

Constrained shortest paths in networks
by Chi-Hung Shek

A thesis submitted in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY in Electrical Engineering
Montana State University
© Copyright by Chi-Hung Shek (1975)

Abstract:
Shortest path problems in networks constitute a branch of optimization theory wherein system network models with paths which are characterized by path lengths are of concern.

This work is an extension of the classical shortest path problem; here additional cost constraints are imposed on the network model.

A dynamic programming solution approach is first formulated. It is solved by an iterative scheme in which the cost bound variable is incremented by a unit at the end of each iteration where the shortest paths with path costs not exceeding the cost bound variable are found. The concept of "stepping cost" is introduced representing the path cost at which a change of shortest path length occurs. A theorem is proved on the procedure to locate the "immediate stepping cost" and the associated stepping-cost path length; thus, an acceleration of the iterative scheme is feasible. As a result, a combinatorial algorithm is developed with computational effort being the order of the square of the number of stepping costs.

Further extensions to networks with more than one path cost constraint are presented.

The constrained shortest path algorithms are applied to flow networks and to networks with probabilistic weighted branch costs. FORTRAN computer programs are given to implement the algorithms.

CONSTRAINED SHORTEST PATHS IN NETWORKS

by

CHI-HUNG SHEK

A thesis submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Donald A. Pierre
Chairman, Examining Committee

Paul E. Ulrich
Head, Major Department D.N.M

Henry L Parsons
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

June, 1975

ACKNOWLEDGMENTS

The author would like to thank the Electrical Engineering Department of Montana State University for the financial support throughout his entire Ph.D. program.

Also, the author would like to thank Dr. Donald A. Pierre for his guidance, moral support, and patience without which the preparation of this thesis would have been impossible. Thanks are expressed to Dr. D. A. Rudberg and Dr. R. M. Johnson for their review of material and their many constructive suggestions for the thesis.

The author would like to thank Peggy Humphrey for typing the manuscript and for her suggestions in the arrangement of this thesis.

Finally, the author would like to thank his parents and the rest of his family for their encouragement and support.

TABLE OF CONTENTS

	Page
VITA	ii
ACKNOWLEDGMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	x
CHAPTER I: INTRODUCTION AND PRELIMINARY CONCEPTS	1
1.1 Introduction	1
1.2 Fundamental Definitions	5
1.3 The Problem Statement	7
1.4 The Dynamic Programming Formulation	8
1.5 A Historical Review	10
1.6 Some Solution Methods	11
1.6.1 The Dynamic Programming Approach	11
1.6.2 Dijkstra's Method	14
1.6.3 The Matrix Method	18
1.6.4 Observations	19
1.7 Contribution of the Research	20
CHAPTER II: THE CONSTRAINED SHORTEST PATH	22
2.1 Introduction	22
2.2 The Constrained Shortest Path	22
2.3 Solution Formulation	23
2.4 Proof of Validity	25
2.5 The Stepping Costs	28
2.6 The Constrained Shortest Path Algorithm	31
2.7 The Extended Algorithm	33
2.8 Networks With Several Cost Weights	40
CHAPTER III: THE COMBINATORIAL ALGORITHM	43
3.1 Introduction	43
3.2 The Immediate Stepping Cost	43
3.3 The Combinatorial Theorem	47
3.4 The Combinatorial Algorithm	48
3.5 A Comparison	52
3.6 Networks With Doubly-Labelled Branch Cost Weights	55
3.7 The Constrained Shortest Path Algorithm for Networks with Two Branch Cost Weights	67
3.8 An Example	71

TABLE OF CONTENTS (cont'd.)

	Page
CHAPTER IV: SOME OPTIMAL PATHS IN PROBABILISTIC WEIGHTED NETWORKS	82
4.1 Introduction	82
4.2 Probability Distribution of the Shortest Path	82
4.3 Some Optimal Paths	90
4.4 Stochastically Dependent Weights	96
CHAPTER V: SHORTEST PATHS IN FLOW NETWORKS	99
5.1 Introduction	99
5.2 Maximum Flow in Networks	100
5.3 The Minimum Cost Flow	106
5.4 The Minimum Cost Flow Problem with Transit-Time Delay	112
5.5 The Maximum Capacity Reliability Constrained Path	118
CHAPTER VI: COMPUTER PROGRAM AND COMPUTATIONAL EXPERIENCE	121
6.1 The Computer Program	121
6.2 A Maximum Capacity Reliability Constrained Path Problem	134
6.3 A Min-Cost Flow Problem With Transit-Time Delay	137
6.4 A Message Transmission Problem	145
CHAPTER VII: CONCLUSION AND FUTURE RESEARCH SUGGESTIONS	150
7.1 Conclusion	150
7.2 Future Research Suggestions	151
APPENDIX	154
LITERATURE CITED	193

LIST OF TABLES

Table		Page
3.1	Computational Effort Using the Combinatorial Algorithm. .	54
3.2	The Set of 1-5 Paths of the Network in Fig. 3.4a.	63
3.3	The Set of 1-5 Paths of the Network in Fig. 3.7a.	81
6.1	The Branch Weights of the Network in Fig. 6.7	136
6.2	The Path Capacities and Reliabilities	138
6.3	Computational Results	138
6.4	The Stepping Delays of the Network in Fig. 6.8.	141
6.5	The Stepping Delays of the Network in Fig. 6.8 with (3,4) Deleted	142
6.6	The Stepping Delays of the Network of Fig. 6.8 with (1,4), (3,4) Deleted.	143
6.7	The Set of 1→5 Paths.	149
6.8	The Set of 1→i Stepping-Cost Paths.	149

LIST OF FIGURES

Figure		Page
1.1	Illustration of graph fundamentals.	7
2.1	Concatenation of paths.	24
2.2	A five-node network with $W_{ij} = (d_{ij},h_{ij})$.	29
2.3	$\|\pi_5(c)\|$ of the five-node network in Fig. 2.2.	29
2.4a	A network G with $W_{ij} = (d_{ij},h_{ij})$.	37
2.4b	An approximate network G^1 of G in Fig. 2.4a with unit metric.	37
2.4c	Constrained shortest 1-5 paths of G, G^1 of Figs. 2.4a and 2.4b.	38
3.1	A five-node undirected network with $W_{ij} = (d_{ij},h_{ij})$ . .	44
3.2	The set of stepping costs and the stepping-cost path lengths of the network in Fig. 3.1. For c = 5, $K_1 = 1$, $K_2 = 1$, $K_3 = 0$, $K_4 = 1$, $K_5 = 0$.	45
3.3	The computational effort vs. the number of nodes in the network using the combinatorial algorithm	55
3.4a	A 5-node network with $W_{ji} = (d_{ji},h^1_{ji},h^2_{ji})$	61
3.4b	The 1-5 stepping costs.	62
3.5	A 5-node network with $W_{ji} = (d_{ji},h^1_{ji},h^2_{ji})$	72
3.6a	$\|\pi^*_3(c)\|$	75
3.6b	$\|\pi^*_4(c)\|$	75
3.6c	$\|\pi^*_2(c)\|$	75
3.6d	$\|\pi^*_5(c)\|$	77
3.6e	$\|\pi^*_5(c)\|$	77

LIST OF FIGURES (cont'd)

Figure		Page
3.6f	$\lvert\pi_5^*(c)\rvert$	77
3.6g	$\lvert\pi_2^*(c)\rvert$	77
3.6h	$\lvert\pi_2^*(c)\rvert$	78
3.6i	$\lvert\pi_5^*(c)\rvert$	78
3.6j	$\lvert\pi_5^*(c)\rvert$	78
3.7a	The 1-5 stepping costs.	79
3.7b	The 1→3 stepping costs.	80
3.7c	The 1→4 stepping costs.	80
3.7d	The 1→2 stepping costs.	80
4.1	A 5-node 12-branch network.	86
4.2	A confidence level curve and the feasible region.	93
4.3	The maximal confidence level ℓ_0 path.	95
5.1	A flow pattern where $W_{ij} = (f_{ij},c_{ij})$.	101
5.2	A minimum cost s→t augmentation path where $W_{ij} = (f_{ij}, c_{ij},h_{ij})$.	107
5.3a	$(i,j) \in B$, $(j,i) \notin B$.	110
5.3b	Branches generated from Fig. 5.3a	110
5.4a	$(i,j) \in B$, $(j,i) \notin B$.	110
5.4b	Branches generated from Fig. 5.4a	110
5.5a	$(i,j),(j,i) \in B$	110
5.5b	Branches generated from Fig. 5.5a	110

LIST OF FIGURES (cont'd)

Figure		Page
5.6a	$(i,j),(j,i) \in B$	111
5.6b	Branches generated from Fig. 5.6a	111
5.7a	$(i,j),(j,i) \in B$	111
5.7b	Branches generated from Fig. 5.7a	111
5.8a	$(i,j) \in B$	111
5.8b	Branches generated from Fig. 5.8a	111
5.9	A $\pi_{s,t}$ with one backward branch	114
5.10	A $\pi_{s,t}$ with two backward branches	115
5.11	A $\pi_{s,t}$ with q backward branches	117
6.1	Block diagram for the computer program.	121
6.2	A sample routine for NETWRK where the procedure to find $\{\lvert\pi_i\rvert\}$, the unconstrained shortest path lengths is illustrated	123
6.3	Flow chart for DYNAMC	124
6.4	Flow chart for CMBINT	128
6.5	Flow chart for PATH	132
6.6	A sample routine for Subroutine SELECT where the maximum confidence level ℓ_0 1→i path is sought.	133
6.7	A 5-node network with $W_{ji} = (c_{ji},p'_{ji})$	135
6.8	A 5-node network with $W_{ji} = (h_{ji},d_{ji},c_{ji})$	139
6.9	Final flow pattern.	144
6.10	A 5-node communication network model with $W_{ji} = (h_{ji}, \mu_{ji},\sigma^2_{ji})$	145

ABSTRACT

Shortest path problems in networks constitute a branch of optimization theory wherein system network models with paths which are characterized by path lengths are of concern.

This work is an extension of the classical shortest path problem; here additional cost constraints are imposed on the network model.

A dynamic programming solution approach is first formulated. It is solved by an iterative scheme in which the cost bound variable is incremented by a unit at the end of each iteration where the shortest paths with path costs not exceeding the cost bound variable are found. The concept of "stepping cost" is introduced representing the path cost at which a change of shortest path length occurs. A theorem is proved on the procedure to locate the "immediate stepping cost" and the associated stepping-cost path length; thus, an acceleration of the iterative scheme is feasible. As a result, a combinatorial algorithm is developed with computational effort being the order of the square of the number of stepping costs.

Further extensions to networks with more than one path cost constraint are presented.

The constrained shortest path algorithms are applied to flow networks and to networks with probabilistic weighted branch costs. FORTRAN computer programs are given to implement the algorithms.

CHAPTER I

INTRODUCTION AND PRELIMINARY CONCEPTS

1.1 Introduction

Because of the existence and availability of digital computers, the past few decades have seen a rapid growth in the application of scientific methods to solve problems for which there was virtually no hope of solution before. Optimization theory has emerged as a very important discipline common to many diverse problem areas.

Loosely speaking, optimization can be defined as finding the best way to carry out an action or to produce a result. More accurately, it can be said that one wishes to define a system of some type, to identify its variables and the conditions they must satisfy, to define a measure of effectiveness of the system, and then to seek the state of the system that gives the most desirable measure of effectiveness.

Problems in optimal control usually involve a measure of effectiveness in the form of a time integral, and their solution requires that a set of boundary-value equations be solved, in general. With appropriate modifications, the control problem can often be posed such that the scope of solution methods includes dynamic programming. The problem of finding a minimal value of a function of several variables requires the application of mathematical programming. When only linear algebraic equations are involved in the optimizing function and constraints, linear programming is used; otherwise, nonlinear programming is applicable.

The Simplex Method, originated by G. B. Dantzig [1], is a widely used linear programming method. Recent research has shown that large problems are likely to be decomposable into a number of smaller sized linear optimization problems; problems with several thousand variables and constraints have been solved successfully.

Nonlinear programming solutions are found through various search techniques. Concepts such as steepest descent, the Lagrange multiplier, Newton-Raphson search, and conjugate-directions are incorporated in some of the popularly used methods [35].

Dynamic programming, originated by R. Bellman [2,3], is applicable when the system can be modelled as a multistage decision process where a recurrence relation is obtained to relate the optimal decision of one stage to that of the next, so that the remaining decision is expressible in terms of the previous decision and the state of the system. Whereas both linear and nonlinear programming are suitable for solving systems described by well-behaved functions of the system state, dynamic programming is well suited for problems with discontinuities because of the discrete nature of the technique.

Of all the system optimization problems, network optimization has been considered as a highly important subdiscipline. In general, all problems in the field have the following in common: They comprise components and parts which are interconnected, and the interconnection can be described by a graph. The constituents or variables in the

networks are interdependent yet some of them are loosely coupled, thus allowing simplified solution. Certain constraints and goals are prescribed. The objective is, by and large, the optimization of such items as speed, cost, or reliability.

In transportation networks where commodities are transported through the network between specified pairs of source and destination nodes, the maximum flow problem [4] has an objective to find the optimal flow pattern so that as much flow as possible can be shipped without causing an overflow of the network capacity; the minimum cost flow problem [5] concerns the cost of shipping as the measure of effectiveness when a set of flow requirements between the source and destination pairs is prescribed. The placement of power lines [6], pipelines [7,8] are special kinds of Steiner's problem where the objective is to minimize the total interconnection length (or cost) subject to a limited budget.

As small size products are in continuous demand, the placement of logic units, the interconnection of wires [9,10], and the ordering of logic boards [11] of complicated electronic devices to minimize total wire length become nontrivial problems for which only heuristic solution methods exist at present.

Computer networks, which link data processors at different locations to form communication networks, are used to facilitate high speed information handling and transmission in addition to resource sharing. Rapid advances [12,13,14,15,16,17,18,19,20] have been seen in optimiza-

tion objective areas such as flow control, routing, message switching, delay, and reliability.

The problem of finding shortest paths in networks is a type of multistage decision process where the dynamic programming solution approach has been most efficient [21,22]. The problem first appears as a commuting problem [23] in which given a road map the shortest route from one city to another is sought. Under appropriate definitions, the shortest path has been used as an analytic tool to relate network performance; for example, packets or messages are forwarded to their destinations along the "shortest" route [24], "shortest" in the sense of minimal time-delay [16], maximal reliability [25,26], or maximal utilization of network capability [27]. The ordering of logic boards in a complicated electronic system, so that the total interconnection wire length is minimized, turns out to be a problem of finding the shortest path between a unique pair of source and sink nodes of a network model [11].

Shortest paths with special features have been studied in some works. In [28,29], consideration is given to the shortest path distribution in a probabilistic weighted network; in [30], the shortest path problem is formulated in a network with time-dependent weights; in [31], a study that corresponds to an optimal traveling scheme through a transportation network with time restrictions for the accessibility and parking on certain nodes is treated; the problem of finding shortest

paths that must pass through a given subset of nodes is discussed in [32]; and in [33], the optimal route to traverse a road network with penalties or prohibitions in making a turn is found as an application of the shortest path problem.

In this thesis, shortest paths that satisfy a constraint are treated. Section 1.2 introduces the problem of finding the constrained shortest path in a network, with the dynamic programming formulation stated in Section 1.3. Section 1.4 cites some of the basic graph definitions. Pertinent history and some of the solution approaches for the unconstrained shortest path problem are listed in Sections 1.5 and 1.6. The contribution of this research work is outlined in Section 1.7.

1.2 Fundamental Definitions

A network G(V,B) of n nodes and m branches is characterized by the set of nodes V, and the set of branches B, where each member of B connects a distinct pair of nodes in V. Let v_i be the ith node, and b_j be the jth branch, so that $V = \{v_1, v_2, \ldots, v_n\}$, $B = \{b_1, b_2, \ldots, b_m\}$.

Let (i,j) denote a branch b_q if v_i is the initial node of b_q, and v_j is the terminal node of b_q. In a directed network, b_q is said to be directed from v_i to v_j; if the network is undirected, b_q is said to be incident at v_i and v_j.

A weighted network is one in which numbers, called weights, are associated with the branches of the network. Throughout this work,

each branch (i,j) of B is assumed to be assigned a pair of weights, d_{ij} and h_{ij}, represented by $W_{ij} = (d_{ij},h_{ij})$, which we call the length and cost of (i,j), respectively. In real networks, the weights may represent things such as distance (wire, road), amount of flow (commodity), cost, capacity (road, communication channel), probability measure (reliability), or delay (time to traverse the branch).

A chain is a sequence of nodes and branches, say $v_{i_1}, b_1, v_{i_2}, b_2, \ldots, v_{i_{k-1}}, b_{k-1}, v_{i_k}$, where each branch b_j incidents at $v_{i_{j+1}}$ and v_{i_j}, $j = 1, 2, \ldots, k-1$. A loop is an unidirectional chain where $i_1 = i_k$. A path π is an unidirectional chain where the nodes are distinct. $|\pi|$ is used to denote the length of π which is the sum of the lengths of the constituent branches. $C(\pi)$ is used to denote the cost of π which is the sum of the costs of the constituent branches.

An $i \to j$ path, denoted as $\pi_{i,j}$, is a path from node i to node j. When paths from a common initial node, say node 1, are under consideration, π_j represents a $1 \to j$ path.

A network G(V,B) is said to be connected if, for any node pair v_i and v_j, an $i \to j$ path exists.

A tree of undirected network is a subset of branches that connects the set of nodes V and that has the property that one and only one path exists between each pair of nodes.

A cut $(X,\bar{X})$ is the set of branches (i,j) where $i \in X$, $j \in \bar{X}$, and X is a subset of V with $\bar{X}$ denoting its node complement V-X.

In Fig. 1.1, a directed network G with 5 nodes, and 8 branches, is shown. G is connected. The branches (1,5), (4,5), (5,2) are undirected. The set of 1→5 paths are {(1,5)}, {(1,2),(2,5)}, {(1,2),(2,3),(3,5)}, {(1,2),(2,3),(3,4),(4,5)}. If $X = \{v_1\}$, $\bar{X} = \{v_2,v_3,v_4,v_5\}$, $(X,\bar{X}) = \{(1,2),(1,5)\}$. If $X = \{v_1,v_4\}$, $(X,\bar{X}) = \{(1,2),(1,5),(4,5)\}$.

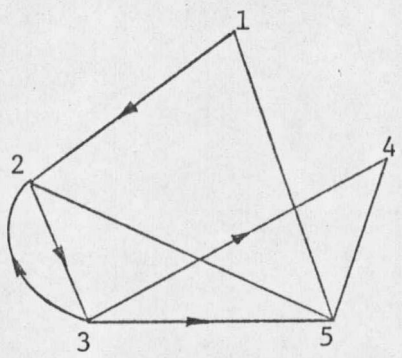


Fig. 1.1. Illustration of graph fundamentals.

1.3 The Problem Statement

Given a directed network G(V,B), a set of length weights $\{d_{ij}\}$,† a set of nonnegative cost weights $\{h_{ij}\}$, and a nonnegative constant c, find the set of shortest 1→i paths π_i, i = 2, 3, ..., n, if they exist, such that $C(\pi_i) \leq c$, for all $i \neq 1$.

If $\{\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_q}\}$ represents the set of 1→i paths, and π_i is the shortest 1-i path such that $C(\pi_i) \leq c$, then the length $|\pi_i|$ of π_i

---

†The branch lengths may be negative. However, a practical assumption is that all loop lengths must be nonnegative.

is such that

$$|\pi_i| = \underset{j}{\text{minimum}}\{|\pi_{i_j}|, \text{ such that } C(\pi_{i_j}) \leq c\} \qquad (1\text{-}3\text{-}1)$$

Equation (1-3-1) contains a minimizing operation that selects the minimum value of $|\pi_{i_j}|$ for π_{i_j} that satisfies the path cost requirement to be the shortest $1 \to i$ path length.

Intuitively, one may find the optimal path by straightforward path enumeration. However, when the size of the network is large, which is quite usual for practical network models, a large number of feasible paths exist, and naive path enumeration may become a tedious task.†

1.4 The Dynamic Programming Formulation

The principle of optimality, when applied to the problem in Section 1.3, yields a set of equations,††

$$|\pi_s(c)| = \underset{t \neq s, h_{ts} \leq c}{\text{minimum}} \{d_{ts} + |\pi_t(c-h_{ts})|\}, \text{ for } s = 2, 3, \ldots, n \qquad (1\text{-}4\text{-}1)$$

$$|\pi_1(c)| = 0 \qquad (1\text{-}4\text{-}2)$$

subject to the condition,

$$|\pi_s(c)| \leq |\pi_s(c')|, \text{ for } c' \leq c, \; s = 2, 3, \ldots, n \qquad (1\text{-}4\text{-}3)$$

---

†For a network with n nodes, the number of paths between a pair of nodes with n-2 intermediate nodes may be as large as (n-2)!.

††A derivation of equation (1-4-1) is presented in Chapter II.

where c is a nonnegative constant, and $\pi_s(c)$ is the shortest 1-s path with a path cost that is no bigger than c.

Equation (1-4-1) is a set of n-1 equations which relate each of the optimal 1→s paths to the rest of the optimal paths as a function of the cost bound c . Equation (1-4-2) simply points out the fact that the optimal path from node 1 to itself is a trivial path with zero path length. Inequality (1-4-3) is a statement of the fact that the optimal 1→s paths are nonincreasing functions of the cost bound variable c.

The multistage decision process of finding the shortest paths is such that whatever the initial cost bound c (state) and the last intermediate node (decision) are, the remaining decisions of intermediate nodes must contribute an optimal policy with regard to the cost bound resulting from the decision of the last intermediate node.

If all branch costs of the network assume zero value, a degenerate problem results: The unconstrained 1→s shortest paths π_s's are sought. Equations (1-4-1) and (1-4-2) become

$$|\pi_s| = \underset{t \neq s}{\text{minimum}}\{d_{ts} + |\pi_t|\}, \text{ for } s \neq 1 \qquad (1\text{-}4\text{-}4)$$

$$|\pi_1| = 0 \qquad (1\text{-}4\text{-}5)$$

The variable c is eliminated, since any feasible 1→s path has zero path cost. The n-1 unknown variables are sought. A large number of solution methods exist in the literature to solve (1-4-4); some of them are presented in the latter part of this chapter.

1.5 A Historical Review

One of the most efficient procedures to find the unconstrained shortest paths was described by Dijkstra [34], and by Whiting and Hillier [23] for networks with nonnegative branch lengths. For networks with possible negative branch lengths, a modified network can be found by simply adding a fixed positive amount to each of the branch lengths in the original network, so that all branch lengths become nonnegative, and Dijkstra's method is then applicable. Bellman [21], Ford [36], and Moore [37] proposed a set of functional equations by the application of the principle of optimality to relate the set of unconstrained shortest paths from a common initial node to all other terminal nodes. The equations are then solved by an iterative scheme. Yen [38,39] modified the above method, reducing computational effort by an integral factor.

While the above methods are able to find the set of $1 \to s$ shortest paths, for all $s \neq 1$, Floyd [40], Shimbel [41] published a matrix procedure, based on the work of Warshall [42], to find the unconstrained shortest paths between all node pairs. Yen [43] reduced the amount of computational effort by repetitive application of Dijkstra's algorithm. Winograd and Hoffman [44] suggested a pseudomultiplication method through network partitioning. Hu [45], Land *et al.* [46], Mills [47], taking advantage of the loosely-connected nature of many large order networks, suggested a decomposition method which reduces the problem to a set of problems of smaller size.

When the shortest path is not available because of possible network restrictions, the second shortest path is sought. Hoffman and Pavley [48] proposed a method to find the second-best path between a node pair by enumerating all of the possible deviation paths of the shortest path, and by selecting the best to be the second shortest path. Bellman and Kalaba's [50] functional formulation is a generalized expression. Dreyfus [51] suggested a general expression for the kth best path. Based on the work of Murty [52] and Yen [53], Lawler's enumeration method [54] for the kth best path is obviously more efficient.

Some of the unconstrained shortest path methods are presented in the following section as they contribute to the solution of the problem in Section 1.3.

1.6 Some Solution Methods

1.6.1 The Dynamic Programming Approach

The principle of optimality states that: Whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the initial decision.

In finding the set of 1→s shortest paths, according to the principle of optimality, if the initial decision is to take branch (t,s) to be the last branch in the 1→s path, then the remaining portion of

the path, namely, the 1→t path, must be the shortest 1→t path. The system of equations

$$\begin{aligned} |\pi_s| &= \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|\}, \text{ for } s \neq 1 \\ |\pi_1| &= 0 \end{aligned} \tag{1-6-1-1}$$

thus constitute a necessary condition for the set of path lengths $\{|\pi_s|\}$ to be optimal. In fact, it is shown in Chapter II as a special case that the solution to equation (1-6-1-1) is also unique, so that the equation is indeed both a necessary and a sufficient condition for the set $\{|\pi_s|\}$ to be the shortest 1→s path lengths.

Although these equations are interlinked in such a fashion that they cannot be solved recursively, they may be solved by defining a new sequence $\{|\pi_s|^{(k)}\}$ in the following iterative scheme,

$$\begin{aligned} |\pi_s|^{(0)} &= d_{1s}, \text{ for } s \neq 1 \\ |\pi_1|^{(0)} &= 0 \end{aligned} \tag{1-6-1-2}$$

and

$$\begin{aligned} |\pi_s|^{(k+1)} &= \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|^{(k)}\}, \text{ for } s \neq 1 \\ |\pi_1|^{(k+1)} &= 0 \end{aligned} \tag{1-6-1-3}$$

for k = 0, 1, ... Then the sequence $\{|\pi_s|^{(k)}\}$ converges in a monotonically decreasing fashion to the sequence $\{|\pi_s|\}$ as $k\to\infty$. In fact, k need never be determined beyond the value n-2. Observe that, for $s \neq 1$,

$$|\pi_s|^{(1)} = \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|^{(0)}\} \leq d_{1s} = |\pi_s|^{(0)} \qquad (1\text{-}6\text{-}1\text{-}4)$$

since t = 1 is an admissible choice; and that

$$|\pi_s|^{(2)} = \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|^{(1)}\} \leq \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|^{(0)}\}$$

$$= |\pi_s|^{(1)} \qquad (1\text{-}6\text{-}1\text{-}5)$$

Hence, by induction, the sequence $\{|\pi_s|^{(k)}\}$ is such that $|\pi_s|^{(k)} \leq |\pi_s|^{(k-1)}$ for $k \geq 1$ and $s \neq 1$.

From equation (1-6-1-1), the shortest 1→s path length is

$$|\pi_s| = \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|\} \leq d_{1s} = |\pi_s|^{(0)} \qquad (1\text{-}6\text{-}1\text{-}6)$$

and

$$|\pi_s| = \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|\} \leq \underset{t\neq s}{\text{minimum}}\{d_{ts} + |\pi_t|^{(0)}\}$$

$$= |\pi_s|^{(1)}, \text{ for } s \neq 1 \qquad (1\text{-}6\text{-}1\text{-}7)$$

Through induction, the sequence $\{|\pi_s|^{(k)}\}$

$$|\pi_s|^{(0)} \geq |\pi_s|^{(1)} \geq \cdots \geq |\pi_s|^{(k)} \geq |\pi_s| \qquad (1\text{-}6\text{-}1\text{-}8)$$

The fact that the sequence in equation (1-6-1-3) will terminate for $k \leq n-2$ follows from the fact that $|\pi_s|^{(k)}$ is the shortest 1→s path

that is restricted to contain at most k+1 branches, and there can be at most n-1 branches in any loopless path.†

1.6.2 Dijkstra's Method

In essence, this method is an extension of the dynamic programming formulation when the branch lengths of the network are nonnegative.

DEFINITION 1.1. The α-restricted shortest 1→s path, denoted as $\pi_{s,\alpha}$, is the shortest 1→s path of which the intermediate nodes are restricted to be members of a nonnull subset α of V = {1, 2, ..., n}. Node 1 is an essential member of α.

Again, a set of equations that governs the α-restricted shortest 1→s paths are found by the application of the principle of optimality.

$$|\pi_{s,\alpha}| = \underset{t\neq s, t\in\alpha}{\text{minimum}}\{d_{ts} + |\pi_{t,\alpha}|\}, \text{ for } s \neq 1^{\dagger\dagger}$$
$$|\pi_{1,\alpha}| = 0 \qquad (1\text{-}6\text{-}2\text{-}1)$$

The only way that equation (1-6-2-1) differs from equation (1-6-1-1) is the restriction that the minimization operation is now taken over a subset α of V. In other words, when α = V, the two equations become equal.

---

†A loop with negative length would be detected if convergence is not attained for k = n-2.

††The subset α may or may not contain s.

Dijkstra's method is a direct consequence of the following theorem:

Theorem 1.1 Given a nonnull proper subset α of the set of nodes $V = \{1, 2, \ldots, n\}$, where node 1 is an essential member, and that $|\pi_{s,\alpha}| = |\pi_s|$ for all $s \in \alpha$, then there exists a node, say β, $\beta \notin \alpha$, such that $|\pi_{\beta,\alpha}| = |\pi_\beta|$. Also, β is the node that satisfies

$$|\pi_{\beta,\alpha}| = \underset{q \notin \alpha}{\text{minimum}} |\pi_{q,\alpha}| \qquad (1\text{-}6\text{-}2\text{-}2)$$

Assertion 1.1 If α_1, α_2 are subsets of V, and $\alpha_1 \supseteq \alpha_2$, then $|\pi_{s,\alpha_1}| \leq |\pi_{s,\alpha_2}|$ for all $s \neq 1$.

Lemma 1.1 For any subset α_1 of V, if $q \in \alpha_1$, then $|\pi_{q,\alpha_1}| = |\pi_{q,\alpha_1 - q}|$.

Proof. From the last assertion we have $|\pi_{q,\alpha_1}| \leq |\pi_{q,\alpha_1 - q}|$.

If, however, the path π_{q,α_1} contains the node q as one of its intermediate nodes, then π_{q,α_1} can be expressed as a concatenation of $\pi_{q,\alpha_1 - q}$ and some loop containing node q. Since the branch lengths are nonnegative, the loop is of nonnegative length, and it must follow that $|\pi_{q,\alpha_1}| \geq |\pi_{q,\alpha_1 - q}|$. The conclusion is that $|\pi_{q,\alpha_1}| = |\pi_{q,\alpha_1 - q}|$.

Lemma 1.2 For any node r where $r \notin \alpha$, and a node q, $q \neq r$, then

$$|\pi_{r,\alpha \cup q}| = \text{minimum}\{|\pi_{r,\alpha}|, d_{qr} + |\pi_{q,\alpha}|\} \qquad (1\text{-}6\text{-}2\text{-}3)$$

Proof. According to equation (1-6-2-1),

$$|\pi_{r,\alpha\ q}| = \underset{t\neq r,t\ \alpha\ q}{\text{minimum}}\ \{d_{tr} + |\pi_{t,\alpha\ q}|\}$$

$$= \text{minimum}\{\text{minimum}\{d_{tr} + |\pi_{t,\alpha\ q}|\},\ d_{qr} + |\pi_{q,\alpha\ q}|\} \qquad (1\text{-}6\text{-}2\text{-}4)$$

By the definition of α, and that $t \quad \alpha$, $|\pi_{t,\alpha\ q}| = |\pi_t| = |\pi_{t,\alpha}|$; this with the result of Lemma 1.2 reduces equation (1-6-2-4) to

$$|\pi_{r,\alpha\ q}| = \text{minimum}\{\text{minimum}\{d_{tr} + |\pi_{t,\alpha}|\},\ d_{qr} + |\pi_{q,\alpha}|\}$$

$$= \text{minimum}\{|\pi_{r,\alpha}|,\ d_{qr} + |\pi_{q,\alpha}|\} \qquad (1\text{-}6\text{-}2\text{-}5)$$

Proof of Theorem 1.1.

Let r, p, and q be distinct nodes not belonging to the subset α. Extending the result of Lemma 1.2, equation (1-6-2-5) is

$$|\pi_{r,\alpha\ q\ p}| = \text{minimum}\{\text{minimum}\{d_{tr} + |\pi_{t,\alpha\ q\ p}|\},\ d_{qr} + |\pi_{q,\alpha\ q\ p}|,$$

$$d_{pr} + |\pi_{p,\alpha\ q\ p}|\}$$

$$= \text{minimum}\{|\pi_{r,\alpha}|,\ d_{qr} + |\pi_{q,\alpha\ p}|,\ d_{pr} + |\pi_{p,\alpha\ q}|\}$$

$$= \text{minimum}\{|\pi_{r,\alpha}|,\ d_{qr} + |\pi_{q,\alpha}|,\ d_{qr} + d_{pq} + |\pi_{p,\alpha}|,$$

$$d_{pr} + |\pi_{p,\alpha}|,\ d_{pr} + d_{qp} + |\pi_{q,\alpha}|\} \qquad (1\text{-}6\text{-}2\text{-}6)$$

By induction, if Q is the set such that $Q = V\text{-}\alpha = \{\ell_1, \ell_2, \ldots, \ell_m\}$, then

$$|\pi_{r,\alpha \cup Q}| = |\pi_r| = \underset{\ell_j \notin \alpha}{\text{minimum}}\{|\pi_{r,\alpha}|,\ |\pi_{\ell_j,\alpha}| + \text{length of any } \ell_j \to r \text{ path through the set } Q\} \qquad (1\text{-}6\text{-}2\text{-}7)$$

The node r has been arbitrarily chosen. Instead, let r be the node such that

$$|\pi_{r,\alpha}| = \underset{q \notin \alpha}{\text{minimum}} |\pi_{q,\alpha}| \qquad (1\text{-}6\text{-}2\text{-}8)$$

Because the branch lengths are nonnegative, so is any $\ell_j \to r$ path length, and equation (1-6-2-7) becomes

$$|\pi_r| = |\pi_{r,\alpha}|$$

and β of equation (1-6-2-2) is equal to r.

Equations (1-6-2-2) and (1-6-2-3) automatically suggest an algorithm to find the set of unconstrained shortest $1 \to s$ paths: Given the set α, and the α-restricted $1 \to s$ paths, a new member of α, node β, can be found by equation (1-6-2-2). With this node β, the set of $(\alpha \cup \beta)$-restricted shortest $1 \to s$ paths for all $s \notin \alpha \cup \beta$ can be found by equation (1-6-2-3). Since β is a new member of α, $\alpha \cup \beta$ replaces α. New members of α can then be found by repetitive application of the procedure until α becomes V.

Dijkstra's Algorithm

Step 1. $\alpha = \{1\}$, $k = 1$, $|\pi_{r,\alpha}| = d_{1r}$, for $r \neq 1$.

Step 2. Find β_k, where $\beta_k \notin \alpha$, such that

$$|\pi_{\beta_k,\alpha}| = \underset{r \notin \alpha}{\text{minimum}} |\pi_{r,\alpha}|.$$

Step 3. $\alpha = \alpha \cup \beta_k$. If $\alpha \neq V$, go to Step 4; otherwise terminate.

Step 4. For all $r \notin \alpha$

$$|\pi_{r,\alpha}| = \text{minimum}\{|\pi_{r,\alpha-\beta_k}|, d_{\beta_k r} + |\pi_{\beta_k,\alpha}|\}$$

Let $k = k+1$. Return to Step 2.

The computation requires n^2 comparisons and $n^2/2$ additions, while the dynamic programming method mentioned before requires up to n^3 comparisons and additions. Dijkstra's method is the most efficient method developed so far.

A combinatorial algorithm for the constrained shortest path which is similar to the rationale of Dijkstra's method is presented in Chapter III.

1.6.3 The Matrix Method

The matrix method originated by Warshall, and presented in an algorithmic form by Floyd, is most elegant in that it is simple to program and shortest paths between all node pairs are found simultaneously.

Let $\pi^k_{i,j}$ denote the shortest $i \to j$ path subject to the condition that it may only pass through nodes 1, 2, ..., k. $\pi^0_{i,j}$ is the branch

(i,j), if it exists; otherwise a large value is assigned for the path length.

Again the principle of optimality asserts that given the set of optimal i→j paths $\pi^k_{i,j}$, then $\pi^{k+1}_{i,j}$ must be either equal to $\pi^k_{i,j}$ or equal to the concatenation of $\pi^k_{i,k+1}$ and $\pi^k_{k+1,j}$. Since if $\pi^{k+1}_{i,j}$ does not contain node k+1, $\pi^{k+1}_{i,j}$ must equal $\pi^k_{i,j}$; if $\pi^{k+1}_{i,j}$ contains node k+1, the portion of i→j path from i to k+1 must be the shortest path, and the other portion of i→j path from k+1 to j must also be the shortest path.

The matrix algorithm starts with $|\pi^0_{i,j}| = d_{ij}$. The minimization

$$|\pi^{k+1}_{i,j}| = \text{minimum}\{|\pi^k_{i,j}|, |\pi^k_{i,k+1}| + |\pi^k_{k+1,j}|\} \qquad (1\text{-}6\text{-}3\text{-}1)$$

is in effect for all $i,j \neq k+1$.

Equation (1-6-3-1) is performed for k = 0, 1, ..., n-3. The set of shortest i-j paths is obtained when k = n-3.

1.6.4 Observations

The methods presented in this section to find the unconstrained shortest paths are efficient and easy to program; moreover, from these methods, constrained shortest path algorithms to solve the problem in Section 1.3 may be derived as an extension.

The dynamic programming method finds the optimal paths through the solution of a set of equations where the solution approach, based on

the shortest paths that may contain at most k branches, is to generate the shortest paths that may contain k+1 branches in an iterative manner. The matrix method is similar in that, based on the shortest paths that may contain a subset of nodes as intermediate nodes, the shortest paths that may contain an additional intermediate node are found iteratively. Dijkstra's method is a labelling scheme where nodes are labelled in a sequential manner.

Lawler [54] suggested an algorithm that requires computational effort in the order of kn^3 to find the first k best paths between a given node pair. The method is essentially an enumeration type. Intuitively one may apply the scheme to find the $k+1^{th}$ best path if all of the first k best paths fail to satisfy the path cost requirement. However, no practical upper bound on the computational effort can be guaranteed, and the method is not recommended for further study.

1.7 Contribution of the Research

In this work the problem of constrained shortest paths is investigated. Several existing methods for the unconstrained shortest path problem from which possible extensions to the solution of the constrained shortest path problem are studied.

A set of necessary and sufficient conditions for constrained shortest paths is first established through an application of the principle of optimality. Two methods of solving constrained shortest

path problems are introduced, as briefly described in the following two paragraphs.

An iterative scheme to solve the set of equations for the case where branch costs are integer valued is presented first. This scheme is extended to the case where branch costs may be any real values by using discrete approximations; this results in suboptimal solutions wherein the degree of closeness to optimality is specified by a given error bound. It is shown that the computational effort is inversely proportional to this error bound.

The second solution method involves the notion of "stepping costs" which is introduced in this thesis to accelerate convergence of the iterative scheme. As a result, a combinatorial algorithm is generated which locates the stepping costs and the associated constrained shortest paths in a sequential manner. No integer valued limitation for the branch costs is required. If the number of stepping costs is mn, the computational effort (number of additions or comparisons) is found to be of the order of $(mn)^2$.

Certain optimal paths in probabilistic weighted networks are found to be constrained shortest paths. The design of computer networks may be facilitated by a study of some constrained shortest paths. Applications in other areas are presented through computational examples.

CHAPTER II

THE CONSTRAINED SHORTEST PATH

2.1 Introduction

There exists a wide variety of literature on the problem of finding unconstrained shortest paths in networks. However, in practice, one may be interested in finding a shortest path which, in addition to being shortest in length, satisfies certain requirements. In this chapter, a particular kind of shortest path, one for which there is a cost that is defined as the sum of the costs of the constituent branches and that satisfies a given cost bound, is investigated. We will call such an optimal path the cost constrained shortest path, or in short, the constrained shortest path -- optimal in the sense of being shortest in path length while the path cost satisfies a cost bound. Furthermore, if c represents the cost bound, the phrase c-constrained shortest path will denote such an optimal path.

2.2 The Constrained Shortest Path

Given is a network in which each of the branches (i,j) is weighted with the pair (d_{ij},h_{ij}), where d_{ij} and h_{ij} are, respectively, the length and the cost to traverse the branch. Let the cost of a path be the summation of the constituent branch costs. Then the problem is, given a nonnegative real value as the cost bound for any allowed path, find the shortest path from node 1 to node s, such that the path cost is no bigger than the cost bound.

If $\{\pi_{s_1}, \pi_{s_2}, \ldots, \pi_{s_q}\}$ is the set of 1→s paths, and c the cost bound, the shortest 1→s path π_s that satisfies the cost bound is the path with

$$\text{length of } \pi_s = \underset{j\in(1,2,\ldots,q)}{\text{minimum}} \{\text{length of } \pi_{s_j} \text{ such that cost of } \pi_{s_j} \leq c\} \qquad (2\text{-}2\text{-}1)$$

2.3 Solution Formulation

Define d_{ts} to be the length of the branch (t,s),

h_{ts} to be the cost of the branch (t,s),

$\pi_s(c)$ to be the c-constrained shortest 1→s path, and

$|\pi|$ to be the length of the path π.

Now consider the particular case where the terminal node of interest is node s. If (t,s) is the last branch in the path $\pi_s(c)$, where $t \neq s$, then the principle of optimality asserts that the optimal 1→s path must be the union of the branch (t,s), and the $(c-h_{ts})$-constrained 1→t path $\pi_t(c-h_{ts})$. $\pi_s(c)$ can be written as

$$\pi_s(c) = \pi_t(c-h_{ts}) \cup (t,s) \qquad (2\text{-}3\text{-}1)$$

and the length $|\pi_s(c)|$ of $\pi_s(c)$ is then

$$|\pi_s(c)| = |\pi_t(c-h_{ts})| + d_{ts} \qquad (2\text{-}3\text{-}2)$$

Also,

$$|\pi_s(c)| \leq |\pi_j(c-h_{js})| + d_{js}, \text{ for } j \neq t,\ h_{js} \leq c \qquad (2\text{-}3\text{-}3)$$

This situation is illustrated by the following figure:

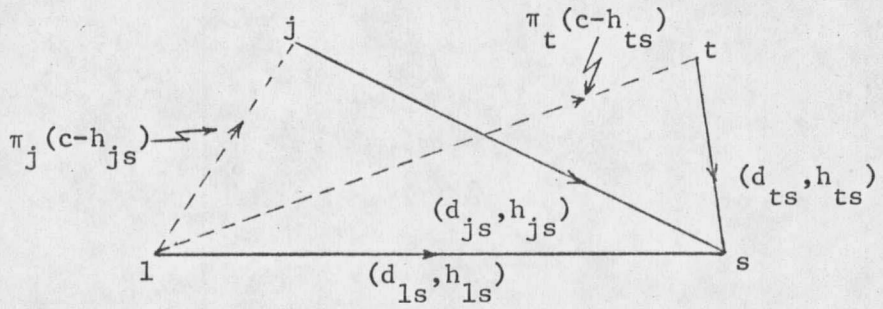


Fig. 2.1. Concatenation of paths.

Note that equation (2-3-2) can also be true for t = 1. For under this situation, the optimal 1→s path is simply the direct branch (1,s) with length d_{1s}, cost h_{1s}, and $\pi_1(c-h_{1s})$ being a trivial path from node 1 to itself with zero path length.

The above arguments suggest the optimal 1→s path, for s ≠ 1, can be represented by the system of equations,

$$|\pi_s(c)| = \underset{t\neq s,\, h_{ts}\leq c}{\text{minimum}} \{d_{ts} + |\pi_t(c-h_{ts})|\}^{\dagger} \qquad (2\text{-}3\text{-}4)$$

---

†In fact, without loss of generality, the condition $h_{ts} \leq c$ can be dropped, since any path with a negative cost bound does not exist, and a very large value can be assigned for its length in computation.

$$|\pi_1(c)| = 0 \tag{2-3-5}$$

The fact that $|\pi_s(c)|$ is a nonincreasing function of the cost bound c is expressed as the following inequality:

$$|\pi_s(c)| \leq |\pi_s(c')|, \text{ for } c' \leq c \tag{2-3-6}$$

Equations (2-3-4) to (2-3-6) form a set of necessary conditions that govern the n-1 shortest distance functions $|\pi_s(c)|$'s in terms of the cost variable c.

2.4 Proof of Validity

In the previous section, it has been shown that the set of optimal path lengths satisfy the necessary conditions (2-3-4) to (2-3-6). If the solution of equations (2-3-4) to (2-3-6) is unique, then it must also be the set of optimal path lengths. The proof of uniqueness of the solution is given in the following.

For a given cost variable c, let $\{|\pi_i^*(c)|\}$, $\{|\pi_i(c)|\}$ be two sets of solutions. Suppose k is the label where the finite difference $|\pi_i(c)| - |\pi_i^*(c)|$ achieves its maximum. Then $|\pi_k(c)|$, $|\pi_k^*(c)|$ are both finite valued, and there must be an r and s, where $r \neq k$, $s \neq k$, with the property that

$$|\pi_k(c)| = |\pi_r(c-h_{rk})| + d_{rk} \leq |\pi_s(c-h_{sk})| + d_{sk} \tag{2-4-1}$$

$$|\pi_k^*(c)| = |\pi_s^*(c-h_{sk})| + d_{sk} \leq |\pi_r^*(c-h_{rk})| + d_{rk} \qquad (2\text{-}4\text{-}2)$$

From (2-4-1) and (2-4-2), the following inequality is obtained:

$$|\pi_k(c)| - |\pi_k^*(c)| \leq |\pi_s(c-h_{sk})| - |\pi_s^*(c-h_{sk})| \qquad (2\text{-}4\text{-}3)$$

Since $|\pi_k^*(c)|$ is finite, and $|\pi_s^*(c-h_{sk})|$, being the length of part of the optimal $1 \to k$ path, must be finite, too. Then there must be a label p, $p \neq k$[†], such that

$$|\pi_s^*(c^s)| = |\pi_p^*(c^s-h_{ps})| + d_{ps} \qquad (2\text{-}4\text{-}4)$$

where $c^s \triangleq c-h_{sk}$.

Following arguments similar to the ones above, the following inequality is in order,

$$|\pi_s(c^s)| - |\pi_s^*(c^s)| \leq |\pi_p(c^s-h_{ps})| - |\pi_p^*(c^s-h_{ps})| \qquad (2\text{-}4\text{-}5)$$

and $|\pi_p^*(c^s-h_{ps})|$ must also be finite. In accordance, a sequence of distinct labels s, p, ..., with the property that

$$|\pi_k(c)| - |\pi_k^*(c)| \leq |\pi_s(c^s)| - |\pi_s^*(c^s)| \leq |\pi_p(c^p)| - |\pi_p^*(c^p)| \leq \ldots \qquad (2\text{-}4\text{-}6)$$

can be found, where again $c^p \triangleq c^s-h_{ps}$, and that the corresponding costs form a nonincreasing sequence

---

[†]If $p = k$, the condition in (2-3-6) will be violated.

$$c = c^k \geq c^s \geq c^p \geq \ldots \tag{2-4-7}$$

Eventually, the label 1 must occur in the sequence so that differences in (2-4-6) remain finite, and (2-4-6) becomes

$$|\pi_k(c)| - |\pi_k^*(c)| \leq |\pi_s(c^s)| - |\pi_s^*(c^s)| \leq \ldots \leq |\pi_1(c^1)| - |\pi_1^*(c^1)| = 0 \tag{2-4-8}$$

which shows that, for an arbitrary cost variable c, each finite member of the set of $\{\pi_i(c)\}$ is no greater than the corresponding member of the set $\{\pi_i^*(c)\}$. We would like to show that the reverse is also true in the following.

Similarly, let ℓ be the label where $|\pi_i(c)| - |\pi_i^*(c)|$ achieves its minimum. It can be found that there exists a sequence of inequalities

$$0 = \ldots \leq |\pi_\ell(c)| - |\pi_\ell^*(c)| \leq \ldots \leq |\pi_k(c)| - |\pi_k^*(c)| \leq \ldots = 0 \tag{2-4-9}$$

The above procedure establishes the fact that the corresponding members of the sets $\{|\pi_i^*(c)|\}$ and $\{|\pi_i(c)|\}$ are of equal value, assuming they are finite.

To complete the proof, suppose $|\pi_k(c)|$ is of infinite value, while $|\pi_k^*(c)|$ is finite. From (2-3-4), $|\pi_k^*(c)|$ can be expressed by

$$|\pi_k^*(c)| = |\pi_s^*(c-h_{sk})| + d_{sk} = |\pi_p^*(c^s-h_{ps})| + d_{ps} + d_{sk}$$

$$= |\pi_1^*(c^1)| + d_{1q} + \ldots + d_{pd} + d_{sk} = d_{1q} + \ldots + d_{ps} + d_{sk} \qquad (2\text{-}4\text{-}10)$$

But in the same way, the right-hand side of (2-4-10) can be an expression for $|\pi_k(c)|$ having a finite value, which leads to a contradiction of the infinite-value assumption.

2.5 The Stepping Costs

Before the computational method is introduced in the next section, we would like to present some of the properties of the constrained shortest paths of networks through the concept of "Stepping Costs."

Let the set of 1→s paths, denoted as $\{\pi_{s_i}\}$, be divided into subsets according to their path costs, so that L(c) is the subset of 1-s paths the costs of which are equal to c. With this division of paths, the c-constrained shortest 1-s path $\pi_s(c)$ is then the one whose length $|\pi_s(c)|$ satisfies

$$|\pi_s(c)| = \underset{\pi_s \in L(c'),c' \leq c}{\text{minimum}} \{|\pi_{s_i}|\} \qquad (2\text{-}5\text{-}1)$$

which can be rewritten as

$$|\pi_s(c)| = \text{minimum}\{\underset{\pi_{s_i} \in L(c)}{\text{minimum}}\{|\pi_{s_i}|\}, \underset{\pi_{s_i} \in L(c'),c'<c}{\text{minimum}}\{|\pi_{s_i}|\}\} \qquad (2\text{-}5\text{-}2)$$

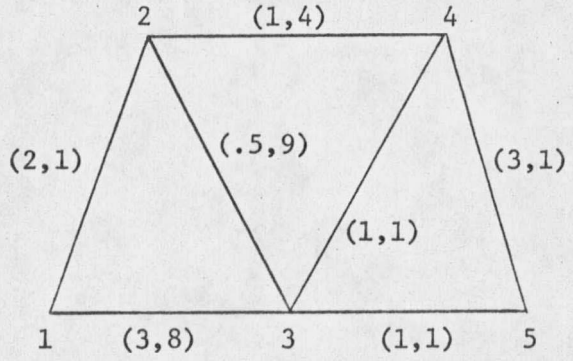


Fig. 2.2. A five-node network with $W_{ij} = (d_{ij}, h_{ij})$.

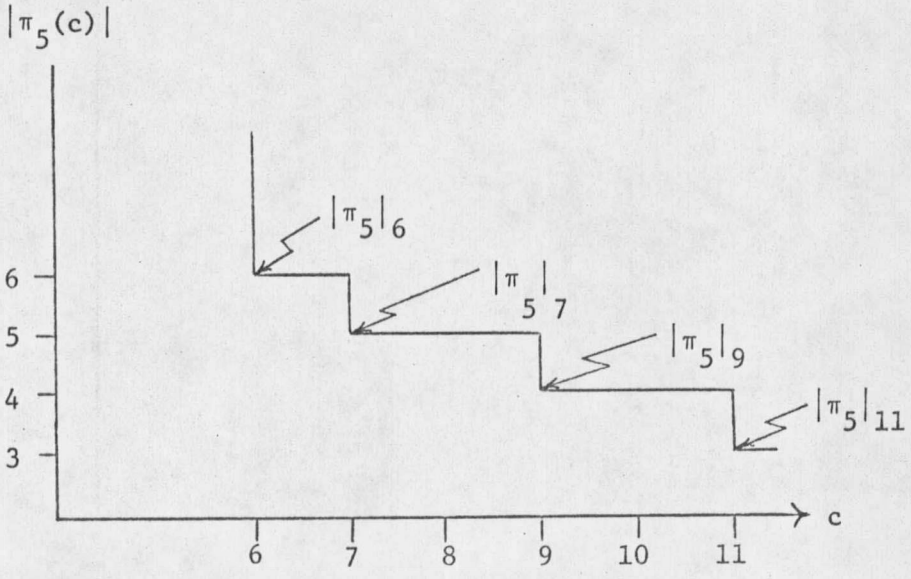


Fig. 2.3. $|\pi_5(c)|$ of the five-node network in Fig. 2.2.

To simplify notation, define $|\pi_s|_c = \underset{\pi_{s_i} \in L(c)}{\text{minimum}}\{|\pi_{s_i}|\}$; (2-5-2) is then

$$|\pi_s(c)| = \underset{c'<c}{\text{minimum}}\{|\pi_s|_c,\ |\pi_s(c')|\}, \qquad (2\text{-}5\text{-}3)$$

which relates the c-constrained shortest path to the c'-constrained shortest path.

It becomes obvious that, through (2-5-3), $|\pi_s(c)|$ is a non-increasing piecewise linear function of the cost variable c.

To illustrate, a five-node network is presented in Fig. 2.2, with $|\pi_5(c)|$ plotted in Fig. 2.3.

DEFINITION 2.1. Stepping costs are those costs for which an incremental increase in cost results in a decrease in shortest path length; the corresponding path is called a stepping-cost path.

For example, in Fig. 2.3, the stepping costs are 6, 7, 9, and 11; the 1-5 stepping-cost path at cost equal to 11 is the sequence of branches (1,2), (2,3), (3,5).

Property 1. At a stepping cost c_o, $|\pi_s|_{c_o} = |\pi_s(c_o)|$.

Property 2. For each finite constrained shortest path $\pi_s(c)$, there exists a stepping cost c_o, $c_o \leq c$, such that $|\pi_s|_{c_o} = |\pi_s(c)|$.

For c being a stepping cost, the above statement is trivially true. If c is not a stepping cost, by Definition 2.1, the cost c', $c' < c$, is such that $|\pi_s(c')| = |\pi_s(c)|$. The cost c' can then be tested to see if

it is a stepping cost. Since $|\pi_s(c)|$ is finite, by (2-5-3), eventually there must exist a stepping cost c_o, such that $|\pi_s|_{c_o} = |\pi_s(c_o)| = \ldots = |\pi_s(c')| = |\pi_s(c)|$.

2.6 The Constrained Shortest Path Algorithm

In order to provide a computationally efficient method for finding the constrained shortest path length, we shall assume the branch costs h_{ts}'s are restricted to assume only nonnegative integer values.† (For less restriction, a nonnegative real unit can be used instead of the integer-valued restriction.) Accordingly, constrained shortest paths with costs no bigger than a member of the set $\{0, 1, 2, \ldots, \tau\}$, denoted as S_I, are of interest, where τ represents the largest integer cost bound.

Equations (2-3-4) through (2-3-6) for $c \in S_I$ can now be rewritten as

$$|\pi_s(c)| = \underset{t\neq s, h_{ts}\leq c}{\text{minimum}} \{d_{ts} + |\pi_t(c-h_{ts})|\}, \text{ for } s \neq 1 \qquad (2\text{-}6\text{-}1)$$

$$|\pi_s(c)| = 0 \qquad (2\text{-}6\text{-}2)$$

$$|\pi_s(c)| \leq |\pi_s(c')|, \text{ for } c' \leq c \qquad (2\text{-}6\text{-}3)$$

First, consider a special case where all branch costs are zero; the constrained shortest paths will then become the unconstrained ones. The set of unconstrained shortest paths can be found using the successive

†This assumption is removed in the next chapter.

approximation method discussed in Chapter I, i.e.,

$$|\pi_s|^{(0)} = \infty, \text{ for } s \neq 1 \qquad (2\text{-}6\text{-}4)$$

$$|\pi_s|^{(k+1)} = \underset{t \neq s}{\text{minimum}}\{d_{ts} + |\pi_t|^{(k)}\}, \text{ for } s \neq 1 \qquad (2\text{-}6\text{-}5)$$

and iterates with $k = 0, 1, 2, \ldots$

A similar approach can be used for the general case. The algorithm is described by the following steps.

Step 1. $c = 0$; $|\pi_s(c-1)| = \infty$, for $s \neq 1$.

Step 2. $|\pi_s(c)|^{(0)} = |\pi_s(c-1)|$, for $s \neq 1$; $k = 0$.

Step 3. $|\pi_s(c)|^{(k+1)} = \underset{t \neq s, h_{ts} \leq c}{\text{minimum}} \{d_{ts} + f_t\}$, for $s \neq 1$, where

$$f_t = \begin{cases} |\pi_t(c)|^{(k)}, & \text{if } h_{ts} = 0 \\ |\pi_t(c-1)|, & \text{if } h_{ts} > 0 \end{cases}$$

Step 4. If $|\pi_s(c)|^{(k+1)} = |\pi_s(c)|^{(k)}$ for all $s \neq 1$, go to Step 5; otherwise $k = k+1$, go to Step 3.

Step 5. $|\pi_s(c)| = |\pi_s(c)|^{(k+1)}$, for $s \neq 1$; if $c < \tau$, $c = c+1$, go to Step 2; otherwise stop.

Essentially, Step 2 through Step 4 are the same successive approximation scheme described in Chapter I; except in Step 3, the minimizing operation is taken over the set of shortest path lengths with costs no greater than c, c-1, ..., 0, depending on the value h_{ts}. The $|\pi_s(c)|$'s

are found by iterating on the inner loop of Steps 2 through 4, until the maximum cost bound τ is reached. Since the amount of computation in the inner loop is bounded from above by n^3, the total amount of computation is bounded from above by τn^3. However, in general, there are relatively few branches with zero cost, and the amount of computation is well under τn^3. In fact, it is quite often true that all branches have nonzero costs, in which case $|\pi_s(c)|$ can be found by two passes on Step 3, and the computation is reduced to $2\tau n^2$.

The computation associated with (2-3-4) can often be reduced by observing that no finite valued shortest $1 \to s$ path exists for cost less than $\min_{t \neq s} h_{ts}$, since if (t,s) is the last branch in the optimal path, the path cost must be no less than the branch cost h_{ts}; and t can be any label other than s. Thus, no finite valued shortest paths exist for cost less than $c_{min} \triangleq \text{minimum}\{\min_{t \neq s} h_{ts}\} = \text{minimum}_{t,s,t \neq s} h_{ts}$. In Step 1 c can have c_{min} as its starting value instead of zero.

2.7 The Extended Algorithm

The constrained shortest path problem analyzed above falls short of wide-range applicability in that the branch costs are restricted to be integer-valued or to be multiples of some real unit, a restriction which limits the applicability of the algorithm to a small subset of physical system networks models. Hence, an effort is made in this section to determine if a compromise exists between the algorithm and

the network model. It first seems reasonable to modify the algorithm so that the integer-valued branch cost requirement is no longer necessary. Depending on how this is done, the computational effort may greatly increase, and the method may not offer an advantage over the naive path enumeration approach. So we turn our attention to the aspect of network remodeling.

In essence, we will try to find suboptimal paths by solving for the constrained shortest path of an approximate network through the algorithm described in the last section. The degree of accuracy of the approximation versus the amount of computational effort will then be discussed.

Let G^{α} denote a network that is similar to the original network G for which the constrained shortest paths are required. G^{α} has the same network structure as that of G, i.e., the two networks (G,G^{α}) have the same set of nodes and the same set of branches. Also, each branch of G^{α} is associated with the same length as that of G. The only difference between the two networks is, for each branch, the branch costs are related by a multiplicative factor α, such that $h^{\alpha}_{ij} = h_{ij}/\alpha$, where h^{α}_{ij}, h_{ij} are, respectively, the $(i,j)^{th}$ branch cost of network G^{α}, G.

Because of the similarity of the two networks, one tends to conjecture that the L-constrained shortest 1→s path of G is the same as the (L/α)-constrained shortest 1→s path of G^{α}. In fact, this conjecture

is true. The proof is straightforward and will be omitted here. The network G^{α} is said to be equivalent to G.

Now let G^{α} be modified; let the branch cost h^{α}_{ij} be taken as a truncated multiple of the unit α, so that $h^{\alpha}_{ij} = \alpha[h_{ij}/\alpha]$, where $[\cdot]$ denotes the largest integer not exceeding the value $\cdot$. G^{α} is no longer an equivalent network of G, unless, of course, the branch costs h_{ij}'s are integral multiplies of α. However, if α is small, $h^{\alpha}_{ij} \approx h_{ij}$, since the truncation error is bounded from above by the quantity α, which approaches zero as α approaches zero. G^{α} can then be called an approximate network of G. With this multiplicity nature, the constrained shortest path algorithm is applicable to G , where the cost variable c increments a unit α for each step of iteration. The constrained shortest paths of G^{α} are then a set of suboptimal paths of the original network G.

One would observe that a largest common divisor exists for a set of real values. When taken to be the cost increment α, G^{α} becomes equivalent to G, and the suboptimal paths are therefore the optimal paths of G. This choice of α appears plausible at first sight but falls short of reality in that this value of α could be so small that computational effort could become formidably large.

In general, a starting unit size α would be chosen which results in an amount of computation well under a given bound. Then a decrease in the unit α by an integral factor would be made and the procedure

repeated until either the computational bound is exceeded or satisfactory suboptimal paths are obtained.

It remains to find the relationship between the unit and the approximation accuracy. To facilitate this, a network G, along with its approximate network G^1 are shown in Fig. 2.4a and Fig. 2.4b. The constrained shortest 1→5 path lengths $|\pi_5(c)|$, $|\pi_5^1(c)|$ of G, G^1, respectively, are plotted in Fig. 2.4c.

First, observe that $|\pi_5^1(c)|$ is less than or equal to $|\pi_5(c)|$ for any cost variable c. This occurs because some path costs are reduced through truncation, whereas no path costs are increased in G.

Secondly, for every stepping cost of G^1, the corresponding stepping cost in network G occurs at a larger value of c.

Due to discontinuities in the path length, it is not practical to define the degree of approximation by the amount of change in shortest path lengths; instead, the amount of change of path cost will be used.

DEFINITION 2.2 The degree of approximation, of the network G^α to G by truncating the branch cost of G to integral multiples of the unit α, is the maximum decrease of the costs of paths which correspond to the set of stepping costs of G^α.

Let $\{c_{s_j}^\alpha\}$ be the set of 1→s stepping costs in G^α, $\{\pi_{s_j}\}$ be the corresponding set of 1→s paths, and $\{c_{s_j}\}$ be the corresponding set of

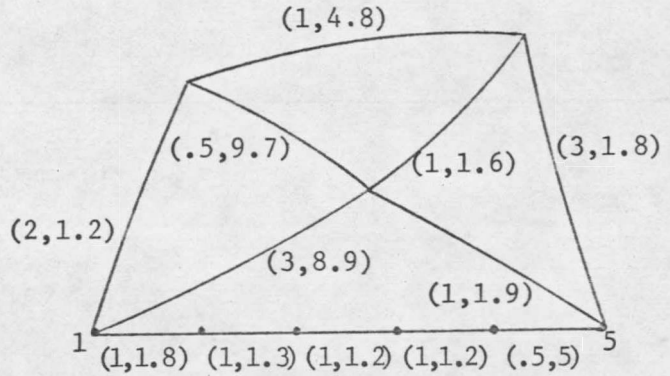


Fig. 2.4a. A network G with $W_{ij} = (d_{ij}, h_{ij})$.

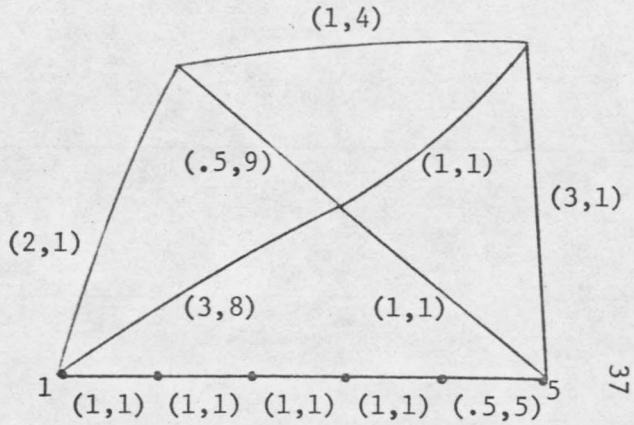


Fig. 2.4b. An approximate network G^1 of G in Fig. 2.4a with unit metric.

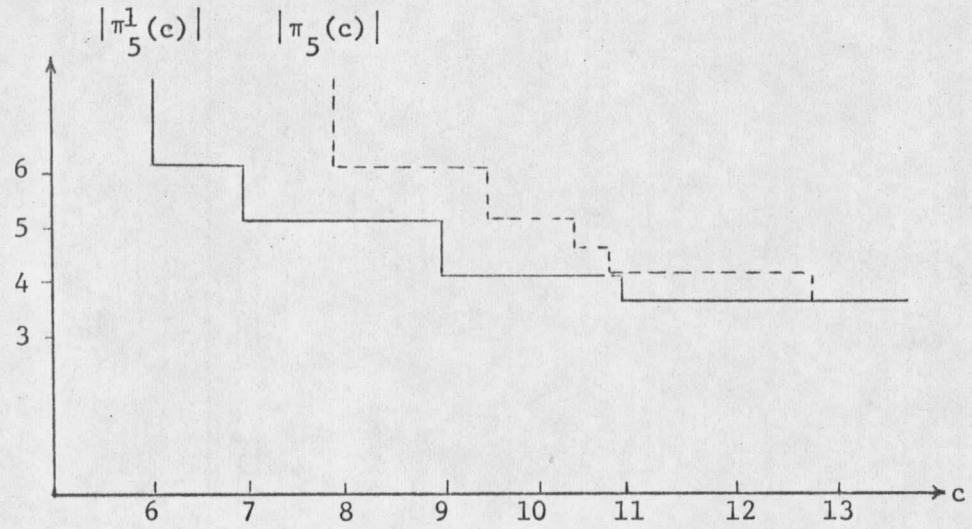


Fig. 2.4c. Constrained shortest 1↦5 paths of G, G^1 of Figs. 2.4a and 2.4b.

path costs in G. We denote the 1→s degree of approximation by A^{α}_{1s}. A^{α}_{1s} is accordingly

$$A^{\alpha}_{1s} \triangleq \underset{j}{\text{minimum}}\{c_{s_j} - c^{\alpha}_{s_j}\} \qquad (2\text{-}7\text{-}1)$$

If A^{α}_{1s} is less than a given error bound, then G^{α} is said to be within satisfactory accuracy as far as the constrained shortest 1→s path is concerned.

The extended algorithm is displayed as follows:

Step 1. Initialize α, ε.

Step 2. Generate network G^{α}.

Step 3. Evaluate $|\pi_{s_j}|$ of G^{α} through the constrained shortest path algorithm, for all $s \neq 1$, for $c = \{0, \alpha, 2\alpha, \ldots, \alpha[\tau/\alpha]\}$.

Step 4. Evaluate $A^{\alpha}_{1s} = \underset{j}{\text{maximum}}\{c_{s_j} - c^{\alpha}_{s_j}\}$.

Step 5. If $A^{\alpha}_{1s} \leq \varepsilon$, terminate; otherwise go to Step 6.

Step 6. $\alpha = \alpha/2$; go to Step 2.

2.8 Networks With Several Cost Weights

Shortest path problems of networks with several cost weights are natural extensions of the previously treated constrained shortest path problem.

Suppose there are K nonnegative cost weights, h_{ji}^1, h_{ji}^2, ..., h_{ji}^K, associated with each branch (j,i) of the network, so that the joint weight W_{ji} of (j,i) can be written as $W_{ji} = (d_{ji}, h_{ji}^1, h_{ji}^2, \ldots, h_{ji}^K)$. The problem is to find, for a given node pair and a set of cost bounds, $\tau_1, \tau_2, \ldots, \tau_K$, such that the shortest path π satisfies

$$C_k(\pi) \leq \tau_k, \text{ for } k = 1, 2, \ldots, K \qquad (2\text{-}8\text{-}1)$$

where

$$C_k(\pi) \triangleq \sum_{(j,i)\in\pi} h_{ji}^k \qquad (2\text{-}8\text{-}2)$$

is called the type k path cost.

Similar to the arguments for the constrained shortest path of a network with a single branch cost, let $\pi_i(c_1, c_2, \ldots, c_K)$ denote the shortest 1→i path such that the type k path cost is not greater than the nonnegative value c_k, k = 1, 2, ..., K; $\pi_i(c_1, c_2, \ldots, c_K)$ then is the solution of the following set of equations,

$$|\pi_i(c_1, c_2, \ldots, c_K)| = \underset{j\neq i, h_{ji}^k \leq c_k, \text{ for } k = 1, 2, \ldots, K}{\text{minimum}}$$

$$\{d_{ji} + |\pi_j(c_1 - h_{ji}^1, c_2 - h_{ji}^2, \ldots, c_k - h_{ji}^k, \ldots, c_K - h_{ji}^K)|\},$$

$$\text{for } i \neq 1 \qquad (2\text{-}8\text{-}3)$$

$$|\pi_1(c_1, c_2, \ldots, c_K)| = 0 \qquad (2\text{-}8\text{-}4)$$

$$|\pi_i(c_1, c_2, \ldots, c_K)| \leq |\pi_i(c_1', c_2', \ldots, c_K')|, \text{ for } c_k' \leq c_k,$$

$$k = 1, 2, \ldots, K \qquad (2\text{-}8\text{-}5)$$

Equations (2-8-3) through (2-8-5) express the shortest $1 \rightarrow i$ path of type k path cost not greater than c_k in terms of other shortest $1 \rightarrow j$ paths of type k path cost not greater than c_k, for $k = 1, 2, \ldots, K$. Hence, if the path costs assume only integer values, the constrained shortest path algorithm can be applied in an iterative manner to find $\pi_i(c_1, c_2, \ldots, c_K)$, for $i \neq 1$, for $c_k = \{0, 1, 2, \ldots, \tau_k\}$, which is shown as follows:

Step i. $c_k = 0$, for $k = 1, 2, \ldots, K$; $|\pi_s(c_1, c_2, \ldots, c_K)| = \infty$ if there is a $c_k < 0$, for $s \neq 1$. Let $k = 0$.

Step ii. $|\pi_s(c_1, c_2, \ldots, c_K)|^{(k)} = |\pi_s(c_1-1, c_2-1, \ldots, c_K-1)|$, for $s \neq 1$.

Step iii. $|\pi_s(c_1, c_2, \ldots, c_K)|^{(k+1)} = \underset{t \neq s, h_{ts}^j \leq c_j, \text{for } j=1,2,\ldots,K}{\text{minimum}}$

$\{d_{ts} + |\pi_t(c_1-h^1_{ts}, c_2-h^2_{ts}, \ldots, c_K-h^K_{ts}|^{(k)}\}$, for $s \neq 1$.

Step iv. If $|\pi_s(c_1, c_2, \ldots, c_K)|^{(k+1)} = |\pi_s(c_1, c_2, \ldots, c_K)|^{(k)}$ for all $s \neq 1$, go to Step 1; otherwise $k = k+1$, return to Step iii.

Step 1. If $c_K < \tau_K$, $c_K = c_K+1$, return to Step ii; otherwise go to Step 2.

Step 2. If $c_{K-1} < \tau_{K-1}$, $c_{K-1} = c_{K-1}+1$, $c_K = 0$, return to Step ii; otherwise go to Step 3.

.
.
.

Step k. If $c_{K-k+1} < \tau_{K-k+1}$, $c_{K-k+1} = c_{K-k+1}+1$, $c_i = 0$, for $i = K, K+1, \ldots, K-k+2$, return to Step ii; otherwise go to Step k+1.

.
.
.

Step K. If $c_1 < \tau_1$, $c_1 = c_1+1$, $c_2 = c_3 = \cdots = c_K = 0$, return to Step ii; otherwise stop.

Essentially, the above steps form K do-loops with the resulting computational effort being the order of $\tau^K n^2$ (assuming $\tau = \tau_1 = \tau_2 = \ldots = \tau_K$), where n^2 is the computational effort required in carrying out Steps 3 and 4. An alternative solution is discussed in Chapter III.

CHAPTER III

THE COMBINATORIAL ALGORITHM

3.1 Introduction

The constrained shortest path algorithm presented in Chapter II is an iterative scheme in which the cost bound variable c is incremented by a unit at the end of each iteration where stepping costs, if they exist, along with the corresponding stepping-cost paths are found. The same amount of computation, of order of n^2, is needed for each iteration step without prior knowledge of whether there exist any stepping costs for the current cost bound.

In this chapter, a combinatorial approach is employed to locate the stepping costs and the corresponding stepping-cost paths so that the cost bound variable may be increased to the value of the next stepping cost. Hopefully, in doing this, we will have a more efficient algorithm to solve the problem of Section 1.3.

3.2 The Immediate Stepping Cost

Let c_i^j denote the jth $1 \to i$ stepping cost, and π_i^j denote the corresponding j^{th} $1 \to i$ stepping-cost path, so that $c_i^1 < c_i^2 < \ldots$, for $i = 1, 2, \ldots, n$. By the definition of stepping cost, it is obvious that $|\pi_i^1| > |\pi_i^2| > \ldots$ The situation is illustrated by the following network and its set of constrained shortest paths.

Define, for all i, K_i to be the number of $1 \to i$ stepping costs not exceeding the current cost bound c.

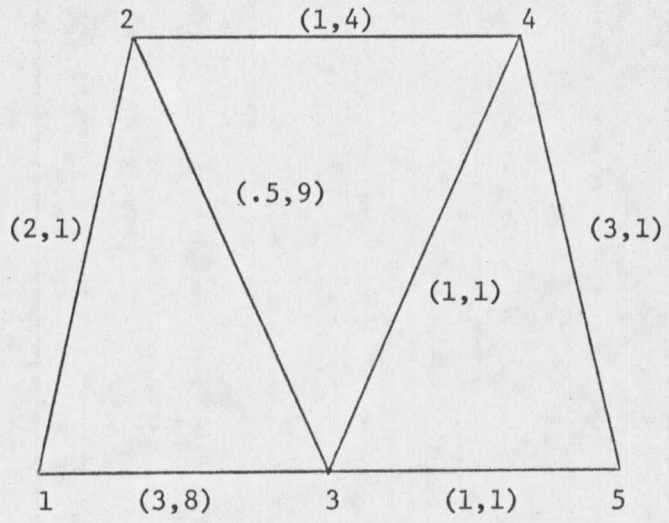


Fig. 3.1. A five-node undirected network with $W_{ij} = (d_{ij}, h_{ij})$.

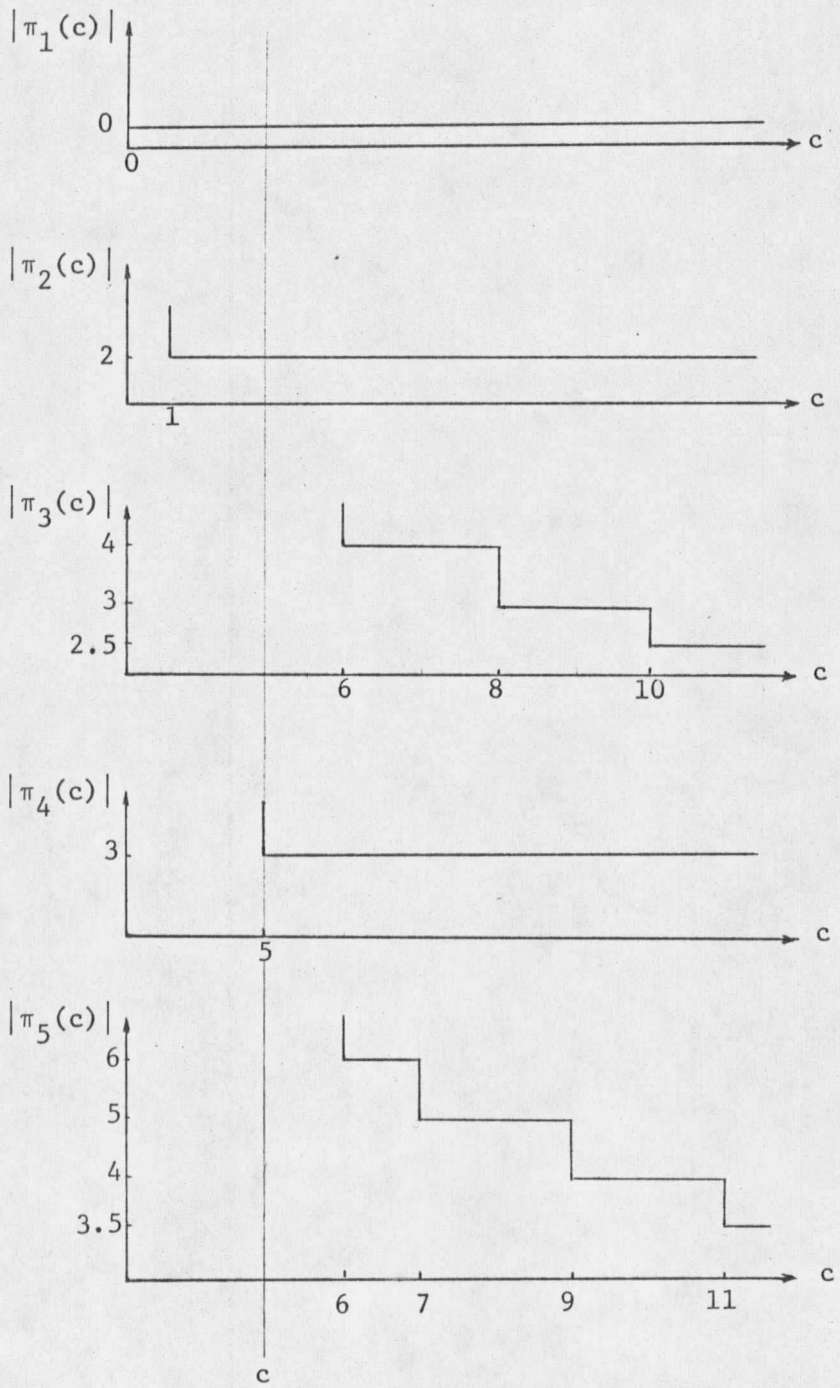


Fig. 3.2. The set of stepping costs and the stepping-cost path lengths of the network in Fig. 3.1. For c = 5, $K_1 = 1$, $K_2 = 1$, $K_3 = 0$, $K_4 = 1$, $K_5 = 0$.

For $\neq 1$, define the set of nodes S_i to be

$$S_i \triangleq \{k, \text{ such that } |\pi_k^{K(k)}| + d_{ki} < |\pi_i^{K_i}|, \text{ where } K(k) \leq K_k \text{ is the smallest integer satisfying the inequality}\} \quad (3\text{-}2\text{-}1)$$

and define the real constant c_i^* to be

$$c_i^* \triangleq \underset{k \in S_i}{\text{minimum}}\{c_k^{K(k)} + h_{ki}\}^{\dagger} \quad (3\text{-}2\text{-}2)$$

Assertion 3.1 If k is the last intermediate node on the $(K_i+1)^{th}$ $1 \to i$ stepping-cost path $\pi_i^{K_i+1}$, then the $1 \to k$ path in $\pi_i^{K_i+1}$ is a stepping-cost path.

Lemma 3.1 $c_i^{K_i+1} \leq c_i^*$

Proof. Let k be the node where c_i^* attains its minimum value, so that $c_k^{K(k)} + h_{ki} = c_i^*$. Now $\pi_k^{K(k)} + d_{ki}$ is a feasible $(K_i+1)^{th}$ $1 \to i$ stepping-cost path, so $c_i^{K_i+1} \leq c_k^{K(k)} + h_{ki} = c_i^*$.

DEFINITION 3.1. The immediate stepping cost (abbreviated as IMSC) is the stepping cost that is not less than the current cost bound c, having no other stepping cost between c and IMSC.

---

†In case S_i is null, c_i^* may be assigned an infinite value.

The goal at this point is to find IMSC, and have the cost bound variable c increased to this value, instead of having to use a step-by-step computation.

Assertion 3.2 $\text{IMSC} = \min_i c_i^{K_i+1}$

3.3 The Combinatorial Theorem

Theorem 3.1 $\text{IMSC} = c_r^*$ where r is the node such that

$$c_r^* = \text{minimum}_i\ c_i^* \tag{3-3-1}$$

By the definition of IMSC, we have $\text{IMSC} = \min_i c_i^{K_i+1}$. Moreover, from the result of Lemma 3.1, $\text{IMSC} = \min_i c_i^{K_i+1} \leq \min_i c_i^* = c_r^*$. It remains to be proved that $\text{IMSC} \geq c_r^*$.

Lemma 3.2 If the immediate stepping cost is associated with a node, say node i, and k is the last intermediate node on the $(K_i+1)^{\text{th}}$ $1 \to i$ stepping-cost path, then $k \in S_i$; moreover, $c_i^{K_i+1} \geq c_i^*$.

Proof. From Assertion 3.1, the $1 \to k$ path in $\pi_i^{K_i+1}$ is a stepping-cost path. Let c_k denote the cost of the $1 \to k$ path; then $c_k \leq c_k^{K_k}$.

For if not, $c_k = c_i^{K_i+1} - h_{ki} \leq c_i^{K_i+1} = \text{IMSC}$, then by the definition of IMSC, the $1 \to k$ path must be a stepping cost path with stepping cost not exceeding $c_k^{K_k}$ which contradicts the $c_k > c_k^{K_k}$ assumption. Now that $c_k \leq c_k^{K_k}$, and that the $1 \to k$ path is a stepping-cost path, by the assump-

tion of the lemma and the definition of S_i, $k \in S_i$. Let K(k) be the integer $\leq K_k$ such that $c_k = c_k^{K(k)}$.

Accordingly,

$$c_i^{K_i+1} = c_k + h_{ki} = c^{K(k)} + h_{ki} \geq c_i^* \qquad (3\text{-}3\text{-}2)$$

The rest of the proof of Theorem 3.1 becomes immediate after Lemma 3.2 is established. Assume the immediate stepping cost is associated with a node, say node i; then the definition of node r in the theorem statement asserts that

$$c_r^* = \min_i c_i^* \leq c_i^* \leq c_i^{K_i+1} = \text{IMSC} \qquad (3\text{-}3\text{-}3)$$

3.4 The Combinatorial Algorithm

The results of the arguments presented so far can be summarized as follows:

1) No stepping cost exists between the open real interval (c,IMSC)--by the definition of IMSC. So that we can have c advanced to the value IMSC^*, a value less than IMSC.

2) An immediate stepping cost has been found with value IMSC = $\text{minimum}_i\{\text{minimum}_{k \in S_i}\{c_k^{K(k)} + h_{ki}\}\}$.

3) Since c_i^* is the candidate value for the $(K_i+1)^{th}$ stepping cost, as far as the algorithm is concerned, we may let $c_i^{K_i+1} = c_i^* = \underset{k \in S_i}{\text{minimum}} \{c_k^{K(k)} + h_{ki}\}$.

Rewrite (3-2-1), (3-2-2) for further reference

$$S_i \triangleq \{k, \text{ such that } |\pi_k^{K(k)}| + d_{ki} < |\pi_i^{K_i}|, \text{ where } K(k) \leq K_k \text{ is the smallest integer satisfying the inequality}\} \quad (3\text{-}4\text{-}1)$$

$$c_i^{K_i+1} \triangleq \underset{k\ S_i}{\text{minimum}} \{c_k^{K(k)} + h_{ki}\} \quad (3\text{-}4\text{-}2)$$

and define the minimizing last intermediate node i^* of S_i such that

$$|\pi_{i^*}^{K(i^*)}| + d_{i^*i} = \underset{k \in S_i, c_k^{K(k)} + h_{ki} = c_i^{K_i+1}}{\text{minimum}} \{|\pi_k^{K(k)}| + d_{ki}\} \quad (3\text{-}4\text{-}3)$$

Then

$$\text{IMSC} = \underset{i}{\text{minimum}}\ c_i^{K_i+1} \quad (3\text{-}4\text{-}4)$$

and IMSC is associated with the node, say node r, if r is such that

$$c_r^{K_r+1} = \underset{i}{\text{minimum}}\ c_i^{K_i+1} \quad (3\text{-}4\text{-}5)$$

and

$$|\pi_{r^*}^{K(r^*)}| + d_{r^*r} = \underset{i, \text{ such that } c_i^{K_i+1} = c_r^{K_r+1}}{\text{minimum}} \{|\pi_{i^*}^{K(i^*)}| + d_{i^*i}\} \quad (3\text{-}4\text{-}6)$$

The algorithm is a straightforward implementation of equations (3-4-1) through (3-4-6).

Let $\{\pi_i\}$ denote the set of unconstrained shortest paths of the network, using any appropriate methods mentioned in Chapter I.

Step 1. Initialization.

Let $I = \{2, 3, \ldots, n\}$, $K_1 = 1$, $|\pi_1^1| = 0$, $c_1^1 = 0$.

For all $i \in I$, let $K_i = 0$, $|\pi_i^0| = \infty$, $S_i = \{1\}$, $c_i^{K_i+1} = h_{1i}$, $i^* = 1$, $K(i^*) = 1$.

Step 2. Locate the node, r, with which the IMSC is associated. r is such that

$$c_r^{K_r+1} = \underset{i \in I}{\text{minimum}}\ c_i^{K_i+1}$$

and

$$|\pi_r^{K_r+1}| = |\pi_{r^*}^{K(r^*)}| + d_{r^*r} = \underset{i, c_i^{K_i+1} = c_r^{K_r+1}}{\text{minimum}} \{|\pi_{i^*}^{K(i^*)}| + d_{i^*i}\}$$

Step 3. If $|\pi_r^{K_r+1}| \leq |\pi_r|$, $I = I - r$.

Step 4. If $I = \phi \triangleq$ null set, terminate.

Step 5. For all nodes $i, i \neq r$, and $r \notin S_i$:

i) If $|\pi_r^{K_r+1}| + d_{ri} \geq |\pi_i^{K_i}|$ go to Step 6; otherwise, go to ii.

ii) $r \in S_i$; and further if

iii) $c_r^{K_r+1} + h_{ri} < c_i^{K_i+1}$, then $c_i^{K_i+1} = c_r^{K_r+1} + h_{ri}$, $i^* = r$,

and $K(i^*) = K_r+1$; or if

iv) $c_r^{K_r+1} + h_{ri} = c_i^{K_i+1}$, and $|\pi_r^{K_r+1}| + d_{ri} < |\pi_{i*}^{K(i^*)}| + d_{i^*i}$,

then $i^* = r$, and $K(i^*) = K_r+1$.

Step 6. i) $K_r = K_r+1$; for all $k \in S_r$, if there exists a smallest integer $Kl \leq K_k$ such that $|\pi_k^{Kl}| + d_{kr} < |\pi_r^{K_r}|$, then k remains in S_r, $K(k) = Kl$; otherwise $S_r = S_r - k$.

ii) $c_r^{K_r+1} = \underset{k \in S_r}{\text{minimum}}\{c_k^{K(k)} + h_{kr}\}$

iii) $r^* \in S_r$ is such that for all $k \in S_r$ where $c_k^{K(k)} + h_{kr} = c_r^{K_r+1}$, $|\pi_{r*}^{K(r^*)}| + d_{r^*r} \leq |\pi_k^{K(k)}| + d_{kr}$.

Step 7. Go to Step 2.

Steps 2 through 6 form one sequence of steps to find an immediate stepping cost. When the last stepping cost has been found, the node is exempted from further consideration in Step 3. The algorithm terminates when all stepping costs have been found -- when the set I becomes null, and essentially no nodes are left to contribute to form another stepping cost. Step 2 locates the immediate stepping cost and evaluates the stepping-cost path length. Step 5 updates the sets S_i, $i \neq r$, and the nominated last intermediate node i^* for the $(K_i+1)^{th}$ $1 \to i$ stepping-cost path. Step 6 updates the set S_r, the nominated last intermediate node r^*, and increments K_r.

The computational effort is analyzed as follows: Step 2 requires 2n comparisons to locate node r. For each node $i \neq r$, four comparisons are needed in Step 5. The computational effort in Step 6 depends on the size of K_k where $k \in S_r$. Assume there are at most m stepping costs for each node k in S_r, then stage 1 of Step 6 requires at most nm+n comparisons, stage 2 requires at most n comparisons, and stage 3 requires at most 2n comparisons. So the total number of comparisons needed in finding one stepping cost is bounded from above by $(9+m)n \simeq nm$. Since there are at most nm stepping costs, the algorithm will terminate with the maximum number of comparisons of the order of $(nm)^2$.

3.5 A Comparison

The constrained shortest path algorithm of Chapter II for networks with integer-valued branch costs requires of the order of $2\tau n^2$ comparisons where τ is the maximum value of the stepping costs. Since a stepping-cost path may contain as many as n-1 branches, if h_{max} is the maximum value of the branch costs, then τ is bounded above by nh_{max}. The combinatorial algorithm requires computational effort on the order of m^2n^2 where m is the average number of stepping costs for each node. It is obvious that both τ and m depend on the network topology and on the distribution of branch weights. Perhaps only through experience or intuitive judgment can one prefer one method over the other.

However, the combinatorial algorithm has a definite advantage over the other in that no integer-type limitation is imposed, whereas only suboptimal paths are attainable by the constrained shortest path algorithm for networks with real branch costs.

The computational effort of the combinatorial algorithm is recorded for complete networks (networks with a branch between every distinct node pair) of different sizes whose branch weights are generated by a uniform random number generating routine with a maximum value of the branch weights being 5. The average number of comparison operations is taken over 5 randomly generated networks of the same size. Table 3.1 contains the computational effort of networks of starting from 14 nodes, up to 21 nodes. The average number of comparison operations over the theoretical upper bound is plotted in Fig. 3.3.

The ratio is greater than one when the network size is less than 15. This is because the number of comparison operations associated with indexing (which is included as part of the computer code but is not part of the theoretical upper bound) dominates when the number of stepping costs is small. The ratio is a decreasing function of the network size which indicates that the theoretical upper bound is generally conservative as the network size increases. Since the networks are generated in a random fashion, whereas the number of stepping costs depends on the network topology and the branch weights, no attempt has been made to find the asymptotic value of the ratio.

Table 3.1. Computational Effort Using the Combinatorial Algorithm. I1 = number of comparisons required, I2 = number of stepping costs, $\bar{I1}$ = average of I1, $\bar{I2}$ = average of I2.

Run No.	14-node		15-node		16-node		17-node		18-node		20-node		21-node	
	I1	I2	I1	I2	I1	I2	I1	I2	I1	I2	I1	I2	I1	I2
1	2481	40	3971	60	4563	62	6502	83	8153	101	12977	147	11337	119
2	2941	45	5484	81	5570	76	5170	68	9226	113	7764	88	13998	137
3	3999	65	4568	65	4912	66	6071	79	6790	85	9421	107	15160	156
4	3210	50	2992	44	4637	66	6314	83	6568	79	8388	91	10395	107
5	3930	64	3812	57	4660	65	5228	70	5631	71	6774	76	12124	129
$(\bar{I1},\bar{I2})$	3312	53	4166	61	4868	67	5857	77	7274	90	9065	102	12603	130
$\frac{\bar{I1}}{\bar{I2}(\bar{I2}+9)}$	1.015		.964		.956		.890		.819		.804		.702	

3.6 Networks With Doubly-Labelled Branch Cost Weights

In this section shortest paths of networks where each branch (i,j) has two associated cost weights, h_{ij}^1, h_{ij}^2, are treated. Let $W_{ij} = (d_{ij}, h_{ij}^1, h_{ij}^2)$ denote the joint weight of (i,j). Similar to the definition of stepping costs of networks with a single branch cost weight, a stepping cost for the present network is defined.

DEFINITION 3.2. (c_1, c_2) is a $1 \to i$ stepping cost of networks with doubly-labelled cost weights, if and only if c_1 and c_2 are, respectively, the type 1 and type 2 cost of a $1 \to i$ path, $\pi_i(c_1, c_2)$, and for any

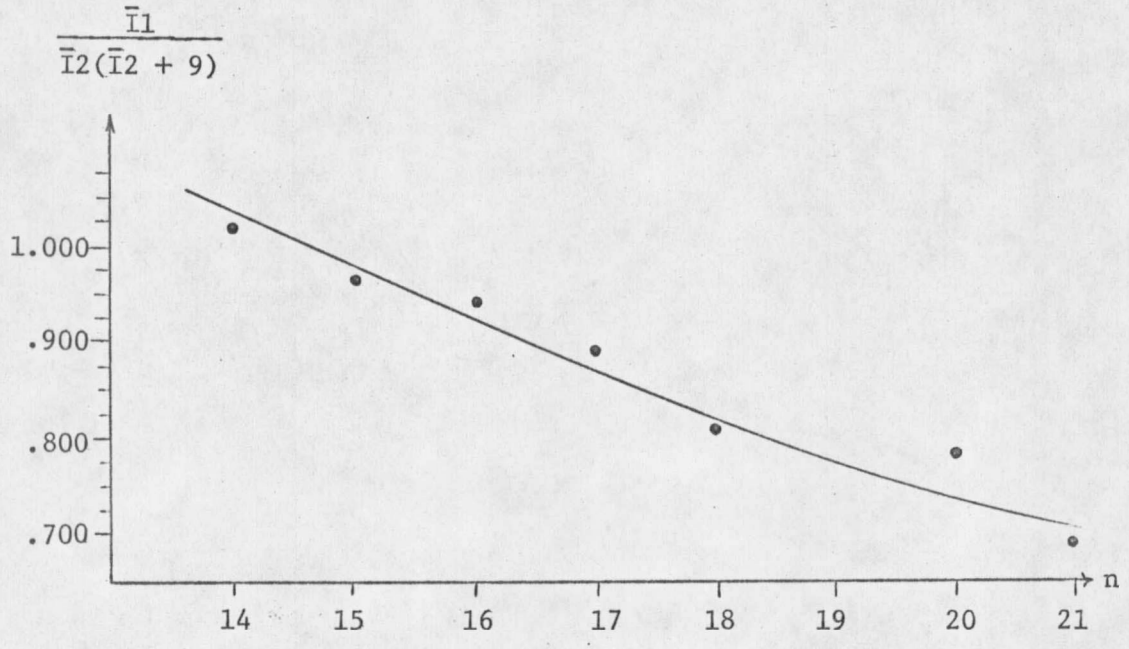


Fig. 3.3. The computational effort vs. the number of nodes in the network using the combinatorial algorithm.

$c_1' < c_1$, and $c_2' < c_2$, if follows that

$$|\pi_i(c_1,c_2)| < |\pi_i(c_1,c_2')| \tag{3-6-1}$$

and that

$$|\pi_i(c_1,c_2)| < |\pi_i(c_1',c_2)| \tag{3-6-2}$$

The associated 1→i path, $\pi_i(c_1,c_2)$ is called a 1→i stepping-cost path. Besides, c_2 and (c_2,c_1) are said to be associated with c_1, as far as the stepping cost is concerned, and vice versa.

The combinatorial approach is to locate each stepping cost in a sequential manner such that the type 2 1→i costs associated with c_1, are to be located only when all c_1'-associated type 2 1→i costs have been located, for $c_1' < c_1$.

Let K_{1i} be the number of type 1 1→i costs $\tilde{c}_i^1, \tilde{c}_i^2, \ldots, \tilde{c}_i^{K_{1i}}$ where a nonnull number of associated type 2 1→i costs have been located. Let K_{2i}' be the number of located type 2 costs associated with a type 1 cost $\tilde{c}_i^{K_1(i)}$, $K_1(i) \leq K_{1i}$. And let $c_2^{K_2(i)}$ denote the $K_2(i)^{th}$ type 2 cost associated with $K_1(i)$, with $K_2(i) \leq K_{2i}'$. In particular, let K_{2i} be the number of type 2 1→i costs $c_i^1, c_i^2, \ldots, c_i^{K_{2i}}$ associated with $\tilde{c}_i^{K_{1i}}$. Note that each of the terms $K_2(i)$, K_{2i}', and K_{2i} has no meaning by itself; a certain type 1 cost has to be specified to make it meaningful. And let $\pi_i^{j,k}$ denote the kth 1→i

stepping-cost path associated with $\tilde{c}_i^j$. We assume that no other $1 \to i$ stepping cost exists with its associated type 1 cost less than $\tilde{c}_i^{K_{1i}}$; and no other $1 \to i$ stepping cost exists with its type 1 cost equal to $\tilde{c}_i^{K_{1i}}$, and its type 2 cost less than $c_i^{K_{2i}}$.

The approach then is to locate the immediate stepping cost which is defined as follows.

DEFINITION 3.3. The immediate stepping cost (c_{IM}^1, c_{IM}^2) is the stepping cost which is not in the set of located stepping costs, and for which there is no other stepping cost, say (c_1, c_2), that satisfies either

$$c_1 < c_{IM}^1$$

or

$$c_1 = c_{IM}^1 \text{ and } c_2 < c_{IM}^2$$

Now define, for all $i \neq 1$, $|\pi_i^*(c)|$ to be a function of c such that

$$|\pi_i^*(c)| \triangleq \underset{K_1(i) \leq K_{1i}}{\text{minimum}} \; |\pi_i^{K_1(i), K_2(i)}| \qquad (3\text{-}6\text{-}3)$$

where $c_i^{K_2(i)}$, $K_2(i) \leq K'_{2i}$, is the type 2 $1 \to i$ cost associated with $K_1(i)$ such that

$$c \in [c_i^{K_2(i)}, c_i^{K_2(i)+1}]$$

Define, for all $i \neq 1$, the set of nodes S_i,

$S_i \triangleq \{k$ if there exists a smallest integer $K_1(k) \leq K_{1k}$ and its associated smallest integer $K_2(k) \leq K'_{2k}$ such that

$$|\pi_k^{K_1(k),K_2(k)}| + d_{ki} < |\pi_i^*(c_k^{K_2(k)} + h_{ki}^2)|\} \qquad (3\text{-}6\text{-}4)$$

Essentially, if $k \in S_i$, then k is a feasible last intermediate node such that $\pi_k^{K_1(k),K_2(k)} \cup (k,i)$ is a possible next $1 \rightarrow i$ stepping cost, with type 1 path cost $\tilde{c}_k^{K_1(k)} + h_{ki}^1$ and type 2 path cost $c_k^{K_2(k)} + h_{ki}^2$, which satisifes (3-6-1) and (3-6-2). $K_1(k)$ is chosen to be the smallest possible so that the next $1 \rightarrow i$ stepping cost is of smallest type 1 cost, and $K_2(k)$ is the associated smallest possible integer, such that the type 2 cost of the next $1 \rightarrow i$ stepping cost is as small as possible.

Define for all $i \neq 1$, the values c_{1i}^*, c_{2i}^* to be

$$c_{1i}^* \triangleq \underset{k \in S_i}{\text{minimum}}\{\tilde{c}_k^{K_1(k)} + h_{ki}^1\} \qquad (3\text{-}6\text{-}5)$$

$$c_{2i}^* \triangleq \underset{k \in S_i}{\text{minimum}}\{c_k^{K_2(k)} + h_{ki}^2\} \qquad (3\text{-}6\text{-}6)$$

where the above minimization is restricted by,

$$\tilde{c}_k^{K_1(k)} + h_{ki}^1 = c_{1i}^*$$

and define $i^* \in S_i$ to be the node such that

$$|\pi_{i^*}^{K_1(i^*),K_2(i^*)}| + d_{i^*i} = \underset{k\in S_i}{\text{minimum}}\{|\pi_k^{K_1(k),K_2(k)}| + d_{ki}\} \qquad (3\text{-}6\text{-}7)$$

where the minimization is restricted by,

$$\tilde{c}_k^{K_1(k)} + h_{ki}^1 = c_{1i}^*$$

and

$$c_k^{K_2(k)} + h_{ki}^2 = c_{2i}^*$$

Hence, for all possible last intermediate nodes for the next 1→i stepping-cost path, c_{1i}^* is the smallest possible type 1 path cost; c_{2i}^*, associated with c_{1i}^*, is the smallest possible type 2 path cost; and i^* is the minimizing last intermediate node to result in a next 1→i stepping cost (c_{1i}^*, c_{2i}^*).

Assertion 3.3 If $c_{1i}^* = \tilde{c}_i^{K_{1i}}$, then there exists a next 1→i stepping cost with type 1 cost $\tilde{c}_i^{K_{1i}}$, and with type 2 cost $c_i^{K_{2i}+1} \leq c_{2i}^*$. If $c_{1i}^* > \tilde{c}_i^{K_{1i}}$ then there exists a next 1→i stepping cost with type 1 cost $c_i^{K_{1i}+1} \leq c_{1i}^*$.

Define the node r, along with its associated c_{1r}^* and c_{2r}^*, such that

$$c_{1r}^* = \underset{i}{\text{minimum}}\{c_{1i}^*\} \qquad (3\text{-}6\text{-}8)$$

$$c_{2r}^{*} = \underset{i, c_{1i}^{*}=c_{1r}^{*}}{\text{minimum}} \{c_{2i}^{*}\} \qquad (3\text{-}6\text{-}9)$$

and that

$$|\pi_{r^*}^{K_1(r^*),K_2(r^*)}| + d_{r^*r} = \underset{i}{\text{minimum}}\{|\pi_{i^*}^{K_1(i^*),K_2(i^*)}| + d_{i^*i}\} \qquad (3\text{-}6\text{-}10)$$

where the above minimization is restricted by,

$$c_{1i}^{*} = c_{1r}^{*}$$

and

$$c_{2i}^{*} = c_{2r}^{*}$$

By the above definition, r is the node with a possible next stepping cost of smallest type 1 path cost. And of all other nodes with a possible next stepping cost of the same type 1 path cost, the type 2 path cost associated with node r is the smallest one. Finally, of all other nodes with the same possible stepping cost, the length of the stepping-cost path associated with node r is the smallest one.

Shown in Fig. 3.4a is a 5-node network with its 1→5 paths listed in Table 3.2, along with the associated path lengths and both type 1 and type 2 path costs. The 1→5 stepping costs are shown in Fig. 3.4b.

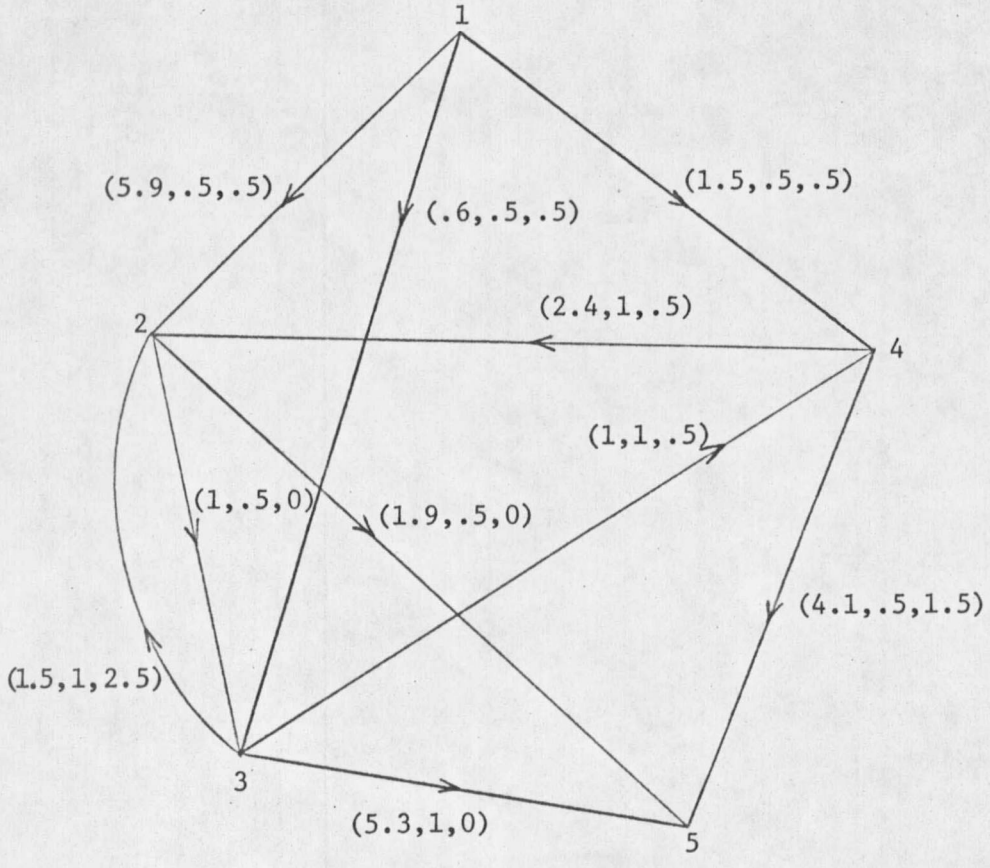


Fig. 3.4a. A 5-node network with $W_{ji} = (d_{ji}, h^1_{ji}, h^2_{ji})$.

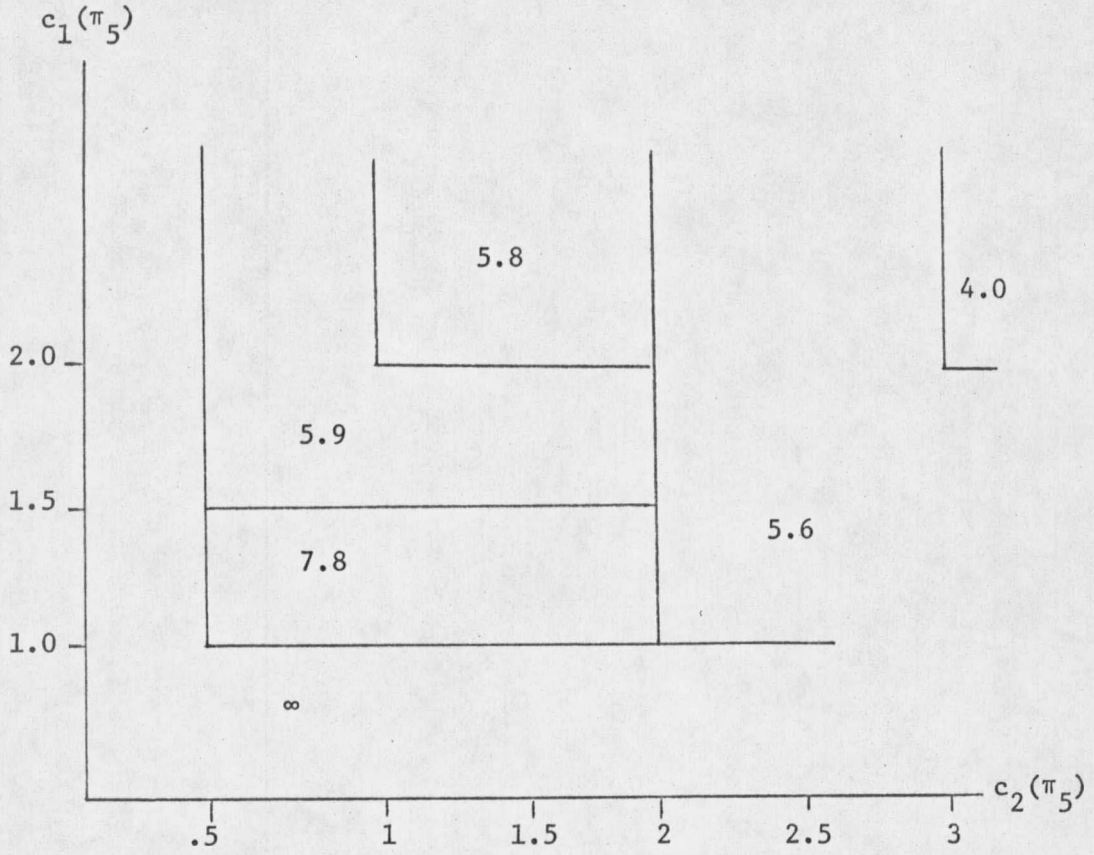


Fig. 3.4b. The 1→5 stepping costs.

Table 3.2. The Set of 1→5 Paths of the Network in Fig. 3.4a.

π_5	$\lvert\pi_5\rvert$	$c_1(\pi_5)$	$c_2(\pi_5)$
1-2-5	7.8	1.0	.5
1-3-5	5.9	1.5	.5
1-4-5	5.6	1.0	2.0
1-2-3-5	12.2	2.0	.5
1-3-2-5	4.0	2.0	3.0
1-3-4-5	5.7	2.0	2.5
1-4-2-5	5.8	2.0	1.0
1-2-3-4-5	12.0	2.5	2.5
1-3-4-2-5	5.9	3.0	1.5
1-4-2-3-5	10.2	3.0	1.0

For example, the IMSC, say (2, 1), is not associated with node 5 until the 1→5 stepping costs (1.5, .5), (1, .5), and (1, 2) are located.

Theorem 3.2 Let c^1_{IM} denote the type 1 cost of the IMSC; then

$$c^1_{IM} = c^*_{1r} \tag{3-6-11}$$

where r is the node that satisfies (3-6-8) to (3-6-10).

It is obvious that, for all i, $c^1_{IM} \leq c^*_{1i}$ because, for each nonnull S_i, there exists a possible new 1→i stepping-cost path with type 1 path cost not smaller than c^*_{1i}.

It remains to be shown that $c^1_{IM} \geq c^*_{1r}$ to complete the proof.

Lemma 3.3 If IMSC is associated with a node, say node i, and k is the last intermediate node on the new 1→i stepping-cost path, then $k \in S_i$; moreover, $c^1_{IM} \geq c^*_{1i}$.

The proof is essentially the same as that of Lemma 3.2.

Proof of Lemma 3.3.

It is obvious that the 1→k path in the IMSC path is itself a stepping-cost path. Let A_k denote the type 1 path cost of this path; then $A_k \leq \tilde{c}^{K_{1k}}_k$.

We have $A_k = c^1_{IM} - h^1_{ki} \leq c^1_{IM}$. Then by the definition of IMSC, and the definition of K_{1k} as the number of type 1 costs each of which

has a nonzero number of associated stepping costs, the 1→k path must be a stepping-cost path with type 1 cost not exceeding $\tilde{c}_k^{K_{1k}}$; hence $A_k \leq \tilde{c}_k^{K_{1k}}$. Thus, by the assumption of the Lemma and the definition of S_i, $k \in S_i$. Let $K_1(k)$ be the integer $\leq K_{1k}$ such that $A_k = \tilde{c}_k^{K_1(k)}$. Accordingly, by (3-6-5),

$$c_{IM}^1 = A_k + h_{ki}^1 = c_k^{K_1(k)} + h_{ki}^1 \geq c_{1i}^* \tag{3-6-12}$$

Proof of Theorem 3.2.

Assume IMSC is associated with node i; then from (3-6-8) and (3-6-12),

$$c_{1r}^* = \underset{i}{\text{minimum}}\ c_{1i}^* \leq c_{1i}^* \leq c_{IM}^1 \tag{3-6-13}$$

Once c_{IM}^1 is found, we seek to find c_{IM}^2, the associated type 2 cost of the immediate stepping cost.

Theorem 3.3 $\quad c_{IM}^2 = c_{2r}^*$

Proof.

It is obvious that, if i is such that $c_{1i}^* = c_{1r}^* = c_{IM}^1$, $c_{IM}^2 \leq c_{2i}^*$. The problem that remains is to show that $c_{IM}^2 \geq c_{2i}^*$.

Lemma 3.4 If IMSC is associated with node i, and k is the last intermediate node on the new 1→i stepping-cost path, then $c^2_{IM} \geq c^*_{2i}$.

Proof of Lemma 3.4.

From Lemma 3.3, $K \in S_i$. Let B_k denote the type 2 path cost of the 1→k path; then $B_k \leq c_k^{K'_{2k}}$, where K'_{2k} is the number of 1→k stepping costs associated with the type 1 cost $c_k^{K_1(k)} = c^1_{IM} - h^1_{ki}$. The fact that $B_k = c^2_{IM} - h^2_{ji} \leq c^2_{IM}$ shows, by the same argument as that of Lemma 3.3, that $B_k \leq c_k^{K'_{2k}}$. Hence, there is an integer $K_2(k) \leq K'_{2k}$ such that $B_k = c_k^{K_2(k)}$, and

$$c^2_{IM} = B_k + h^2_{ki} = c_k^{K_2(k)} + h^2_{ki} \geq c^*_{2i}$$

with the final result, $c^2_{IM} \geq c^*_{2i}$.

The results of Theorem 3.2 and Theorem 3.3 are summarized as follows:

1) The immediate stepping cost IMSC is characterized by IMSC = (c^*_{1r}, c^*_{2r}), where c^*_{1r}, c^*_{2r} are defined by equations (3-6-8) and (3-6-9).

2) The node r with which the IMSC associates can be located by (3-6-8) to (3-6-10).

3) The 1→r IMSC path and the associated length is defined in (3-6-10).

Once the IMSC is located, the set S_i, and the values c^*_{1i}, c^*_{2i}, i^*, are to be updated by using (3-6-3) through (3-6-7), and further IMSC are sought until none exists.

3.7 The Constrained Shortest Path Algorithm for Networks with Two Branch Cost Weights

The procedure to solve the constrained shortest path problem of networks with two branch cost weights is illustrated in the following:

Step 1. Initialization

$I = \{2, 3, \ldots, n\}$, $K_{11} = K_{21} = 1$, $|\pi_1^{1,1}| = 0$, $\check{c}_1^1 = 0$, $c_2^1 = 0$.

For $i \in I$, $K_{1i} = 1$, $K_{2i} = 0$, $|\pi_i^{1,1}| = \infty$, $\check{c}_i^1 = \infty$, $c_i^1 = \infty$,

$S_i = \phi$, $K_1(i) = K_2(i) = 1$, $i^* = 1$, $c^*_{1i} = c^*_{2i} = \infty$,

$|\pi_i^*(c)| = \infty$, for $c \geq 0$.

For i such that $h^1_{1i} < \infty$, $\check{c}_i^1 = h^1_{1i}$, $c_2^1 = h^2_{1i}$, $|\pi_i^{1,1}| = d_{1i}$,

$S_i = \{1\}$, $c^*_{1i} = h^1_{1i}$, $c^*_{2i} = h^2_{1i}$; $|\pi_i^*(c)| = d_{1i}$, for $c \geq c^*_{2i}$.

Step 2. Locate node r, such that r is the node that satisfies

$$c^*_{1r} = \underset{i \in I}{\text{minimum}}\ c^*_{1i} \tag{3-7-1}$$

$$c^*_{2r} = \underset{i \in I,\, c^*_{1i} = c^*_{1r}}{\text{minimum}}\ \{c^*_{2i}\} \tag{3-7-2}$$

$$|\pi_{r^*}^{K_1(r^*),K_2(r^*)}| + d_{r^*r} = \underset{i\in I}{\text{minimum}}\{|\pi_{i^*}^{K_1(i^*),K_2(i^*)}| + d_{i^*i}\} \tag{3-7-3}$$

where the above minimization is restricted by,

$$c_{1i}^* = c_{1r}^*$$

and

$$c_{2i}^* = c_{2r}^*$$

Step 3. If $c_{1r}^* \leq \tilde{c}_r^{K_{1r}}$, go to Step 4; otherwise $K_{1r} = K_{1r}+1$, $K_{2r} = 0$, and $\tilde{c}_r^{K_{1r}} = c_{1r}^*$.

Step 4. $$|\pi_r^{K_{1r},K_{2r}+1}| = |\pi_{r^*}^{K_1(r^*),K_2(r^*)}| + d_{r^*r} \tag{3-7-4}$$

If $$|\pi_r^{K_{1r},K_{2r}+1}| < |\pi_r|,^{\dagger} \; I = I - r. \tag{3-7-5}$$

Step 5. If $I = \phi$, terminate.

Step 6. For all nodes i such that $r \notin S_i$,

i) If $|\pi_r^{K_{1r},K_{2r}+1}| + d_{ri} < |\pi_i^*(c_{2r}^* + h_{ri}^2)|$, go to Step 7; otherwise continue with ii.

ii) Let $A_i \triangleq c_{1r}^* + h_{ri}^1$, $B_i \triangleq c_{2r}^* + h_{ri}^2$. If $A_i < c_{1r}^*$, then $c_{1i}^* = A_i$, $c_{2i}^* = B_i$, $i^* = r$, $K_1(i^*) = K_{1r}$, and $K_2(i^*) = K_{2r}+1$.

$^{\dagger}\pi_r$ is the precomputed unconstrained shortest 1→r path.

iii) Or, if $A_i = c^*_{1i}$, and $B_i < c^*_{2i}$, then $c^*_{2i} = B_i$, $i^* = r$, $K_1(i^*) = K_{1r}$, and $K_2(i^*) = K_{2r}+1$.

iv) Or, if $A_i = c^*_{1i}$, $B_i = c^*_{2i}$, and

$$|\pi_r^{K_{1r},K_{2r}+1}| + d_{ri} < |\pi_{i^*}^{K_1(i^*),K_2(i^*)}| + d_{i^*i},$$ then

$i^* = r$, $K_1(i^*) = K_{1r}$, and $K_2(i^*) = K_{2r}+1$.

Step 7. $K_{2r} = K_{2r}+1$; $c_r^{K_{2r}} = c^*_{2r}$.

i) $$|\pi^*_r(c)| = \text{minimum}\{|\pi^*_r(c)|, |\pi_r^{K_{1r},K_{2r}}|\}, \text{ for } c \geq c^*_{2r} \qquad (3\text{-}7\text{-}6)$$

ii) For all nodes $k \in S_r$, if there exists a smallest integer $K_1(k) \leq K_{1k}$, and an associated smallest integer $K_2(k) \leq K'_{2k}$, where K'_{2k} is the number of $1 \to k$ type 2 costs associated with $K_1(k)$, such that

$$|\pi_k^{K_1(k),K_2(k)}| + d_{kr} < |\pi^*_r(c_{2k}^{K_2(k)} + h^2_{kr})| \qquad (3\text{-}7\text{-}7)$$

k remains in S_r; otherwise $S_r = S_r - k$.

iii) $$c^*_{1r} = \underset{k\in S_i}{\text{minimum}}\{\tilde{c}_k^{K_1(k)} + h^1_{kr}\} \qquad (3\text{-}7\text{-}8)$$

$$c^*_{2r} = \underset{k\in S_i}{\text{minimum}}\{c_k^{K_2(k)} + h^2_{kr}\} \qquad (3\text{-}7\text{-}9)$$

where the above minimization is restricted by

$$\tilde{c}_k^{K_1(k)} + h^1_{kr} = c^*_{1r}$$

$r^* \in S_r$ is such that

$$|\pi_{r^*}^{K_1(r^*),K_2(r^*)}| + d_{r^*r} = \underset{k\in S_r}{\text{minimum}}\{|\pi_k^{K_1(k),K_2(k)}| + d_{kr}\} \tag{3-7-10}$$

where the above minimization is restricted by,

$$\tilde{c}_{1k}^{K_1(k)} + h_{kr}^1 = c_{1r}^*$$

and

$$c_{2k}^{K_2(k)} + h_{kr}^2 = c_{2r}^*$$

Step 8. Return to Step 2.

The computational effort for the above procedure is evaluated as follows.

Assume there are at most m type 1 stepping costs associated with each node, and m type 2 costs associated with each type 1 stepping cost; hence, for each node, the number of stepping costs is bounded from above by m^2.

Equations (3-7-1) through (3-7-3) require at most 3n comparisons. The computational effort in Steps 3 through 5 is trivial. In procedure i of Step 6, the function π_i^* is itself a stepping function, and hence, requires at most m comparisons. Procedures ii through iv of Step 6 require at most 3 comparisons. Thus Step 6 needs at most n(m+3) comparisons. For (3-7-6) in Step 7, at most m^2 comparisons are

required. In (3-7-7), m comparisons are needed to locate $\pi_r^*(c_{2k}^{K_2(k)} + h_{kr}^2)$, for a given $K_2(k)$; and since at most m^2 stepping costs are associated with node k, m^3 comparisons are needed for each k in S_r. Hence the number of comparisons is bounded from above by nm^3. Equations (3-7-8) through (3-7-10) again require at most 3n comparisons.

The total number of comparisons to locate one stepping cost is found to be $nm^3 + 3n + m^2 + n(m+3) + 3n = n(m^3 + 9 + m) + m^2 \approx nm^3$. Since there exists at most nm^2 stepping costs, the algorithm terminates within the order of n^2m^5 comparisons, which is of the order of the square of the total number of stepping costs.

3.8 An Example

The combinatorial approach for finding shortest paths in a doubly-labelled cost weighted network is illustrated here by finding the set of stepping costs and the associated stepping-cost path for the network in Fig. 3.5.

To start the procedure, we know there is the trivial 1→1 path with one and only one stepping cost, i.e., (0, 0). So we have $K_{11} = K_{21} = 1$, $|\pi_1^{1,1}| = 0$, $\tilde{c}_1^1 = 0$, $c_1^1 = 0$. And for all $i \neq 1$, no stepping cost is known, $K_{1i} = 1$, $K_{2i} = 0$, $|\pi_i^{K_{1i},K_{2i}}| = \infty$, c_1^1 can be assigned an infinite value, $|\pi_i^*(c)| = \infty$ for $c \geq 0$.

According to (3-6-3) and (3-6-4), only node 1 satisfies (3-6-4). Hence $S_i = \{1\}$ if (1, i) exists, with $K_1(1) = K_2(1) = 1$. Thus,

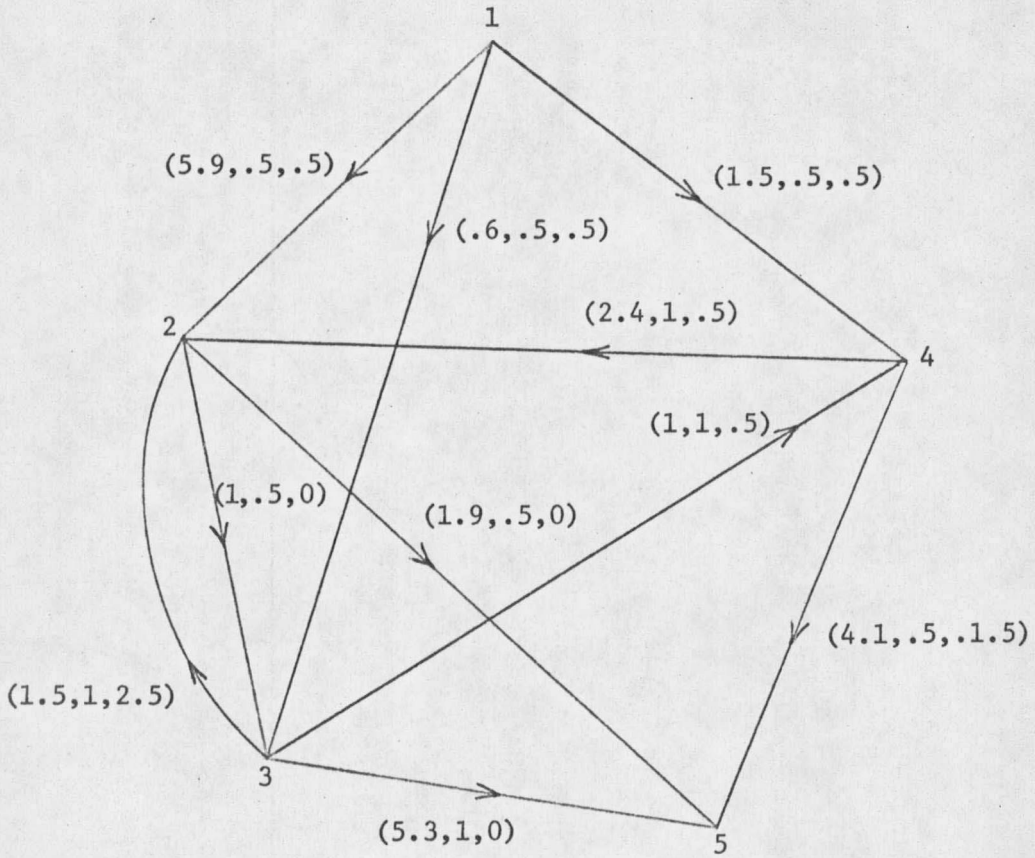


Fig. 3.5. A 5-node network with $W_{ji} = (d_{ji}, h^1_{ji}, h^2_{ji})$.

for $i = 2, 3, 4$, $S_i = \{1\}$, $\tilde{c}_i^1 = c_{1i}^* = h_{1i}^1$, $c_i^1 = c_{2i}^* = h_{1i}^2$; $S_5 = \phi$.

The first IMSC is found by evaluating (3-6-8) to (3-6-10). Node 3 is found to be the minimizing node, since in (3-6-10)

$$\text{minimum}\{d_{12}, d_{13}, d_{14}\} = \text{minimum}\{5.9, .6, 1.5\} = .6 = d_{13}$$

while $c_{23}^* = c_{22}^* = c_{24}^* = .5$, and $c_{13}^* = c_{12}^* = c_{14}^* = .5$. Hence, IMSC = (.5, .5) with the associated path length = .6. The curve $|\pi_3^*(c)|$ is thus modified as shown in Fig. 3.6a.

Note that S_3 becomes null since node 1 is no longer in S_3. c_{13}^* and c_{23}^* are assigned infinite values. Node 3 becomes a member in the set of S_i's, so $S_2 = \{1,3\}$, $S_4 = \{1,3\}$, $S_5 = \{3\}$, with $K_1(3) = K_2(3) = 1$. Hence, from (3-6-5) and (3-6-6),

$$c_{12}^* = \min\{h_{12}^1, \tilde{c}_3^1 + h_{32}^1\} = \min\{.5, .5 + 1\} = .5$$

$$c_{14}^* = \min\{h_{14}^1, \tilde{c}_3^1 + h_{34}^1\} = \min\{.5, .5 + 1\} = .5$$

$$c_{15}^* = \tilde{c}_3^1 + h_{35}^1 = 1.5$$

$$c_{22}^* = h_{12}^2 = .5$$

$$c_{24}^* = h_{14}^2 = .5$$

Further comparisons by (3-6-10) shows that the second IMSC is associated with node 4, with IMSC = (.5, .5). $|\pi_4^*(c)|$ is shown in Fig. 3.6b.

The second IMSC does not contribute node 3 to a stepping-cost path, so S_3 remains null;† and S_4 becomes null. $S_2 = \{1,3,4\}$, $S_5 = \{3,4\}$. From (3-6-5),

$$c_{12}^* = \min\{h_{12}^1,\ \tilde{c}_3^1 + h_{32}^1,\ \tilde{c}_4^1 + h_{42}^1\} = \min\{.5,1.5,1.5\} = .5$$

$$c_{15}^* = \min\{\tilde{c}_3^1 + h_{35}^1,\ \tilde{c}_4^1 + h_{45}^1\} = \min\{1.5,1\} = 1.$$

Thus, the third IMSC is associated with node 2, with IMSC = (.5, .5). $|\pi_2^*(c)|$ is shown in Fig. 3.6c.

S_3 and S_4 remain null, $S_2 = \{3,4\}$, and $S_5 = \{2,3,4\}$. Again,

$$c_{12}^* = \min\{\tilde{c}_3^1 + h_{32}^1,\ \tilde{c}_4^1 + h_{42}^1\} = \min\{1.5,1.5\} = 1.5$$

$$c_{15}^* = \min\{\tilde{c}_2^1 + h_{25}^1,\ \tilde{c}_3^1 + h_{35}^1,\ \tilde{c}_4^1 + h_{45}^1\} = \min\{1,1.5,1\} = 1$$

$$c_{22}^* = \min\{c_3^1 + h_{32}^2,\ c_4^1 + h_{42}^2\} = \min\{3,1\} = 1$$

$$c_{25}^* = \min\{c_2^1 + h_{25}^2,\ c_4^1 + h_{45}^2\} = \min\{.5,2\} = .5$$

Hence, the fourth IMSC is associated with node 5, with IMSC = (1, .5). $|\pi_5^*(c)|$ is shown in Fig. 3.6d.

---

†In fact $|\pi_3^{1,1}| = |\pi_3|$, so node 3 is exempt from further consideration by (3-7-5).

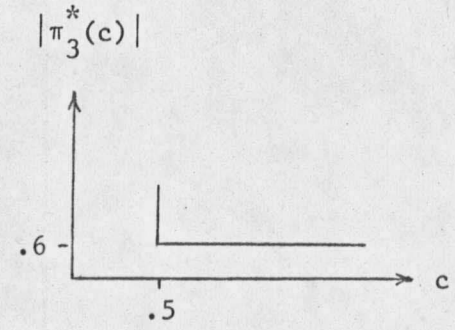


Fig. 3.6a. $|\pi_3^*(c)|$.

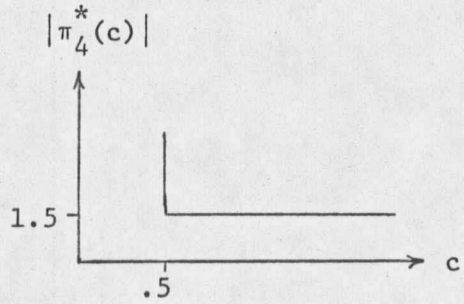


Fig. 3.6b. $|\pi_4^*(c)|$.

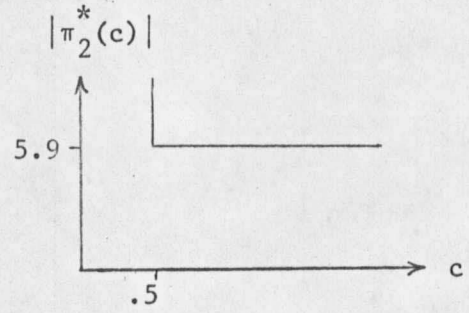


Fig. 3.6c. $|\pi_2^*(c)|$.

Node 2 is no longer in S_5, $S_5 = \{3,4\}$. S_2, c^*_{12}, and c^*_{22} remain unchanged; $c^*_{15} = \min\{\tilde{c}^1_3 + h^1_{35}, \tilde{c}^1_4 + h^1_{45}\} = \min\{1.5,1\} = 1$; $c^*_{25} = c^1_4 + h^2_{45} = 2$. So the fifth IMSC is still associated with node 5, with IMSC = (1, 2). $|\pi^*_5(c)|$ is updated and shown in Fig. 3.6e.

Now $S_5 = \{3\}$ with $c^*_{15} = 1.5$, $c^*_{25} = c^1_3 + h^2_{35} = .5$. S_2 remains unchanged. Hence the sixth IMSC is associated with node 5, with IMSC = (1.5, .5). $|\pi^*_5(c)|$ is updated and shown in Fig. 3.6f.

S_5 becomes null while $S_2 = \{3,4\}$ is the only nonnull set. Hence the seventh IMSC is associated with node 2, with IMSC = (1.5, 1). $|\pi^*_2(c)|$ is updated and shown in Fig. 3.6g.

Now $S_2 = \{3\}$, and 2 becomes a member of S_5 with $S_5 = \{2\}$.

$$c^*_{12} = \tilde{c}^1_3 + h^1_{32} = 1.5$$

$$c^*_{22} = c^1_3 + h^2_{32} = .5 + 2.5 = 3$$

$$c^*_{15} = \tilde{c}^2_2 + h^1_{25} = 1.5 + .5 = 2$$

$$c^*_{25} = c^1_2 + h^2_{25} = 1 + 0 = 1$$

Hence, the eighth IMSC is associated with node 2, with IMSC = (1.5, 3). $|\pi^*_2(c)|$ is updated and shown in Fig. 3.6h.

S_2 becomes null. The ninth IMSC is automatically associated with the only nonnull set, S_5. IMSC = (2, 1). $|\pi^*_5(c)|$ is updated and shown in Fig. 3.6i.

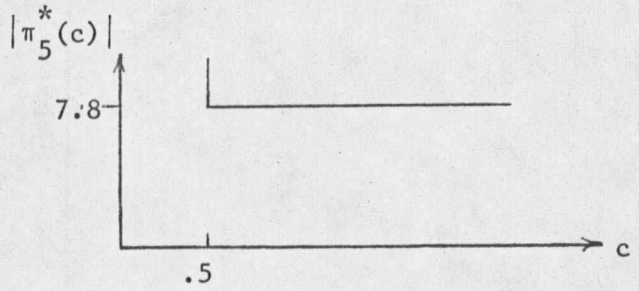


Fig. 3.6d. $|\pi_5^*(c)|$.

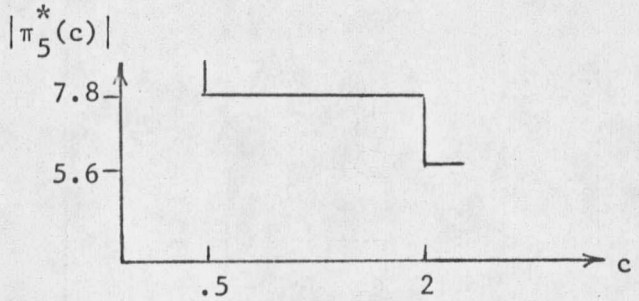


Fig. 3.6e. $|\pi_5^*(c)|$.

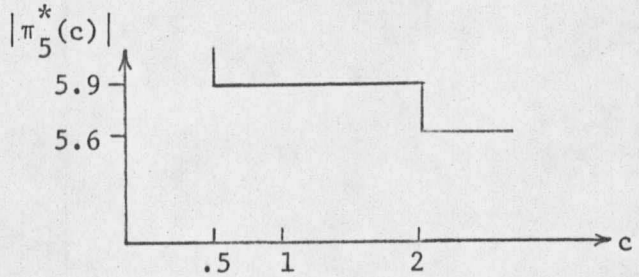


Fig. 3.6f. $|\pi_5^*(c)|$.

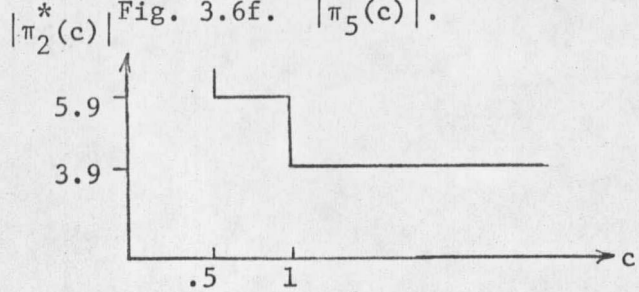


Fig. 3.6g. $|\pi_2^*(c)|$.

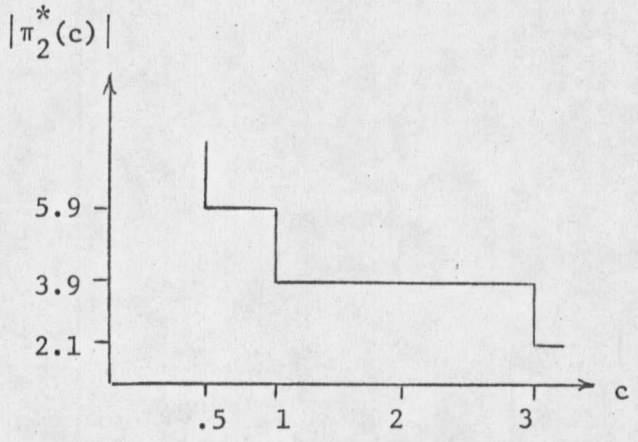


Fig. 3.6h. $|\pi_2^*(c)|$.

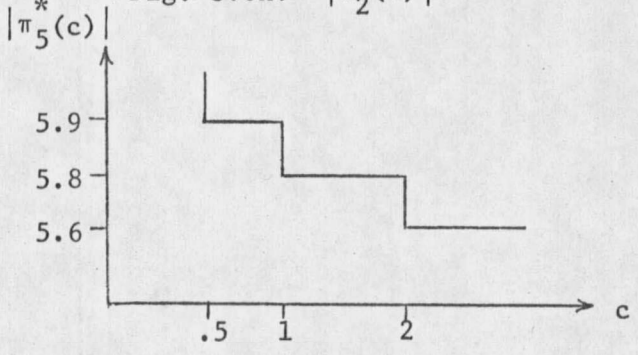


Fig. 3.6i. $|\pi_5^*(c)|$.

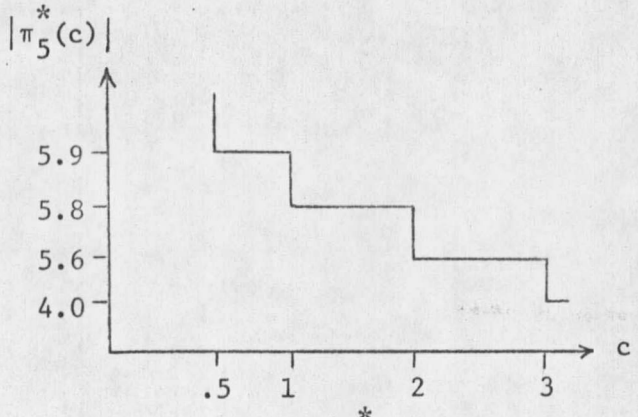


Fig. 3.6j. $|\pi_5^*(c)|$.

Finally, $S_5 = \{2\}$ with $c^*_{15} = \tilde{c}^2_2 + h^1_{25}$, $c^*_{25} = c^2_2 + h^2_{25} = 3 + 0 = 3$. IMSC = (2, 3). $|\pi^*_5(c)|$ is updated and shown in Fig. 3.6j.

The set of stepping costs and the associated stepping-cost path lengths are shown in Fig. 3.7. The 1→5 paths are listed in Table 3.3.

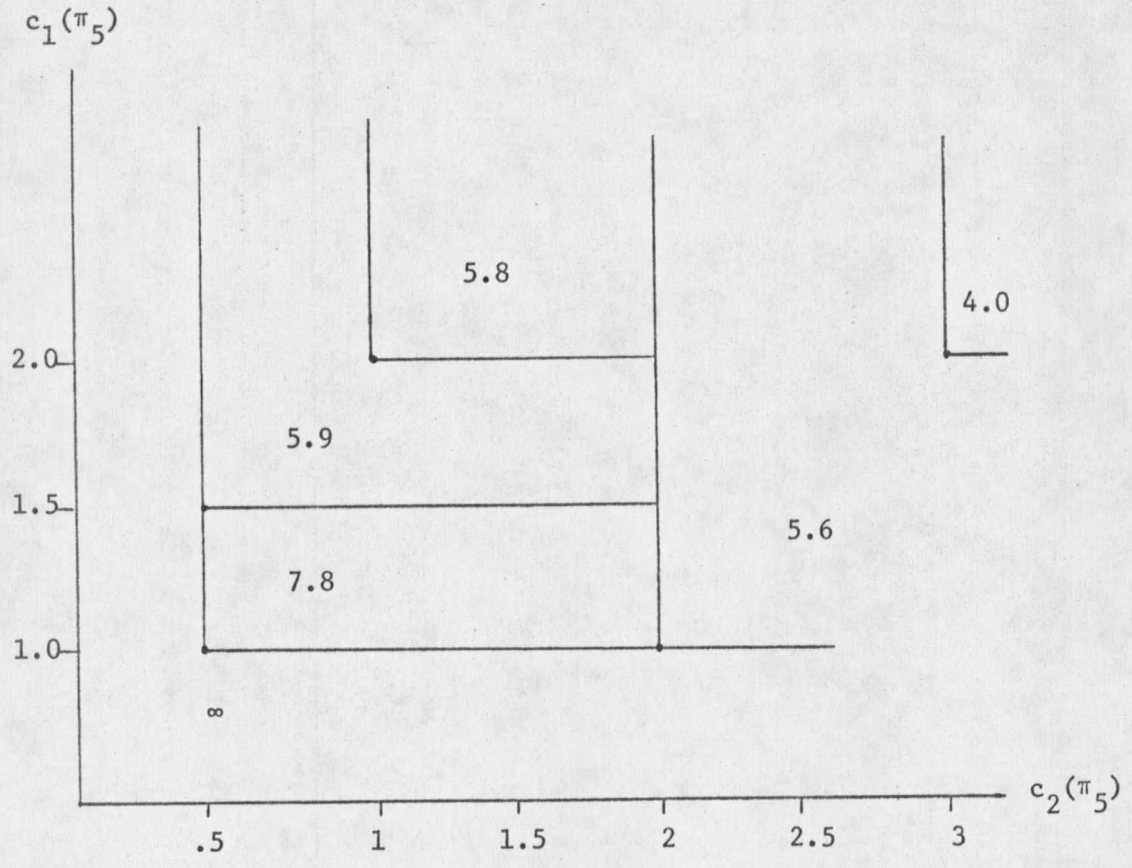


Fig. 3.7a. The 1→5 stepping costs.

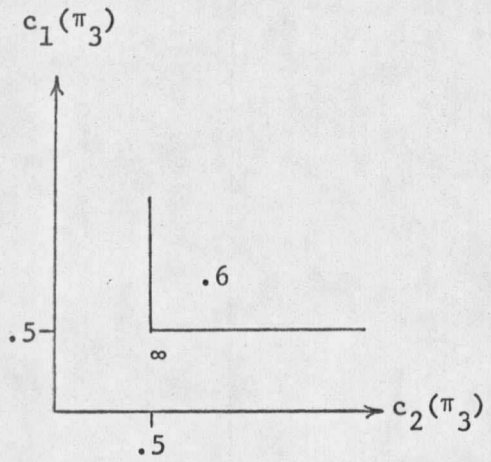


Fig. 3.7b. The 1→3 stepping costs.

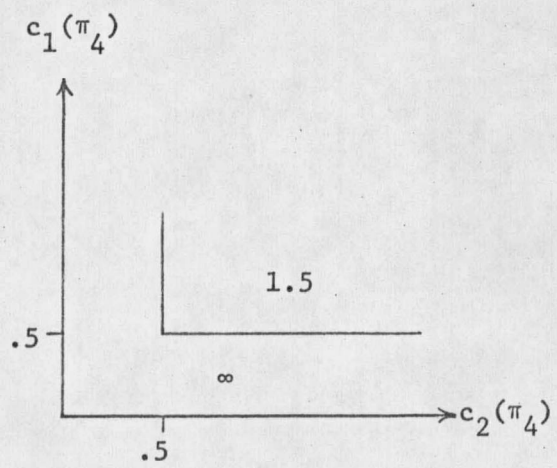


Fig. 3.7c. The 1→4 stepping costs.

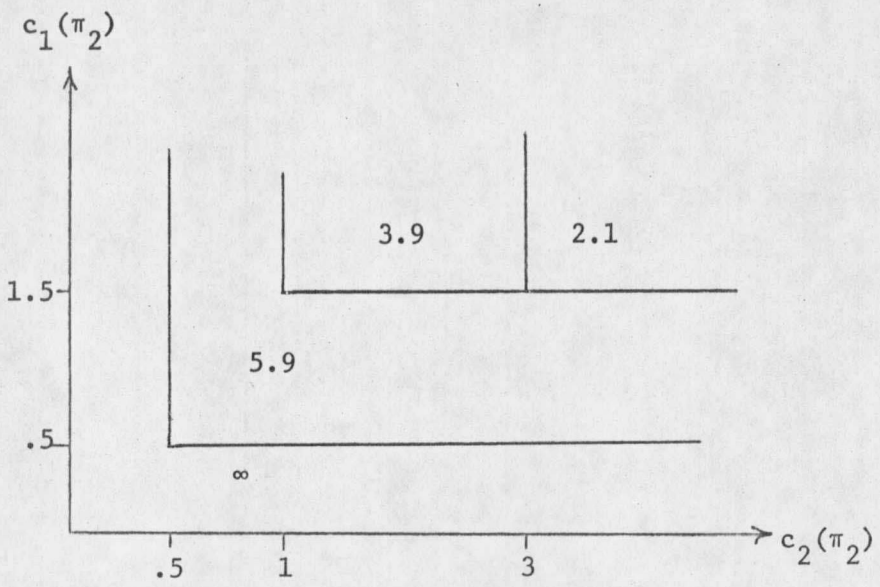


Fig. 3.7d. The 1→2 stepping costs.

Table 3.3. The Set of 1→5 Paths of the Network in Fig. 3.7a.

π_5	$\lvert\pi_5\rvert$	$c_1(\pi_5)$	$c_2(\pi_5)$
1-2-5	7.8	1.0	.5
1-3-5	5.9	1.5	.5
1-4-5	5.6	1.0	2.0
1-2-3-5	12.2	2.0	.5
1-3-2-5	4.0	2.0	3.0
1-3-4-5	5.7	2.0	2.5
1-4-2-5	5.8	2.0	1.0
1-2-3-4-5	12.0	2.5	2.5
1-3-4-2-5	5.9	3.0	1.5
1-4-2-3-5	10.2	3.0	1.0

CHAPTER IV

SOME OPTIMAL PATHS IN PROBABILISTIC WEIGHTED NETWORKS

4.1 Introduction

The general problem of finding the constrained shortest paths in deterministic weighted networks has been discussed in earlier chapters. In practical situations, however, the length of a branch, which may be a measure of the cost or the time needed to traverse that branch, is often a random variable. It may be random by nature, or because of errors in measurements. Hence, the shortest paths are a function of random events characterized by probability distributions. We would like to examine properties of such networks and to introduce definitions of some types of optimal paths. Finally, methods to find such optimal paths are discussed and shown to be applications of constrained shortest path algorithms.

4.2 Probability Distribution of the Shortest Path

Let W_i be a random branch length associated with branch b_i, $i = 1, 2, \ldots, m$, of an n-node m-branch network. Assume that W_i is continuously distributed and that the joint probability density function $P_w(w_1, w_2, \ldots, w_m)$ of the random vector $\underline{W} \triangleq (W_1, W_2, \ldots, W_m)^T$, of which $\underline{W} = (w_1, w_2, \ldots, w_m)^T$ is a sample value, is given. Let $\pi \triangleq \{\pi_1, \pi_2, \ldots, \pi_q\}$ be the set of 1→s paths of the network, and let $|\pi| \triangleq (|\pi_1|, |\pi_2|, \ldots, |\pi_q|)^T$ be the random vector representing the set of 1→s path lengths. Then the length of the shortest 1→s path, denoted as M, can

be written as

$$M = \underset{i}{\text{minimum}}\{|\pi_i|\} = \underset{i}{\text{minimum}}\{\sum_{b_j \in \pi_i} W_j\} \tag{4-2-1}$$

Since the W_j's are random variables, the shortest 1→s path length M is also a random variable characterized by its probability distribution function $F(\ell)$, where

$$F(\ell) \triangleq \text{Prob}[M \leq \ell] \tag{4-2-2}$$

is the cumulative probability that M is not greater than a real value ℓ.

To compute $F(\ell)$ we must first find the probability density function $P_{|\pi|}(x_1, x_2, \ldots, x_q)$ of $|\pi|$ where $\underline{x} = (x_1, x_2, \ldots, x_q)^T$ is the sample vector of $|\pi|$. $F(\ell)$ can then be easily found as

$$F(\ell) = 1 - \int_{\ell}^{\infty} \int_{\ell}^{\infty} \cdots \int_{\ell}^{\infty} P_{|\pi|}(x_1, x_2, \ldots, x_q) dx_1\, dx_2 \cdots dx_q \tag{4-2-3}$$

The rest of this section is devoted to the evaluation of $P_{|\pi|}(x_1, x_2, \ldots, x_q)$.

The sample value $\underline{x}$ of the 1→s path length $|\pi|$ can be expressed as

$$\underline{x} = A\underline{w} \tag{4-2-4}$$

where $A = [A_{ij}]$ is a $q \times m$ matrix of zeroes and ones such that $A_{ij} = 1$ if and only if the branch $b_j \in \pi_i$.

Let $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T$, $\underline{\beta} = (\beta_1, \beta_2, \ldots, \beta_q)^T$, and let $\Phi_w(\underline{\alpha})$, $\Phi_{|\pi|}(\underline{\beta})$ be the characteristic functions of W and $|\pi|$, respectively.

By definition,

$$\Phi_w(\underline{\alpha}) = E\{\exp(j\underline{\alpha}^T\underline{W})\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\exp(j\sum_{i=1}^{m}\alpha_i w_i)\cdot$$

$$P_w(w_1, w_2, \ldots, w_m)dw_1\, dw_2 \cdots dw_m \qquad (4\text{-}2\text{-}5)$$

and

$$\Phi_{|\pi|}(\underline{\beta}) = E\{\exp(j\underline{\beta}^T|\pi|)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\exp(j\sum_{i=1}^{q}\beta_i x_i)\cdot$$

$$P_{|\pi|}(x_1, x_2, \ldots, x_q)dx_1\, dx_2 \cdots dx_q \qquad (4\text{-}2\text{-}6)$$

Then, since $\underline{x} = A\underline{w}$ and since the expectation of a function of a random variable can be computed as the integral of that function and the original variable,

$$\Phi_{|\pi|}(\underline{\beta}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\exp(j\sum_{i=1}^{q}\beta_i\sum_{k=1}^{m}A_{ik}w_k)\cdot$$

$$P_w(w_1, w_2, \ldots, w_m)dw_1\, dw_2 \cdots dw_m$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\exp\{j\sum_{k=1}^{m}(\sum_{i=1}^{q}\beta_i A_{ik})w_k\}\cdot$$

$$P_w(w_1, w_2, \ldots, w_m)dw_1\, dw_2 \cdots dw_m$$

$$= \Phi_w(A^T\underline{\beta}) \qquad (4\text{-}2\text{-}7)$$

Now that the characteristics function $\Phi_{|\pi|}(\underline{\beta})$ of $|\pi|$ is known, we can attempt to find the probability density function of $|\pi|$ by inversion. However, although $\Phi_{|\pi|}(\underline{\beta})$ is well defined, the corresponding density may

not be unless the rank r of A is q. Suppose r < q; then, instead of finding the density function of $|\pi|$, we must find the density function $P_{|\hat{\pi}|}$ of $|\hat{\pi}| = (|\pi_{i_1}|, |\pi_{i_2}|, \ldots, |\pi_{i_r}|)^T$ where $|\hat{\pi}|$ is a set of independent variables of $|\pi|$. The density function of $|\hat{\pi}|$ is the inversion of its characteristic function $\Phi_{|\hat{\pi}|}(\hat{\underline{\beta}})$ which is now

$$\Phi_{|\hat{\pi}|}(\hat{\underline{\beta}}) = \Phi_w(\hat{A}^T\hat{\underline{\beta}}) \tag{4-2-8}$$

where $\hat{\underline{\beta}} \triangleq (\beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_r})$, and $\hat{A}$ is the submatrix of A corresponding to a set of linearly independent rows $i_1, i_2, \ldots, i_r$ of A.

The relation between the set of independent and dependent path lengths can be expressed by a $q \times r$ matrix $A' = [A'_{ji_k}]$ obtained by Gauss reducing the matrix A, written as

$$x_j = \sum_{k=1}^{r} A'_{ji_k} x_{i_k}, \text{ for } j = 1, 2, \ldots, q \tag{4-2-9}$$

Finally, $F(\ell)$ is found as

$$F(\ell) = 1 - \iint \cdots \int_{\Omega} P_{|\hat{\pi}|}(x_{i_1}, x_{i_2}, \ldots, x_{i_r}) dx_{i_1} dx_{i_2} \cdots dx_{i_r} \tag{4-2-10}$$

where Ω is the convex region defined by the set of linear inequalities

$$x_j = \sum_{k=1}^{r} A'_{ji_k} x_{i_k} \geq \ell, \text{ for } j = 1, 2, \ldots, q \tag{4-2-11}$$

For a complete network of n nodes and n^2 branches, the number of 1→s paths may be as many as

$$1 + (n - 2) + (n - 2)(n - 3) + \ldots + (n-2)!$$

So, in general, there are many more paths than branches, justifying the conclusion that in a complete network, most of the 1→s paths are linearly dependent.

Example 4.1

In Fig. 4.1, a 5-node and 12-branch directed network is shown. The set of 1→5 paths are listed in the following:

$\pi_1 = (1,2),(2,5)$

$\pi_2 = (1,3),(3,5)$

$\pi_3 = (1,4),(4,5)$

$\pi_4 = (1,2),(2,3),(3,5)$

$\pi_5 = (1,2),(2,4),(4,5)$

$\pi_6 = (1,3),(3,2),(2,5)$

$\pi_7 = (1,3),(3,4),(4,5)$

$\pi_8 = (1,4),(4,2),(2,5)$

$\pi_9 = (1,4),(4,3),(3,5)$

$\pi_{10} = (1,2),(2,3),(3,4),(4,5)$

$\pi_{11} = (1,2),(2,4),(4,3),(3,5)$

$\pi_{12} = (1,3),(3,2),(2,4),(4,5)$

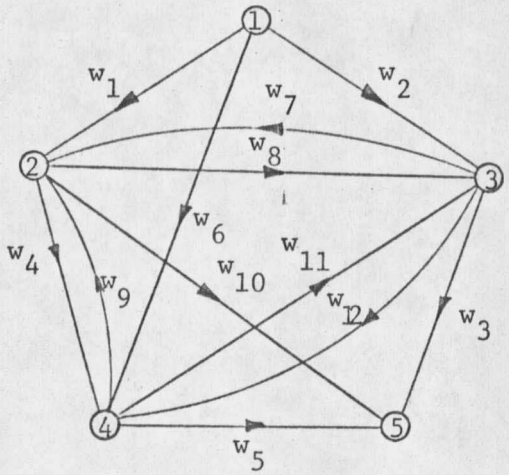


Fig. 4.1. A 5-node 12-branch network.

$\pi_{13} = (1,3),(3,4),(4,2),(2,5)$

$\pi_{14} = (1,4),(4,2),(2,3),(3,5)$

$\pi_{15} = (1,4),(4,3),(3,2),(2,5)$

The A matrix is then

		W_j										
	j / i	1	2	3	4	5	6	7	8	9	10	11	12
	1	1									1		
	2		1	1									
	3					1	1						
	4	1		1					1				
	5	1			1	1							
	6		1					1			1		
	7		1			1							1
X_i	8						1			1	1		
	9			1			1					1	
	10	1				1			1				1
	11	1		1	1							1	
	12		1		1	1		1					
	13		1							1	1		1
	14			1			1		1	1			
	15						1	1			1	1	

After Gauss reduction the equation $\underline{Aw} = \underline{Ix}$ becomes

w_j

x_i \ w_j	1	2	3	4	5	6	7	8	11	10	9	12
1	1									1		
2		1								1	1	1
4			1							-1	-1	-1
5				1							1	
6					1					-1	-1	
3						1				1	1	
7							1				-1	-1
8								1			1	1
9									1			1
10												
11												
12												
13												
14												
15												

$\underline{w}' =$

x_j

x_i \ x_j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1							1							
2			-1				1	-1							
4		1	1				-1	1							
5	-1		-1		1			1							
6			1					-1							
3								1							
7			1			1	-1	-1							
8	-1	-1	-1	1			1	1							
9			-1				1		1						
10		1		-1			-1			1					
11			1		-1				-1		1				
12	1				-1	-1						1			
13			1				-1	-1					1		
14	1			-1				-1						1	
15		1				-1			-1						1

$\underline{x}$

(4-2-13)

where $\underline{w}' = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_{11}, w_{10}, w_9, w_{12})^T$, and I is a 15 × 15 identity matrix. It is found for the example that the

rank r of A is r = 9. Let us select as a linear independent set of 1→5 paths

$$\hat{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, \pi_7, \pi_8, \pi_9)^T$$

and let $\pi' = (\pi_{10}, \pi_{11}, \pi_{12}, \pi_{13}, \pi_{14}, \pi_{15})^T$ denote the linearly dependent set of 1→5 paths. From the lower half submatrix of the right-hand side of the above equation (4-2-13), define a matrix B.

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

Then $\pi' = B\hat{\pi}$ (4-2-14)

and Ω is simply the region defined by

$$\begin{bmatrix} I_9 \\ \cdots \\ B \end{bmatrix} \hat{\pi} \geq \begin{bmatrix} \ell \\ \vdots \\ \ell \end{bmatrix} \qquad (4\text{-}2\text{-}15)$$

where I_9 is a 9 × 9 identity matrix.

4.3 Some Optimal Paths

In the previous section, an exact analysis of the shortest path in a probabilistic weighted network was presented. It is found that given the probabilistic characteristics of the network, the probabilistic density function of the shortest path can be calculated by a multiple integration. The evaluation of the integration becomes much more difficult when there exist some dependent paths, such that a convex region is needed to specify the region of integration. The calculation may become simpler if the branch weights are discretely distributed and the branch weights can only assume a value from a small set of real values. Frank [28] treated the analysis problem by statistical methods. Even though the probabilistic characteristics of the shortest path is completely known, the problem of finding the shortest path is still not yet solved.

Knowing that the shortest path length is indeed a random variable, additional deterministic objectives are needed in order to define an optimal path. Examples of such deterministic objectives are as follows: (1) the optimal path such that the expectation of the path length is smallest; (2) the minimum mean path such that the path length variance is limited; (3) the minimum mean path such that the probability of the path length exceeding a given amount will not be freater than a given fraction; (4) the optimal path such that the probability of the path length exceeding a given amount is minimized. These deterministic paths will

be discussed in the following for a network with a set of stochastically independent random branch weights.

DEFINITION 4.1. The minimal mean 1→s path is the path which is minimal in the mean value of the 1→s path lengths.

The minimal mean 1→s path is the path with minimal path length on the average and is trivial to obtain since it is simply the shortest 1→s path of the averaged network; that is, the network whose branches are labeled by the mean value of the original weight.

The biggest drawback of this minimal mean path is the absence of knowledge of its variance. It may be that since the variance of the path length is so large that one is forced to accept a path with a larger mean value but having much less variance. Thus, minimal mean paths that satisfy a certain variance constraint are of practical interest.

Similar to the concept of the constrained shortest paths in a network with a length and a cost associated with each branch, let the mean value and the variance of the random branch weight in the probabilistic weighted network be the length and cost weights, respectively, associated with each branch of a deterministic weighted network of the same network topology. The cost constrained shortest paths of this network are then the variance constrained minimal mean paths.

Let the symbols μ and σ^2 denote the mean and the variance, respectively. Let $\pi(\sigma)$ denote the constrained minimal mean 1→s path such that the variance of the path length is not greater than σ^2.

4.3.1 The Minimal Mean Path of Confidence Level q_0 at ℓ_0

DEFINITION 4.2. The minimal mean path $\hat{\pi}$ of confidence level q_0 at ℓ_0 is the minimal mean path such that the probability of the path length exceeding ℓ_0 is not greater than a positive fraction q_0.

When the random branch weights are normally distributed, then the probability condition that $\hat{\pi}$ has to satisfy can be written as

$$\text{Prob}[|\hat{\pi}| > \ell_0] = 1 - N\{(\ell_0 - \text{mean of } |\hat{\pi}|)/\text{standard deviation of } |\hat{\pi}|\} \leq q_0 \qquad (4\text{-}3\text{-}1\text{-}1)$$

where

$$N(x) \triangleq \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-y^2/2)dy \qquad (4\text{-}3\text{-}1\text{-}2)$$

is the cumulative distribution function of a normalized zero-mean and unit-variance Gaussian random variable. Define $N^{-1}(\cdot)$ as the inverse function of $N(\cdot)$, and let

$$N_0 \triangleq N^{-1}(1 - q_0) \qquad (4\text{-}3\text{-}1\text{-}3)$$

Let μ_0 and σ_0^2 denote, respectively, the mean and variance of $|\hat{\pi}|$; then (4-3-1-1) is

$$\frac{\ell_0 - \mu_0}{\sigma_0} \geq N_0 \qquad (4\text{-}3\text{-}1\text{-}4)$$

Thus we have changed the probabilistic requirement into its deterministic equivalent of which the mean and variance of the length of the constrained minimal mean path $\hat{\pi}$ must satisfy.

$\hat{\pi}$ can be found by first finding the set of constrained minimal mean paths $\pi(\sigma)$ with variance not greater than σ^2, and then searching over the set of paths $\pi(\sigma)$ that satisfies (4-3-1-4); the one with minimal mean path length is $\hat{\pi}$.

Fig. 4.2 illustrates the evaluation of $\hat{\pi}$. In Fig. 4.2, the minimal mean path with variance not greater than σ_1^2 satisfies (4-3-1-4) is the path with minimal mean.

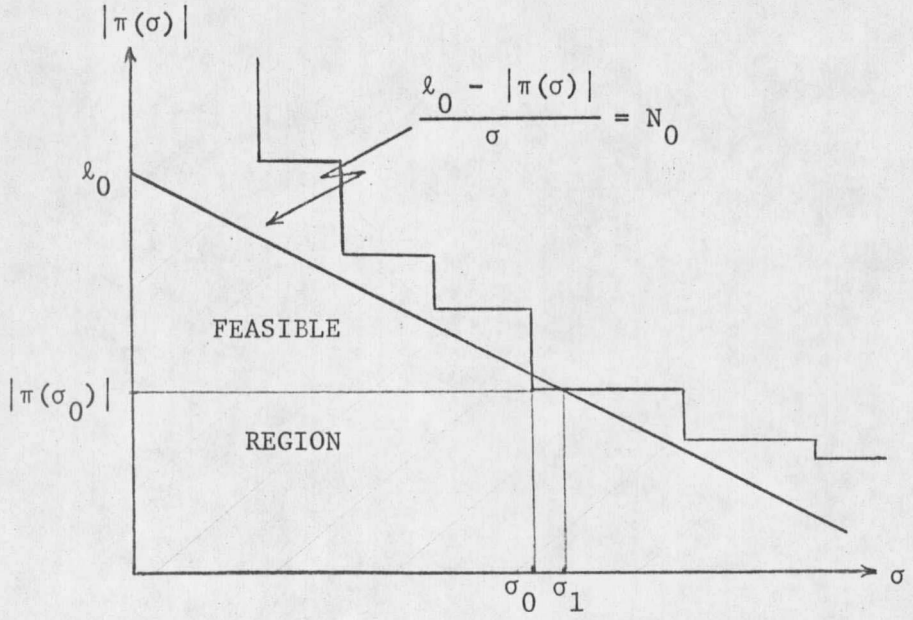


Fig. 4.2. A confidence level curve and the feasible region.

The stepping-deviation path $\pi(\sigma_0)$ is also a candidate for $\hat{\pi}$. In the algorithm presented in previous chapters, only the set of stepping-cost paths are retained. So we can choose $\pi(\sigma_0)$ as $\hat{\pi}$.

Assertion 4.1 Let $\{\pi(\sigma_j)\}$ be the set of stepping-deviation 1→s paths with mean $\mu(\sigma_j)$ and variance σ_j; then the constrained minimal mean 1→s path $\hat{\pi}$ can be found as the stepping-deviation 1→s path with mean $\hat{\mu}$ and its corresponding stepping deviation $\hat{\sigma}$ that satisfies

$$\hat{\mu} = \underset{\pi(\sigma_j)\in S_0}{\text{minimum}} \{\mu(\sigma_j)\}$$

where S_0 is that subset of stepping-deviation paths that satisfy

$$\frac{\ell_0 - \mu(\sigma_j)}{\sigma_j} \geq N_0 \qquad (4\text{-}3\text{-}1\text{-}5)$$

4.3.2 The Maximal Confidence Level ℓ_0 Path

DEFINITION 4.3 The maximal confidence level ℓ_0 path is that path which is minimal in the probability of the path length exceeding ℓ_0.

It is obvious that the maximal confidence level π_0 path must be a variance constrained minimal mean path. For if not, a path with the same variance but with a smaller mean would result in a smaller probability of the path length exceeding π_0. This is true because N(x) is a strictly increasing function of x.

Secondly, the maximal confidence level ℓ_0 path must be a stepping-deviation path. For if not, there exists a stepping-deviation path with the same mean but with a smaller deviation, again resulting in a smaller probability of the path length exceeding ℓ_0. So if $\{\pi(\sigma_j)\}$ is the set of stepping-deviation 1→s paths with mean $\mu(\sigma_j)$, deviation σ_j, the maximal confidence level ℓ_0 path $\hat{\pi}$ with the mean $\hat{\mu}$, variance $\hat{\sigma}^2$ is the stepping-deviation path such that

$$\frac{\ell_0 - \hat{\mu}}{\hat{\sigma}} = \max_j \frac{\ell_0 - \mu(\sigma_j)}{\sigma_j} \qquad (4\text{-}3\text{-}1\text{-}6)$$

The selection of $\hat{\pi}$ is illustrated in Fig. 4.3.

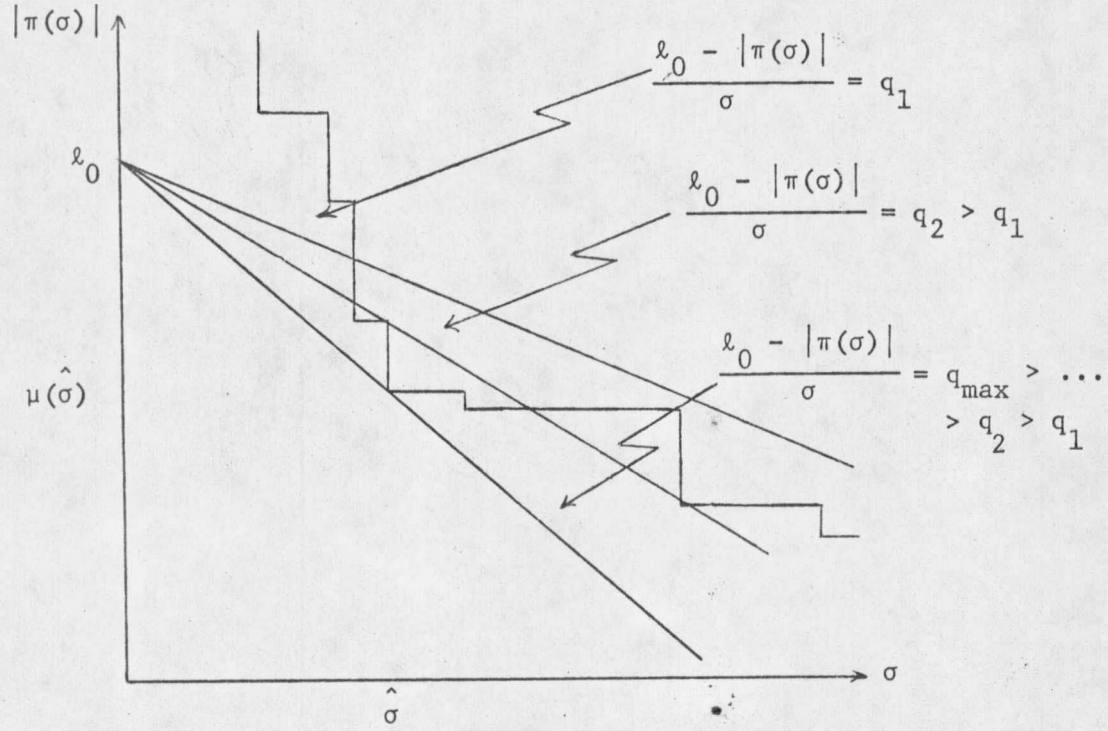


Fig. 4.3. The maximal confidence level ℓ_0 path.

In Fig. 4.3 the confidence level curves starting from the right-hand side to the left are in an order of increasing confidence level. The maximum confidence level q_{max} is found at variance $\hat{\sigma}^2$ with the associated path $\pi(\hat{\sigma})$ and mean $\mu(\hat{\sigma})$.

The assumption of the branch weights being normally distributed can be relaxed by the fact that under very general conditions, sums of random variables are approximately normally distributed [55].

4.4 Stochastically Dependent Weights

In the preceding treatment of certain optimal paths the branch weights are assumed stochastically independent. For networks whose branch weights may be stochastically dependent, a 1→s path still has the sum of its constituent branch means as the mean of the path length; however, the path variance, depending on the choice of constituent branches, and the stochastically dependence relationship between the branches, may not be the simple sum of the constituent branch variances. This creates a new dimension for constrained shortest paths problems and is recommended for future research.

Let π denote a path in an m-branch network with an associated random weight W_j for each branch b_j. Express $|\pi|$, the length of π, as

$$|\pi| = \sum_{j=1}^{m} a_j W_j \qquad (4\text{-}4\text{-}1)$$

where

$a_j = 1$ if $b_j \in \pi$; otherwise $a_j = 0$.

Suppose the mean and variance for W_j's are known, and the covariance between any two random weights are also known. By definition,

$$\sigma_j^2 = E[W_j - E(W_j)]^2 = E[W_j^2] - E^2[W_j] \tag{4-4-2}$$

$$\sigma_{ij} = E[W_i - E(W_i)][W_j - E(W_j)] = E[W_iW_j] - E[W_i]E[W_j] \tag{4-4-3}$$

Then $E[|\pi|]$ is

$$E[|\pi|] = E[\sum_{j=1}^{m} a_jW_j] = \sum_{j=1}^{m} a_jE[W_j] \tag{4-4-4}$$

$$E[|\pi|^2] = E[\sum_{i=1}^{m} a_i^2W_i^2 + 2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} a_ia_jW_iW_j] = \sum_{i=1}^{m} a_i^2E[W_i^2]$$

$$+ 2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} a_ia_jE[W_iW_j] \tag{4-4-5}$$

Hence, the variance σ_π^2 of $|\pi|$ is

$$\sigma_\pi^2 = E[|\pi|^2] - E^2[|\pi|] = \sum_{i=1}^{m} a_i^2\sigma_i^2 + 2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} a_ia_j\sigma_{ij} \tag{4-4-6}$$

Thus

$$\sigma_\pi^2 = \sum_{i=1}^{m} a_i^2\sigma_i^2 = \text{sum of the constituent branch weight variances} \tag{4-4-7}$$

only if

$$\sigma_{ij} = 0, \text{ for all } b_i, b_j \in \pi$$

which justifies the above statement.

CHAPTER V

SHORTEST PATHS IN FLOW NETWORKS

5.1 Introduction

A flow network is a network where each branch (i,j) is associated with a nonnegative weight c_{ij} called the branch capacity which represents the maximal amount of commodity that can arrive at node j from node i along (i,j) per unit time. The commodity may be messages in a communication network, traffic in a road network, or oil in a pipeline network, etc. A natural question is: What is the maximal amount of commodity flow that can be shipped between an initial-destination node pair through the network? Or, if an additional cost weight is assigned to the branch representing the unit cost of commodity flow along (i,j), then a question is: Given the requirement of flow from one node to another, how should the commodity be routed through the network so that the total cost is minimized?

Ford and Fulkerson [56] contributed most of the fundamental work for the maximum network flow problem in Section 2 and the well-known problem of constructing flow at minimum cost of Section 3. Section 4 treats the minimum cost flow problem with an additional transit delay requirement where the constrained shortest paths approach contributes to a suboptimal solution. Section 5 discusses some practical paths in flow networks.

5.2 Maximum Flow in Networks

Let nodes s and t be, respectively, the source and sink of the commodity flow in a directed network G(V,B). Let F denote the set of flows $\{f_{ij}\}$ associated with the set of branches in G(V,B); F is called a flow pattern. The flow pattern F is said to be feasible if it satisfies

$$\sum_{j\in V} f_{ij} - \sum_{k\in V} f_{ki} = \begin{cases} f^*_{st} & , \text{ if } i = s \\ 0 & , \text{ if } i \neq s,t \\ -f^*_{st} & , \text{ if } i = t \end{cases} \qquad (5\text{-}2\text{-}1)$$

and

$$c_{ij} \geq f_{ij} \geq 0 \qquad (5\text{-}2\text{-}2)$$

where $V = \{1, 2, \ldots, n\}$ is the set of nodes in G(V,B), and f^*_{st} is the amount of s→t flow.

Equation (5-2-1) expresses the assumption that flow is restricted to be conserved at all nodes except nodes s and t. Equation (5-2-2) indicates that the flow f_{ij} in each branch is one directional -- along the direction of the branch, and is bounded from above by the associated branch capacity. Fig. 5.1 illustrates a flow pattern of value 2.

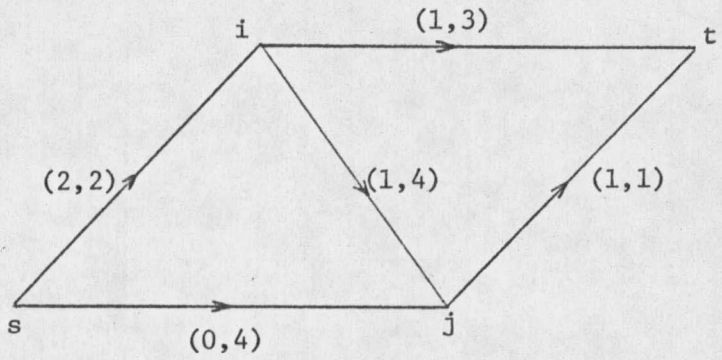


Fig. 5.1. A flow pattern where $W_{ij} = (f_{ij}, c_{ij})$.

Let $\pi_{s,t}$ denote a chain from node s to node t. With respect to $\pi_{s,t}$, the flow f_{ij} of branch (i,j) is said to be a forward flow if (i,j) is a forward branch in $\pi_{s,t}$; f_{ij} is said to be a backward flow if (i,j) is a backward branch in $\pi_{s,t}$. The forward flows in $\pi_{s,t} = \{(s,j),(j,i),(i,t)\}$ in Fig. 5.1 are $f_{sj} = 0$ and $f_{it} = 1$; the backward flow is $f_{ji} = 1$.

DEFINITION 5.1. Residual capacity $\gamma(\pi_{s,t})$ of an s→t chain $\pi_{s,t}$ in a directed network is a nonnegative value defined as

$$\gamma(\pi_{s,t}) \triangleq \text{minimum}\{\underset{\substack{\text{backward branches}\\(i,j)\in\pi_{s,t}}}{\text{minimum}}\{f_{ij}\}, \underset{\substack{\text{forward branches}\\(a,b)\in\pi_{s,t}}}{\text{minimum}}\{c_{ab}-f_{ab}\}\} \tag{5-2-3}$$

A chain $\pi_{s,t}$ is said to be saturated if $\gamma(\pi_{s,t}) = 0$, and to be unsaturated otherwise. If a chain $\pi_{s,t}$ is not saturated, then one can

add the amount of flow $\gamma(\pi_{s,t})$ to all forward flows in the chain and subtract the amount of flow $\gamma(\pi_{s,t})$ from all backward flows, thereby obtaining a new flow pattern for which flow is still conserved and branch capacities are not exceeded. The new flow pattern then corresponds to a larger amount of s→t flow that is transmitted from node s to node t.

DEFINITION 5.2. An s→t augmentation path $\pi_{s,t}$ is an unsaturated s→t chain.

A procedure to find the maximum s→t flow automatically follows:

Step 1. Let $f_{ij} = 0$, for all $(i,j) \in B$.

Step 2. Find an s→t augmentation path $\pi_{s,t}$; if none exists, terminate.

Step 3. Increase the flow in the augmentation path $\pi_{s,t}$ by $\gamma(\pi_{s,t})$; return to Step 2.

DEFINITION 5.3. A cut $(X,\bar{X})$ is the set of branches (i,j) where $i \in X$, $j \in \bar{X}$, and X is a subset of V with $\bar{X}$ denoting its node complement V-X. $(X,\bar{X})$ is called an s→t cut if $s \in X$, $t \in \bar{X}$, and is denoted as A_{st}.

For example, {(s,i),(s,j)}, {(i,t),(j,t)}, {(s,i),(i,j),(i,t)}, and {(s,i),(j,t)} are cuts of the network in Fig. 5.1 with X being, respectively, {s}, {s,i,j}, {s,i}, and {s,j}; in addition, they are s→t cuts.

DEFINITION 5.4. The value $C(X,\bar{X})$ of a cut $(X,\bar{X})$ is the sum of the branch capacities of the cut. $C(A_{st})$ denotes the value of an s→t cut.

The values of cuts for the above example are respectively 6, 4, 11, and 3.

A famous theorem which specifies the maximum s→t flow value in terms of the network characteristics is proved independently by Ford and Fulkerson [56], Dantzig, et al [57], and Elias, et al [58] and is presented in the following.

Theorem 5.1. The Max-Flow Min-Cut Theorem. Let τ_{st} be the value of the minimum s→t cut, that is,

$$\tau_{st} = \min_{k}\{C(A_{st}^{k})\} \tag{5-2-4}$$

where $\{A_{st}^{k}\}$ is the set of s→t cuts; then for any feasible flow pattern F with s→t flow value f_{st}^{*},

$$\underset{F}{\text{maximum}}\{f_{st}^{*}\} = \underset{k}{\text{minimum}}\{C(A_{st}^{k})\} = \tau_{st} \tag{5-2-5}$$

Lemma 5.1. $f_{st}^{*} \leq \tau_{st}$

Proof.

From (5-2-1),

$$f^*_{st} = \sum_{j \in V} f_{sj} - \sum_{k \in V} f_{ks} \tag{5-2-6}$$

and

$$0 = \sum_{j \in V} f_{ij} - \sum_{k \in V} f_{ki} \text{ , for } i \neq s,t \tag{5-2-7}$$

Thus, for any nonnull proper subset X of V such that $s \in X$ and $t \in \bar{X}$,

$$\sum_{i \in X} \left(\sum_{j \in V} f_{ij} - \sum_{k \in V} f_{ki} \right) = f^*_{st} \tag{5-2-8}$$

or

$$\sum_{i \in X} \left(\sum_{j \in X} f_{ij} + \sum_{j \in \bar{X}} f_{ij} - \sum_{k \in X} f_{ki} - \sum_{k \in \bar{X}} f_{ki} \right) = f^*_{st} \tag{5-2-9}$$

The first and third terms in (5-2-9) cancel, and (5-2-9) is therefore

$$\sum_{i \in X} \left(\sum_{j \in \bar{X}} f_{ij} - \sum_{k \in \bar{X}} f_{ki} \right) = f^*_{st} \tag{5-2-10}$$

However, since by (5-2-2), all flows are nonnegative, $\sum_{i \in X} \sum_{k \in \bar{X}} f_{ki} \geq 0$; therefore (5-2-10) is

$$f^*_{st} \leq \sum_{i \in X} \sum_{j \in \bar{X}} f_{ij} \tag{5-2-11}$$

Using (5-2-11) and the fact that $f_{ij} \leq c_{ij}$,

$$f^*_{st} \leq \sum_{i \in X} \sum_{j \in \bar{X}} c_{ij} \tag{5-2-12}$$

However, the right-hand side is the value of an s t cut, and X is arbitrary, so

$$f^*_{st} \leq \tau_{st} \tag{5-2-13}$$

Proof of the Max-Flow Min-Cut Theorem.

In the procedure shown before to construct the maximum flow pattern, an s→t augmentation path is sought on which additional s→t flow is routed. We want to show that when the procedure terminates, $f^*_{st} \geq \tau_{st}$.

Define a set of nodes L such that

$$s \in L \tag{5-2-14}$$

and

$$j \in L, \text{ if there is an } i \in L \text{ such that } c_{ij} > f_{ij} \text{ or } f_{ji} > 0 \tag{5-2-15}$$

We know that $t \in \bar{L}$ because otherwise there would be an s→t augmentation path contradicting the assumption that the procedure has terminated. Hence $\bar{L}$ is not empty. From (5-2-10),

$$f^*_{st} = \sum_{i \in L} \sum_{j \in \bar{L}} f_{ij} - \sum_{i \in L} \sum_{k \in \bar{L}} f_{ki} \tag{5-2-16}$$

But from the definition of L if $(i,j) \in (L,\bar{L})$, the $f_{ij} = c_{ij}$, and if $(k,i) \in (\bar{L},L)$, then $f_{ki} = 0$; hence

$$\sum_{i\in L}\sum_{j\in \bar{L}} f_{ij} = \sum_{i\in L}\sum_{j\in \bar{L}} c_{ij} \qquad (5\text{-}2\text{-}17)$$

And

$$\sum_{i\in L}\sum_{k\in \bar{L}} f_{ki} = 0 \qquad (5\text{-}2\text{-}18)$$

So

$$f^{*}_{st} = \sum_{i\in L}\sum_{j\in \bar{L}} c_{ij} \geq \tau_{st} \qquad (5\text{-}2\text{-}19)$$

which completes the proof.

5.3 The Minimum Cost Flow

In more practical situations one has supply of a commodity at a certain node in a flow network, demand at another, and the objective is to satisfy the demand from the supply at minimum cost.

Let an additional weight h_{ij} be associated with each branch representing the unit cost of flow transmitted through the branch.

The problem is to minimize

$$\sum_{(i,j)\in B} h_{ij}\cdot f_{ij} \qquad (5\text{-}3\text{-}1)$$

where f_{ij} is the flow of (i,j) satisfying the conservation and capacity requirements in (5-2-1) and (5-2-2); f^{*}_{st} in (5-2-1) represents the demand of s→t flow.

DEFINITION 5.5. The minimum cost s→t augmentation path $\pi_{s,t}$ is an unsaturated s→t chain with minimum cost which is the sum of the branch costs over the forward branches minus the corresponding sum over the backward branches in the chain, and is the minimum cost of sending one unit of additional s→t flow through the network.

In Fig. 5.2 is shown a minimum cost s→t augmentation path $\pi_{s,t}$ = {(s,1),(1,2),(2,t)}. The forward branches are (s,1),(2,t); the backward branch is (1,2). $\pi_{s,t}$ is of cost 2 - 1 + 4 = 5 and residual capacity = 1. The s→t chain {(s,1),(1,2),(2,3),(3,t)} is also unsaturated but with cost 6.

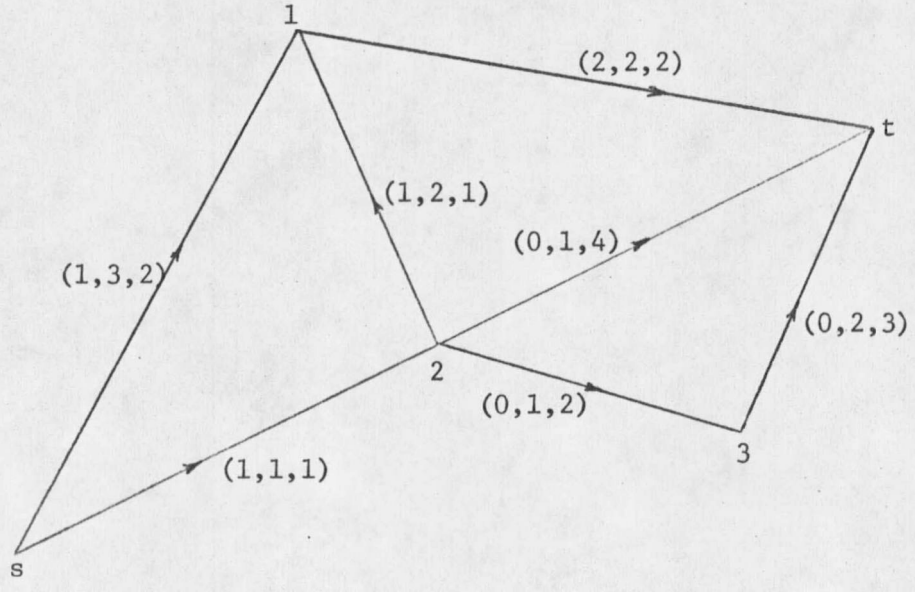


Fig. 5.2. A minimum cost s→t augmentation path where $W_{ij} = (f_{ij}, c_{ij}, h_{ij})$.

Assertion 5.1 Let F be the minimum cost set flow pattern of requirement f^*_{st}; then the flow pattern obtained by sending additional flow $\gamma(\pi_{s,t})$ on $\pi_{s,t}$ results in the minimum cost flow pattern of an amount $f^*_{st} + \gamma(\pi_{s,t})$.

A procedure similar to the one for the maximum flow problem is displayed as follows.

Step 1. Set $f_{ij} = 0$ for all $(i,j) \in B$.

Step 2. Find the minimum cost s→t augmentation path $\pi_{s,t}$. If none exists, terminate.

Step 3. $\gamma = \min\{\gamma(\pi_{s,t}), f^*_{st} - f_{st}\}$; increase s→t flow along $\pi_{s,t}$ by the amount γ.

Step 4. If $f_{st} < f^*_{st}$ return to Step 2, otherwise terminate.

The problem that remains is how to find $\pi_{s,t}$ in Step 2.

Theorem 5.2 Let G(V,B') be a network generated from G(V,B) such that G(V,B') contains the same set of nodes, and $(i,j) \in B'$ if either $(i,j) \in B$ and $f_{ij} < c_{ij}$, $f_{ji} = 0$, or $(j,i) \in B$ and $f_{ji} > 0$. Define the cost h'_{ij} of (i,j) in G(V,B') as

$$h'_{ij} = \begin{cases} h_{ij}, & \text{if } f_{ij} < c_{ij} \text{ and } f_{ji} = 0 \\ -h_{ji}, & \text{if } f_{ji} > 0 \end{cases} \qquad (5\text{-}3\text{-}2)$$

Then the minimum cost s→t path $\pi'_{s,t}$ of G(V,B') is the minimum cost s→t augmentation path $\pi_{s,t}$ of G(V,B).

Lemma 5.1 Any s→t path of G(V,B') is an s→t augmentation path of G(V,B).

Proof.

Consider a branch (i,j) in an s→t path of G(V,B'). From Theorem 5.2, it must be either that $f_{ij} < c_{ij}$ and $f_{ji} = 0$, or that $f_{ji} > 0$; these are exactly the conditions that (i,j) may be a member of an s→t augmentation path.

The branch cost h'_{ij} of (i,j) in an s→t path of G(V,B') corresponds to the cost of sending an additional unit flow along the forward branch in $\pi_{s,t}$ if h'_{ij} is positive, or the saving of reducing a unit flow from the backward branch in $\pi_{s,t}$ if h'_{ij} is negative. Hence, the minimum cost s→t path of G(V,B') is the minimum cost s→t augmentation path of G(V,B).

The combinations of branch generation from G(V,B) are shown in Figs. 5.3a through 5.8b.

In Figs. 5.3a and 5.5a, $f_{ij} < c_{ij}$, then $(i,j) \in B'$ with unit flow cost equal to h_{ij}; and if $(i,j) \in \pi'_{s,t}$, then $(i,j) \in \pi_{s,t}$ is a forward branch where additional flow may be sent through (i,j). Furthermore, if $f_{ij} > 0$, then $(j,i) \in B'$ with unit flow cost equal to $-h_{ij}$; if $(j,i) \in \pi'_{s,t}$, then $(i,j) \in \pi_{s,t}$ and is a backward branch where the

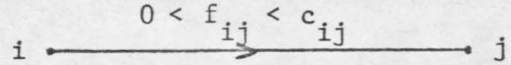


Fig. 5.3a. $(i,j) \in B$, $(j,i) \notin B$.

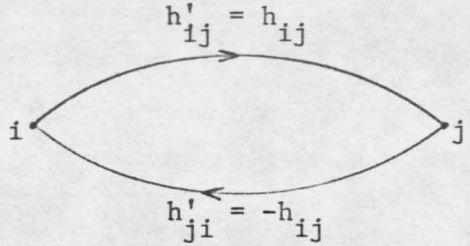


Fig. 5.3b. Branches generated from Fig. 5.3a.

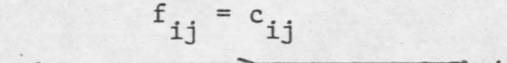


Fig. 5.4a. $(i,j) \in B$, $(j,i) \notin B$.

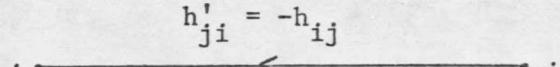


Fig. 5.4b. Branches generated from Fig. 5.4a.

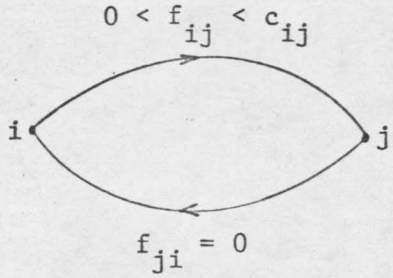


Fig. 5.5a. $(i,j),(j,i) \in B$.

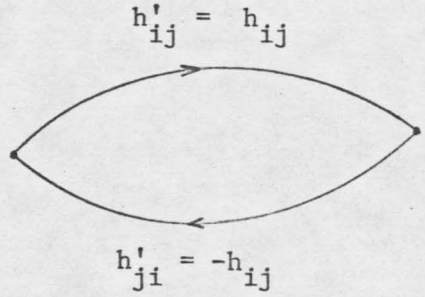


Fig. 5.5b. Branches generated from Fig. 5.5a.

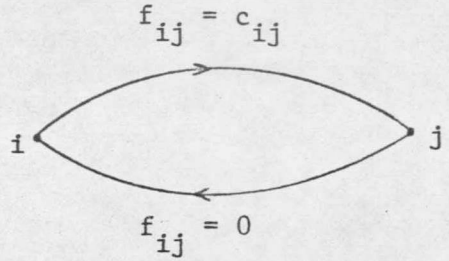


Fig. 5.6a. $(i,j),(j,i) \in B$.

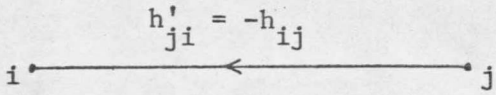


Fig. 5.6b. Branches generated from Fig. 5.6a.

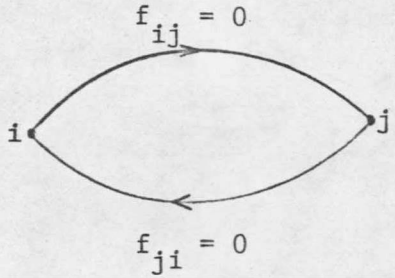


Fig. 5.7a. $(i,j),(j,i) \in B$.

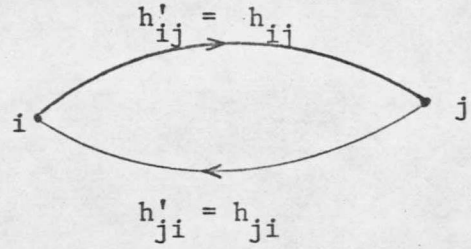


Fig. 5.7b. Branches generated from Fig. 5.7a.

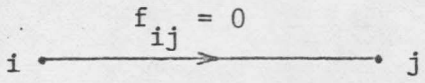


Fig. 5.8a. $(i,j) \in B$.

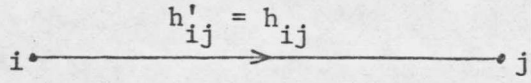


Fig. 5.8b. Branches generated from Fig. 5.8a.

branch flow may be reduced. In Figs. 5.7a and 5.8a, $f_{ji} = 0$, then $(i,j) \in B'$ with unit flow cost equal to h_{ij}, and (i,j) may be a forward branch in $\pi_{s,t}$. In Fig. 5.7a, (j,i) may also be a forward branch in $\pi_{s,t}$. In Figs. 5.4a and 5.6a, $f_{ij} = c_{ij}$, (i,j) is saturated and only reduction of flow is possible; hence, $(j,i) \in B'$ with a negative unit flow cost. If $(j,i) \in \pi'_{s,t}$, then $(i,j) \in \pi_{s,t}$ and is a backward branch.

In Step 2 of the previous scheme to find the minimum cost flow, once G(V,B') is generated from the flow network G(V,B) under the current flow situation, the minimum cost s→t path $\pi'_{s,t}$ then corresponds to the minimum cost s→t augmentation path $\pi_{s,t}$.

As a special case, when the s→t flow requirement $f^*_{st} = 1$, and all branch capacities are equal to one unit of flow, the minimum cost flow problem reduces to a shortest path problem.

5.4 The Minimum Cost Flow Problem with Transit-Time Delay

The problem considered in this section is that of finding the minimum cost s→t flow pattern which, besides satisfying the given s→t flow requirement, must also satisfy a transit-time delay constraint. A transit-time delay is defined as the time required for the flow to travel from its source to the sink and is the sum of the transit-time delay on the constituent branches in the flow path.

A weight d_{ij}, in addition to the flow f_{ij}, capacity c_{ij}, and unit flow cost h_{ij}, is associated with each branch of the network representing the time required to traverse the branch (i,j). In flow networks, such as communication networks or computer networks, this weight d_{ij} is a function of the branch residual capacity -- the branch capacity minus the present branch flow, the message processing time which is the time required for decoding or encoding the heading message, reassembling of the message, etc., before it is ready to be transmitted when the channel is available, the arriving pattern of messages, and the propagation time through the branch [20]. Nevertheless, the weight d_{ij} may be assigned a fixed value (say, through estimation), to result in a simplified model so that efficient computation is feasible.

When the minimum cost s→t augmentation path which satisfies the transit-time delay requirement contains only forward branches, additional flow may be sent through this s→t path to yield a new minimum cost flow pattern of larger s→t flow value. However, if the minimum cost s→t augmentation path contains backward branches, care must be taken to assure that the resulting flow paths satisfy the transit-time delay requirement, which is a tedious process. A practical solution is to adopt a suboptimal minimum cost flow scheme that allows only forward branches for the minimum cost s→t augmentation paths so that the constrained shortest path scheme is applicable.

Suppose the minimum cost s→t augmentation path $\pi_{s,t}$ contains some backward branches; flow on the backward branches is to be switched into the succeeding forward path in $\pi_{s,t}$ while letting additional s→t flow to be sent through the preceding forward path in $\pi_{s,t}$, as illustrated by the following figure with $\pi_{s,t} = \{(s,j),(j,i),(i,t)\}$. If $\pi'_{s,t} = \pi_{s,i} \cup (i,j) \cup \pi_{i,t}$ is a previous minimum cost s→t augmentation path through which part of the flow on (i,j) has been sent, then $(s,j) \cup \pi_{j,t}$ and $\pi_{s,i} \cup (i,t)$ are two new s→t forward flow paths that result from sending additional flow through the forward branches (s,j) and (i,t), while reducing the same flow amount from the backward branch (j,i). A question is: Do these paths satisfy the transit-time delay requirement?

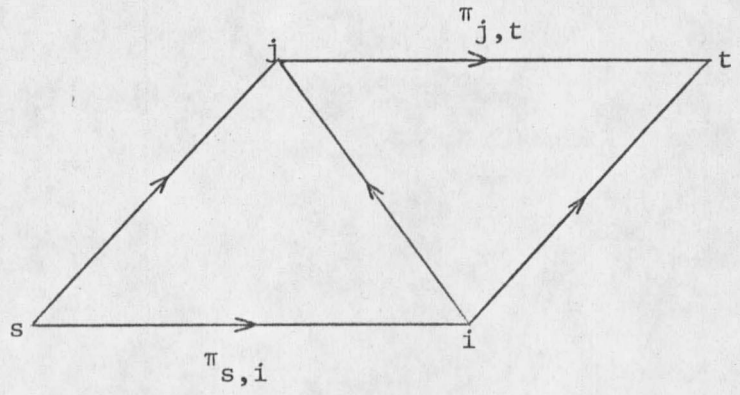


Fig. 5.9. A $\pi_{s,t}$ with one backward branch.

One way of assuring that is to keep a record of the set of existing s→t forward flow paths $\{\pi_{s,t}\}$; then check to see if there exists an s→t flow path, say $\pi^1_{s,t}$, $\pi^1_{s,t} = \pi^1_{s,i} \cup (i,j) \cup \pi^1_{j,t}$, which contains (i,j), such that the new s→t forward flow paths, $(s,j) \cup \pi^1_{j,t}$ and $\pi^1_{s,i} \cup (i,t)$, satisfy the transit-time delay requirement. If no such $\pi^1_{s,t}$ exists, $\pi_{s,t}$ is not feasible.

The procedure becomes tedious when the set $\{\pi_{s,t}\}$ contains a large number of s→t forward flow paths because every path in the set $\{\pi_{s,t}\}$ has to be checked once to discard $\pi_{s,t}$. Also, the checking effort is proportional to the number of backward branches in $\pi_{s,t}$.

Consider the following $\pi_{s,t}$ that contains two backward branches b_1 and b_2. The first step is to find a subset $\{\pi_{s,t}\}_1$ of $\{\pi_{s,t}\}$, a member $\pi^1_{s,t}$ of which contains b_1, $\pi^1_{s,t} = \pi^1_{s,i} \cup b_1 \cup \pi^1_{j,t}$, such that $\pi_{s,j} \cup \pi^1_{j,t}$ satisfies the delay requirement.

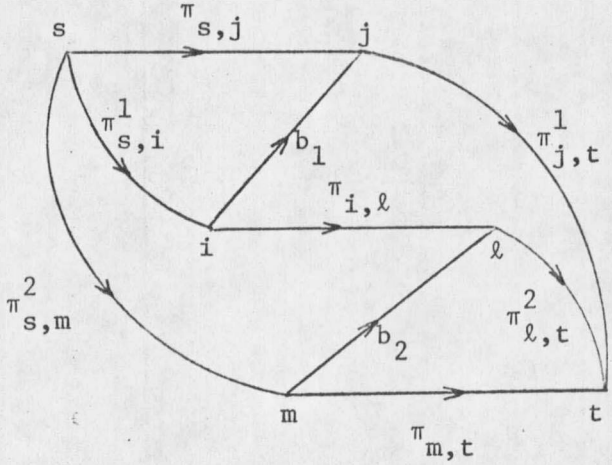


Fig. 5.10. A $\pi_{s,t}$ with two backward branches.

Then find the subset $\{\pi_{s,t}\}_2$ of $\{\pi_{s,t}\}$, a member $\pi^2_{s,t}$ of which contains b_2, $\pi^2_{s,t} = \pi^2_{s,m} \cup b_2 \cup \pi^2_{\ell,t}$, such that $\pi^1_{s,i} \cup \pi_{i,\ell} \cup \pi^2_{\ell,t}$ satisfies the delay requirement, where $\pi^1_{s,i} \subseteq \pi^1_{s,t}$ is a member of $\{\pi_{s,t}\}_1$. Finally, find the subset $\{\pi_{s,t}\}_3$ of $\{\pi_{s,t}\}_2$, a member of which is $\pi^3_{s,t}$; $\pi^3_{s,t}$ contains b_2, such that $\pi^3_{s,m} \cup \pi_{m,t}$ satisfies the delay requirement. $\pi_{s,t}$ is discarded when either $\{\pi_{s,t}\}_1$, $\{\pi_{s,t}\}_2$, or $\{\pi_{s,t}\}_3$ is null, and is said to be feasible otherwise.

A scheme to find the minimum cost s↦t flow pattern with transit-time delay not greater than a given value d_0 is described as follows.

Step 1. Set all branch flows to zero. Let $\{\pi_{s,t}\}$ denote the set of existing s↦t forward flow paths. Let $d(\pi)$ denote the delay of path π.

Step 2. Find a minimum cost s↦t augmentation path $\pi_{s,t}$. If none exists, terminate.

Step 3. Suppose $\pi_{s,t}$ contains no backward branches: If $d(\pi_{s,t}) \leq d_0$, let $\gamma = \infty$, go to Step 8; otherwise return to Step 2.

Step 4. Suppose $\pi_{s,t}$ contains b_1, b_2, ..., b_q, consecutively as backward branches which is shown in Fig. 5.11. Let $b_k = \{i_k^-, i_k^+\}$, $k = 1, 2, \ldots, q$. Let $k = 1$, $i^+_{q+1} = t$ and $i^-_0 = s$, $\{\pi_{s,t}\}_0 = \pi_{s,s}$.

Step 5. Find $\{\pi_{s,t}\}_k$ of $\{\pi_{s,t}\}$, a member $\pi^k_{s,t}$ of which contains b_k, $\pi^k_{s,t} = \pi^k_{s,i_k^-} \cup b_k \cup \pi^k_{i_k^+,t}$, such that $d(\pi^{k-1}_{s,k-1}) + d(\pi_{i^-_{k-1},i^+_k}) + d(\pi^k_{i^+_k,t}) \leq d_0$. If $\{\pi_{s,t}\}_k = \phi$, return to Step 2.

Step 6. $k = k+1$. If $k \leq q$, return to Step 5, otherwise go to Step 7.

Step 7. Let $\gamma = \underset{k}{\text{minimum}}\ f_{i_k^- i_k^+}$

Step 8. $\gamma(\pi_{s,t}) = \text{minimum}\{\gamma, \underset{\substack{\text{forward branches} \\ (i,j) \in \pi_{s,t}}}{\text{minimum}} (c_{ij} - f_{ij})\}$

Step 9. Increase $s \to t$ flow on $\pi_{s,t}$. Let the set of $s \to t$ forward flow paths associated with $\pi_{s,t}$ be new members of $\{\pi_{s,t}\}$.

Step 10. $f_{st} = f_{st} + \gamma(\pi_{s,t})$. If $f_{st} < f_{st}^*$, return to Step 2; otherwise terminate.

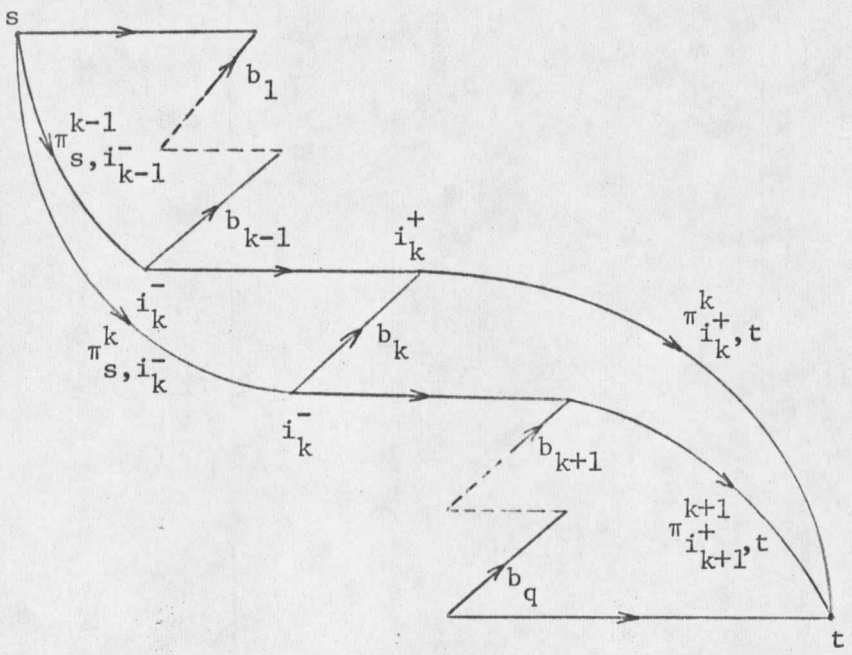


Fig. 5.11. A $\pi_{s,t}$ with q backward branches.

The above procedure is a path enumeration process and often requires a large amount of computational effort. However, if no backward branches are allowed in the minimum cost s→t augmentation path, then an efficient computational scheme exists to find a suboptimal solution to the minimum cost flow problem with transit-time delay constraint.

The scheme is to associate each branch (i,j) of the G(V,B') network with a transit-time delay weight d_{ij}; then the d_0-constrained minimal cost s→t path of G(V,B') is the minimum cost s→t augmentation forward path of G(V,B) which satisfies the delay requirement. The new flow pattern obtained by sending an additional amount of s→t flow through this path is a suboptimal solution to the problem.

5.5 The Maximum Capacity Reliability Constrained Path

This problem involves a network where each branch has an associated capacity weight and a reliability weight which may be interpreted as the probability that the branch becomes available, or unavailable, and the objective is to find the path with maximum capacity that satisfies a given path reliability requirement.

Frisch [19,49] discussed the evaluation of the optimal path by a ranking process in which the kth subnetwork that corresponds to the kth maximal capacity path is generated in a sequential manner, until

the subnetwork yields the maximal capacity path that satisfies the given reliability requirement.

Instead, an approach to find the above mentioned optimal path through the application of the constrained shortest path algorithm is suggested in the following.

Let p'_{ij} be the reliability weight of the branch (i,j) representing the probability that the branch becomes available; then the constraint that a path will become available with probability no less than a given fraction p'_0 is expressed as

$$p(\pi) = \prod_{(i,j)\in\pi} p'_{ij} \geq p'_0 \tag{5-5-1}$$

or equivalently,

$$-\ln p(\pi) = -\sum_{(i,j)\in\pi} \ln p'_{ij} \leq -\ln p'_0 \tag{5-5-2}$$

Hence, define

$$p_0 \triangleq -\ln p'_0 \tag{5-5-3}$$

and

$$p_{ij} \triangleq -\ln p'_{ij} \tag{5-5-4}$$

to be the equivalent reliability weight. The path constraint is

$$\sum_{(i,j)\in\pi} p_{ij} \leq p_0 \tag{5-5-6}$$

The maximum capacity path with reliability no less than a fraction p_0' is then the p_0-constrained maximum capacity path where the sum of the equivalent reliability weight is to be not greater than p_0.

According to the principle of optimality, the set of constrained maximum capacity 1→i paths $\pi_i(p)$ are governed by the equations

$$|\pi_i(p)| = \underset{j\neq i, p_{ji} \leq p}{\text{maximum}}\{\text{minimum}\{c_{ji}, |\pi_j(p - p_{ji})|\}\}, \text{ for } i \neq 1 \quad (5\text{-}5\text{-}7)$$

$$|\pi_1(p)| = \infty \quad (5\text{-}5\text{-}8)$$

$$|\pi_i(p)| \leq |\pi_i(p')|, \quad \text{for } p' \geq p \quad (5\text{-}5\text{-}9)$$

Equations (5-5-7) through (5-5-9) differ from (2-3-4) to (2-3-6) only in that a minimization operation appears instead of the addition in (2-3-4) and a maximization operation replaces the minimization in (2-3-4). The fact that the constrained maximum capacity path is non-decreasing is illustrated by (5-5-9). Equation (5-5-8) states the fact that the trivial 1→1 optimal path has infinite capacity.

Due to the similarity of (5-5-7) to (5-5-9), the constrained maximum capacity paths can be easily found through an appropriate modification of the constrained shortest path algorithm.

CHAPTER VI

COMPUTER PROGRAM AND COMPUTATIONAL EXPERIENCE

6.1 The Computer Program

A computer program is written to solve the constrained shortest paths problem. The computer program consists of four phases each of which is represented by a subroutine as shown in the following block diagram.

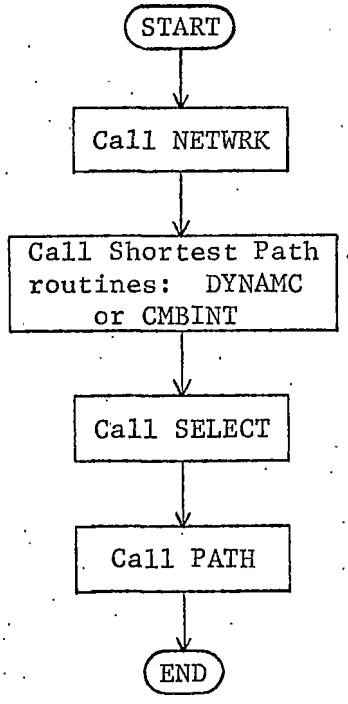


Fig. 6.1. Block diagram for the computer program.

Subroutine NETWRK handles the construction of the network. It reads in the network parameters -- the number of nodes in the network, the set of branch weights, and the optimal path criterion. It then

evaluates the set of unconstrained shortest paths which are to be used in the shortest path routines as the stepping criterion. Two shortest path routines, subroutine CMBINT and DYNAMC, are presented where CMBINT finds the constrained shortest paths in a sequential manner,† and DYNAMC employs the dynamic programming approach to find the constrained shortest paths of a network with integer-valued branch costs in an iterative fashion. Subroutine SELECT selects the appropriate stepping-cost path according to the given optimal path criterion, and subroutine PATH traces the optimal path.

Flow-charts for subroutines NETWRK, DYNAMC, CMBINT, and PATH are presented in Figs. 6.2, 6.3, 6.4, and 6.5, respectively. Fig. 6.6 contains a sample routine to find the maximum confidence ℓ_0 1→i path.

Nomenclature of symbols used in the flow charts:

n = number of nodes in the network

D = (d_{ji}) is the branch distance matrix

H = (h_{ji}) is the branch cost matrix

π_i = unconstrained shortest 1→i path

c_i^j = j^{th} 1→i stepping cost

π_i^j = j^{th} 1→i stepping-cost path

---

†CMBINT treats networks with a single branch cost weight. The constrained shortest path algorithm for networks with two branch cost weights is similar in rationale to that of CMBINT, and its program is included in Appendix A.

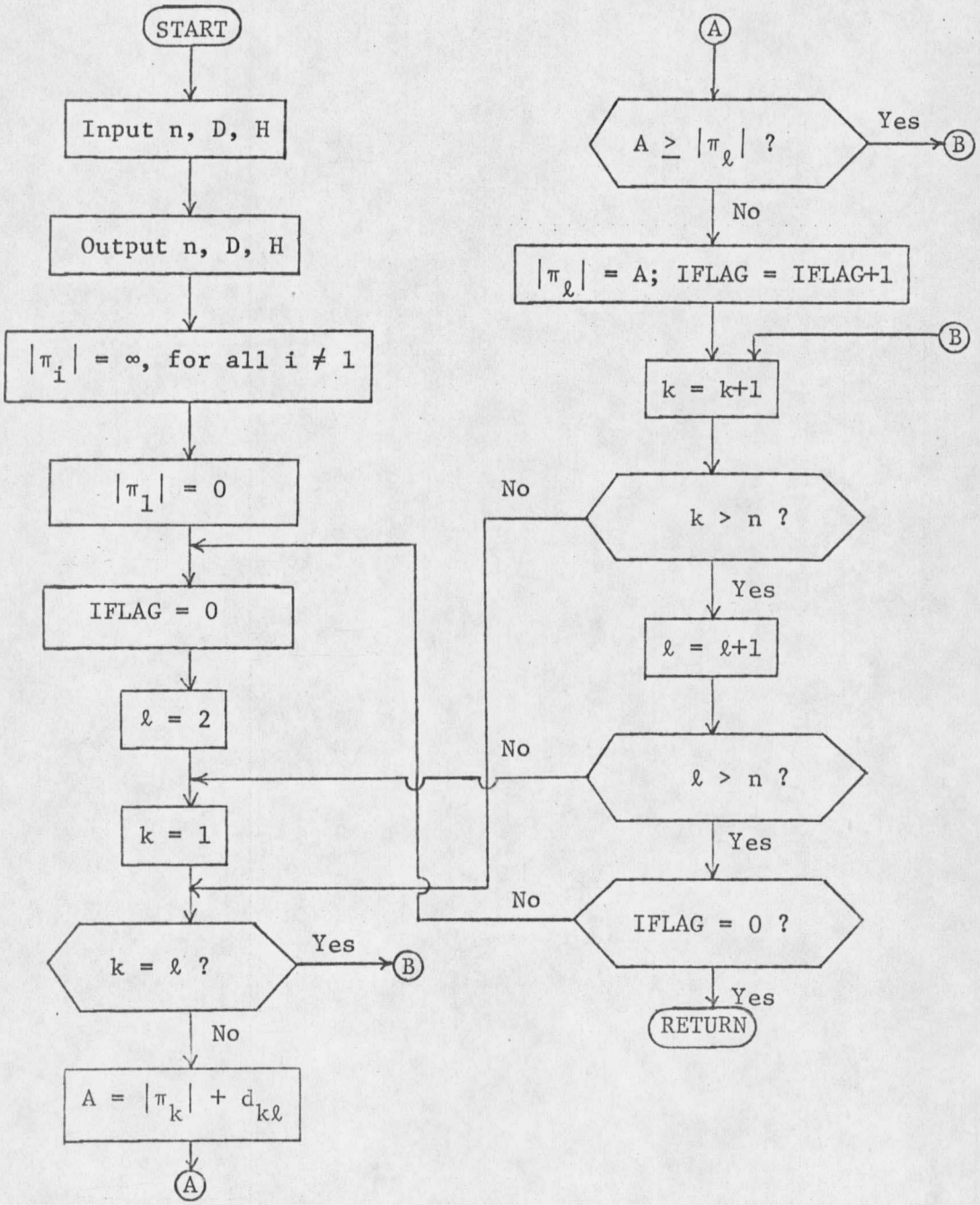


Fig. 6.2. A sample routine for NETWRK where the procedure to find $\{|\pi_i|\}$, the unconstrained shortest path lengths, is illustrated.

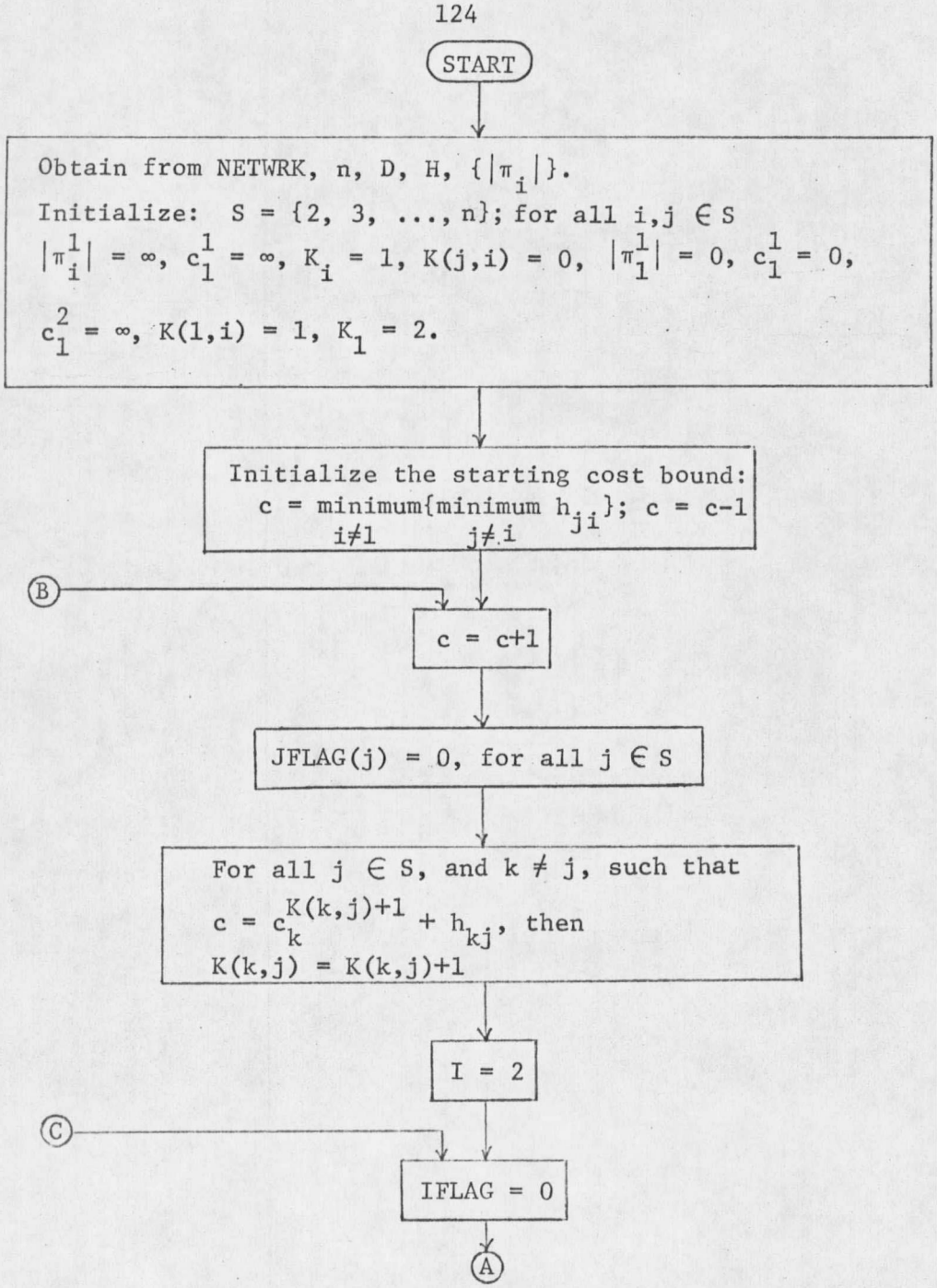


Fig. 6.3. Flow chart for DYNAMC.

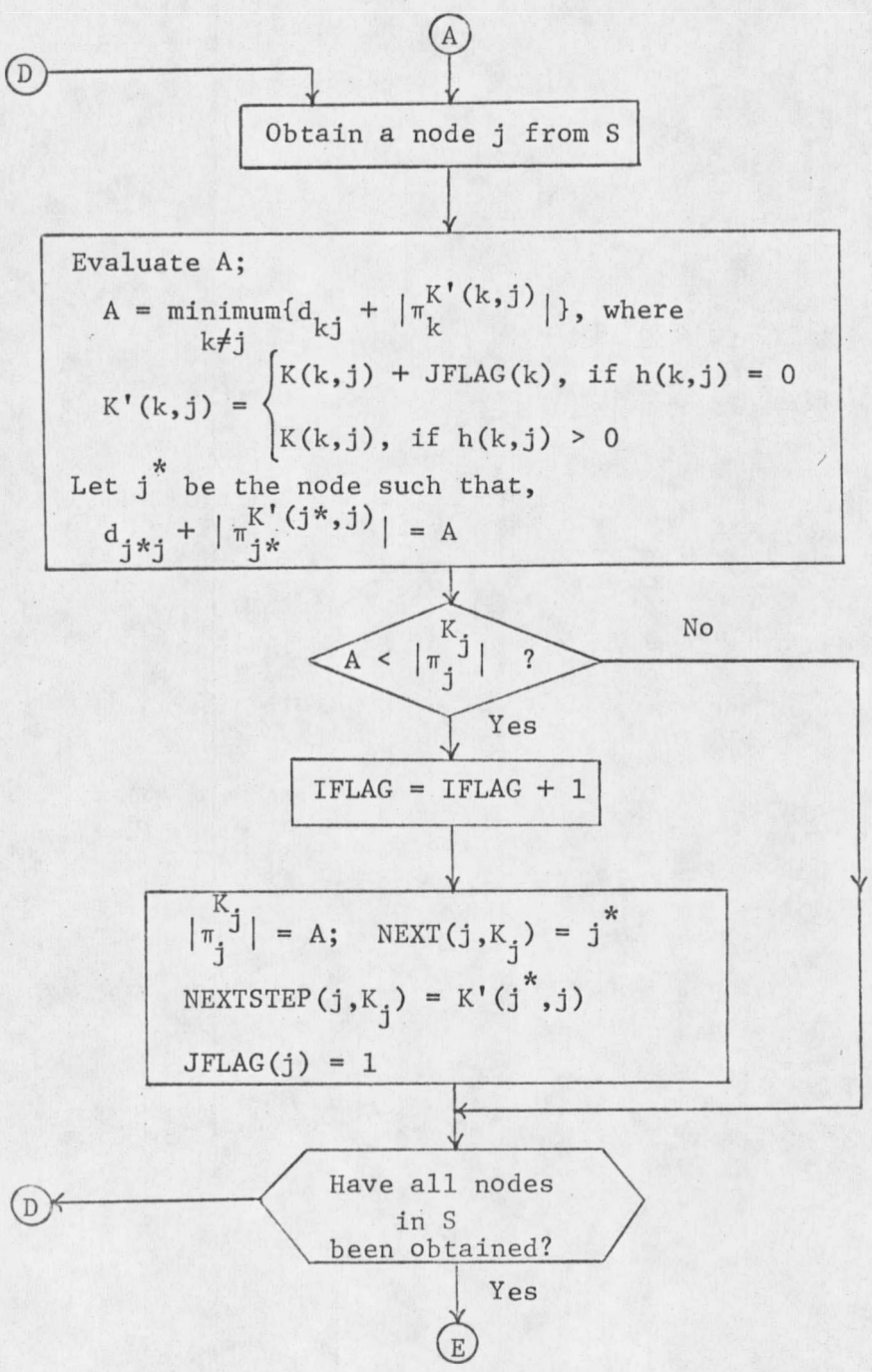


Fig. 6.3. (continued).

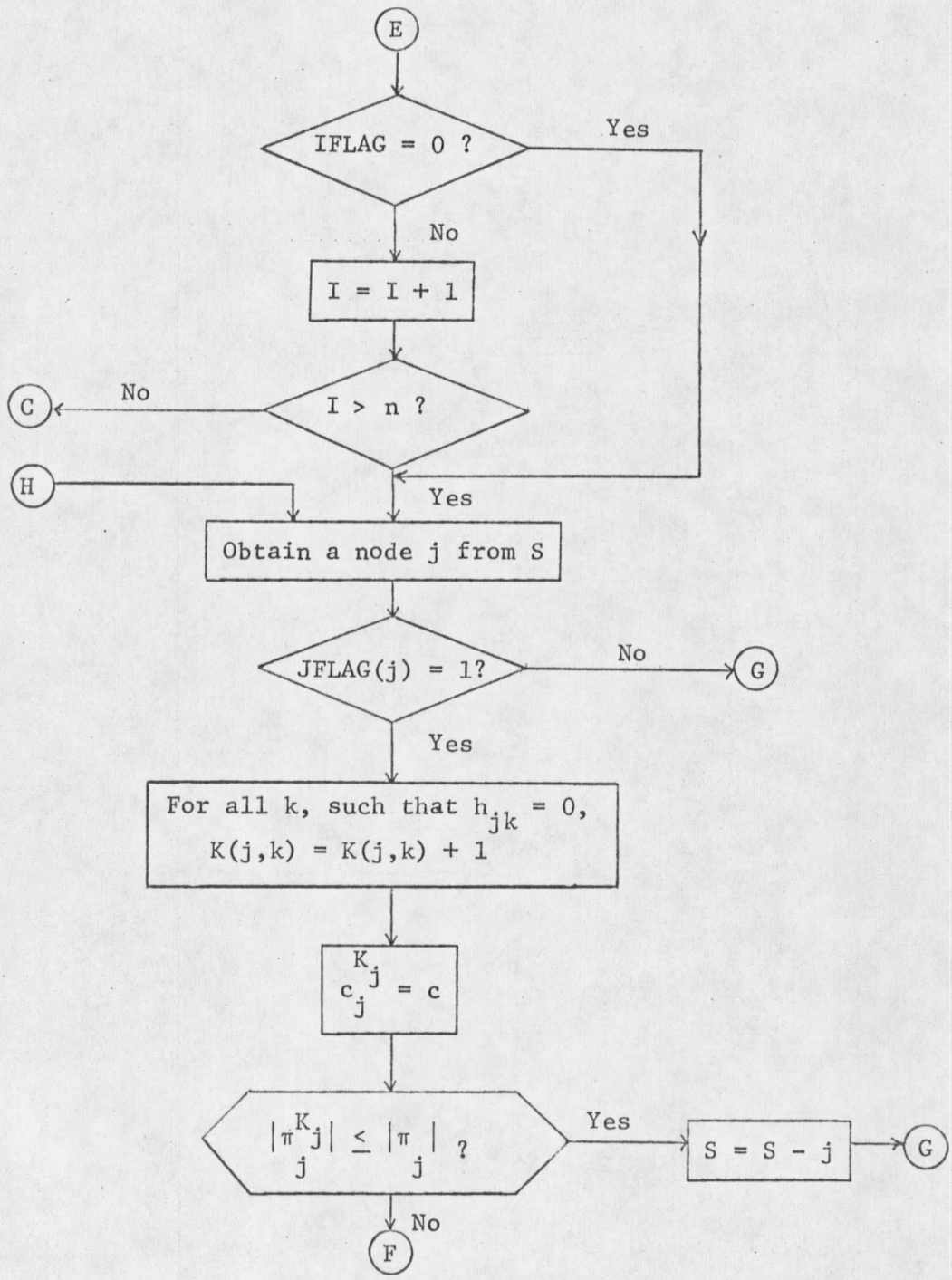


Fig. 6.3. (continued).

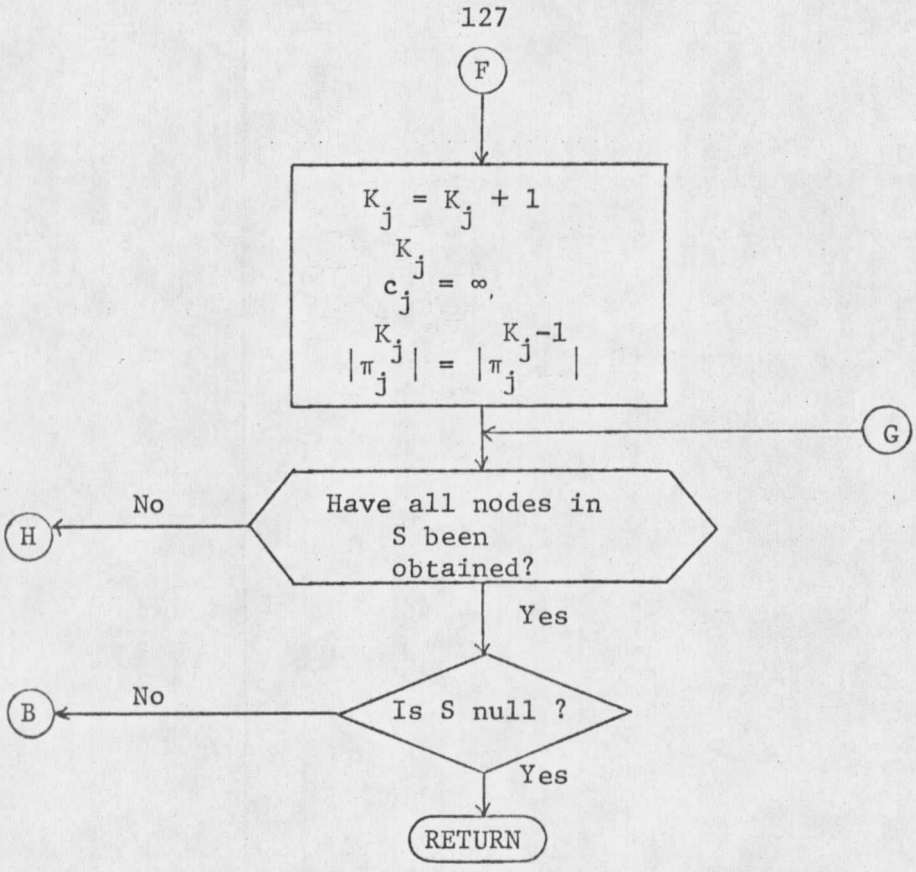


Fig. 6.3. (continued).

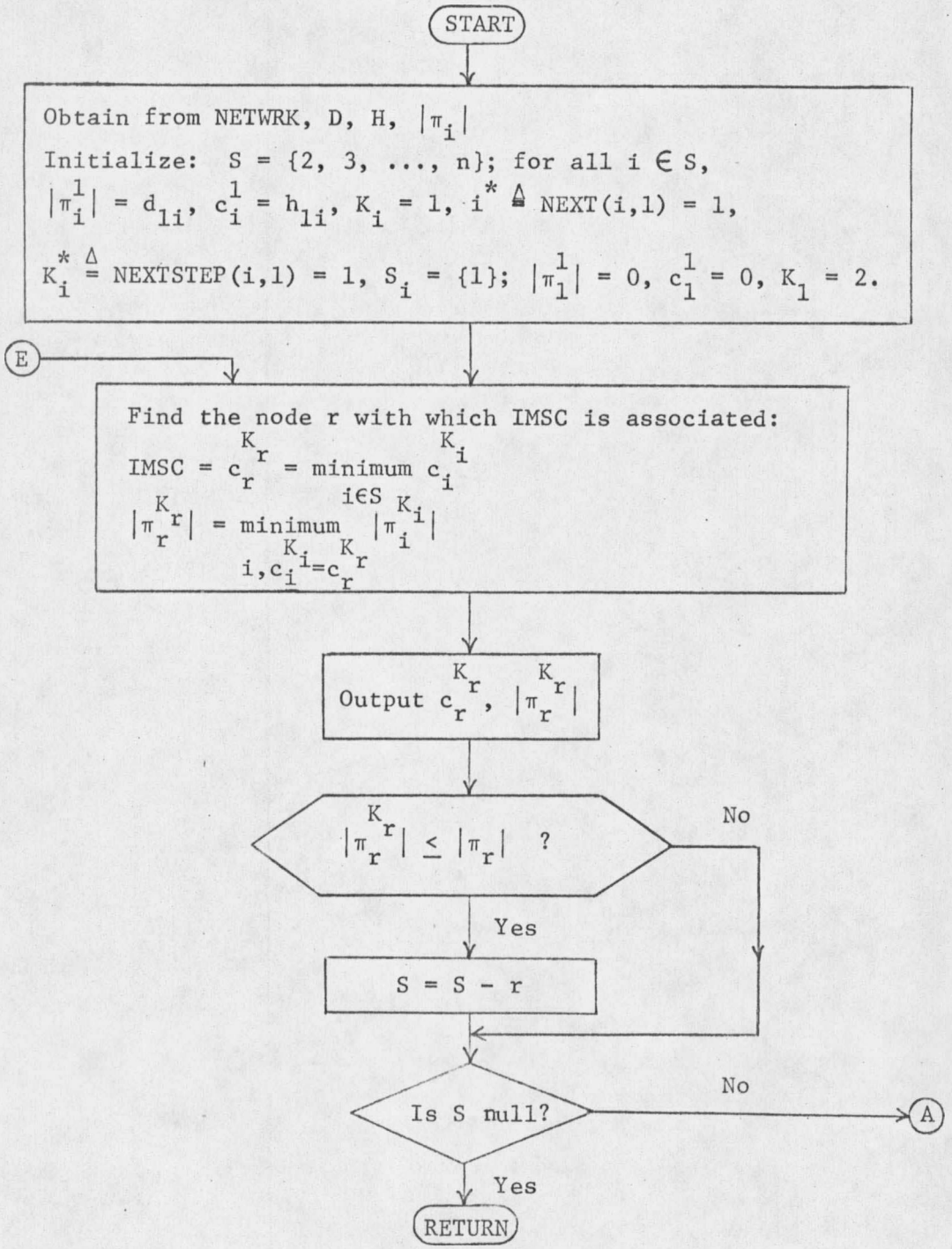


Fig. 6.4. Flow chart for CMBINT.

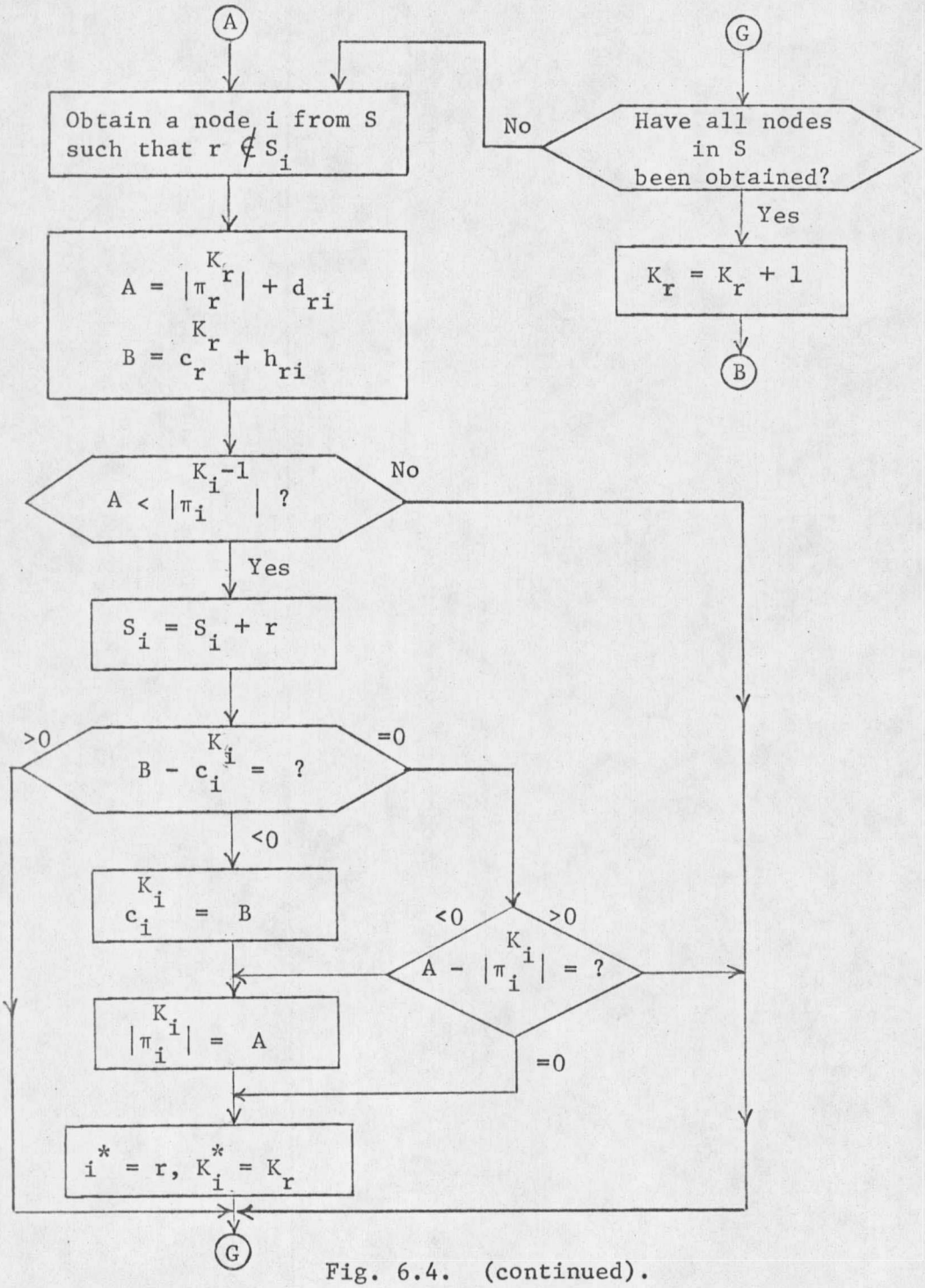


Fig. 6.4. (continued).

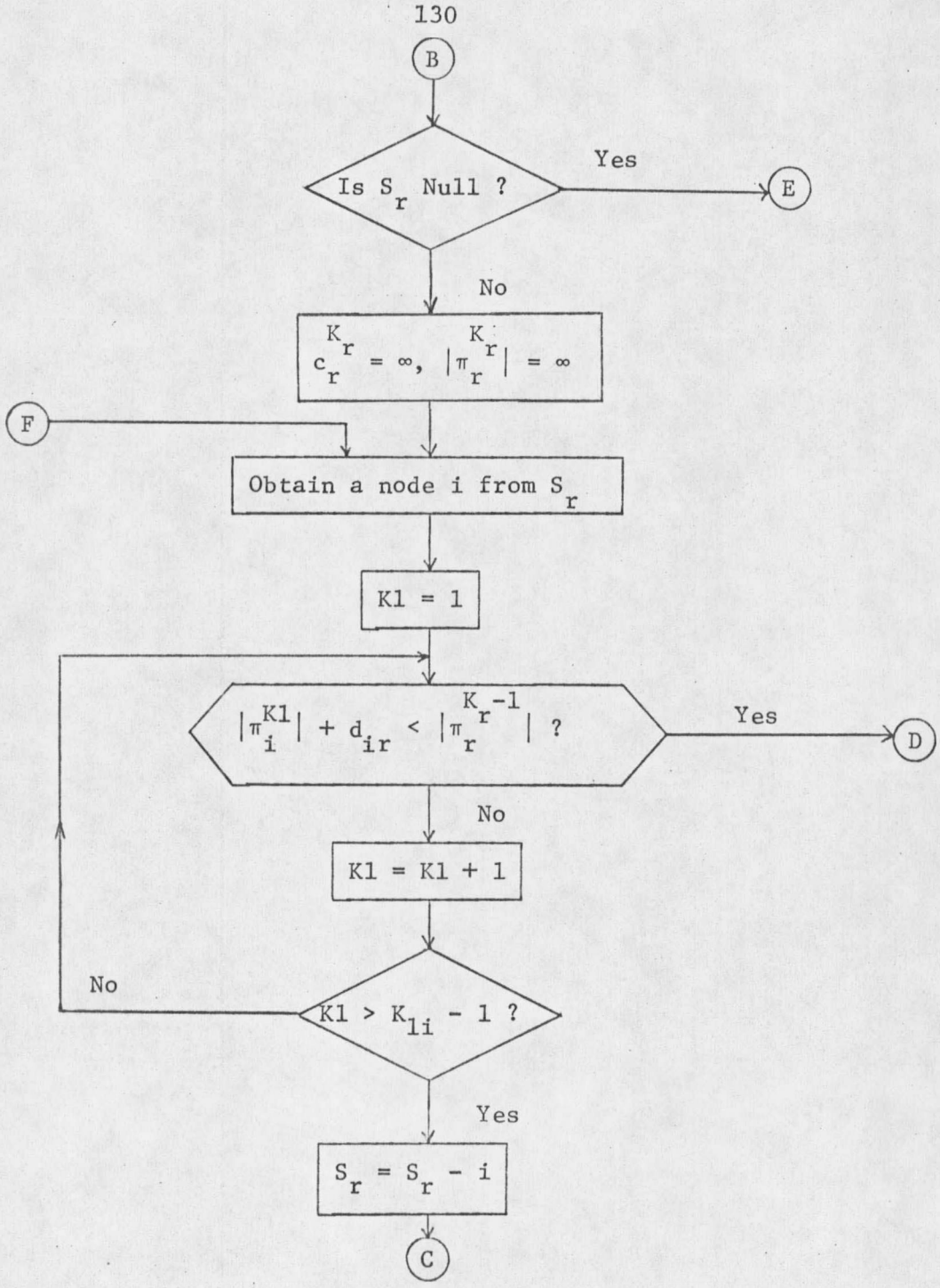


Fig. 6.4. (continued).

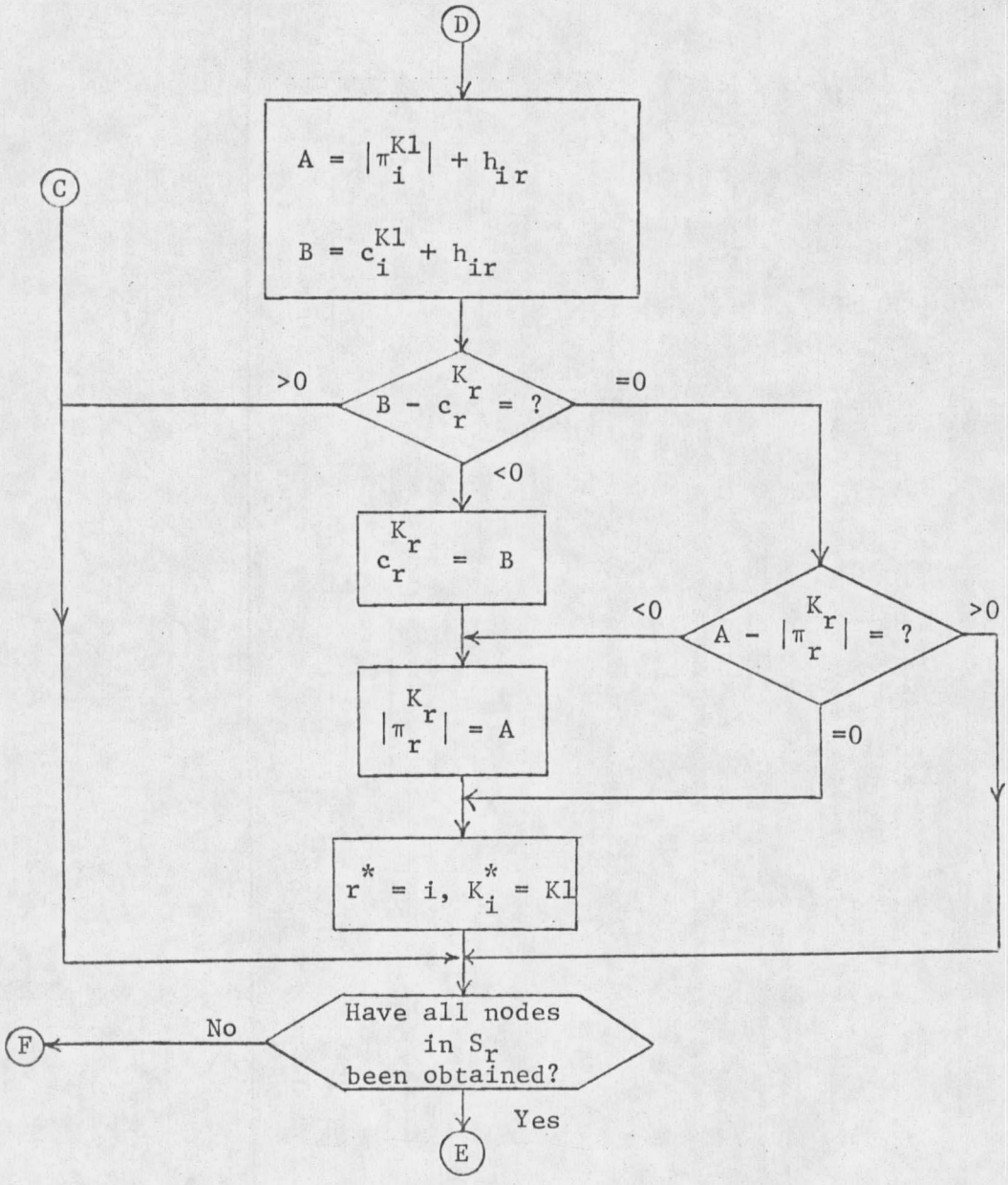


Fig. 6.4. (continued).

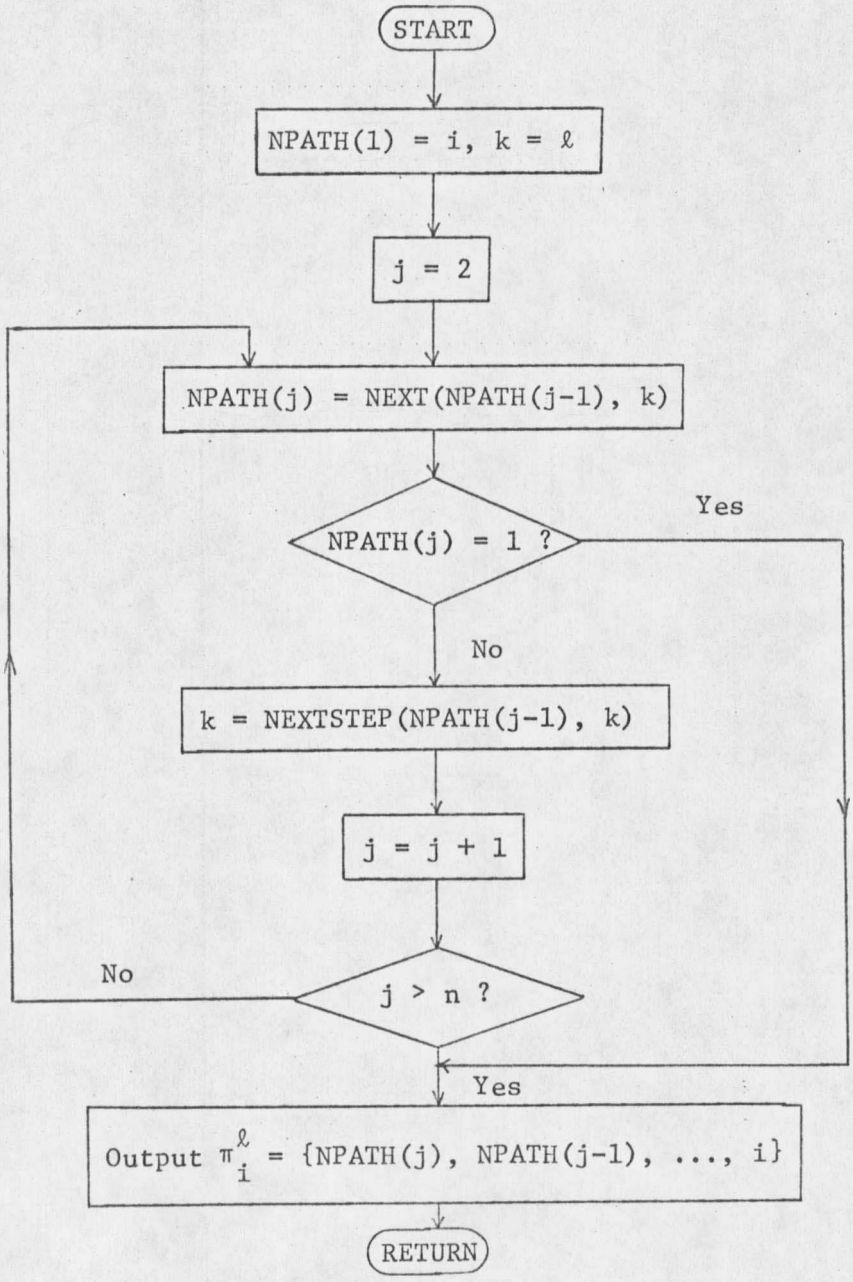


Fig. 6.5. Flow chart for PATH.

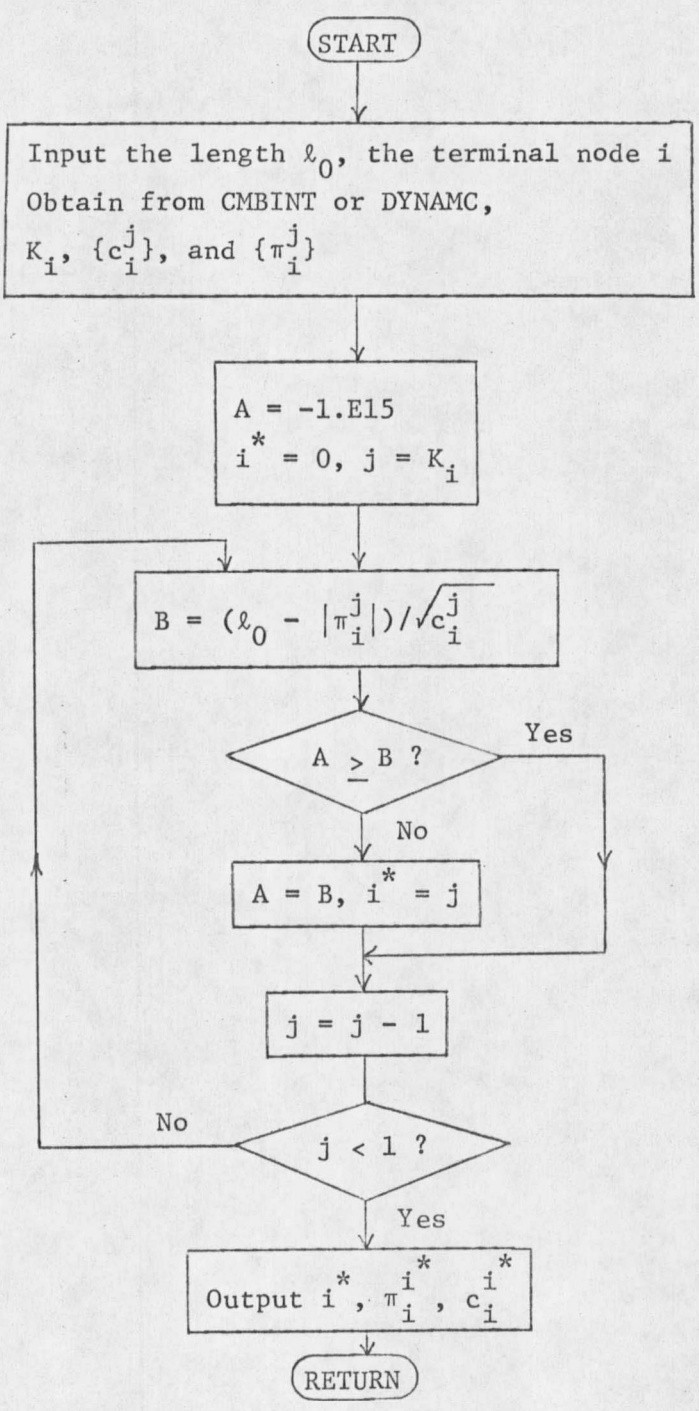


Fig. 6.6. A sample routine for Subroutine SELECT where the maximum confidence level ℓ_0 1→i path is sought.

$\pi_i^{K_i}$ = K_i^{th} $1 \to i$ stepping-cost path to be found

$K(j,i)$ is such that $\pi_j^{K(j,i)} \cup (j,i)$ is the candidate K_i^{th} $1 \to i$ stepping-cost path

$j^* \triangleq \text{NEXT}(i,K_i)$ and $K_j^* \triangleq \text{NEXTSTEP}(i,K_i) = K(j^*,i)$ are such that $\pi_i^{K_i} = \pi_{j^*}^{K_j^*} \cup (j,i)$ is the K_i^{th} $1 \to i$ stepping-cost path

$S_i = \{k$, such that there is a smallest integer $K(k,i) \leq K_k$ such that $|\pi_k^{K(k,i)}| + d_{ki} < |\pi_i^{K_i-1}|\}$

6.2 A Maximum Capacity Reliability Constrained Path Problem

The maximum capacity $1 \to 5$ path is to be found for the following 5-node network with reliability of path not smaller than .20. The joint branch weight of (i,j) is $W_{ji} = (c_{ji}, p'_{ji})$.

The equivalent reliability weights p_{ji}'s are defined as,

$$p_{ji} = -\ln p'_{ji}$$

So that the equivalent path reliability constraint p_0 is

$$p_0 = -\ln(.20) = 1.6$$

The network weights are listed in Table 6.1.

The values p_{ji}'s are rounded to integral multiples of .1, as shown in Table 6.1. Thus the dynamic programming solution approach is feasible.

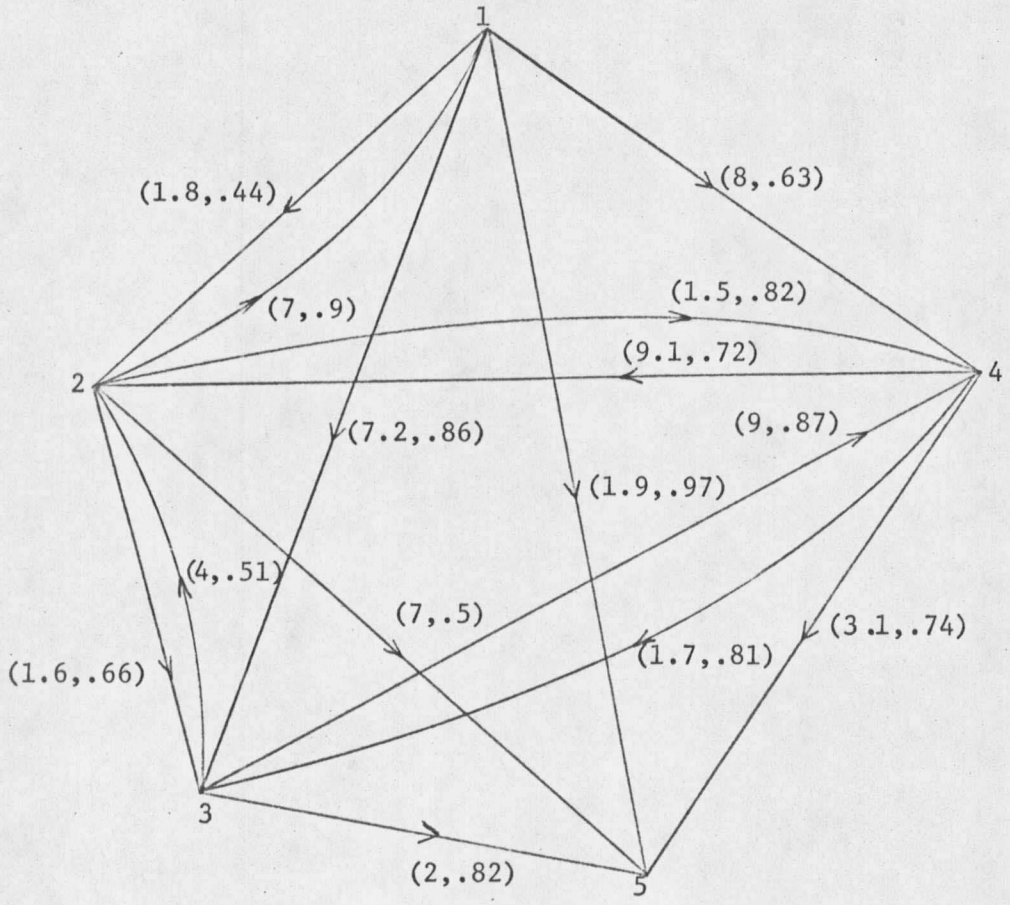


Fig. 6.7. A 5-node network with $W_{ji} = (c_{ji}, p'_{ji})$.

Table 6.1. The Branch Weights of the Network in Fig. 6.7.

W_{ji} / (j,i)	c_{ji}	p'_{ji}	p_{ji}	$\frac{1}{10}[\frac{p_{ji}}{.1} + .5]$
(1,2)	1.8	.44	.82	.8
(1,3)	7.2	.86	.15	.2
(1,4)	8.0	.63	.46	.5
(1,5)	1.9	.97	.03	.0
(2,1)	7.0	.90	.11	.1
(2,3)	1.6	.66	.42	.4
(2,4)	1.5	.82	.20	.2
(2,5)	7.0	.50	.69	.7
(3,2)	4.0	.51	.67	.7
(3,4)	9.0	.87	.14	.1
(3,5)	2.0	.82	.20	.2
(4,2)	9.1	.72	.33	.3
(4,3)	1.7	.81	.21	.2
(4,5)	3.1	.74	.30	.3

The set of 1→5 paths and the associated path capacities and the rounded path equivalent reliabilities are listed in Table 6.2.

The computational result of the ℓth 1→i stepping-reliability path with capacity, $|\pi_i^{\ell}|$, equivalent reliability (rounded), $c(\pi_i)$, last intermediate node, k, and integer K1, where the $K1^{th}$ 1→k stepping-reliability path forms the path π_i^{ℓ}, are listed in Table 6.3.

Since the equivalent path reliability constraint is $p_0 = 1.6$, the optimal 1→5 path, from Table 6.3, is the stepping-reliability path with equivalent reliability = 1.3. The capacity of the optimal path is 7.0.

To trace the optimal path, we have k = 2 as the last intermediate node with K1 = 1, so π_2^1 is the 1→2 path. The next to last intermediate node is 4 with K1 = 1, and so on. The optimal path is 1-3-4-2-5.

6.3 A Min-Cost Flow Problem With Transit-Time Delay

The following 5-node network represents a communication network where each branch between two nodes is a communication channel with a known capacity. Suppose the message time to traverse a branch is known and given as a constant branch parameter. The network is to be analyzed in terms of message transmission cost. Given an initial-terminal node pair, say (1,5), and .6 units of message requirement between the two nodes, what is the minimum cost flow pattern of sending the message through the network so that the message will arrive at the terminal node within a given time period, say 14 units of time?

Table 6.2. The Path Capacities and Reliabilities

i	π_i	Path Capacity	Path Equivalent Reliability (Rounded to Integral Multiple of .1)
1	(1,5)	1.9	0.0
2	(1,2,5)	1.8	1.5
3	(1,3,5)	2.0	.4
4	(1,4,5)	3.1	.8
5	(1,2,3,5)	1.6	1.4
6	(1,2,4,5)	1.5	1.3
7	(1,3,2,5)	4.0	1.6
8	(1,3,4,5)	3.1	.6
9	(1,4,2,5)	7.0	1.5
10	(1,4,3,5)	1.7	.9
11	(1,2,3,4,5)	1.6	1.6
12	(1,2,4,3,5)	1.5	1.4
13	(1,3,2,4,5)	1.5	1.4
14	(1,3,4,2,5)	7.0	1.3
15	(1,4,2,3,5)	1.6	1.4
16	(1,4,3,2,5)	1.7	2.1

Table 6.3. Computational Results.

i	$c(\pi_i^\ell)$	$\lvert\pi_i^\ell\rvert$	ℓ	k	Kl
5	.0	1.9	1	1	1
3	.2	7.2	1	1	1
4	.3	7.2	1	3	1
5	.4	2.0	2	3	1
4	.5	8.0	2	1	1
2	.6	7.2	1	4	1
5	.6	3.1	3	4	1
2	.8	8.0	2	4	2
5	1.3	7.0	4	2	1

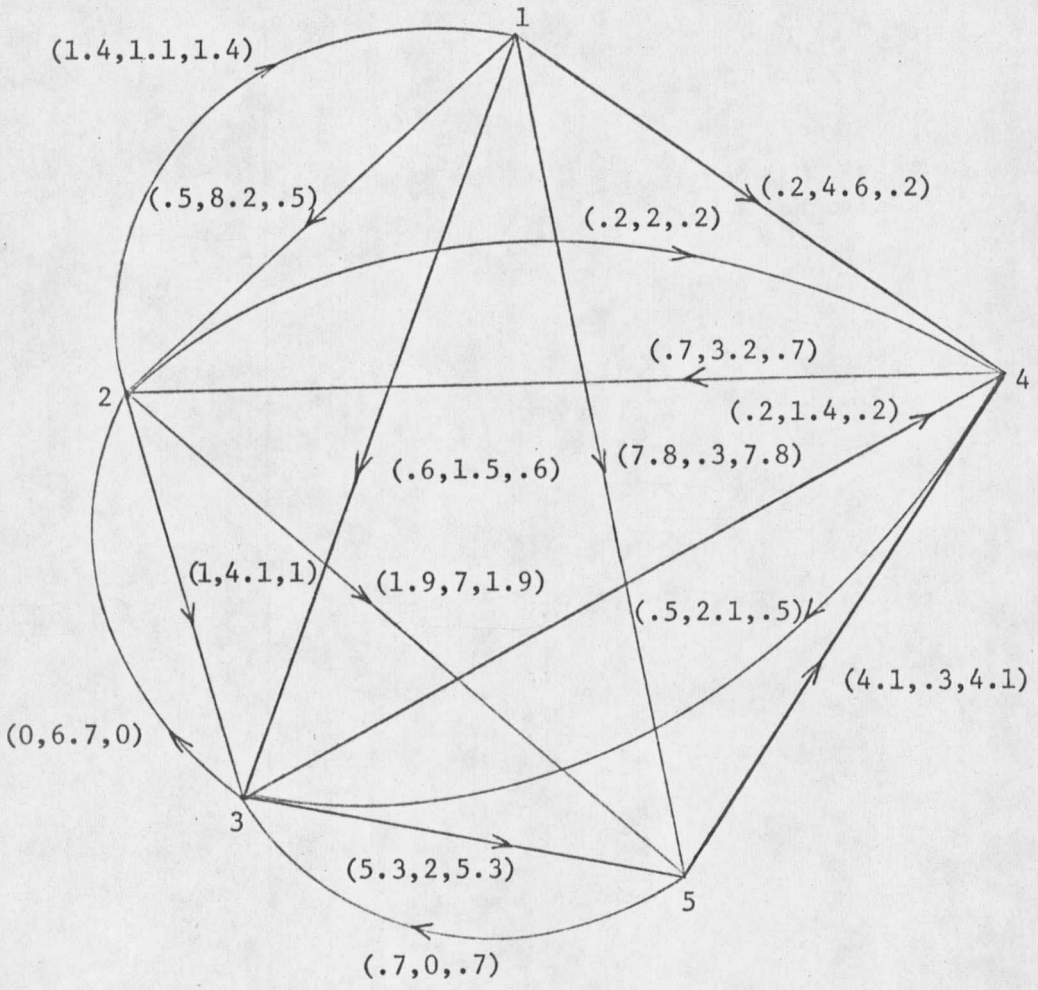


Fig. 6.8. A 5-node network with $W_{ji} = (h_{ji}, d_{ji}, c_{ji})$.

The problem is solved by applying the constrained shortest path algorithm to find the minimum cost 1→5 path with time delay of no more than 14. And then the message is sent through this path in the amount of the path residual capacity. The saturated branches are deleted from the network, and further minimum cost 1→5 paths are located sequentially, until the 1→5 flow requirement is satisfied or no more 1→5 paths exist.

The 1→i stepping-delay paths are shown in Table 6.4, where $\pi_i^{\ell} = \pi_k^{Kl} \cup (k,i)$, and $d(\pi_i^{\ell})$ represents the time delay of the ℓth 1→i stepping-delay path. The optimal 1→5 path is obviously the fifth stepping-delay path. The path is traced as follows:

$$\pi_5^5 = \pi_2^2 \cup (2,5) = \pi_4^2 \cup (4,2) \cup (2,5) = \pi_3^2 \cup (3,4) \cup (,42)$$

$$\cup (2,5) = \{(1,3),(3,4),(4,2),(2,5)\}$$

The path residual capacity is found to be .2 and that amount of flow is sent through the optimal path. Since $c_{34} = .2$, (3,4) is saturated and deleted from further consideration.

The set of 1→i stepping-delay paths are listed in Table 6.5. Note that the third stepping-delay path no longer appears in Table 6.4, since the path is {(1,3),(3,4),(4,5)}, where (3,4) is saturated.

Table 6.4. The Stepping Delays of the Network in Fig. 6.8.

i	$d(\pi_i^{\ell})$	$\lvert\pi_i^{\ell}\rvert$	ℓ	k	Kl
5	.3	7.8	1	1	1
3	.4	8.5	1	5	1
3	1.5	.6	2	1	1
5	1.7	5.9	2	3	2
4	1.8	8.7	1	3	1
4	2.9	.8	2	3	2
5	3.2	4.9	3	4	2
4	4.6	.2	3	1	1
5	4.9	4.3	4	4	3
2	5.0	9.4	1	4	1
2	6.1	1.5	2	4	2
2	7.8	.9	3	4	3
2	8.2	.5	4	1	1
5	13.1	3.4	5	2	2
5	14.8	2.8	6	2	3
5	15.2	2.4	7	2	4

Table 6.5. The Stepping Delays of the Network in Fig. 6.8 with (3,4) Deleted.

i	$d(\pi_i^{\ell})$	$\lvert\pi_i^{\ell}\rvert$	ℓ	k	K1
5	.3	7.8	1	1	1
3	.4	8.5	1	5	1
3	1.5	.6	2	1	1
5	1.7	5.9	2	3	2
4	4.6	.2	1	1	1
5	4.9	4.3	3	4	1
2	7.1	8.5	1	3	1
2	7.8	.9	2	4	1
2	8.2	.5	3	1	1
5	14.8	2.8	4	2	2
5	15.2	2.4	5	2	3

The optimal 1→5 path is the third stepping-delay path in Table 6.5.

$$\pi_5^3 = \pi_4^1 \cup (4,5) = (1,4) \cup (4,5)$$

The path residual capacity is found to be .2, and that amount of flow is again sent through the optimal path with (1,4) saturated.

The set of 1→i stepping-delay paths are again found and listed in Table 6.6.

Table 6.6. The Stepping Delays of the Network of Fig. 6.8 with (1,4), (3,4) Deleted.

i	$d(\pi_i^\ell)$	$\lvert\pi_i^\ell\rvert$	ℓ	k	Kl
5	.3	7.8	1	1	1
3	.4	8.5	1	5	1
3	1.5	.6	2	1	1
5	1.7	5.9	2	3	2
2	7.1	8.5	1	3	1
2	8.2	.5	2	1	1
4	9.1	8.7	1	2	1
4	10.2	.7	2	2	2
5	10.5	4.8	3	4	2
5	15.2	2.4	4	2	2

The optimal 1→5 path is the third stepping-delay path in Table 6.6.

$$\pi_5^3 = \pi_4^2 \cup (4,5) = \pi_2^2 \cup (2,4) \cup (4,5) = (1,2) \cup (2,4) \cup (4,5)$$

The path residual capacity is again .2, and that amount of flow is again sent through the optimal path.

Since .6 units of 1→5 flow have been sent, the flow requirement is met. And the optimal flow pattern is shown in the following figure.

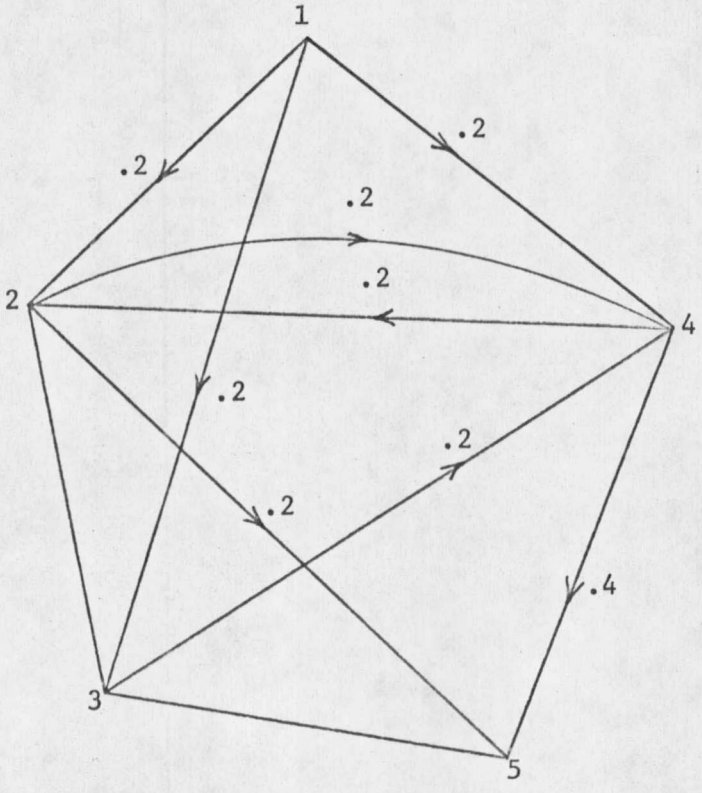


Fig. 6.9. Final flow pattern.

6.4 A Message Transmission Problem

Consider a 5-node communication network model shown in Fig. 6.10, for which an optimal message transmission path is to be found from node 1 to node 5.

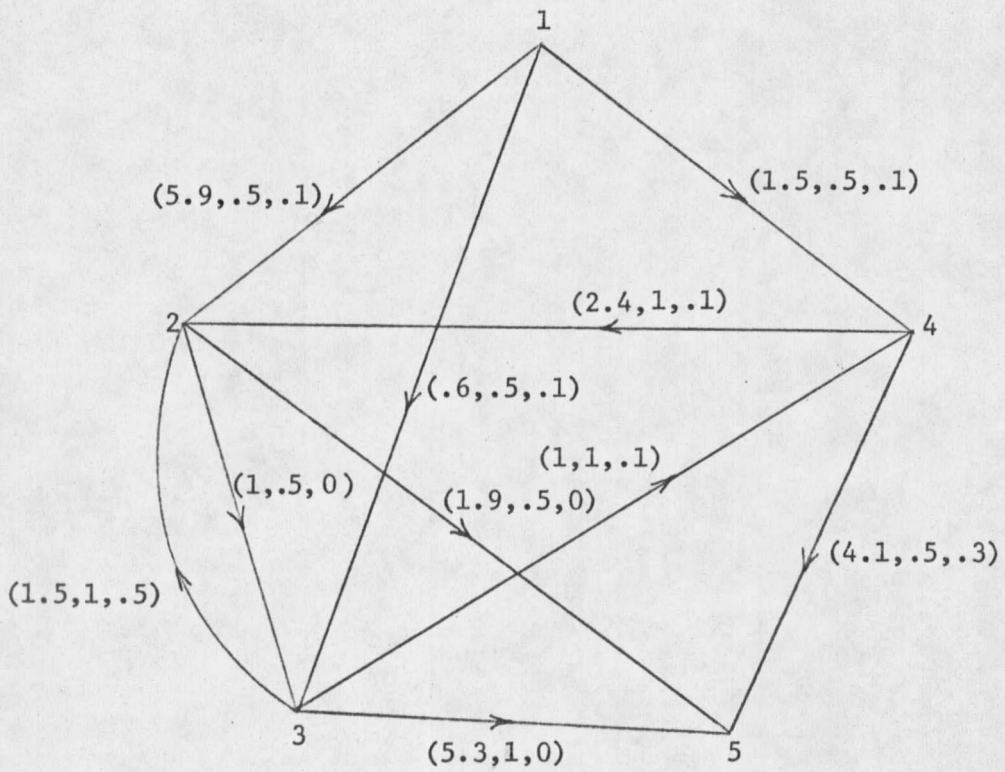


Fig. 6.10. A 5-node communication network model with $W_{ji} = (h_{ji}, \mu_{ji}, \sigma^2_{ji})$.

In Fig. 6.10, W_{ji} represents the joint branch weight of (j,i), where h_{ji} is the cost of transmitting a message through the branch, and μ_{ji}, σ^2_{ji} are, respectively, the mean and variance of the message traverse time through the branch. The set of branch traverse time weights are assumed to be stochastically independent and normally distributed.

The optimal transmission path is defined to be the minimal cost path such that there exists a 95 percent confidence level that the path traverse time is within 2.5 units of time.

Let π be the optimal path with path cost $|\pi|$, the traverse time $d(\pi)$, the mean of $d(\pi)$, $\mu(\pi)$, and the variance of $d(\pi)$, $\sigma^2(\pi)$. The probabilistic constraint is

$$\text{Prob}[d(\pi) \geq 2.5] \leq .05 \qquad (6\text{-}4\text{-}1)$$

or equivalently,

$$1 - N[(2.5 - \mu(\pi))/\sigma(\pi)] \leq .05 \qquad (6\text{-}4\text{-}2)$$

or

$$(2.5 - \mu(\pi))/\sigma(\pi) \geq N^{-1}(.95) \approx 1.65 \qquad (6\text{-}4\text{-}3)$$

where, similar to (4-3-1-1) and (4-3-1-2), $N(\cdot)$ is the cumulative distribution function of a zero-mean unit-variance normal random variable; $N^{-1}(\cdot)$ is the inverse function.

The optimal path π is then the minimal cost path with its mean and variance of the path traverse time satisfying the deterministic inequality (6-4-3).

The optimal path π can be found using the constrained shortest path algorithm in Section 3.6, by letting the mean and variance weights, respectively, represent the type 1 and type 2 cost weights, and letting the unit cost of message transmission represent the branch length. Let $\{\pi_5^{j,k}\}$ denote the set of 1→5 stepping-cost paths, with type 1 costs $\{\mu_5^j\}$ and type 2 costs $\{(\sigma_5^2)^{j,k}\}$. Thus $|\pi|$ is

$$|\pi| = \underset{j,k,\text{ such that } (2.5-\mu_5^j)/\sigma_5^{j,k} \geq 1.65}{\text{minimum}} \{|\pi_5^{j,k}|\} \qquad (6\text{-}4\text{-}4)$$

where $\sigma_i^{j,k} = \sqrt{(\sigma_i^2)^{j,k}}$ is the standard deviation of the kth 1→i stepping-cost path $\pi_i^{j,k}$ associated with μ_i^j, the jth type 1 1→i stepping cost.

The reason that π is a 1→5 stepping-cost path is as follows: If π is the shortest 1→5 path with mean $\mu(\pi)$ and variance $\sigma^2(\pi)$ of the path traverse time that satisfies (6-4-3), and if $(\mu(\pi), \sigma^2(\pi))$ is not a 1→5 stepping cost, then, according to the definition of stepping cost in Section 3.9, there exists a μ and σ^2 such that (μ, σ^2) is a 1→5 stepping cost with a path length no longer than $|\pi|$, and that $\mu \leq \mu(\pi)$, $\sigma^2 \leq \sigma^2(\pi)$, and that when replacing $\mu(\pi)$ and $\sigma^2(\pi)$, inequality (6-4-3) is still satisfied. Hence π is a 1→5 stepping-cost path, and equation (6-4-4) is justified.

The set of 1→5 paths are listed in Table 6.7 for reference. The computed set of 1→i stepping-cost paths are listed in Table 6.8, where i^* is the last intermediate node for $\pi_i^{j,k}$; and j^*, k^* are such that

$$\pi_i^{j,k} = \pi_{i*}^{j*,k*} \cup (i^*,i)$$

$$\mu_i^j = \mu_{i*}^{j*} + \mu_{i*i} \qquad (6\text{-}4\text{-}6)$$

and

$$(\sigma_i^2)^{j,k} = (\sigma_{i*}^2)^{j*,k*} + \sigma_{i*i}^2 \qquad (6\text{-}4\text{-}7)$$

The 1→5 stepping-cost paths that satisfy the inequality (6-4-3) are $\pi_5^{1,1}$, $\pi_5^{1,2}$, and $\pi_5^{2,1}$. Hence the optimal path is $\pi_5^{1,2}$, with mean of traverse time 1.0, variance .4, and path cost 5.6. From Table 6.8, $\pi_5^{1,2}$ is traced as follows:

$$\pi_5^{1,2} = \pi_4^{1,1} \cup (4,5) = (1,4) \cup (4,5) \qquad (6\text{-}4\text{-}8)$$

Table 6.7. The Set of 1→5 Paths.

π	$\lvert\pi\rvert$	$\mu(\pi)$	$\sigma^2(\pi)$	Prob[$d(\pi) \geq 2.5$]
1-2-5	7.8	1.0	.1	$1 - N[1.5/\sqrt{.1}] \approx 0.0$
1-3-5	5.9	1.5	.1	$1 - N[1.0/\sqrt{.1}] \approx 0.0$
1-4-5	5.6	1.0	.4	$1 - N[1.5/\sqrt{.4}] \approx .01$
1-2-3-5	12.2	2.0	.1	$1 - N[0.5/\sqrt{.1}] \approx .06$
1-3-2-5	4.0	2.0	.6	$1 - N[0.5/\sqrt{.6}] \approx .26$
1-3-4-5	5.7	2.0	.5	$1 - N[0.5/\sqrt{.5}] \approx .24$
1-4-2-5	5.8	2.0	.2	$1 - N[0.5/\sqrt{.2}] \approx .13$
1-2-3-4-5	12.0	2.5	.5	$1 - N[0.0/\sqrt{.5}] = .50$
1-3-4-2-5	5.9	3.0	.3	$1 - N[-.5/\sqrt{.3}] \approx .82$
1-4-2-3-5	10.7	3.0	.2	$1 - N[-.5/\sqrt{.2}] \approx .87$

Table 6.8. The Set of 1→i Stepping-Cost Paths.

i	j	k	μ_i^{j}	$(\sigma_i^2)^{j,k}$	$\lvert\pi_i^{j,k}\rvert$	i^*	j^*	k^*	Prob[$d(\pi_i^{j,k}) \geq 2.5$]
3	1	1	.5	.1	.6	1	1	1	$1 - N[2.0/\sqrt{.1}] \approx 0.0$
4	1	1	.5	.1	1.5	1	1	1	$1 - N[2.0/\sqrt{.1}] \approx 0.0$
2	1	1	.5	.1	5.9	1	1	1	$1 - N[2.0/\sqrt{.1}] \approx 0.0$
5	1	1	1.0	.1	7.8	2	1	1	$1 - N[1.5/\sqrt{.1}] \approx 0.0$
5	1	2	1.0	.4	5.6	4	1	1	$1 - N[1.5/\sqrt{.4}] \approx .01$
5	2	1	1.5	.1	5.9	3	1	1	$1 - N[1.0/\sqrt{.1}] \approx 0.0$
2	2	1	1.5	.2	3.9	4	1	1	$1 - N[1.0/\sqrt{.2}] \approx .01$
2	2	2	1.5	.6	2.1	3	1	1	$1 - N[1.0/\sqrt{.6}] \approx .20$
5	3	1	2.0	.6	4.0	2	2	2	$1 - N[0.5/\sqrt{.6}] \approx .26$

CHAPTER VII

CONCLUSION AND FUTURE RESEARCH SUGGESTIONS

7.1 Conclusion

The general shortest path problem is studied and extended by considering an additional nonnegative cost weight associated with each branch of the network model. A cost constraint is thus imposed on the shortest path, and shortest paths that satisfy a cost constraint are sought.

The problem solution is first formulated through a dynamic programming approach where the problem is solved as a multistage decision process by incrementing the cost bound variable as the stage increases. The approach is easy to implement. However, only suboptimal shortest paths are attainable when the cost weights are of real values. A combinatorial approach is then evolved in an attempt to accelerate the decision process by advancing the cost bound to the stepping costs -- costs at which a step change occurs in the associated path lengths. A reduction in computational effort is obtained where the optimal paths can be found with comparison operations as well as addition operations being in the order of the square of the number of stepping costs.

The approach is extended to include networks with more than one branch cost weight. A combinatorial algorithm for networks with two branch cost weights is presented.

Several optimal paths are discussed in a probabilistic weighted network. When probabilistic constraints are transformed into the deterministic equivalent, the deterministic optimal path can often be found by applying the constrained shortest path algorithm.

Problems in flow networks are studied where a time-delay constraint is imposed on the network flow. A suboptimal flow pattern is achieved which satisfies a given node pair flow requirement; flows are shipped from the supply node to the demand node in minimal cost while the delay of each flow path is no larger than a given transmission time. The problem of finding the maximal capacity path in a network with possible branch failure is solved through a modified form of the constrained shortest path algorithm. Several examples of flow networks and probabilistic weighted networks are solved and discussed.

7.2 Future Research Suggestions

Two constrained shortest path algorithms employing the combinatorial approaches are written and tested by solving practical examples on networks with respectively one and two branch cost weights. Theoretically, the combinatorial approach may be extended to include networks with multiple branch cost weights at the expense of a more complicated solution procedure. It seems reasonable to conjecture that the computational effort for finding the constrained shortest paths in networks with a multiple number of branch cost weights will be of the

order of the square of the number of stepping costs involved. Because of time limitations, the author leaves this for future research endeavors.

The computational effort of the constrained shortest paths in networks with up to two branch cost weights has been found to be in the order of the square of the number of stepping costs. However, no effort has been undertaken to investigate the relation between the order of the network versus the number of stepping costs in order to establish an upper bound in terms of the network size.

Throughout this work, consideration has been exclusively given to networks for which the path cost, where a constraint is imposed, is a simple sum of the constituent branch costs. For example, with probabilistic weighted networks in which some of the branch lengths are stochastically dependent, path length variance is no longer a simple sum of the constituent branch length variances, but depends on the choice of the path branches, and the covariance between them. Future work on such problems may be fruitful.

In treating flow networks, where the minimum cost flow pattern for a given flow requirement between a node pair is considered, the flow paths are restricted to be of traverse time not greater than a given time period. A suboptimal approach is given. The optimal solution involves a checking procedure to assure that the traverse time of a flow path is within the given bound. For shortest paths it is

desirable to generate a solution formulation which includes the checking procedure, thus resulting in a set of solutions which automatically satisfy the traverse time requirement. Such a procedure has not been developed here, and future work toward this area is recommended.

The shortest path problems discussed in this work are such that the simultaneous choice of a set of optimal paths do not affect each other. For example, in a flow network, the flow on one of the optimal paths decreases the effective capacities of the constituent branches; thus a simultaneous choice of other optimal paths is affected. More work in this area is encouraged in the future.

APPENDIX

APPENDIX A

THE PROGRAM LISTINGS

Two programs for finding the constrained shortest paths in networks are listed. They are coded in FORTRAN-IV and were run using the XDS Sigma 7 computer in the Montana State University Computing Center. The first program solves a minimum cost flow problem through the application of the algorithm in Section 3.4 where the flow network is modeled by a single branch cost network. The second program solves a message transmission problem in a communication network with probabilistic weighted transmission time through the network branches. The network is modeled by a two branch costs network with the mean and variance of the message traverse time through the branch representing the type 1 and type 2 branch cost weights. The algorithm in Section 3.7 is applied.

The programs are initiated by inputing the number of nodes in the network, the branch weights, the optimal path criteria, and the terminal node of the optimal path.

The output consists of the set of stepping costs, the associated stepping-cost paths, the last intermediate node of the stepping-cost paths, and its associated stepping cost for tracing the optimal path.

The rationale of the program is adequately disclosed by comment statements in front of each subroutine.

```
C     ****************************************************************
C     THIS PROGRAM EMPLOYS A COMBINATORIAL APPRCACH TO                *
C     FIND A SUBOPTIMAL FLOW PATTERN FOR A S-NODE NETWORK TO          *
C     SATISFY A GIVEN NODE PAIR FLOW REQUIREMENT. THE FLOW            *
C     PATH TRANVERSE TIME ARE TO BE WITHIN A GIVEN TIME REQUIREMENT.  *
C     ****************************************************************
      DIMENSION D(10,10),FINAL(10),F(10,15),H(10,10)
      DIMENSION NSTEP(10),NCOST(10,15),NEXTPT(10,15)
      DIMENSION CAP(10,10),FLOW(10,10),NPATH(10),NESTEP(10,15)
      DIMENSION ID(10),MIR(10,10)
      REAL NCOST,LO
      DATA FLOW,CAP,FST/0./
C
C     ****************************************************************
C     ******* CALL NETWRK FOR NETWORK CHARACTERISTICS *****************
C     ****************************************************************
C
      CALL NETWRK(N,MM,H,CAP,D,ID,MIR,LO,NODE,DELAY)
C
C     ****************************************************************
C     ******* EVALUATE THE UNCONSTRAINED MINIMUM COST PATH ****
C     ******* LENGTHS === FINAL TO BE USED IN SUBROUTINE   ****
C     ******* CMBINT1 AS A STOPPING CRITERION              ****
C     ****************************************************************
C
      WRITE(5,900)
    1 DO 20 I=2,N
   20 FINAL(I)=1.E15
      FINAL(1)=0.
      DO 40 I=1,N
      IFLAG=0
      DO 30 L=2,N
      DO 30 K=1,ID(L)
      M=MIR(L,K)
      A=FINAL(M)+D(M,L)
      IF(A.GE.FINAL(L)) GO TO 30
      FINAL(L)=A
      IFLAG=IFLAG+1
   30 CONTINUE
   40 IF(IFLAG.LE.0) GO TO 50
C
C     ****************************************************************
C     ******** EVALUATE THE CONSTRAINED MINIMUM COST PATHS ****
C     ****************************************************************
C
   50 CALL CMBINT1(N,MM,D,H,F,NSTEP,NCOST,NEXTPT,NESTEP,FINAL)
C
C     ****************************************************************
```

```
C      ******** SELECT THE MINIMUM COST AUGMENTATION PATH ******
CC     ******** IF NOT EXISTS TERMINATE                  ******
CCC    ***************************************************
C
       CALL SELECT(N,NSTEP,NCOST,NODE,NOSTEP,DELAY)
C
CC     ***************************************************
C
       IF(NOSTEP.LE.0)   GO TO 70
C
CCC    ***************************************************
CCCC   ********* TRACE THE OPTIMAL PATH ********************
CCC    ***************************************************
C
       CALL PATH(N,NODE,NOSTEP,D,H,F(NODE,NOSTEP),CAP,FLOW,
      1J,NCOST(NODE,NOSTEP),NEXTPT,NESTEP,NPATH,RESIDUAL )
       WRITE(5,910) NODE,NCOST(NODE,NOSTEP),RESIDUAL,
      1F(NODE,NOSTEP),(NPATH(K),K=J,1,-1)
       B=LO-FST
       IF(RESIDUAL.GT.B) RESIDUAL=B
       DO 60 I=J,2,-1
       FLOW(NPATH(I),NPATH(I-1))=FLOW(NPATH(I),NPATH(I-1))+RESIDUAL
       IF(FLOW(NPATH(I),NPATH(I-1)).LT.CAP(NPATH(I),NPATH(I-1)))
      1GO TO 60
       D(NPATH(I),NPATH(I-1))=1.E15
       H(NPATH(I),NPATH(I-1))=1.E15
   60  CONTINUE
       FST=FST+RESIDUAL
       WRITE(5,984) NODE,FST
C
CCC    ***************************************************
CCCC   ******** FIND ANOTHER OPTIMAL FLOW PATH IF FLOW ********
CCC    ******** REQUIREMENT HAS NOT BEEN SATISFIED     ********
CCC    ***************************************************
C
       IF(FST.LT.LO) GO TO 1
   70  WRITE(5,980)
       DO 80   I=1,N
       WRITE(5,981)
   80  WRITE(5,982) I,(FLOW(I,J),J=1,N)
  980  FORMAT(T10,'THE L TH 1-I STEPPING-DELAY PATH WITH COST',
      1/T10,'F(I,L), DELAY C(I,L), LAST INTERMEDIATE NODE K, AND',
      2/T10,'K1, WHERE THE K1 TH 1-K STEPPING-DELAY PATH FORMS F(I,L)'
      */T10,'IS SUCH THAT:',
      3//T15,'  F(I,L)=F(K,K1)+D(K,I)'
      4/T15,'  C(I,L)=C(K,K1)+H(K,I)'/)
  910  FORMAT(//T2,'THE MINIMUM COST 1-',I2,' AUGMENTATION PATH WITH',
      1/T2,'PATH DELAY ',F5.2,' RESIDUAL CAPACITY',F5.2,' AND PATH ',
```

```
     2/T2,'COST',F5.2,' IS THE SEQUENCE OF NODES:  ',10(I2,3X)//)
 920 FORMAT(T2,'THE PRESENT AMOUNT OF 1=5 FLOW Is:',F5.2//)
 980 FORMAT(T12 ,'******* OPTIMAL FLOW PATTERN *******'////
    1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
 981 FORMAT(//)
 982 FORMAT(3X,I2,5X,10(E7.2,2X))
 984 FORMAT(//T2,'THE PRESENT 1=',I2,' FLOW IS ',F5.2,//1H1)
     STOP
     END
```

```
C     ************************************************************
C     SUBROUTINE CMBINT1 COMPUTES THE SET OF STEPPING COSTS AND   *
C     THE CORRESPONDING STEPPING=COST PATHS OF A NETWORK          *
C     WITH A SINGLE BRANCH COST WEIGHT ASSOCIATED WITH EACH OF THE*
C     BRANCH IN THE NETWORK.                                      *
C     NOMENCLATURE:                                               *
C     D          D(I,J) IS THE LENGTH OF (I,J)                    *
C     H          H(I,J) IS THE COST OF (I,J)                      *
C     NEXTPT     NEXTPT(I,L) IS THE LAST INTERMEDIATE NODE OF     *
C                THE L TH 1=I STEPPING=COST PATH.                 *
C     NSTEP      NSTEP(I)=1 IS THE NUMBER OF EXISTING 1=I STEPP=  *
C                ING COSTS, WHERE THE NSTEP(I) TH ONE IS A CAN=   *
C                DIDATE.                                          *
C     NCOST      NCOST(I,L) IS THE VALUE OF THE L TH 1=I STEPP=   *
C                ING COST.                                        *
C     LOSTEP     IF J IS A FEASIBLE LAST INTERMEDIATE NODE OF THE 1=I
C                PATH, THEN THE PROPOSED LENGTH IS THE SUM OF     *
C                D(J,I) AND THE LENGTH OF THE LOSTEP(J,I)TH       *
C                STEPPING=COST 1=J PATH.                          *
C     NESTEP      IF J=NEXTPT(I,L), THEN THE NESTEP(I,L)TH   1=J  *
C                STEPPING=COST  PATH IS PART OF THE LTH 1= I      *
C                STEPPING=COST PATH.                              *
C     F          F(I,L) IS THE LENGTH OF THE L TH 1=I STEPPING=   *
C                COST PATH.                                       *
C     THE RATIONALE OF THE ALGORITHM IS EXPLAINED IN THE          *
C     FOLLOWING:                                                  *
C     LET S(I) DENOTE  THE SET OF NODES K TOGETHER WITH AN        *
C     INTEGER K1, WHERE K1=LOSTEP(K,I) IS THE SMALLEST            *
C     INTEGER LESS THAN NSTEP(K) SATISFYING THE INEQUALITY:       *
C     F(K,K1)+D(K,I).LT.F(I,NSTEP(I)=1).                          *
C     THE CANDIDATE VALUE OF THE NSTEP(I)TH 1=I STEPPING          *
C     COST IS:                                                    *
C     NCOST(I,NSTEP(I))=MINIMUM  OF NCOST(K,K1)+H(K,I), FOR       *
C     ALL MEMBERS OF S(I).                                        *
C  1. INITIALIZATIONS.                                            *
C  2. THE IMMEDIATE STEPPING COST IS ASSOCIATED WITH NODE R       *
C     IF NCOST(R,NSTEP(R))=MINIMUM OF NCOST(I,NSTEP(I));          *
C     AND F(R,NSTEP(R))=MINIMUM OF F(I,NSTEP(I)), FOR ALL NODE    *
C     I SUCH THAT NCOST(I,NSTEP(I))=NCOST(R,NSTEP(R)).            *
C  3. IF ALL STEPPING COSTS ASSOCIATED WITH NODE R HAVE BEEN      *
C     LOCATED, NODE R IS EXEMPTED FROM FURTHER CONSIDERATION.     *
C  4. IF ALL STEPPING COSTS HAVE BEEN LOCATED, TERMINATE.         *
C  5. FOR ALL I SUCH THAT R IS NOT A MEMBER OF S(I), DO THE       *
C     THE FOLLOWING:                                              *
C     DEFINE:  F123=F(R,NSTEP(R))+D(R,I).                         *
C     IF F123.LT.F(I,NSTEP(I)=1), THEN R BECOMES A MEMBER         *
C     OF S(I), AND LOSTEP(R,I)=NSTEP(R); AND FURTHER IF           *
C     NCOST(R,NSTEP(R))+H(R,I).LT.NCOST(I,NSTEP(I)), THEN         *
```

```
C      F(I,NSTEP(I))=F123, NEXTPT(I,NSTEP(I))=R, AND                          *
C      NESTEP(I,NSTEP(I))=NSTEP(R);   OR IF                                   *
C      NCOST(R,NSTEP)R))+H(R,I).EQ.NCOST(I,NSTEP(I)), AND                     *
C      F123.LT.F(I,NSTEP(I)), THEN                                            *
C      F(I,NSTEP(I))=F123, NEXTPT(I,NSTEP(I))=R, AND                          *
C      NESTEP(I,NSTEP(I))=NSTEP(R).                                           *
C   6. UPDATE NSTEP(R) BY: NSTEP(R)=NSTEP(R)+1.                               *
C      FOR ALL K OF S(R), IF THERE IS A K1.LT.NSTEP(K), SUCH                  *
C      THAT F(K,K1)+D(K,R).LT.F(R,NSTEP(R)=1)                                 *
C      K REMAINS IN S(R) OTHERWISE  K IS DELETED FROM S(R).                   *
C      AND LOSTEP(K,R)=K1.                                                    *
C      THEN NCOST(R,NSTEP(R))= MINIMUM OF NCOST(K,K1)+H(K,R),                 *
C      FOR ALL NODES K IN S(R), WITH K1=LOSTEP(K,R).                          *
C      THE NODE NEXTPT(R,NSTEP(R)) IS THE NODE I, IF NODE I IS                *
C      SUCH THAT F(I,I1)+D(I,R)= MINIMUM OF F(K,K1)+D(K,R), FOR ALL           *
C      NODE K SUCH THAT NCOST(K,K1)+H(K,R)=NCOST(R,NSTEP(R)), WHERE           *
C      I1=LOSTEP(I,R), AND K1=LOSTEP(K,R)                                     *
C   7. THE ABOVE COMPLETES A SET OF OPERATIONS TO FIND AN                     *
C      IMMEDIATE STEPPING COST. PROGRAM IS DIRECTED TO STEP 2                 *
C      UNTIL NO FURTHER STEPPING COSTS CAN BE FOUND.                          *
C      ***************************************************************
      SUBROUTINE CMBINT1(N,MM,D,H,F,NSTEP,NCOST,NEXTPT,NESTEP,FINAL)
      DIMENSION D(10,10),F(10,15),H(10,10)
      DIMENSION NSTEP(10),NCOST(10,15),NEXTPT(10,15)
      DIMENSION LOSTEP(10,10),CAP(10,10),FLOW(10,10)
      DIMENSION FINAL(10),LABEL(10),NESTEP(10,15)
      DIMENSION IN(10),NIN(10),INTER(10,10),NINTER(10,10)
      REAL NCOST
C     ***************************************************
C     ************** STEP 1: INITIALIZATION **************
C     ***************************************************
      NSTEP(1)=2
      F(1,1)=0
      NCOST(1,1)=0
      LABEL(1)=1
      NIN(1)=0
      LPER=2
      DO 300 I=2,N
      LABEL(I)=I
      NSTEP(I)=1
      F(I,1)=D(1,I)
      NCOST(I,1)=H(1,I)
      NEXTPT(I,1)=1
      NESTEP(I,1)=1
      LOSTEP(1,I)=1
      IN(I)=1
      INTER(I,1)=1
      NIN(I)=N-2
```

```
      K=0
      DO 300 J=2,N
      IF(J.EQ.I) GO TO 300
      K=K+1
      NINTER(I,K)=J
  300 CONTINUE
C     ************************************************************
C     ************** STEP 2: LOCATE IMMEDIATE STEPPING COST
C     ************************************************************
      WRITE(5,900)
  310 A123=1.E15
      LR=LPER
      B123=F(LABEL(LR),NSTEP(LABEL(LR)))
      DO 320 J123=LPER,N
      I=LABEL(J123)
      IF(NCOST(I,NSTEP(I))-A123)315,317,320
  315 A123=NCOST(I,NSTEP(I))
  316 LR=J123
      B123=F(I,NSTEP(I))
      GO TO 320
  317 IF(F(I,NSTEP(I)).LT.B123)GO TO 316
  320 CONTINUE
      LABELR=LABEL(LR)
      J123=NEXTPT(LABELR,NSTEP(LABELR))
      WRITE(5,990) LABELR,NCOST(LABELR,NSTEP(LABELR)),
     1F(LABELR,NSTEP(LABELR)),NSTEP(LABELR),NEXTPT(LABELR,
     2NSTEP(LABELR)),NESTEP(LABELR,NSTEP(LABELR))
C     ************************************************************
C     ************** STEP 3: EXEMPT LABELR FROM FURTHER **
C     **************                CONSIDERATION          **
C     ************************************************************
      IF(F(LABELR,NSTEP(LABELR)).GT.FINAL(LABELR)) GO TO 325
      M123=LABEL(LPER)
      LABEL(LPER)=LABELR
      LABEL(LR)=M123
      LPER=LPER+1
C     ************************************************************
C     ************** STEP 4: TEST OF PROGRAM TERMINATION ***
C     ************************************************************
      IF(LPER.GT.N) GO TO 430
C     ************************************************************
C     ************** STEP 5: FOR I.NE.LABELR, UPDATE THE ***
C     **************           STEPPING COSTS            ****
C     ************************************************************
  325 CONTINUE
      IF(NIN(LABELR).LE.0) GO TO 380
      J=1
  330 I=NINTER(LABELR,J)
```

```
      IF(I.NE.LABEL(LPER-1)) GO TO 331
      NINTER(LABELR,J)=NINTER(LABELR,NIN(LABELR))
      NIN(LABELR)=NIN(LABELR)-1
      J=J-1
      GO TO 370
  331 B123=F(LABELR,NSTEP(LABELR))+D(LABELR,I)
      IF(NSTEP(I).GT.1) GO TO 335
      IF(B123-1.E15) 340,370,370
  335 IF(B123-F(I,NSTEP(I)-1)) 340,370,370
  340 LOSTEP(LABELR,I)=NSTEP(LABELR)
      IN(I)=IN(I)+1
      INTER(I,IN(I))=LABELR
      NINTER(LABELR,J)=NINTER(LABELR,NIN(LABELR))
      NIN(LABELR)=NIN(LABELR)-1
      J=J-1
      A123=NCOST(LABELR,NSTEP(LABELR))+H(LABELR,I)
      IF(A123-NCOST(I,NSTEP(I))) 350,360,370
  350 NCOST(I,NSTEP(I))=A123
  355 NEXTPT(I,NSTEP(I))=LABELR
      NESTEP(I,NSTEP(I))=NSTEP(LABELR)
      F(I,NSTEP(I))=B123
      GO TO 370
  360 CONTINUE
      IF(B123.LT.F(I,NSTEP(I))) GO TO 355
  370 J=J+1
      IF(J-NIN(LABELR)) 330,330,380
C     ********************************************************
CC    ************** STEP 6: UPDATE THE STEPPING COST FOR
CC    **************                NODE LABELR
C     ********************************************************
  380 NSTEP(LABELR)=NSTEP(LABELR)+1
      IF(NSTEP(LABELR).LE.MM) GO TO 375
      GO TO 430
  375 NCOST(LABELR,NSTEP(LABELR))=1.E15
      IF(IN(LABELR).LE.0) GO TO 310
      J=1
  390 K=INTER(LABELR,J)
      DO 400 K1=LOSTEP(K,LABELR),NSTEP(K)-1
      B123=F(K,K1)+D(K,LABELR)
      IF(B123.LT.F(LABELR,NSTEP(LABELR)-1)) GO TO 410
  400 CONTINUE
  405 NIN(K)=NIN(K)+1
      NINTER(K,NIN(K))=LABELR
      INTER(LABELR,J)=INTER(LABELR,IN(LABELR))
      IN(LABELR)=IN(LABELR)-1
      J=J-1
      GO TO 420
  410 A123=NCOST(K,K1)+H(K,LABELR)
```

```
      LOSTEP(K,LABELR)=K1
      ICO=ICO+1
      IF(A123=NCOST(LABELR,NSTEP(LABELR))) 415,418,420
  415 NCOST(LABELR,NSTEP(LABELR))=A123
  416 NEXTPT(LABELR,NSTEP(LABELR))=K
      NESTEP(LABELR,NSTEP(LABELR))=K1
      F(LABELR,NSTEP(LABELR))=B123
      GO TO 420
  418 CONTINUE
      IF(B123.LT.F(LABELR,NSTEP(LABELR))) GO TO 416
  420 J=J+1
      IF(J=IN(LABELR)) 390,390,310
  430 DO 450 J=1,N=1
      I=LABEL(J)
  450 NSTEP(I)=NSTEP(I)=1
  900 FORMAT(/T3,'I',3X,'C(I,L)',3X,'F(I,L)',4X,'L',4X,'K',4X,'K1'
     1//)
  990 FORMAT(T2,I2,3X,F6.2,3X,F5.2,4X,3(I2,3X))
      RETURN
      END
```

```
C
C     ************************************************************************
C     SUBROUTINE NETWRK READS IN THE NETWORK CHARACTERISTICS,              *
C     RETURNS MATRICES D, H, CAP FOR FINDING A SUBOPTIMAL FLOW             *
C     PATTERN FOR THE GIVEN NODE PAIR.                                     *
C     NOMENCLATURE:                                                        *
C     N        IS THE NUMBER OF NODES IN THE NETWRK.                       *
C     NODE     IS THE TERMINAL NODE OF FLOWS FROM THE GIVEN                *
C              INITIAL NODE ONE.                                           *
C     LO       IS THE FLOW REQUIREMENT BETWEEN THE INITIAL AND TERMINAL    *
C              NODE.                                                       *
C     DELAY    IS THE MAXIMUM VALUE OF TRANSIT-TIME DELAY ALLOWED FOR      *
C              EACH FORWARD FLOW PATH.                                     *
C     MM       IS THE MAXIMUM NUMBER OF STEPPING COSTS FOR EACH NODE,      *
C              MM=15 FOR THIS PARTICULAR EXAMPLE.                          *
C     IR, IC, D1, D2, D3 ARE A SET OF DATA FOR EACH BRANCH, WHERE          *
C              IR IS THE INITIAL NODE, IC IS THE TERMINAL NODE, D1         *
C              IS THE UNIT COST OF FLOW THROUGH  THE BRANCH, D2 IS THE     *
C              TRANSIT TIME DELAY FOR THE FLOW TO TRAVERSE THE BRANCH,     *
C              D3 IS THE CAPACITY OF THE BRANCH.                           *
C     DATA CARD ONE: FIELD ONE      N                                      *
C                    FIELD TWO      MM                                     *
C                    FIELD THREE    NODE                                   *
C                    FIELD FOUR     LO                                     *
C                    FIELD FIVE     DELAY                                  *
C              FORMAT (2I1,3F4.2)                                          *
C     EACH SUCCEEDING CARD AFTER CARD ONE:                                 *
C                    FIELD ONE      IR                                     *
C                    FIELD TWO      IC                                     *
C                    FIELD THREE    D1                                     *
C                    FIELD FOUR     D2                                     *
C                    FIELD FIVE     D3                                     *
C              FORMAT (8(2I1,3F3.1))                                       *
C                                                                          *
C     ************************************************************************
      SUBROUTINE NETWRK(N,MM,H,CAP,D,ID,MIR,LO,NODE,DELAY)
      DIMENSION  D(10,10),CAP(10,10),ID(10),MIR(10,10),H(10,10)
      DIMENSION D1(8),D2(8),IR(8),IC(8),D3(8)
      REAL LO
      READ(105,950) N,MM,NODE,LO,DELAY
      WRITE(5,970)NODE,LO,DELAY
      DO 08 I=1,N
      DO 05 J=1,N
      H(I,J)=1.E15
      CAP(I,J)=0.
   05 D(I,J)=1.E15
      D(I,I)=0.
      CAP(I,I)=1.E15
```

```
      ID(I)=0
   08 H(I,I)=0.
   10 READ(105,990) (IR(I),IC(I),D1(I),D2(I),D3(I),I=1,8)
      DO 20 K=1,8
      J=IR(K)
      I=IC(K)
      IF(J.LE.0) GO TO 21
      ID(I)=ID(I)+1
      MIR(I,ID(I))=J
      D(J,I)=D1(K)
      H(J,I)=D2(K)
      CAP(J,I)=D3(K)
   20 CONTINUE
      GO TO 10
   21 CONTINUE
      WRITE(5,1020)
      DO 100 I=1,N
      WRITE(5,1000) I,(D(I,J),J=1,N)
  100 WRITE(5,1040)
      WRITE(5,1010)
      WRITE(5,1030)
      DO 200 I=1,N
      WRITE(5,1000) I,(H(I,J),J=1,N)
  200 WRITE(5,1040)
      WRITE(5,1010)
      WRITE(5,1060)
      DO 300 I=1,N
      WRITE(5,1000) I,(CAP(I,J),J=1,N)
  300 WRITE(5,1040)
      WRITE(5,1010)
  950 FORMAT(3I2,2F5.2)
  970 FORMAT(T2,'PROGRAM TO FIND THE OPTIMAL FLOW PATTERN OF A FLOW'
     1/T2,'NETWORK FOR THE 1=',I2,' FLOW REQUIREMENT OF ',F5.2,
     2/T2,'UNITS OF FLOW. ALL FLOW PATHS ARE REQUIRED TO BE OF TRANSIT'
     3/T2,'TIME DELAY OF NO MORE THAN ',F5.2,' UNITS OF TIME'///)
  990 FORMAT(8(2I1,3F3.1))
 1000 FORMAT(3X,I2,5X,10(E7.2,2X))
 1010 FORMAT(1H1)
 1020 FORMAT(T12,'********* COST MATRIX     **********'////
     1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
 1030 FORMAT(T12,'*********  DELAY MATRIX **********'////
     1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
 1040 FORMAT(/)
 1060 FORMAT(T12 ,'********* CAPACITY MATRIX **********'////
     1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
      RETURN
      END
```

```
C     ************************************************************
C     SUBROUTINE SELECT CHOOSES THE OPTIMAL PATH FOR A GIVEN PATH  *
C     TRAVERSE TIME REQUIREMENT.                                   *
C     ************************************************************
      SUBROUTINE SELECT(N,NSTEP,NCOST,NODE,NOSTEP,DELAY)
      DIMENSION   NSTEP(10),NCOST(10,15)
      REAL NCOST
      DO 10 NOSTEP=NSTEP(NODE),1,-1
      IF(NCOST(NODE,NOSTEP).LE.DELAY)   RETURN
   10 CONTINUE
      RETURN
      END
```

```
C     ***********************************************************
CC    SUBROUTINE PATH TRACES THE OPTIMAL PATH.                  *
C     ***********************************************************
      SUBROUTINE PATH(N,NODE,NOSTEP,D,H,F,CAP,FLOW,J,NCOST,
     1NEXTPT,NESTEP,NPATH,RESIDUAL)
      DIMENSION    NEXTPT(10,15),NESTEP(10,15),NPATH(10),D(10,10)
      DIMENSION CAP(10,10),ID(10),MIR(10,10),H(10,10),FLOW(10,10)
      REAL NCOST
      RESIDUAL=1.E15
      IF(NOSTEP.GT.0)       GO TO 5
      WRITE(5,900)
      RETURN
    5 NPATH(1)=NODE
      LL=NOSTEP
      DO 10 J=2,N
      NPATH(J)=NEXTPT(NPATH(J-1),LL)
      AA=CAP(NPATH(J),NPATH(J-1))-FLOW(NPATH(J),NPATH(J-1))
      IF(RESIDUAL.GT.AA) RESIDUAL=AA
      IF(NPATH(J).EQ.1) RETURN
   10 LL=NESTEP(NPATH(J-1),LL)
  900 FORMAT(//T30,'NO SUCH PATH')
      RETURN
      END
```

PROGRAM TO FIND THE OPTIMAL FLOW PATTERN OF A FLOW NETWORK FOR THE 1= 5 FLOW REQUIREMENT OF .59 UNITS OF FLOW. ALL FLOW PATHS ARE REQUIRED TO BE OF TRANSIT TIME DELAY OF NO MORE THAN 14.30 UNITS OF TIME

********* COST MATRIX **********

	1	2	3	4	5
1	.C0E 00	.50E 00	.60E C0	.20E 00	.78E 01
2	.14E 01	.00E 00	.10E C1	.20E 00	.19E 01
3	.10E 16	.00E 00	.00E C0	.20E 00	.53E 01
4	.10E 16	.70E 00	.50E C0	.00E 00	.41E 01
5	.10E 16	.10E 16	.70E C0	.10E 16	.00E 00

********* DELAY MATRIX **********

	1	2	3	4	5
1	.00E 00	.82E 01	.15E C1	.46E 01	.30E 00
2	.11E 01	.00E 00	.41E C1	.20E 01	.70E 01
3	.10E 16	.67E 01	.00E 00	.14E 01	.20E 00
4	.10E 16	.32E 01	.21E C1	.00E 00	.30E 00
5	.10E 16	.10E 16	.10E C0	.10E 16	.00E 00

********** CAPACITY MATRIX ************

	1	2	3	4	5
1	•10E 16	•50E 00	•60E 00	•20E 00	•78E 01
2	•14E 01	•10E 16	•10E 01	•20E 00	•19E 01
3	•00E 00	•00E 00	•10E 16	•20E 00	•53E 01
4	•00E 00	•70E 00	•50E 00	•10E 16	•41E 01
5	•00E 00	•00E 00	•70E 00	•00E 00	•10E 16

THE L TH 1=I STEPPING=DELAY PATH WITH COST
F(I,L), DELAY C(I,L), LAST INTERMEDIATE NODE K, AND
K1, WHERE THE K1 TH 1=K STEPPING=DELAY PATH FORMS F(I,L)
IS SUCH THAT:

$$F(I,L)=F(K,K1)+D(K,I)$$
$$C(I,L)=C(K,K1)+H(K,I)$$

I	C(I,L)	F(I,L)	L	K	K1
5	.30	7.80	1	1	1
3	.40	8.50	1	5	1
3	1.50	.60	2	1	1
5	1.70	5.90	2	3	2
4	1.80	8.70	1	3	1
4	2.90	.80	2	3	2
5	3.20	4.90	3	4	2
4	4.60	.20	3	1	1
5	4.90	4.30	4	4	3
2	5.00	9.40	1	4	1
2	6.10	1.50	2	4	2
2	7.80	.90	3	4	3
2	8.20	.50	4	1	1
5	13.10	3.40	5	2	2
5	14.80	2.80	6	2	3
5	15.20	2.40	7	2	4

THE MINIMUM COST 1= 5 AUGMENTATION PATH WITH
PATH DELAY 13.10 RESIDUAL CAPACITY .20 AND PATH
COST 3.40 IS THE SEQUENCE OF NODES: 1 3 4 2 5

THE PRESENT 1= 5 FLOW IS .20

I	C(I,L)	F(I,L)	L	K	K1
5	.30	7.80	1	1	1
3	.40	8.50	1	5	1
3	1.50	.60	2	1	1
5	1.70	5.90	2	3	2
4	4.60	.20	1	1	1
5	4.90	4.30	3	4	1
2	7.10	8.50	1	3	1
2	7.80	.90	2	4	1
2	8.20	.50	3	1	1
5	14.80	2.80	4	2	2
5	15.20	2.40	5	2	3

THE MINIMUM COST 1= 5 AUGMENTATION PATH WITH
PATH DELAY 4.90 RESIDUAL CAPACITY .20 AND PATH
COST 4.30 IS THE SEQUENCE OF NODES: 1 4 5

THE PRESENT 1= 5 FLOW IS .40

I	C(I,L)	F(I,L)	L	K	K1
5	.30	7.80	1	1	1
3	.40	8.50	1	5	1
3	1.50	.60	2	1	1
5	1.70	5.90	2	3	2
2	7.10	8.50	1	3	1
2	8.20	.50	2	1	1
4	9.10	8.70	1	2	1
4	10.20	.70	2	2	2
5	10.50	4.80	3	4	2
5	15.20	2.40	4	2	2

THE MINIMUM COST 1= 5 AUGMENTATION PATH WITH
PATH DELAY 10.50 RESIDUAL CAPACITY .20 AND PATH
COST 4.80 IS THE SEQUENCE OF NODES: 1 2 4 5

THE PRESENT 1= 5 FLOW IS .59

******* OPTIMAL FLOW PATTERN *******

	1	2	3	4	5
1	.00E 00	.19E 00	.20E 00	.20E 00	.00E 00
2	.00E 00	.00E 00	.00E 00	.19E 00	.20E 00
3	.00E 00	.00E 00	.00E 00	.20E 00	.00E 00
4	.00E 00	.20E 00	.00E 00	.00E 00	.39E 00
5	.00E 00	.00E 00	.00E 00	.00E 00	.00E 00

```
C     ****************************************************************
C     THIS PROGRAM COMPUTES THE MINIMAL COST 1-5 PATH OF              *
C     A 5-NODE NETWORK WHERE EACH BRANCH OF THE NETWORK               *
C     IS ASSOCIATED WITH A RANDOMLY DISTRIBUTED TRANSIT TIME WEIGHT.  *
C     THE MINIMAL COST 1-5 PATH IS TO BE SUCH THAT THE PROBABILITY    *
C     OF THE TRANSIT TIME EXCEEDING A GIVEN LENGTH OF TIME IS NOT     *
C     GREATER THAN A GIVEN FRACTION.                                  *
C     IN CMBINT2 THE TERM "LENGTH" REPRESENTS COST, AND THE TERM      *
C     "COST" REPRESENTS THE TRAVERSE TIME.                            *
C     ****************************************************************
C
      DIMENSION D(10,10),ID(10),MIR(10,10),F1(10,8,8),H1(10,10),
     1H2(10,10),NSTEP1(10),NSTEP2(10,8),NCOST1(10,8),NCOST2(10,8,8),
     2FINAL(10),K11STEP(10,8,8),K12STEP(10,8,8),NEXTPT1(10,8,8),NPATH
     3(10)
      REAL NCOST1,NCOST2,LO
C
      CALL NETWRK(N,MM,H1,H2,D,ID,MIR,FINAL,LO,NODE,FRACTION)
C
      CALL CMBINT2(N,MM,H1,H2,D,NSTEP1,NSTEP2,NCOST1,NCOST2,F1,
     1NEXTPT1,K11STEP,K12STEP,FINAL)
C
      CALL SELECT(N,NSTEP1,NSTEP2,NCOST1,NCOST2,F1,NODE,NOSTEP1,
     1NOSTEP2,FRACTION,LO,AA)
C
      IF(NOSTEP2.GT.0)GO TO 500
      WRITE(5,920)
      STOP
  500 CALL PATH(N,NODE,NOSTEP1,NOSTEP2,NEXTPT1,K11STEP,K12STEP,
     1NPATH,J)
      WRITE(5,930) NCOST1(NODE,NOSTEP1),NCOST2(NODE,NOSTEP1,NOSTEP2)
     1,F1(NODE,NOSTEP1,NOSTEP2),(NPATH(K),K=J,1,-1)
  920 FORMAT(T15,'NO SUCH PATH')
  930 FORMAT(//T2,'THE OPTIMAL PATH WITH MEAN OF TRAVERSE TIME ',
     1F5.2,/T2,' VARIANCE OF TRAVERSE TIME ',F5.2,' AND PATH COST ',
     2F5.2,/T2,'IS THE SEQUENCE OF NODES:  ',10(I2,3X)///)
      STOP
      END
```

```
C     ************************************************************
C     SUBROUTINE NETWRK READS IN THE NETWORK CHARACTERISTICS,     *
C     RETURNS MATRICES D, H1, H2, ID, MIR TO THE MAIN PROGRAM.    *
C     NOMENCLATURE:                                               *
C     N       IS THE NUMBER OF NODES IN THE NETWRK.               *
C     NODE    IS THE TERMINAL NODE OF THE OPTIMAL PATH TO BE FOUND.*
C     LO      IS THE UPPER BOUND FOR THE OPTIMAL PATH TRANSIT TIME.*
C     FRACTION IS THE MAXIMUM PROBABILITY FOR THE OPTIMAL PATH TANSIT
C     TIME TO EXCEED THE VALUE LO.                                *
C     MM      IS THE MAXIMUM NUMBER OF STEPPING TIME  FOR EACH NODE,
C             MM=8   FOR THIS PARTICULAR EXAMPLE.                 *
C     IR, IC, D1, D2, D3 ARE A SET OF DATA FOR EACH BRANCH, WHERE *
C             IR IS THE INITIAL NODE, IC IS THE TERMINAL NODE, D1 *
C             IS THE BRANCH COST,     D2 IS THE MEAN OF THE BRANCH
C             TRANSIT TIME, D3 IS THE STANDARD DEVIATION OF THE   *
C             BRANCH TRANSIT TIME                                 *
C     DATA CARD ONE: FIELD ONE      N                             *
C                    FIELD TWO      MM                            *
C                    FIELD THREE    NODE                          *
C                    FIELD FOUR     LO                            *
C                    FIELD FIVE     FRACTION                      *
C             FORMAT (2I1,3F4.2)                                  *
C     EACH SUCCEEDING CARD AFTER CARD ONE:                        *
C                    FIELD ONE      IR                            *
C                    FIELD TWO      IC                            *
C                    FIELD THREE    D1                            *
C                    FIELD FOUR     D2                            *
C                    FIELD FIVE     D3                            *
C             FORMAT (8(2I1,3F3.1))                               *
C     ************************************************************
      SUBROUTINE NETWRK(N,MM,H1,H2,D,ID,MIR,FINAL,LO,NODE,FRACTION)
      DIMENSION   D(10,10),H2(10,10),ID(10),MIR(10,10),H1(10,10)
      DIMENSION FINAL(10), D1(8),D2(8),IR(8),IC(8),D3(8)
      REAL LO
C     ****************************************
C     ***** INPUT NETWORK CHARACTERISTICS *****
C     ****************************************
      READ(105,950) N,MM,NODE,LO,FRACTION
      WRITE(5,970)NODE,LO,FRACTION
      DO 08 I=1,N
      DO 05 J=1,N
      H1(I,J)=1.E15
      H2(I,J)=1.E15
      FINAL(I)=1.E15
   05 D(I,J)=1.E15
      D(I,I)=0.
      H2(I,I)=0.
```

```
      ID(I)=0
   08 H1(I,I)=0.
   10 READ(105,990) (IR(I),IC(I),D1(I),D2(I),D3(I),I=1,8)
      DO 20 K=1,8
      J=IR(K)
      I=IC(K)
      IF(J.LE.0) GO TO 21
      ID(I)=ID(I)+1
      MIR(I,ID(I))=J
      D(J,I)=D1(K)
      H1(J,I)=D2(K)
      H2(J,I)=D3(K)
   20 CONTINUE
      GO TO 10
   21 CONTINUE
C     **************************************************
CC    ****** FIND THE UNCONSTRAINED SHORTEST  PATHS ***
C     **************************************************
      FINAL(1)=0.
      DO 40 I=1,N
      IFLAG=0
      DO 30 L=2,N
      DO 30 K=1,ID(L)
      M=MIR(L,K)
      A=FINAL(M)+D(M,L)
      IF(A.GE.FINAL(L)) GO TO 30
      FINAL(L)=A
      IFLAG=IFLAG+1
   30 CONTINUE
   40 IF(IFLAG.LE.0) GO TO 50
   50 CONTINUE
C     ***********************************
CC    ******* OUTPUT MATRICES ********
C     ***********************************
      WRITE(5,1020)
      DO 100 I=1,N
      WRITE(5,1000) I,(D(I,J),J=1,N)
  100 WRITE(5,1040)
      WRITE(5,1010)
      WRITE(5,1030)
      DO 200 I=1,N
      WRITE(5,1000) I,(H1(I,J),J=1,N)
  200 WRITE(5,1040)
      WRITE(5,1010)
      WRITE(5,1060)
      DO 300 I=1,N
      WRITE(5,1000) I,(H2(I,J),J=1,N)
  300 WRITE(5,1040)
```

```
      WRITE(5,1010)
C     **********************
CC    ******* FORMATS *******
C     **********************
  950 FORMAT(3I2,2F5.2)
  970 FORMAT(T2,'PROGRAM TO FIND THE MIN-COST 1=',I2,' PATH SUCH THAT'
     1/T2,'THE PROBABILITY OF THE PATH TRANSIT TIME EXCEEDING A GIVEN'
     2/T2,'VALUE ',F5.2,' BE NOT GREATER THAN A GIVEN'
     3/T2,'FRACTION 1-N(',F5.2,')'///)
  990 FORMAT(8(2I1,3F3.1))
 1000 FORMAT(3X,I2,5X,10(E7.2,2X))
 1010 FORMAT(1H1)
 1020 FORMAT(T12,'********* COST   MATRIX   **********'////
     1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
 1030 FORMAT(T12,'*********  TRANSIT MEAN MATRIX ************'////
     1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
 1040 FORMAT(/)
 1060 FORMAT(T12 ,'********* TRANSIT VARIANCE MATRIX ************'////
     1T15,'1',8X,'2',8X,'3',8X,'4',8X,'5'//)
      RETURN
      END
```

```
C     ************************************************************
C     SUBROUTINE CMBINT2 COMPUTES THE SET OF STEPPING COSTS AND THE
C     ASSOCIATED STEPPING-COST PATHS OF A DOUBLY-LABELLED COST
C     NETWORK IN A SEQUENTIAL MANNER.
C     NOMENCLATURE:
C N             IS THE NUMBER OF NODES IN THE NETWORK.
C D(J,I)        IS THE LENGTH WEIGHT OF THE BRANCH (J,I).
C H1(J,I)       IS THE TYPE 1 COST OF THE BRANCH (J,I).
C H2(J,I)       IS THE TYPE 2 COST OF THE BRANCH (J,I).
C NCOST1(I,J)     IS THE J TH TYPE 1 1-I STEPPING COST.
C NCOST2(I,J,K)   IS THE K TH TYPE 2 1-I COST ASSOCIATED WITH
C                 NCOST1(I,J).
C NSTEP1(I,J)     IS THE NUMBER OF TYPE 1 1-I STEPPING COST.
C NSTEP2(I,J,K)   IS THE K TH TYPE 2 1-I COST ASSOCIATED WITH
C                 NCOST1(I,J) WHICH IS TO BE FOUND.
C F1(I,J,K)       IS THE LENGTH OF THE STEPPING-COST PATH ASSOCIATED
C                 WITH THE STEPPING COST (NCOST1(I,J),NCOST2(I,J,K)).
C N1(I)           IS THE NOMINATED VALUE FOR THE TYPE 1 1-I COST
C                 OF THE NEXT 1-I STEPPING COST.
C N2(I)           IS THE NOMINATED VALUE FOR THE TYPE 2 1-I COST
C                 OF THE NEXT 1-I STEPPING COST.
C F(I)            IS THE NOMINATED VALUE FOR THE NEXT 1-I STEPPING
C                 COST PATH LENGTH.
C NEXTPT1(I,J,K)  IS THE LAST INTERMEDIATE NODE OF F1(I,J,K).
C LET I1=NEXTPT1(I,J,K), J1=K11STEP(I,J,K), AND K1=K12STEP(I,J,K)
C                 THEN J1,K1 ARE SUCH THAT F1(I,J,K)=F1(I1,J1,K1)+
C                 D(I1,I), NCOST1(I,J)=NCOST1(I1,J1)+H1(I1,I), AND
C                 NCOST2(I,J,K)=NCOST2(I1,J1,K1)+H2(I1,I).
C LET F*(I,C) = MINIMUM OF F1(I,K1(I),K2(I)), WHERE K1(I).LE.
C NSTEP1(I), AND K2(I) IS SUCH THAT THE INTERVAL BETWEEN
C NCOST2(I,K1(I),K2(I)) AND NCOST2(I,K1(I),K2(I)+1) INCLUDES THE
C VALUE C.
C FO(I,J) IS THE J TH STEPPING VALUE OF F*(I,C), NCOSTO(I,J) IS THE
C J TH STEPPING COST OF F*(I,C), AND NSTEPO(I) IS THE
C NUMBER OF STEPPING COSTS ASSOCIATED WITH F*(I,C).
C
C DEIFNE THE SET OF POSSIBLE LAST INTERMEDIATE NODES, S(I), FOR   THE
C PATH F1(I,NSTEP1(I),NSTEP2(I,NSTEP1(I))) TO BE THE NODES K SUCH
C THAT K BELONGS TO S(I) IF THERE EXISTS A SMALLEST INTEGER K1(K)
C .LE.NSTEP1(K), AND AN ASSOCIATED K2(K).LE.NSTEP2(K,NSTEP1(K1(K)))
C SUCH THAT F1(K,K1(K),K2(K))+D(K,I).LT.FO(I,J), WHERE
C J.LE.NSTEPO(I) IS SUCH THAT THE INTERVAL BETWEEN NCOSTO(I,J)
C AND NCOSTO(I,J+1) INCLUDES THE POINT NCOST2(K,K1(K),K2(K)).
C
C THE RATIONALE OF THE ALGORITHM IS AS FOLLOWS:
C STEP 1: INITIALIZATIONS.
C STEP 2: THE IMSC IS ASSOCIATED WITH NODE R IF R IS SUCH THAT
C       1.) N1(R)=MINIMUM OF N1(I), FOR ALL NODES I,K
```

```
C        2.) N2(R)=MINIMUM OF N2(I), FOR ALL NODES I SUCH THAT                  *
C            N1(I)=N1(R), AND
C        3.) F(R)=MINIMUM OF F(I), FOR ALL NODES I SUCH THAT
C            N1(I)=N1(R), AND N2(R)=N2(I).
C        4.) IF N1(R).GT.NCOST1(R,NSTEP1(R)), THEN
C            NSTEP1(R)=NSTEP1(R)+1,
C            NSTEP2(R,NSTEP1(R))=1,
C            NEXTPT1(R,NSTEP1(R),1)=NEXTPT1(R,NSTEP1(R)=1,
C                                   NSTEP2(R,NSTEP1(R)=1)),
C            K11STEP(R,NSTEP1(R),1)=K11STEP(R,NSTEP1(R)=1,
C                                   NSTEP2(R,NSTEP1(R)=1)),
C            K12STEP(R,NSTEP1(R),1)=K12STEP(R,NSTEP1(R)=1,
C                                   NSTEP2(R,NSTEP1(R)=1)).
C        5.) F1(R,NSTEP1(R),NSTEP2(R,NSTEP1(R)))=F(R).
C  STEP 3: EXAMPT NODE R FROM FURTHER CONSIDERATION IF ALL
C  STEP 4: TERMINATE IF ALL STEPPING COSTS HAVE BEEN LOCATED.
C          STEPPING COSTS ASSOCIATED WITH NODE R HAVE BEEN LOCATED.
C  STEP 5:          FOR ALL NODES I SUCH THAT R IS NOT A MEMBER OF S(I).
C        1.) LET J BE THE J TH STEPPING COST NCOST0(I,J) OF F*(I,C),
C            SUCH THAT THE INTERVAL NCOST0(I,J) AND NCOST0(I,J+1)
C            INCLUDES THE POINT N2(R)+H2(R,I), THEN, IF
C            F(R)+D(R,I).LT.FC(I,J), NODE R BECOMES A MEMBER OF
C            S(I); OTHERWISE GO TO STEP 6.
C        2.) LET A=N1(R)+H1(R,I), B=N2(R)+H2(R,I), AND C=F(R)+D(R,I), IF
C        3.) A.LT.N1(I), THEN N1(I)=A, N2(I)=B, F(I)=C,
C            NEXTPT1(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=R,
C            K11STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP1(R),
C            K12STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP2(R,
C                                                     NSTEP1(R)),
C        4.) OR IF A.EQ.N1(I), AND B.LT.N2(I) , THEN
C            N2(I)=B, F(I)=C,
C            NEXTPT1(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=R,
C            K11STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP1(R)
C            K12STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP2(R,
C        5.) OR IF A.EQ.N1(I), B=N2(I), AND C.LT.F(I), THEN
C            F(I)=C, NEXTPT1(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=R,
C            K11STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP1(R),
C            K12STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP2(R,
C                                                     NSTEP1(R)).
C  STEP 6:          FOR NODE R
C        1.) NSTEP2(R,NSTEP1(R))=NSTEP2(R,NSTEP1(R))+1
C        2.) F*(R,C)=MINIMUM OF F*(R,C) AND F(R), WHERE C IS SUCH
C            THAT C.GE.N2(R).
C        3.) FOR ALL NODES K BEING A MEMBER OF S(R), IF THERE EXISTS
C            A SMALLEST INTEGER K1(K), AND AN ASSOCIATED SMALLEST
C            INTEGER K2(K), SUCH THAT K1(K).LE.NSTEP1(K), K2(K).LE.
C            NSTEP2(K,K1(K)), AND THAT
C            F1(K,K1(K),K2(K))+D(K,I).LT.FC(R,J), WHERE J IS SUCH
```

```
C            THAT THE INTERVAL NCOSTC(I,J) AND NCOSTC(I,J+1)                *
C            INCLUDES THE POINT NCOST2(K,K1(K),K2(K)), THEN                 *
C            K REMAINS IN S(R).                                             *
C            OTHERWISE S(R)=S(R)-K                                          *
C         4.) N1(R)=LARGE VALUE, N2(R)=LARGE VALUE, AND                     *
C            F(R)=LARGE VALUE.                                              *
C         5.) N1(R)=MINIMUM OF NCOST1(K,K1(K))+H1(K,R), FOR ALL K IN S(R).  *
C         6.) N2(R)=MINIMUM OF NCOST2(K,K1(K),K2(K))+H2(K,R), FOR ALL       *
C            NODE K SUCH THAT NCOST1(K,K1(K))+H1(K,R)=N1(R).                *
C         7.) F(R)=MINIMUM OF F1(K,K1(K),K2(K))+D(K,R), FOR ALL K           *
C            SUCH THAT NCOST1(K,K1(K))=N1(R),NCOST2(K,K1(K),K2(K))=N2(R).   *
C   STEP 7: RETURN TO STEP 2.                                               *
C         THE ABOVE STEPS FORM ONE COMPLETE SEQUENCE TO LOCATE ONE          *
C         IMSC=THE IMMEDIATE STEPPING COST.                                 *
C       *************************************************************************
C
      SUBROUTINE CMBINT2(N,MM,H1,H2,D,NSTEP1,NSTEP2,NCOST1,NCOST2,
     1F1,NEXTPT1,K11STEP,K12STEP,FINAL)
      DIMENSION D(10,10),ID(10),MIR(10,10),F0(10,8),F1(10,8,8),
     1F(10),H1(10,10),H2(10,10),N1(10),N2(10),NSTEP0(10),
     2NSTEP1(10),NSTEP2(10,8),NCOSTC(10,8),NCOST1(10,8),NCOST2
     3(10,8,8),FINAL(10),LABEL(10),INTER(10,10),NOTINTER(10,10),
     4IN(10),NIN(10),K11STEP(10,8,8),K12STEP(10,8,8),NEXTPT1(10,8,8),
     5LCSTEP(10,10),NPATH(10),F3(8),F4(8)
      REAL NCOSTC,NCOST1,NCOST2,LC,N1,N2
C     ************************************************************
C     ************** STEP 1: INITIALIZATION **********************
C     ************************************************************
      NSTEP2(1,1)=2
      NSTEP1(1)=1
      N1(1)=0.
      N2(1)=0.
      F(1)=0.
      NSTEP0(1)=1
      F1(1,1,1)=0.
      F0(1,1)=0.
      NCOSTC(1,1)=0.
      NCOST1(1,1)=0.
      NCOST2(1,1,1)=0.
      NIN(1)=0
      LABEL(1)=1
      DO 300 I=2,N
      LABEL(I)=I
      NSTEP0(I)=2
      NSTEP1(I)=1
      NSTEP2(I,1)=1
      N1(I)=1.E15
```

```
      N2(I)=1.E15
      FO(I,1)=1.E15
      FO(I,2)=1.E15
      F(I)=1.E15
      F1(I,1,1)=1.E15
      NCOSTO(I,1)=0.
      NCOSTO(I,2)=1.E15
      NCOST1(I,1)=1.E15
      NCOST2(I,1,1)=1.E15
      NEXTPT1(I,1,1)=1
      K11STEP(I,1,1)=1
      K12STEP(I,1,1)=1
      LOSTEP(1,I)=1
      IN(I)=0
      NIN(I)=N-2
      K=0
      DO 290 J=2,N
      IF(J.EQ.I)GO TO 290
      K=K+1
      NOTINTER(I,K)=J
  290 CONTINUE
      IF(H1(1,I).GE.NCOST1(I,1))GO TO 300
      NCOST1(I,1)=H1(1,I)
      NCOST2(I,1,1)=H2(1,I)
      N1(I)=H1(1,I)
      N2(I)=H2(1,I)
      F(I)=D(1,I)
      F1(I,1,1)=D(1,I)
      IN(I)=1
      INTER(I,1)=1
  300 CONTINUE
      LPER=2
C     ************************************************************
C     ****** STEP 2: LOCATE NODE LZ WITH WHICH THE IMSC IS ******
C     ******              ASSOCIATED                        ******
C     ************************************************************
      WRITE(5,910)
  310 LR=LPER
      A123=N1(LABEL(LR))
      B123=N2(LABEL(LR))
      C123=F(LABEL(LR))
      DO 320 J123=LPER,N
      I=LABEL(J123)
      IF(N1(I)-A123)317,315,320
  315 IF(N2(I)-B123)318,316,320
  316 IF(F(I)-C123)319,320,320
  317 A123=N1(I)
  318 B123=N2(I)
```

```
  319 C123=F(I)
      LR=J123
  320 CONTINUE
      LZ=LABEL(LR)
      IF(N1(LZ).LE.NCOST1(LZ,NSTEP1(LZ)))GO TO 322
      NSTEP1(LZ)=NSTEP1(LZ)+1
      NCOST1(LZ,NSTEP1(LZ))=N1(LZ)
      NSTEP2(LZ,NSTEP1(LZ))=1
      NEXTPT1(LZ,NSTEP1(LZ),1)=NEXTPT1(LZ,NSTEP1(LZ)-1,NSTEP2(
     1LZ,NSTEP1(LZ)-1))
      K11STEP(LZ,NSTEP1(LZ),1)=K11STEP(LZ,NSTEP1(LZ)-1,NSTEP2
     1(LZ,NSTEP1(LZ)-1))
      K12STEP(LZ,NSTEP1(LZ),1)=K12STEP(LZ,NSTEP1(LZ)-1,NSTEP2(
     1LZ,NSTEP1(LZ)-1))
  322 CONTINUE
      F1(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))=F(LZ)
      NCOST2(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))=N2(LZ)
      J123=NEXTPT1(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))
      J1=K11STEP(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))
      J2=K12STEP(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))
      WRITE(5,990) LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)),N1(LZ),N2(LZ),
     1F(LZ),J123,J1,J2
C     ************************************************************
CCC   ****** STEP 3: EXAMPT LZ FROM FURTHER ********************
CCC   ******           CONSIDERATION           ********************
C     ************************************************************
      IF(F(LZ).GT.FINAL(LZ))GO TO 325
      M123=LABEL(LPER)
      LABEL(LPER)=LZ
      LABEL(LR)=M123
      LPER=LPER+1
C     ************************************************************
CCC   ****** STEP 4: TEST OF PROGRAM TERMINATION ****************
CCC   ************************************************************
C
      IF(LPER.GT.N)GO TO 430
C
CCC   ************************************************************
CCC   ******* STEP 5: FOR I.NE.LZ, UPDATE N1(I),N2(I),F(I)  ***
CCC   ************************************************************
  325 CONTINUE
      IF(NIN(LZ).LE.0)GO TO 380
      J=1
  330 I=NOTINTER(LZ,J)
      IF(I.NE.LABEL(LPER-1)) GO TO 331
      NOTINTER(LZ,J)=NOTINTER(LZ,NIN(LZ))
      NIN(LZ)=NIN(LZ)-1
      J=J-1
```

```
      GO TO 370
  331 B123=N2(LZ)+H2(LZ,I)
      C123=F(LZ)+D(LZ,I)
      DO 335 K=1,NSTEPO(I)
      IF(NCOSTO(I,K).GT.B123)GO TO 337
  335 CONTINUE
  337 IF(C123=FC(I,K=1))340,370,370
  340 LOSTEP(LZ,I)=NSTEP1(LZ)
      IN(I)=IN(I)+1
      INTER(I,IN(I))=LZ
      NOTINTER(LZ,J)=NOTINTER(LZ,NIN(LZ))
      NIN(LZ)=NIN(LZ)=1
      J=J=1
      A123=N1(LZ)+H1(LZ,I)
      IF(A123=N1(I))360,350,370
  350 IF(B123=N2(I))362,352,370
  352 IF(C123=F(I)) 363,370,370
  360 N1(I)=A123
  362 N2(I)=B123
  363 F(I)=C123
      NEXTPT1(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=LZ
      K11STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP1(LZ)
      K12STEP(I,NSTEP1(I),NSTEP2(I,NSTEP1(I)))=NSTEP2(LZ,NSTEP1(LZ))
  370 J=J+1
      IF(J=NIN(LZ))330,330,380
C     **********************************************************
CC    ******* STEP 6: UPDATE THE STEPPING COST FOR NODE LZ ****
C     **********************************************************
  380 NCOST1(LZ,NSTEP1(LZ))=N1(LZ)
      NSTEP2(LZ,NSTEP1(LZ))=NSTEP2(LZ,NSTEP1(LZ))+1
      IF(NSTEP2(LZ,NSTEP1(LZ)).LE.MM)GO TO 379
      WRITE(5,540)
      GO TO 430
  379 IFLAG=0
      K1=1
C     ***************************
CC    ******* MODIFY F=(LZ,C) ******
C     ***************************
      DO 383 K=1,NSTEPO(LZ)
      IF(NCOSTO(LZ,K).LT.N2(LZ))GO TO 382
      IF(FO(LZ,K).LT.F(LZ))GO TO 382
      IF(IFLAG.GT.0)GO TO 383
  381 CONTINUE
      F3(K1)=F(LZ)
      F4(K1)=N2(LZ)
      K1=K1+1
      IFLAG=IFLAG+1
      GO TO 383
```

```
  382 CONTINUE
      F3(K1)=FO(LZ,K)
      F4(K1)=NCOSTO(LZ,K)
      K1=K1+1
      IF(NCOSTO(LZ,K).GE.N2(LZ))GO TO 383
      IF(NCOSTO(LZ,K+1).LE.N2(LZ))GO TO 383
      IF(FO(LZ,K+1).GE.F(LZ))GO TO 383
      GO TO 381
  383 CONTINUE
C     ************************************
CC    ******* UPDATE N1(LZ),N2(LZ),F(LZ) *****
C     ************************************
      F4(K1)=1.E15
      F3(K1)=F3(K1-1)
      NSTEPO(LZ)=K1
      DO 385 K=1,K1
      FO(LZ,K)=F3(K)
  385 NCOSTO(LZ,K)=F4(K)
      N2(LZ)=1.E15
      N1(LZ)=1.E15
      F(LZ)=1.E15
      IF(IN(LZ).LE.0)GO TO 310
      J=1
  390 K=INTER(LZ,J)
      DO 400 K1=LOSTEP(K,LZ),NSTEP1(K)
      DO 400 K2=1,NSTEP2(K,K1)-1
      B123=NCOST2(K,K1,K2)+H2(K,LZ)
      DO 400 K3=1,NSTEPO(LZ)
      IF(B123.GE.NCOSTO(LZ,K3))GO TO 400
      IF((F1(K,K1,K2)+D(K,LZ)).LT.FO(LZ,K3-1))GO TO 410
  400 CONTINUE
      NIN(K)=NIN(K)+1
      NOTINTER(K,NIN(K))=LZ
      INTER(LZ,J)=INTER(LZ,IN(LZ))
      IN(LZ)=IN(LZ)-1
      J=J-1
      GO TO 420
  410 A123=NCOST1(K,K1)+H1(K,LZ)
      C123=F1(K,K1,K2)+D(K,LZ)
      LOSTEP(K,LZ)=K1
      IF(A123-N1(LZ))417,411,420
  411 IF(B123-N2(LZ))418,412,420
  412 IF(C123-F(LZ))419,420,420
  417 N1(LZ)=A123
  418 N2(LZ)=B123
  419 F(LZ)=C123
      NEXTPT1(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))=K
      K1STEP(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))=K1
```

```
      K12STEP(LZ,NSTEP1(LZ),NSTEP2(LZ,NSTEP1(LZ)))=K2
  420 CONTINUE
      J=J+1
      IF(J-IN(LZ))390,390,310
  430 DO 450 J=1,N
      I=LABEL(J)
      DO 450 K=1,NSTEP1(I)
      IF(J.EQ.N.AND.K.EQ.NSTEP1(I))GO TO 450
      NSTEP2(I,K)=NSTEP2(I,K)=1
  450 CONTINUE
  990 FORMAT(T2,3(I2,3X),3(F5.2,3X),3(I2,3X))
  910 FORMAT(T2,'THE J TH 1=I TYPE 1 STEPPING COST, C1(I,J), THE'
     1/T2,'1=I TYPE 2 COST,C2(I,J,K), ASSOCIATED WITH C1(I,J),'
     2/T2,'AND THE 1=I STEPPING=COST PATH, F1(I,J,K), ASSOCIATED'
     3/T2,'WITH C1(I,J) AND C2(I,J,K), ARE INTERRELATED SUCH THAT:'//
     4T2,'F(I,J,K)=F(I1,J1,K1)+D(I1,I)',
     5/T2,'C1(I,J)=C1(I1,J1)+H1(I1,I)'/
     6T2,'C2(I,J,K)=C2(I1,J1,K1)+H2(I1,I)'///
     3T2,' I',4X,'J',4X,'K',6X,'C1',6X,'C2',6X,'F',4X,'I1',3X,
     4'J1',3X,'K1'//)
  920 FORMAT(T15,'NO SUCH PATH')
  940 FORMAT(T15,'THE NUMBER OF STEPPING COSTS HAS EXCEEDING'
     1/T15,'PROVIDED STORAGE LIMIT'//)
      RETURN
      END
```

```
C     ***********************************************************************
C     SUBROUTINE SELECT SELECTS THE SHORTEST STEPPING-COST PATH              *
C     F1(I,J,K) SUCH THAT THE TYPE 1 AND TYPE 2 COSTS SATISFY THE            *
C     CONSTRAINT:                                                            *
C     (LO=NCOST1(I,J)/SQRT(NCOST2(I,J,K)))*GT*FRACTION*                      *
C     ***********************************************************************
      SUBROUTINE SELECT(N,NSTEP1,NSTEP2,NCOST1,NCOST2,F1,NODE,
     1NOSTEP1,NOSTEP2,FRACTION,LO,AA)
      DIMENSION NSTEP1(10),NSTEP2(10,8),NCOST1(10,8),NCOST2(10,8,8),
     1F1(10,8,8)
      REAL NCOST1,NCOST2,LO
      A=1.E15
      NOSTEP1=0
      NOSTEP2=0
      DO 20 I=NSTEP1(NODE),1,-1
      B=LO*NCOST1(NODE,I)
      DO 10 J=NSTEP2(NODE,I),1,-1
      C=B/SQRT(NCOST2(NODE,I,J))
      IF(C*GT*FRACTION) GO TO 15
   10 CONTINUE
      GO TO 20
   15 IF(A*LE*F1(NODE,I,J))GO TO 20
      A=F1(NODE,I,J)
      NOSTEP1=I
      NOSTEP2=J
      AA=C
   20 CONTINUE
      RETURN
      END
```

```
C     ***********************************************************
CC    SUBROUTINE TRACES THE OPTIMAL PATH F1(NODE,NOSTEP1,NOSTEP2)  *
C     ***********************************************************
      SUBROUTINE PATH(N,NODE,NOSTEP1,NOSTEP2,NEXTPT1,K11STEP,K12STEP,
     1NPATH, J)
      DIMENSION NEXTPT1(10,8,8),K11STEP(10,8,8),K12STEP(10,8,8),
     1NPATH(10)
      REAL NCOST1,NCOST2
      NPATH(1)=NODE
      L1=NOSTEP1
      L2=NOSTEP2
      DO 10 J=2,N
      NPATH(J)=NEXTPT1(NPATH(J-1),L1,L2)
      IF(NPATH(J).EQ.1) RETURN
      L1=K11STEP(NPATH(J-1),L1,L2)
   10 L2=K12STEP(NPATH(J-1),L1,L2)
      RETURN
      END
```

PROGRAM TO FIND THE MIN=COST 1= 5 PATH SUCH THAT
THE PROBABILITY OF THE PATH TRANSIT TIME EXCEEDING A GIVEN
VALUE 2.50 BE NOT GREATER THAN A GIVEN
FRACTION 1-N(2.00).

********* COST MATRIX **********

	1	2	3	4	5
1	.00E 00	.59E 01	.60E 00	.15E 01	.10E 16
2	.10E 16	.00E 00	.10E 01	.10E 16	.19E 01
3	.10E 16	.15E 01	.00E 00	.10E 01	.53E 01
4	.10E 16	.24E 01	.10E 16	.00E 00	.41E 01
5	.10E 16	.10E 16	.10E 16	.10E 16	.00E 00

********* TRANSIT MEAN MATRIX **********

	1	2	3	4	5
1	•0CE 00	•50E 00	•50E C0	•50E 00	•10E 16
2	•10E 16	•00E 00	•50E C0	•10E 16	•50E 00
3	•10E 16	•10E 01	•00E C0	•10E 01	•10E 01
4	•10E 16	•10E 01	•10E 16	•00E 00	•50E 00
5	•10E 16	•10E 16	•10E 16	•10E 16	•00E 00

********** TRANSIT VARIANCE MATRIX **********

	1	2	3	4	5
1	•00E 00	•10E 00	•10E 00	•10E 00	•10E 16
2	•10E 16	•00E 00	•00E 00	•10E 16	•00E 00
3	•10E 16	•50E 00	•00E 00	•10E 00	•00E 00
4	•10E 16	•10E 00	•10E 16	•00E 00	•30E 00
5	•10E 16	•10E 16	•10E 16	•10E 16	•00E 00

THE J TH 1=I TYPE 1 STEPPING COST, C1(I,J), THE
1=I TYPE 2 COST,C2(I,J,K), ASSOCIATED WITH C1(I,J),
AND THE 1=I STEPPING=COST PATH, F1(I,J,K), ASSOCIATED
WITH C1(I,J) AND C2(I,J,K), ARE INTERRELATED SUCH THAT:

F(I,J,K)=F(I1,J1,K1)+D(I1,I)
C1(I,J)=C1(I1,J1)+H1(I1,I)
C2(I,J,K)=C2(I1,J1,K1)+H2(I1,I)

I	J	K	C1	C2	F	I1	J1	K1
3	1	1	.50	.10	.60	1	1	1
4	1	1	.50	.10	1.50	1	1	1
2	1	1	.50	.10	5.90	1	1	1
5	1	1	1.00	.10	7.80	2	1	1
5	1	2	1.00	.40	5.60	4	1	1
5	2	1	1.50	.10	5.90	3	1	1
5	2	1	1.50	.20	3.90	4	1	1
5	2	2	1.50	.60	2.10	3	1	1
5	3	1	2.00	.20	5.80	2	2	1
5	3	2	2.00	.60	4.00	2	2	2

THE OPTIMAL PATH WITH MEAN OF TRAVERSE TIME 1.00
VARIANCE OF TRAVERSE TIME .40 AND PATH COST 5.60
IS THE SEQUENCE OF NODES: 1 4 5

LITERATURE CITED

1 Dantzig, G. B. Linear Programming and Extensions. Princeton, New Jersey: Princeton University Press, 1963.

2 Bellman, R. Dynamic Programming. Princeton, New Jersey: Princeton University Press, 1957.

3 _____, and Dreyfus, S. Applied Dynamic Programming. Princeton, New Jersey: Princeton University Press, 1959.

4 Ford, L. R., Jr., and Fulkerson, D. R. Flows in Networks. Princeton, New Jersey: Princeton University Press, 1962.

5 Yaged, B. "Minimum Cost Routing for Static Network Problems." Networks, Vol. 1 (1971), 139-172.

6 Stagg, G. W., and Watson, R. M. "Dynamic Programming Transmission Line Design." IEEE International Convention Record, 12, Part 3, 55-61.

7 Zadeh, N. "Construction of Efficient Tree Networks: The Pipeline Problem." Networks, Vol. 3, No. 1 (1972), 1-32.

8 Rothfarb, B.; Frank, H.; Rosenbaum, D. M; Steiglitz, K.; and Kleitman, D. J. "Optimal Design of Offshore Natural-Gas Pipeline Systems." Oper. Res., Vol. 18, No. 6 (1970), 992-1020.

9 Lee, C. Y. "An Algorithm for Path Connections and Its Applications." IRE Trans. Electron. Compt., Vol. EC-10 (1961), 346-365.

10 Hanan, M. "Layout, Interconnection, and Placement." Networks, Vol. 5, No. 1 (1975), 86-88.

11 Cederbaum, I. "Optimal Backboard Ordering Through the Shortest Path Algorithm." IEEE Trans. on Circuits and Systems, Vol. CAS-21, No. 5 (1974), 626-632.

12 Frank, H.; Frisch, I. T.; Van Slyke, R.; and Chou, W. S. "Optimal Design of Centralized Computer Networks." Networks, Vol. 1, No. 1 (1971), 43-57.

13 Zadeh, N. "On Building Minimum Cost Communication Networks Over Time." Networks, Vol. 4, No. 1 (1974), 19-34.

14 Fratta, L.; Gerla, M.; and Kleitman, L. "The Flow Deviation Method: An Approach to Store-and-Forward Network Design." Networks, Vol. 1, No. 2 (1972), 97-134.

15 Fultz, G., and Kleinrock, L. "Adaptive Routing Techniques for Store-and-Forward Computer-Communication Networks." Proc. Int. Conf. Communications, (1971), 39-1-39-8.

16 Frank, H., and Chou, W. S. "Routing in Computer Networks." Networks, Vol. 1, No. 2 (1971), 99-112.

17 Van Slyke, R., and Frank, H. "Network Reliability Analysis-1." Networks, Vol. 1, No. 3 (1972), 279-290.

18 _____. "Recursive Analysis of Network Reliability." Networks, Vol. 3, No. 1 (1973), 81-94.

19 Frisch, I. T. "Optimal Routes in Communication Systems with Channel Capacities and Channel Reliabilities." IEEE Trans. Comm. Systems, CS-11 (1963), 241-244.

20 Frank, H., and Chou, W. S. "Topological Optimization of Computer Networks." Proc. of IEEE, Vol. 60, No. 11 (1972), 1385-1397.

21 Bellman, R. "A Routing Problem." Quart. Appli. Math., No. 16 (1958), 87-90.

22 _____; Cooke, K. L.; and Lockett, J. A. Algorithms, Graphs, and Computers. New York: Academic Press, 1970.

23 Whiting, P. D., and Hillier, J. A. "A Model for Finding the Shortest Route Through a Road Network." Oper. Res. Quart., Vol. 11, No. 5 (1961), 37-40.

24 Davis, D. W., and Barber, D. L. A. Communication Networks for Computers. New York: John Wiley & Sons, 1973.

25 Parikh, S. C., and Frisch, I. T. "Finding the Most Reliable Routes in a Communication System." IEEE Trans. Comm. Systems, CS-11 (1963), 402-407.

26 Wing, O. "Algorithm to Find the Most Reliable Path in a Network." IRE Trans. Circuit Theory, CT-8 (1961), 78.

27 Cederbaum, I. "On Optimal Routing Through Communication Nets." IEEE Trans. on Communication, A-8 (1973), 936-941.

28 Frank, H. "Shortest Paths in a Probabilistic Graph." Oper. Res., 17 (1968), 583-599.

29 _____, and Hakimi, S. L. "Probabilistic Flows Through A Communication Network." IEEE Trans. on Circuit Theory, CT-12 (1965), 413-414.

30 Cooke, K. L., and Halsey, E. "The Shortest Route Through a Network with Time-Dependent Internodal Transit Time." J. Math. Anl. Appl., 14 (1966), 493-498.

31 Halpern, J., and Priess, I. "Shortest Paths with Time Constraints on Movement and Parking." Networks, Vol. 4, No. 3 (1974), 241-254.

32 Saksena, J. P., and Kumar, S. "The Routing Problem with 'K' Specified Nodes." Oper. Res., Vol. 14 (1966), 909-913.

33 Kibby, F. R., and Potts, R. B. "The Minimum Route Problem with Turn Penalties Prohibition." Trans. Res., 3 (1969), 397-408.

34 Dijkstra, E. W. "A Note on Two Problems in Connections with Graphs." Nemerische Math., 1 (1959), 269-271.

35 Pierre, D. A. Optimization Theory with Applications. New York: John Wiley & Sons, 1969.

36 Ford, L. R., Jr. Network Flow Theory. The Rand Corporation, 1956.

37 Moore, E. F. "The Shortest Path Through a Maze." Proc. Int. Sym. on the Theory of Switching, Part 2 (1957).

38 Yen, J. Y. "An Algorithm for Finding Shortest Routes from all Source Nodes to a Given Destination in General Networks." Quart. Appl. Math., Vol. 27, No. 4 (1970), 526-530.

39 _____. "Algorithms for Finding the Shortest Paths Through the General Networks." Second Int. Conf. on Computing Methods in Optimization Problems, San Remo, Italy (1968).

40 Floyd, R. W. "Algorithm 97, Shortest Path." Comm. A.C.M., 5 (1962), 345.

41 Shimbel, A. "Structure in Communication Nets." Proc. of Sym. on Information Networks, Polytechnic Institute of Brooklyn, (Apr., 1954), 12-14.

42 Warshall, S. "A Theorem on Boolean Matrices." J. A.C.M., Vol. 9, (1962), 11-12.

43 Yen, J. Y. "Finding the Lengths of all Shortest Paths in N-Node Nonnegative-Distance Complete Networks Using 1/2N**3 Additions and N**3 Comparisons." J. A.C.M., Vol. 19, No. 3 (1972), 423-424.

44 Hoffman, A. J., and Winograd, S. "Finding All Shortest Distances in a Directed Network." IBM J. Res. Dev., 16 (1972), 412-414.

45 Hu, T. C. "A Decomposition Algorithm for Shortest Paths in a Network." Oper. Res., 16 (1968), 91-102.

46 Land, A. H., and Stairs, S. W. "The Extension of the Cascade Algorithm to Large Graphs." Mgmt. Sci., 14 (1967), 29-33.

47 Mills, G. "A Decomposition Algorithm for the Shortest-Route Problem." Oper. Res., 14 (1966), 279-291.

48 Hoffman, W., and Pavley, R. "A Method for the Solution of the Nth Best Path Problem." J. Assoc. Comp. Mach., 6 (1959), 506-514.

49 Frank, H., and Frisch, I. T. Communication, Transmission, and Transportation Networks. Reading, Massachusetts: Addison-Wesley, 1971.

50 Bellman, R., and Kalaba, R. "On k^{th} Best Policies." J. Siam, 8 (1960), 582-588.

51 Dreyfus, S. E. An Appraisal of Some Shortest Path Algorithms. RM-5433-PR. Santa Monica, California: Rand Corporation, Oct. 1967.

52 Murty, K. G. "An Algorithm for Ranking All the Assignments in Increasing Order of Cost." Oper. Res., Vol. 16 (1968), 682-687.

53 Yen, J. Y. "Finding the k Shortest Loopless Paths in a Network." Mgmt. Sci., Vol. 17, No. 11 (1971), 712-716.

54 Lawler, E. L. "A Procedure for Computing the k Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem." Mgmt. Sci., Vol. 18, No. 7 (1972), 401-405.

55 Fitz, M. Probability Theory and Mathematical Statistics. New York: John Wiley & Sons, 1963.

56 Ford, L. R., and Fulkerson, D. R. "Maximal Flow Through a Network." Can. J. Math., 18 (1956), 399-404.

57 Dantzig, G. B., and Fulkerson, D. R. "On the Max-Flow Min-Cut Theorem of Networks." In "Linear Inequalities and Related Systems." Ann. Math. Studies, 38 (1956), 215-221.

58 Elias, P.; Feinstein, A.; and Shannon, C. E. "A Note: The Maximum Flow Through a Network." IRE Trans. Inf. Theory, IT-2 (1956), 117-119.

MONTANA STATE UNIVERSITY LIBRARIES
3 1762 10011228 1

[redacted]
D378
Sh42
cop.2

Shek, Chi-Hung
Constrained shortest paths in networks

DATE | ISSUED TO
MAR 18 RELord Comp Sci
2 WEEKS USE INTERLIBRARY LOAN
5-8-87
[redacted]
D378
Sh42
cop 2