path) is obtained, δ is reset at zero and the process begins again with the first node. If a new label cannot be generated for a given node, the next node is considered until the terminal node is reached. If the terminal node is reached and cannot be labeled, increase δ by one unit and begin the search again at the first node.

If L is the number of periods desired, the labeling procedure terminates anytime δ reaches the value $L - \pi(t, \tau) + 1$, where $\pi(t, \tau)$ denotes the length of the longest partial path to t which has been generated. Figure 17 is the network of Figure 16 with labels up to those for paths of length 11.

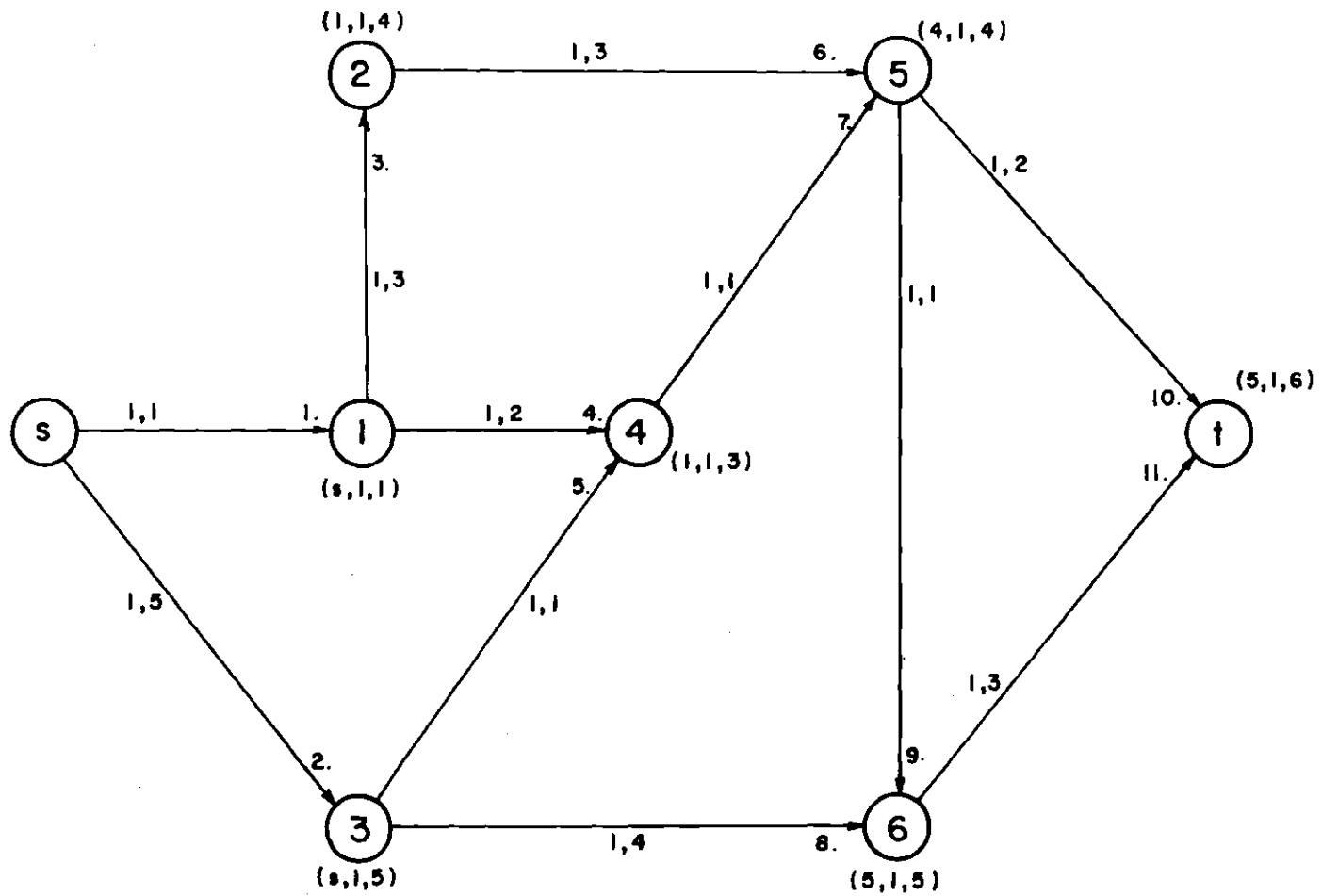


Figure 16. Network With all Shortest Path Labels

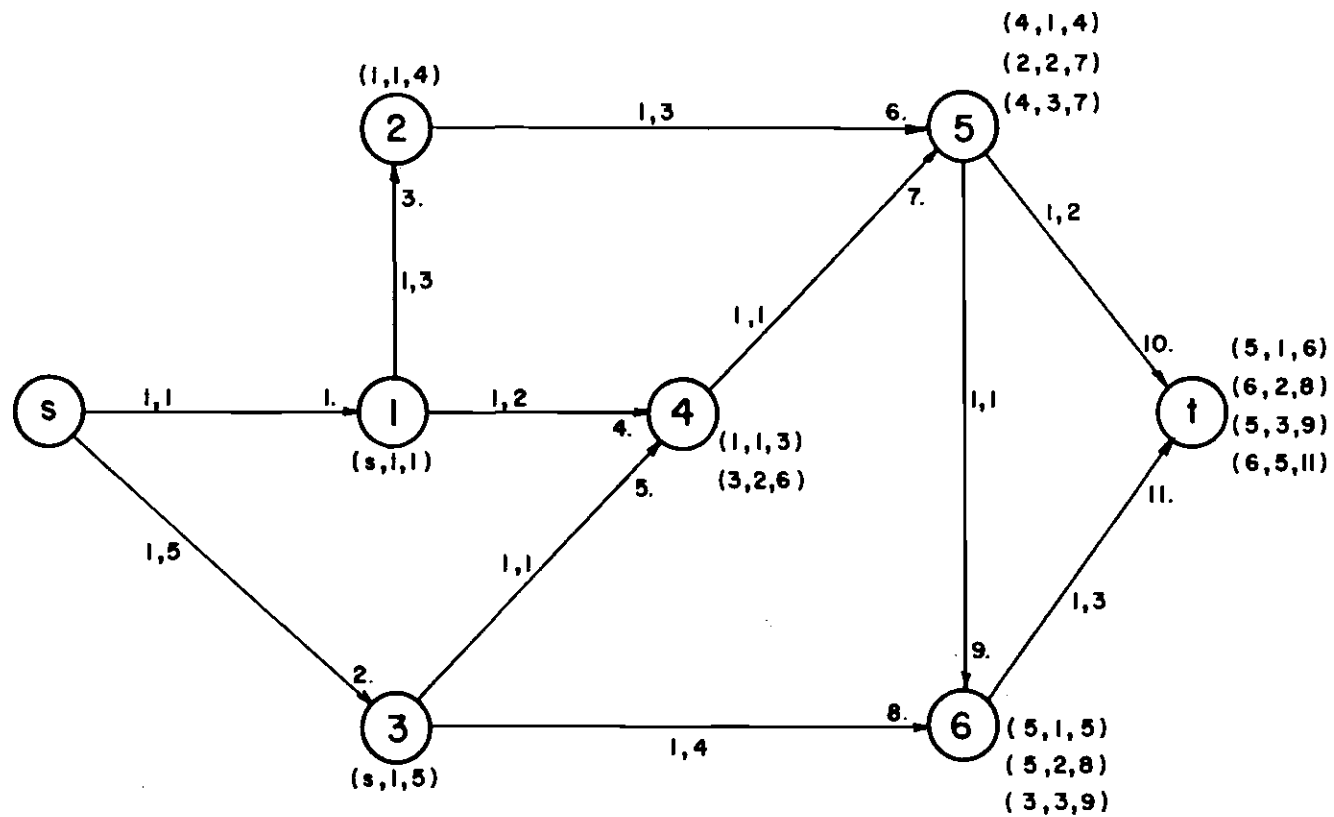


Figure 17. Network With Labels up to Period 11

Step 3. Generate the nodes and arcs of $D_L(G)$ from the labels on $G(N,A)$. Initially include a node for each label on $G(N,A)$, except, include only one node for node t . Call that node (\hat{t}, L) . Each label also designates an arc which will be in $D_L(G)$. The capacity of this arc will be the same as its capacity in $G(N,A)$. For example, if (x, K, τ) is a label on node y , then there will be a node in $D_L(G)$ corresponding to node y at time period τ . Let (s, L) denote the source. There will also be a node corresponding to node x at time period $\tau - a(x, y)$. There will be an arc in $D_L(G)$ joining these two nodes. The capacity of this arc will be $b(x, y)$ and it will have a length equal to $a(x, y)$.

At times we shall identify a node in $D_L(G)$ by an ordered pair of numbers (x, τ) where x denotes the corresponding node in $G(N,A)$ and τ denotes the time period at which the named node appears in $D_L(G)$. Thus we observe the following

Remark 2. If arc j of $D_L(G)$ has end points (x, τ_1) and (y, τ_2) respectively, then the length of j is

$$l(j) = \tau_2 - \tau_1 \quad (27)$$

except for arcs generated from labels $(-, -, \tau \leq L)$ on node t . In this case the length of the arcs will be less than that indicated by (1).

In general then

$$l(j) \leq \tau_2 - \tau_1 \quad (28)$$

Step 4. Reduce the network by deleting nodes and arcs by alternately carrying out the following two steps until one or the other yields no further reductions to the network.

4a. Delete all nodes except the terminal node which have no arcs emanating from them.

4b. Remove all arcs incident to only one node.

This completes a general statement of the algorithm for generating the L-period expanded version of a network. (See Figure 11.)

The pair of numbers in the boxes (nodes) designate the corresponding node of the original network and the time period. The algorithm here generates the same network as the reduced version of the full L-period expansion of $G(N,A)$.

Proof of the Relationship Between $D_L(G)$ and $G(N,A)$

In order to establish the relationship between $G(N,A)$ and $D_L(G)$ we must establish a correspondence between the nodes, arcs, and paths of the two networks.

Nodes. The source node of the two networks are identical and no other node of $D_L(G)$ corresponds to the source node of $G(N,A)$. Likewise, the two terminal nodes are identical. The terminal node in $D_L(G)$ corresponds to node t of $G(N,A)$ at time L and at all previous time periods at which flow could reach t . As indicated previously, node (x,τ) in $D_L(G)$ corresponds to node x of $G(N,A)$ at time period τ .

Lemma 2

If there is an arc (x_{τ_1}, y_{τ_2}) in $D_L(G)$, then there is an arc (x,y) in $G(N,A)$, with $a(x,y) = \tau_2 - \tau_1$.

Proof. The arc in $D_L(G)$ is generated by labeling node y from x . Such a label is produced only if there is an arc (x,y) in $G(N,A)$.
Q.E.D.

Paths. Based on the correspondence between arcs of $D_L(G)$ and $G(N,A)$ and as a consequence of the construction of $D_L(G)$, it follows that any simple path (from source to sink) in $D_L(G)$ corresponds to a (not necessarily simple) path in $G(N,A)$. For completeness we state this as a lemma.

Lemma 3

Let $\{s, x_1, \tau_1; x_2, \tau_2; \dots; x_n, \tau_n; t_{\tau_{n+1}}, \hat{t}_L\}$ be a simple path in $D_L(G)$. Then $\{s, x_1, x_2, \dots, x_n, t\}$ is a path in $G(N,A)$.

Proof. This follows immediately from the definitions of path and simple path, and lemma 2.

Remark 3

If there are no directed cycles in $G(N,A)$ then every simple path in $D_L(G)$ corresponds to a simple path in $G(N,A)$.

Proof. We are assuming all positive arc lengths. Thus, any directed cycle would have positive length. By construction, any cycle in $G(N,A)$ would then be represented by a path in $D_L(G)$ that contained node (x, τ_i) and (x, τ_j) for the node that appeared twice in the cycle in $G(N,A)$ and for two distinct values of τ . If there is no partial path from x back to x , which passes through some other node in between, then no such pairs of nodes, connected by an arc or a sequence of arcs will occur in $D_L(G)$.

This last lemma and remark are not necessary to the development but are given to help characterize the relationship between $D_L(G)$ and

$G(N,A)$. Hence, our original statement allowing the relaxation of the requirement for all positive arc lengths remains valid.

If there are directed cycles in $G(N,A)$ or if the requirement for positive arc lengths in $G(N,A)$ is relaxed, then there is the possibility of a simple path in $D_L(G)$ corresponding to a path in $G(N,A)$ containing a directed cycle. Even so, this will cause no problems as we can derive the flow on $G(N,A)$ by assigning the flow on each path of $D_L(G)$ to its corresponding path in $G(N,A)$ and then deleting from each path in $G(N,A)$ which has flow assigned, its directed cycles. This will produce a new path flow in $G(N,A)$ in which each path is no longer than the corresponding original path in which it is contained.

Obviously, removing flow on cycles in the network will not reduce the net flow into the sink.

Let us consider the arc-path formulation of the maximum flow problem defined on $D_L(G)$.

Let P^L be the set of all simple paths from s to t in $D_L(G)$ and let P be the corresponding set of simple paths in $G(N,A)$. Also let \underline{X}^L denote the vector of path flows defined on $\{P^L\}$ and let \underline{X} denote the corresponding vector of path flows defined on $\{P\}$. The maximum flow problem is to find $\underline{X} \geq 0$ to

$$\text{maximize: } v = x_1^L + \dots + x_{n_L}^L$$

$$\text{subject to: } \sum_{j=1}^{n_L} a_{1,j} x_j^L \leq b_1^L$$

$$\underline{X}^L \geq 0,$$

where $A = (a_{ij})$ is the arc-path incidence matrix and B_i^L is the capacity of the i^{th} arc in $D_L(G)$.

The corresponding vector of path flows defined on $G(N,A)$ is given by

$$x_j = h(P_j) = \sum h^L(P_j^L) = \sum x_j^L,$$

where the summation is taken over all simple paths in $\{P^L\}$ associated with the simple path P_j in $\{P\}$. From this definition of corresponding flows on $D_L(G)$ and $G(N,A)$ we can state the following

Lemma 4

If H^L and H are arc-path flows on $D_L(G)$ and $G(N,A)$ respectively, then the values of these two flows, v^L and v are equal, i.e.,

$$v = v^L.$$

The next theorem provides a necessary condition on a flow on $D_L(G)$ for its corresponding flow on $G(N,A)$ to be a maximal flow.

Theorem 3

Let v_{\max} be the maximum feasible flow on $G(N,A)$. Let H^L denote a feasible arc-path flow defined on $D_L(G)$. Let H denote the corresponding arc-path flow on $G(N,A)$ defined as indicated. Then the flow H is a maximum feasible flow on $G(N,A)$ if the value of the flow H^L is v_{\max} and H , as defined above, is a feasible flow on $G(N,A)$.

Proof. This theorem follows directly from lemma 4.

Q.E.D.

As a result of theorem 1 we see how to obtain the arc-path flow on $G(N,A)$ corresponding to a given arc-path flow on $D_L(G)$. An arc-path flow on $D_L(G)$ can be obtained by solving for a node-arc flow and then decomposing that into flows on simple paths of $D_L(G)$.

Theorem 3 gives a necessary condition for a path flow on $D_L(G)$ to produce a corresponding maximum flow on $G(N,A)$. Obviously this condition can be interpreted analogously with respect to a node-arc flow. Theorem 2 provides the condition for a node-arc flow on $D_L(G)$ to have a corresponding feasible flow on $G(N,A)$.

We will now show that the length of the longest simple path between the source and the sink in $D_L(G)$ is equal to L . Thus, any path decomposition of any flow, in particular the solution to (1)-(4), has a longest path value less than or equal to L .

Lemma 5

Let $D_L(G)$ be the L -period expanded version of $G(N,A)$ and let $\{P_j^L\}$ be the set of simple paths from the source to the sink in $D_L(G)$. Let $\ell(P_j^L)$ be the length of the path P_j^L . Then

$$\ell(P_j^L) \leq L, \quad j.$$

Proof. $D_L(G)$ represents the L -period expanded version of $G(N,A)$ for which node t at time period L is the terminal node. In other words, node t at time period L is in $D_L(G)$ but node t at any later time period is not in $D_L(G)$. Let us consider any path in $D_L(G)$ from (s,L) to (\hat{t},L) say $P_j = \{(s,L), (x_1, \tau_1), \dots, (x_n, \tau_n), (\hat{t}, L)\}$. The length of P_j is given by

$$\ell(P_j) = \sum_{(x,y) \in P_j} \ell(x,y) = \sum_{(x,y) \in P_j} a(x,y),$$

Substituting (28), we obtain

$$\ell(P_j) \leq L - \tau_n + \sum_{k=0}^{n-2} (\tau_{n-k} - \tau_{n-k-1}) + \tau_1.$$

After collecting terms, we get

$$\ell(P_j) \leq L$$

Q.E.D.

As a result of this lemma, we know that the maximum path length for any flow on $D_L(G)$ is no greater than L . The following lemma shows that the arc-path flow in $G(N,A)$ corresponding to any arc-path flow in $D_L(G)$ has maximum path length less than or equal to L also.

Recall that we can associate with every path of $D_L(G)$, a unique simple path in $G(N,A)$.

Lemma 6

If we let P_j represent the simple path in $G(N,A)$ associated with P_j^L in $D_L(G)$, then

$$\ell(P_j) \leq \ell(P_j^L).$$

Proof. This follows from the fact that P_j is a subset of the arcs in $G(N,A)$ corresponding to the arcs in P_j^L . The arcs in $G(N,A)$ have length equal to the length of their corresponding arcs in $D_L(G)$.

Q.E.D.

We shall now state and prove an important theorem concerning the relationship between the paths of $G(N,A)$ and those of $D_L(G)$.

Theorem 4

Let $D_L(G)$ be the L -period expanded version of the network $G(N,A)$ and let $\{P^L\}$ be the set of all simple paths in $D_L(G)$ from the source to the sink. Let $\{P_G^L\}$ denote the set of simple paths in $G(N,A)$ corresponding to the paths in $\{P^L\}$. Then all paths in $G(N,A)$ of length L or less are in $\{P_G^L\}$.

Proof. The proof follows immediately from the construction of $D_L(G)$ and the correspondence between paths in $D_L(G)$ and $G(N,A)$.

Let $P_j \equiv \{s, x_1, x_2, \dots, x_n, t\}$ be a path in $G(N,A)$ of length L or less. If we let

$$\begin{aligned}\tau_1 &= a(s, x_1) \\ \tau_2 &= a(x_1, x_2) + \tau_1 \\ &\vdots \\ &\vdots \\ \tau_n &= a(x_{n-1}, x_n) + \tau_{n-1}\end{aligned}$$

then
$$\tau_n = a(s, x_1) + \dots + a(x_{n-1}, t) .$$

Since the length of P_j is assumed to be less than or equal to L , we conclude that

$$\tau_n \leq L.$$

Thus (t, τ_n) appears in the L -period expanded version of $G(N, A)$. It follows then, since the arcs of P_j allow the labels (x_n, τ_n) on the respective arcs of $G(N, A)$, that the corresponding arcs will appear in $D_L(G)$. There will be arcs in $D_L(G)$ corresponding to the arcs of P_j which connect the nodes in $D_L(G)$. Hence there is a path in $D_L(G)$ corresponding to P_j and we conclude that $P_j \in \{P_G^L\}$.

Q.E.D.

Remark 4

As a result of the expansion and reduction techniques used to generate $D_L(G)$ there will be one or more paths in $D_L(G)$ corresponding to each path of length L or less in $G(N, A)$. However, each path in $D_L(G)$ will correspond to a unique path of length L or less in $G(N, A)$.

Lemma 7

If v_{max}^L is the optimal value of (3), subject to (4), (5), and (6), and

$$v_{max}^L \geq v_{max},$$

then the min-max path flow on $G(N, A)$ has maximum path value less than or equal to L .

Proof. Suppose we have an optimal solution to (3)-(6) and

$$v_{max}^L = v_{max}.$$

Then the corresponding flow on $G(N, A)$ is a maximum feasible flow on $G(N, A)$.

By lemma 5, if $\{P_B^L\}$ is the set of paths in any simple path decomposition of the flow on $D_L(G)$, then

$$\ell(P_j^L) \leq L, \forall j \text{ such that } P_j^L \in \{P_B^L\}.$$

The unique set of simple paths in $G(N,A)$ corresponding to $\{P_B^L\}$ will be denoted by $\{P_B\}$ and will form, with flows equal to that of their corresponding paths in $\{P_B^L\}$, a path decomposition of the flow on $G(N,A)$. By lemma 6,

$$\ell(P_j) \leq \ell(P_j^L), \forall j \text{ such that } P_j \in \{P_B\}.$$

Thus, we conclude that

$$\ell(P_j) \leq L, \forall j \text{ such that } P_j \in \{P_B\}.$$

We have exhibited a maximum flow on $G(N,A)$ the length of whose longest path (with flow) is less than or equal to L .

Q.E.D.

This lemma provides a necessary condition on $D_L(G)$ for the min-max path solution to have maximum path length not greater than L . The next lemma provides a necessary and sufficient condition for the solution to have maximum path length greater than or equal to L .

Lemma 8

If $D_{L-1}(G)$ is the $L-1$ period expanded version of $G(N,A)$, then the min-max path flow on $G(N,A)$ has maximum path length greater than

or equal to L if and only if

$$v_{\max}^{L-1} < v_{\max}$$

Proof. Let L^* denote the length of the maximum length path with positive flow in the optimal solution to the min-max path flow problem. We must establish the following two statements:

$$(a) \ L^* \geq L \Rightarrow v_{\max}^{L-1} < v_{\max}$$

$$(b) \ v_{\max} < v_{\max} \Rightarrow L^* \geq L.$$

Let us consider case (a) first. We shall prove the contrapositive of (a). That is, suppose

$$v_{\max}^{L-1} \geq v_{\max}.$$

By applying lemma 7, we conclude that

$$L^* \leq L-1$$

$$\Rightarrow L^* < L.$$

Thus, since the contrapositive is valid, the original statement, (a), is valid.

We now consider case (b). Again we approach the proof by way of the contrapositive.

Suppose that

$$L^* < L,$$

$$\Rightarrow L^* \leq L-1.$$

This means that there exists a maximum path flow on $G(N,A)$ which involves flow only on paths of length $L-1$ or less. Let $\{P\}$ be such a set of paths in $G(N,A)$, then there exists a corresponding set of paths in $D_{L-1}(G)$.

Since by construction, $D_{L-1}(G)$ contains that portion of $G(N,A)$ appearing in paths of length L or less, and corresponding arcs have equal capacities, the maximum flow on paths of length $L-1$ or less, will be feasible on $D_{L-1}(G)$. Thus

$$v_{max}^{L-1} \geq v_{max}$$

It follows then that

$$v_{max}^{L-1} < v_{max} \Rightarrow L^* \geq L.$$

Q.E.D.

We can now summarize the above discussion by stating and proving the following theorem which provides necessary and sufficient conditions for the min-max path flow problem to have optimal maximum chain length of L .

Theorem 5

The network $G(N,A)$ has a min-max path value of L if and only if

$$v_{\max}^L \geq v_{\max}, \quad (29)$$

while

$$v_{\max}^{L-1} < v_{\max}, \quad (30)$$

where v_{\max} is the maximum flow on $G(N,A)$, and v_{\max}^L and v_{\max}^{L-1} are the solutions to (3)-(6) defined on $D_L(G)$ and $D_{L-1}(G)$ respectively.

Proof. Again let L^* denote the actual length of the longest path in the min-max path flow in $G(N,A)$. By lemma 7, if

$$v_{\max}^L \geq v_{\max},$$

then

$$L^* \leq L.$$

By lemma 8, if

$$v_{\max}^{L-1} < v_{\max},$$

then

$$L^* \geq L.$$

Thus, if (29) and (30) hold, we have

$$L^* \leq L \leq L^*$$

$$\Rightarrow L^* = L,$$

then conditions (29) and (30) hold. By lemma 8, if

$$L^* \geq L$$

then

$$v_{\max}^{L-1} < v_{\max}.$$

To show that

$$L^* = L \Rightarrow v_{\max}^L \geq v_{\max},$$

we assume otherwise and demonstrate that this leads to a contradiction.

Suppose

$$v_{\max}^L < v_{\max}$$

then from lemma 8,

$$L^* \geq L+1.$$

But this contradicts the assumption that

$$L^* = L.$$

Thus we conclude that

$$v_{max}^L \geq v_{max}.$$

Q.E.D.

As a consequence of this theorem we can solve the min-max path flow problem on $G(N,A)$ by solving a sequence of maximum flows with bundle constraints on $D_L(G)$. We begin at some value of L and solve (3)-(6) on $D_L(G)$. If $v_{max}^L < v_{max}$, then we increase L by one and continue. As soon as we obtain a value of L for which $v_{max}^L \geq v_{max}$, we terminate the procedure and the min-max path flow has longest path value L . The path flow on $G(N,A)$ which produces this solution is obtained simply by finding any path decomposition of the optimal flow on $D_L(G)$ and converting this, as suggested previously, to a set of flows on simple paths in $G(N,A)$.

If we drop the bundle constraints and add all individual arc capacities, then the maximum flow \hat{v}_{max}^L may exceed v_{max} as the following example shows.

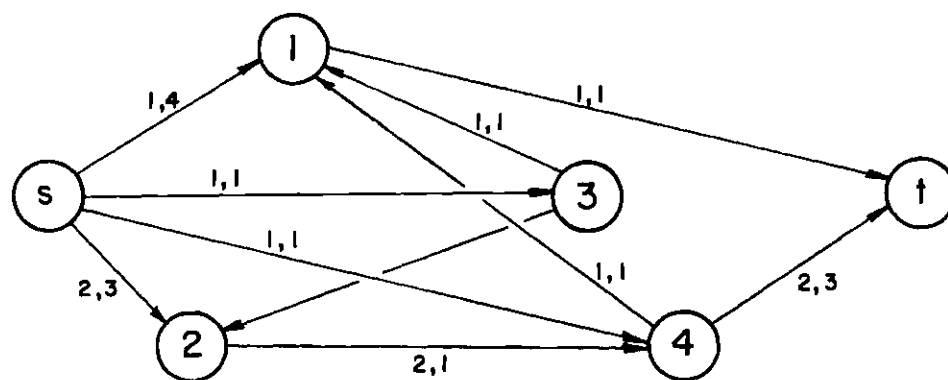
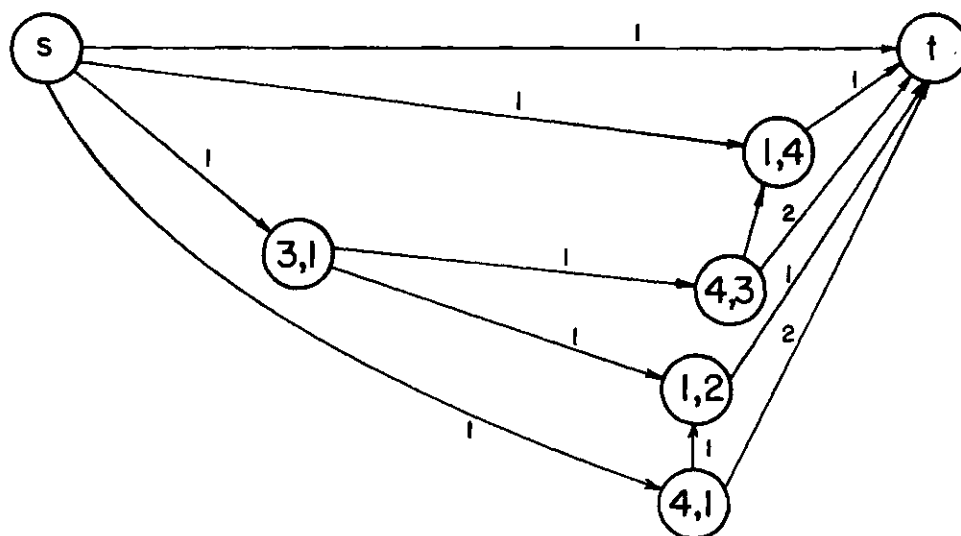


Figure 18. Static Network

Figure 19. Six-Period Expansion $G(N,A)$

The maximum flow, v_{\max} , for this network is $v_{\max} = 3$. Figure 19 is the six-period reduced dynamic expansion of this network. The maximum flow on this network is $\hat{v}_{\max}^6 = 4$, hence

$$\hat{v}_{\max}^6 > v_{\max}.$$

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

This research has been concerned with a study of the min-max path flow problem on a directed network. While this problem has wide application in network problems, it had not, to the author's knowledge, been explicitly treated in the literature.

The objectives of the research were to characterize the structure of the problem within the framework of other network flow problems that have been treated in the literature and to develop a solution algorithm. To more explicitly define the latter objective, we observe that the problem could be solved by enumerating all paths of length L or less and then solving the maximum flow problem on that set of paths. The value of L could then be adjusted and a new linear program be solved. Obviously the procedure terminates when the smallest value of L is found for which the linear programming solution saturates a minimum cut set of the network. The disadvantage of this approach is that the set of paths is quite large and only a very few are likely to be in the solution. This means a large storage is required for problems of any size and many computations are required to test for optimality.

Thus, the objective is to develop an algorithm which will not require the explicit enumeration of all paths but which will generate only paths which are candidates to enter the basis.

Results and Conclusions

The specific results of this research are:

1. The minimum cost flow algorithm does not solve the min-max path flow problem. The min-max path problem is similar in structure to the multicommodity flow problems.
2. The min-max path flow problem does not have all-integer solutions as does the minimum cost flow problem.
3. The computations for solving the min-max path problem are comparable to those for solving arc-path formulations of other network flow problems.
4. An algorithm was developed which utilizes a minimum cost maximum flow algorithm to achieve an initial feasible solution to the problem and then moves toward optimality by solving a sequence of constrained shortest path problems to force flow off the current longest path. A revised simplex type procedure is used after the minimum cost maximum flow problem has been solved.
5. An alternative algorithm was developed which solves the min-max path flow problem by implicitly representing all paths of length L or less on a dynamically expanded network and solving a bundle-constrained maximum flow problem on the expanded network.
6. The path removal algorithm has application to multiterminal maximal flow problems with one or two inadmissible source-sink pairs.
7. The expanded network could be adapted to solve maximal dynamic flows with total arc capacities. Implicit in the statement of the algorithm is the fact that it could also be used to find the maximum

flow in a network on paths of length L or less. There would be no difficulty in adding arc costs in addition to arc lengths. We could then use the algorithm to find minimal cost maximum flows on paths of length L or less.

8. Several areas for future research have been identified, including work toward developing the most efficient computational scheme and applications of the results of this research to other problems.

Other Applications

Some of the results of this research have application to other network flow problems. Consider, for example, the multiterminal maximum flow problem. There are several sources and several sinks, each may have a specific availability or demand. If any source can ship to any sink, the problem can be treated the same way the single source, single sink problem is handled. If, on the other hand, we require that, say source s_1 cannot ship to sink t_2 , and source s_2 cannot ship to sink t_1 , then it is not a single-commodity problem.

Replace node s_i , $i = 1, 2$, with two nodes s_i , s_i' and connect them with arc (s_i, s_i') . Do likewise with nodes t_1 and t_2 . Give each new arc infinite capacity and costs as follows:

$$a(s_1, s_1') = a(t_2, t_2') = \infty$$

$$a(s_2, s_2') = a(t_1, t_1') = 0.$$

We can solve the min-max path flow problem. If a maximum flow exists which satisfies the restrictions on flow specified, then such a solution will be generated by the min-max path flow algorithm.

Minimal Cost, Min-Max Path Flows

If we add arc costs to the min-max path flow problem, we may wish to find the minimal cost min-max path flow. This is easy to handle on expanded network algorithm. This simply alters the objective function in the decomposition approach. The subproblems remain minimal cost maximal flow problems.

Maximal Dynamic Flows With Arc Capacities

In the maximal dynamic flow problem, the arc capacities are replaced with flow rates, or the amount of flow which can enter or leave the arc per unit of time. If we impose on this problem, a total flow capacity, then we can handle these additional constraints as bundle constraints. The decomposition procedure of Chapter IV can then be used. The time expanded network would generally be much larger than that required for the min-max path flow problem however. Further research is needed to determine the possibility of working with the original static network as is the case for the standard maximal dynamic flow problem (12).

Extensions and Future Research

Since a feasible solution to the min-max path can be obtained by solving the max-flow problem without recourse to the simplex method, whereas the elements of B_1^{-1} are required to compute arc lengths for the shortest route problem of phase II, it is worthwhile to investigate the

possibility of computing the elements of B^{-1} on the network, making use of the basic paths in the solution.

The potential for doing this is illustrated by describing the concepts of a procedure and pointing out the theoretical justification of the procedure. Whether the procedure is viable for large problems remains to be seen. In the arc-path formulation of the maximum flow problem, we introduced slack vectors (slack paths) for each arc. Suppose we have a basic feasible solution with B , and B^{-1} such that:

$$\underline{X}_B = B^{-1}\underline{b}, \underline{X}_N = 0$$

Let $A = (a_{ij})$ denote the entire updated tableau for this solution. Then $a_{ij} = B_i^{-1}\underline{p}_j$ is the amount of change in X_{B_i} required by a unit of increase in x_j .

If x_j is the K^{TH} basic variable, then

$$B_i^{-1}\underline{p}_j = \begin{cases} 1, & \text{if } i = K \\ 0, & \text{otherwise} \end{cases}$$

For non-basic variables, from

$$\underline{X}_B = B^{-1}\underline{b} - B^{-1}N\underline{X}_N - B^{-1}S\underline{X}_S$$

where S is the non-basic slack vectors,

$$\underline{X}_{s_i} = B_i^{-1} \underline{b} - B_i^{-1} N \underline{X}_n - B_i^{-1} S \underline{X}_s$$

$$B_{ij}^{-1} S_{kj} = 0, \quad j \neq K; \quad S_k = e_k$$

$$S_{kk} = 1; \quad S_{kj} = 0 \quad K \neq j$$

Thus

$$B_i^{-1} S_k = B_{ij}^{-1} S_{kk} = B_{ij}^{-1},$$

and

$$\underline{X}_{s_i} = B_i^{-1} \underline{b} - \sum B_{ij}^{-1} X_{sj}; \quad \underline{X}_n \equiv 0$$

Hence B_{ij}^{-1} is the amount of change in X_{s_i} if X_{s_j} , the value of the i^{th} slack path is increased while holding all other non-basic paths at zero flow.

Thus, the potential procedure is to perturb the flow on the non-basic slack paths of the network and determine the change required on basic path i in order to keep the flow in balance.

To illustrate, consider the example of Figure 20.

The subscripted lower case letters on the arcs indicate the basic paths which contain the given arc. Slack basic paths are not recorded. The circled letters C_i denote the arc for which the given path is basic

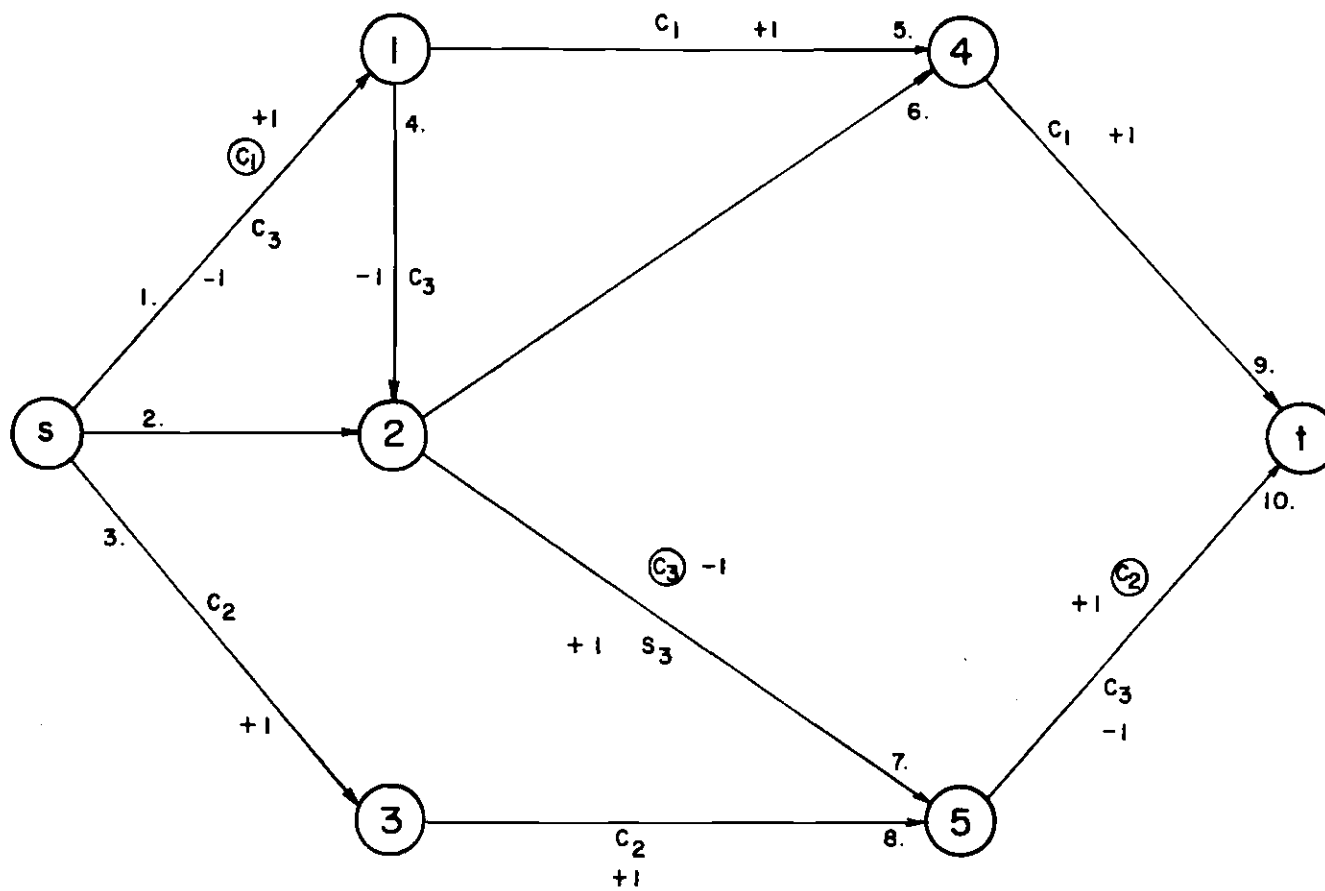


Figure 20. Computation of B^{-1} on the Network

$$B = \begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & 1 & & & 1 & & & \\ & & & & & 1 & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & 1 & & & & & 1 & \\ & & & & & & & 1 & \\ & & & & 1 & & & & 1 \end{bmatrix}$$

The seventh column of B^{-1} is given by

$$B_7^{-1} = [-1 \ 0 \ 1 \ -1 \ 1 \ 0 \ 1 \ 1 \ 1 \ -1]'$$

This can be derived by increasing the flow on the non-basic slack path S_7 and determining the adjustments required to the other basic paths in order that the equations remain satisfied.

The following table shows the basic paths corresponding to the numbered arcs and the change in flow required along each path caused by a unit increase in the value of flow on S_7 . These values are taken from Figure 20.

For more complicated networks, this heuristic procedure becomes unwieldy. The problem of finding the elements of a given column of B^{-1} can be represented as a system of m equations in m unknowns, where m is the number of real paths in the basis.

In the network shown in Figure 21, there are four real chains in the basis.

The simplex basis is given in Table 8. Its inverse is given in Table 9.

Table 7. Column Seven of B^{-1}

Arc	Basic Variable	Flow Change
1	C_1	-1
2	S_2	0
3	S_3	1
4	S_4	-1
5	S_5	1
6	S_6	0
7	C_3	1
8	S_8	1
9	S_9	1
10	C_2	-1

The flow change column corresponds to the seventh column of B^{-1} .

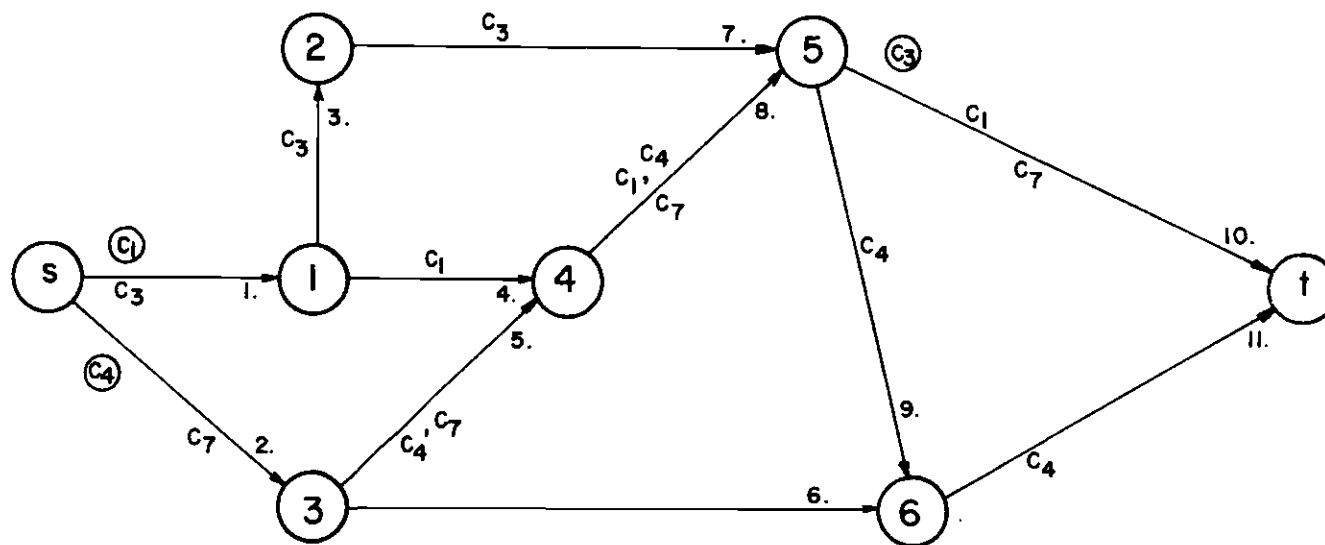


Figure 21. Network With Four Basic Paths

Suppose we want to find the first column of B_1^{-1} . Let $S_1 = B_{11}^{-1}$. From the network, we can write the following equations

$$\begin{aligned} \delta_1 + \delta_{10} &= 1 \\ \delta_2 + \delta_8 &= 0 \\ \delta_1 + \delta_2 + \delta_8 &= 0 \\ \delta_1 + \delta_{10} + \delta_8 &= 0 \end{aligned}$$

The solution is

$$\delta_1 = 0$$

$$\delta_{10} = \delta_2 = 1$$

$$\delta_8 = -1,$$

all other arcs of the network have slack paths as basic. The corresponding value of δ_1 is that necessary to balance the change in flow. It can be achieved by adjusting the flow on X_1 since X_1 represents flow on a slack path and does not interact with other basic variables. From the network, we get

$$\delta_3 = \delta_9 = \delta_{10} = \delta_{11} = 1$$

$$\delta_4 = \delta_5 = \delta_6 = 0 \text{ i } \delta_7 = -1.$$

The vector $\Delta = [\delta_i]' = [0 \ 1 \ 1 \ 0 \ 0 \ 0 \ -1 \ -1 \ 1 \ 1 \ 1]'$.

$$B_1^{-1} = -\Delta$$

Lower Bounds on L for $D_L(G)$

In Chapter IV it was pointed out that it may be necessary to solve the bundle constrained maximum flow problem on $D_L(G)$ for several values of L until we find L such that

$$v_{\max}^{L-1} < v_{\max} = v_{\max}^L .$$

It was pointed out that an important part of the algorithm is the procedure for searching over L. In Chapter II we discussed a necessary condition suggested by Fulkerson that the min-max path decomposition of a given flow has longest path L. A profitable area for further research is the adoption of this concept into a procedure for finding a strong lower bound on L. The following approach should be investigated.

1. Generate $D_L(G)$,
2. For each arc $(x_T, y_{T+a(x,y)})$ in $D_L(G)$, cover arc (x,y) in $G(N,A)$,
3. Delete all uncovered arcs in $G(N,A)$,
4. Solve the maximum flow on the new network. If the maximum flow is less than v_{\max} , then

$$L^* \geq L .$$

5. Increase L by one and return to Step 1. Otherwise,

$$L^* \leq L .$$

After each iteration we have a good starting solution for the next iteration.

The algorithm presented in Chapter III solves the min-max path flow problem by solving a maximum flow problem and then searching for a path to bring into the basis which will tend to reduce the net flow on the set of longest paths in the basis. This algorithm can be viewed as a modification of the parametric linear programming problem.

Suppose we restate the maximum flow problem as follows:

$$\text{maximize: } \sum_{i=1}^h (1 - \delta_i \lambda) x_i$$

$$\text{subject to: } P\underline{X} \leq \underline{b}$$

$$\underline{X} \geq 0$$

Suppose we have an optimal basis B to the maximum flow problem with $\lambda = 0$. Let L_B be the length of the longest path in the basis with positive flow assigned.

We want to determine whether the optimal solution to the min-max path flow problem has longest path of length L_B or less. We define δ_i as before

$$\delta_i = \begin{cases} R & \text{if } \underline{a}'\underline{p}_i \geq L_0 \text{ and } x_i = 0 \\ 1 & \text{if } \underline{a}'\underline{p}_i = L_0 \text{ and } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

We now consider for $\lambda > 0$. We are not interested in solving parametrically on λ , but instead want to resolve for $\lambda = \lambda_0$. We have shown in Chapter III that for any value of $\lambda > 0$, the problem defined above will have a solution which maximizes the value of the flow while minimizing the flow on paths of length L_0 or greater.

Utilizing this approach, it is possible that a candidate to enter the basis may have

$$\pi \underline{p}_j > 1,$$

where π is the vector of simplex multipliers for the maximum flow problem ($\lambda = 0$). That is

$$C_j - Z_j = 1 - \pi \underline{p}_j < 0.$$

What this means is that there may be vectors \underline{p}_j with negative relative cost coefficient which could be brought into the basis. The introduction of such paths may or may not reduce the flow in the network below v_{max} . We might have to solve the auxiliary problem to optimality to ensure that the flow remains at v_{max} .

In Chapter III, the algorithm presented does not consider such vectors as candidates to enter the basis. It would be of value to compare the two approaches from a computational point of view.

It was also shown that in considering paths to enter the basis in order to reduce the net flow on paths of length L_b or less, we need not consider paths of length L_b . It may be, however, that non-basic paths exist whose introduction into the basis will tend to reduce the net flow on paths of length L_b . It would be of interest to investigate further the trade-offs between the two approaches. We can observe, for example that if indeed the optimal solution $L^* = L_b$, then we might detect this more quickly if we do not allow paths of length L_b to enter the basis. On the other hand, if $L^* < L_b$, then any path of length L_b brought into the basis will eventually be removed. Thus, the approach we presented in Chapter III appears to be superior to the standard parametric approach.

Generating Paths of Length L or Less

In Chapter IV, we developed and made use of a technique for representing all paths in a network of length L or less on a dynamically expanded version of the network. The procedure and representation appears to be significantly more efficient than known methods for generating and storing all paths of length L or less or the K shortest routes or paths. Additional investigation is required before definitive statements can be made however.

The use of $D_L(G)$ in place of generating and storing all paths of length L or less should be considered in problems which require such

paths. Further investigation is needed in this area. The Dantzig-Wolfe decomposition method is used with the single subproblem being a shortest route problem. Because of the dominance of certain constraints over others, a modification of the standard decomposite appears to offer some computational advantages.

Rerouting Flow

If one were to delete the bundle constraints from the bundle-constrained maximum flow problem on $D_l(G)$, then the result is a standard maximum flow problem.

A standard maximum flow algorithm can be used to solve the problem. If none of the bundle constraints are violated then the solution is optimal. Otherwise, some bundle constraint is violated.

It would be of interest to investigate a primal-dual approach to rerouting flow through the network. There would be three classes of arcs to consider: those which belong to violated bundles, those which belong to saturated bundles and those which belong to unsaturated bundles. Results here might be useful in the decomposition approach to solving the bundle constrained maximum flow on $D_l(G)$.

APPENDIX A

CHAIN DECOMPOSITION

Let F be an node-arc flow defined on A . Then F can be decomposed into an arc-path flow as follows:

1. Path identification: a labeling procedure is used to identify a path to which flow can be assigned. Suppose we have identified $j - 1$ paths, C_1, C_2, \dots, C_{j-1} . Each path has flow $h(C_k)$. Each time a path is identified, it is assigned the maximum flow possible consistent with the remaining flow on its arcs. The flow assigned to a path is subtracted from the flow on the arcs of the path. Thus, each time a path is identified and flow assigned, at least one arc of the path has its flow reduced to zero. The path will be considered the basic variable corresponding to one such arc. Begin at node s with the label $(-, \infty)$ and label across the network with labels (x, h) until node t is labeled or it is revealed that t cannot be labeled. If t cannot be labeled, then the procedure terminates with the desired path decomposition. The labeling proceeds as follows:
 - 1.a. Select a labeled node x and search for an unlabeled y such that $(x, y) \in A$ and $f'(x, y) > 0$. To initiate, set $f'(x, y) = f(x, y)$ for all $(x, y) \in A$. If no such y can be

found select another node x . If all nodes have been considered, terminate the labeling procedure and go to Set (3).

1.b. If nodes x and y can be found satisfying the above conditions, label node y with the label (x, h) , where

$$h(y) = \text{minimum } \{h(x), f'(x, y)\} .$$

If $f'(x, y)$ is the minimum let $I_{B,j} = \gamma(x, y)$ where $\gamma(x, y)$ is the index of the arc (x, y) and $I_{B,j}$ is the relative position of path C_j in the basis. In other words, vector \underline{P}_j is the $I_{B,j}$ basic vector. We say that \underline{P}_j is basic to arc $I_{B,j}$.

Continue to label until node t is labeled or the procedure terminates with no additional labeling possible. As indicated, if t cannot be labeled, the desired decomposition has been achieved. Otherwise go to Step (2).

2. Flow assignment: Let C_j denote the path identified in Step (1). We must generate the arc-path incidence vector \underline{P}_j and the amount of flow $h(C_j)$. We assume that the arcs of A are indexed.

2.a. Set $I = t$ and initialize \underline{P}_j at $\underline{0}$.

2.b. Let $(N_I, h(I))$ be the label on I . Then $h(C_j) = h(I)$.

2.c. Let k be the index of the arc (N_I, I) , $k = T(N_I, I)$.

Then set $P_{k,j} = 1$. Set $f'(N_I, I) = f'(N_I, I) - h(C_j)$. If $I = S$, remove all labels except on S , and return to Step (1.a.).

3. Completing the basis: Each path \underline{P}_j assigned flow by the above procedure is associated with (basic to) a unique arc of the network $G(N,A)$ while no two are associated with the same arc. If we associate with arcs which have no basic path the slack path (vector) of the constraint corresponding to that arc, the set of real paths and slack paths form a basis for the solution set of problem (1) in Chapter III. This follows from the fact that the above set of vectors could be generated by the simplex method on problem (1) if only chains are brought into the basis which can be brought in at a positive level. The simplex method always produces a basic feasible solution.

BIBLIOGRAPHY

1. Balas, E., "Minimaximal Path in a Disjunctive Pert Network," in Theory of Graphs, International Symposium, Rome 1966, Gordon and Breach, New York, 1967.
2. Berge, C., Theory of Graphs and Its Application, John Wiley and Sons, Inc., New York, 1960.
3. Berge, Claude and A. Ghouila-Houri, Programming Games and Transportation Networks, Translated by Maxine Merrington and C. Ramanujacharyulu, Wiley, New York, 1962.
4. Busacker, R. G. and P. J. Gowen, "A Procedure for Determining a Family of Minimal-Cost Network Flow Patterns," Operations Research Office, Technical Paper 15, 1960.
5. Busacker, R. G. and T. L. Saaty, Finite Graphs and Networks, McGraw-Hill, New York, 1965.
6. Clark, S., A. Krikorian and J. Rausen, "Computing the N Best Loopless Paths in a Network," SIAM Journal on Applied Mathematics, 11, 1963.
7. Dantzig, George B., Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.
8. Dantzig, George B. and P. Wolfe, "The Decomposition Algorithm for Linear Programming," Econometrica, 29, 1961.
9. Edmonds, Jack and D. R. Fulkerson, "Bottleneck Extrema," Journal of Combinatorial Theory, 8, 1970.
10. Elmaghraby, Salah E., "The Theory of Networks and Management Science, Part I," Management Science, 17, 1970.
11. Elmaghraby, Salah E., "The Theory of Networks and Management Science, Part II," Management Science, 17, 1970.
12. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, N. J., 1962.
13. Ford, L. R. and D. R. Fulkerson, "A Suggested Computation for Multicommodity Network Flows," Management Science, 5, 1958.

14. Fulkerson, D. R., "On the Equivalence of the Capacity-Constrained Transshipment Problem and the Hitchcock Problem," The Rand Corporation, Research Memorandum RM-2480, Jan. 13, 1960.
15. Fulkerson, D. R., "An Out-of-Kilter Method for Minimal Cost Flow Problems," SIAM Journal on Applied Mathematics, 9, 1961.
16. Fulkerson, D. R., Private Communication, 19 March 1971.
17. Gale, D., "Transient Flows in Networks," Michigan Mathematics Journal, 6, 1959.
18. Gass, Saul I., Linear Programming, McGraw-Hill, New York, 1969.
19. Geoffrion, Arthur, "Elements of Large-Scale Mathematical Programming," Management Science, 16, 1970.
20. Gross, O., "The Bottleneck Assignment Problem," The Rand Corporation Paper P-1630, March 6, 1959.
21. Hadley, G., Linear Programming, Addison-Wesley Publishing Co., Reading, Mass., 1962.
22. Hammer, P. L., "Time-Minimizing Transportation Problems," Naval Research Logistics Quarterly, 16, 1969.
23. Hu, T. C., Integer Programming and Network Flows, Addison-Wesley, Reading, Mass., 1969.
24. Iri, Masao, Network Flow, Transportation and Scheduling, Theory and Algorithms, Academic Press, New York, 1969.
25. Jarvis, John J., "Optimal Attack and Defense of a Command and Control Communications Network," Ph.D. Dissertation, Johns Hopkins University, 1968.
26. Jarvis, J. J., "On the Equivalence Between the Node-Arc and Arc-Chain Formulation of the Multicommodity Maximal Flow Problem," Naval Research Logistics Quarterly, 16, 1969.
27. Jewell, W. S., "Optimal Flow Through Networks," Ph.D. Dissertation, MIT, Cambridge, June 1958.
28. Jewell, W. S., "Multicommodity Network Solutions," Operations Research Center, Report ORC-66-23, University of California, Berkeley, 1966.
29. Johnson, Ellis L., "Networks and Basic Solutions," Operations Research, 14, 1966.

30. Joksch, H. C., "The Shortest Route Problem with Constraints," *Journal of Mathematical Analysis and Applications*, 14, 1966.
31. Kalaba, R. E. and M. L. Juncosa, "Optimal Design and Utilization of Communication Networks," *Management Science*, 3, 1956.
32. Kaufmann, A., *Graphs, Dynamic Programming and Finite Games*, Academic Press, New York, 1967, Chapter 1.
33. Klein, Morton, "A Primal Method for Minimal Cost Flows," *Management Science*, 14, 1967.
34. Kobayashi, Takashi, "On Maximal Flow Problem in a Transportation Network with a Bundle," *Journal of the Operations Research Society of Japan*, 10, 1968.
35. Pollack, Maurice, "Solutions of the Kth Best Route Through a Network-A Review," *J. Math. Anal. and Appl.*, 3, 1961.
36. Ridley, Tony M., "An Investment Policy to Reduce the Travel Time in a Transportation Network," ORC 65-34, University of California, Berkeley, 1965.
37. Rothfarb, W., N. P. Shein and I. T. Frisch, "Common Terminal Multicommodity Flow," *Operations Research*, 16, 1968.
38. Rothschild, B. and A. Whinston, "On Two-commodity Network Flows," *Operations Research*, 14, 1964.
39. Saigal, Romesh, "Multicommodity Flows in Directed Networks," ORC-67-38, University of California, Berkeley, 1967.
40. Saigal, Romesh, "A Constrained Shortest Route Problem," *Operations Research*, 16, 1968.
41. Sakarovitch, M., "The Multicommodity Maximal Flow Problem," ORC-66-25, University of California, Berkeley, 1966.
42. Sakarovitch, Michel, "The K Shortest Routes and the K Shortest Chains in a Graph," ORC-66-32, University of California, Berkeley, Oct. 1966.
43. Szwarc, W., "On Some Sequencing Problems," *Naval Research Logistics Quarterly*, 15, 1968.
44. Tewarson, R. P., "On the Product Form of Inverses of Sparse Matrices," *SIAM Review*, 8, 1966.
45. Tomlin, J. A., "Minimum-Cost Multicommodity Network Flows," *Operations Research*, 14, 1966.

VITA

Robert Glenn Hinkle was born at Buckhannon, West Virginia, on February 13, 1939, the son of Glenn E. and Ocie V. Hinkle, nee Lang. He attended public schools in West Virginia and was graduated from Fairmont State College, Fairmont, West Virginia, with a B. S. degree in Mathematics in 1961. In 1963 he was awarded the M. S. degree in Mathematics from West Virginia University.

He enrolled in Georgia Institute of Technology in September, 1967, as a participant in the Naval Weapons Laboratory's Full-Time Advanced Study Program. He has been an Operations Research Analyst at the Naval Weapons Laboratory, Dahlgren, Virginia, since 1963.