NETWORK FLOW MODELING
OF
MULTIRESERVOIR DISTRIBUTION SYSTEMS

Final Report
to
Office of Water Resources Research
Department of the Interior
Grant Number 14-31-0001-3656
Project Number B-107-TEX

by

Paul A. Jensen
Gora Bhaumik
William Driscoll

Center for Research in Water Resources
The University of Texas at Austin

CRWR-107

June 1974

ABSTRACT

The operating policy for a multireservoir distribution system is determined by many factors, including benefits and priorities for the current use of water, size and use limitations of the facilities of the system, costs and delays in transmitting water through the system and the current and future weather. Some of these factors have been modelled using a network flow model which allows determination of optimum policies for complex systems. This report describes three extensions to the current modelling capabilities. The first part discusses the water losses which occur in the system and proposes a computerized algorithm which can solve networks with losses. The second part treats the same problem but derives a much more efficient algorithm. The third part considers the uncertainty inherent in estimates of future water supplies and demands and provides a representation for this uncertainty for the network model. The general approach is to calculate a mathematical expression which determines the value of water stored for the future. This work expands the generality of the network flow models as related to water systems and also makes contributions to the general theory of network flow analysis.

Acknowledgments

The authors are indebted to peel in the Systems Engineering Division of Texas Water lopment Board. Of particular support was Mr. Arden Weho was instrumental in the early use of network approaches, Dr. Quentin Martin who is incorporating the network wiains code developed in this study in the planning packages he Water Development Board.

The Center for Water Resources the University of Texas was responsible for providing the mo part of the matching funds for this research. Certainlithout this support and the able aid of the CRWR staff thiork could not have been pursued. The administration and sff of the Mechanical Engineering Department of The Univers of Texas aided considerably in the management of the project, student support, and in secretarial services. This reporvas typed by Ms. Peggy Deinken and Ms. Susan Patterson.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

OPTIMUM OPERATING POLICIES OF A WATER DISTRIBUTION SYSTEM WITH LOSSES

A COMPUTATIONALLY EFFICIENT ALGORITHM FOR THE NETWORK WITH GAINS PROBLEM

LIST OF TABLES

Table Page

OPTIMUM OPERATING POLICIES OF A WATER DISTRIBUTION SYSTEM WITH LOSSES

A COMPUTATIONALLY EFFICIENT ALGORITHM FOR THE NETWORK WITH GAINS PROBLEM

INTRODUCTION

Within the last decade many mathematical models have been proposed to represent multireservoir distribution systems. Comprehensive reviews of this work are presented later in this report. Models have been formulated for the design of the system in terms of locating, sizing and timing of the various facilities, for the determination of the optimum use of limited quantities of water in a reservoir, and for planning the operating policies which best balance the present and future uses of water.

Any model is an abstraction of the real world situation. The least abstract models bear a close resemblance to the real world and hence would generally be expected to have the greatest accuracy in detail and the most useful outputs. Alternatively these models would require the most input data, involve the greatest effort in model preparation and probably they require the greatest computational effort for solution. The most abstract of models would bear little detailed similarity to the real world situation. However to be useful at all an abstract model must bear some resemblance to some important aspect of the situation modelled. Although such a model is inaccurate in many details it is hopefully accurate in the aspect for which it is designed and its output would be expected to be useful for the control of that aspect. Such a model will generally be easier for model preparation and require minimal computational effort for solution. The analyst faced with these tradeoffs when constructing a model will usually find the model with the least abstraction possible within his data and computational limitations.

For some applications computational effort is an important consideration in the choice of the model form. This is true for capacity expansion studies which require the determination of the location, time of construction and size of each of a number of facilities in a complex water system. Planning horizons on the order of fifty years are usual for such problems. In order to evaluate a particular plan it is necessary to determine the operating cost for the system over the time horizon. This has been done by the Texas Water Development Board using a mathematical model of the system for discrete time steps of one month. For a fifty year horizon such a model requires 600 monthly models. Operating costs are determined by applying an optimization algorithm to this model. Even if the model is linear the computational effort and cost required to solve such a large problem is significant. If a large number of expansion alternatives are to be considered computational cost is the limiting factor to the analysis.

For the long range planning function it is not necessary that the model used to estimate operating cost be completely accurate with regard to detailed water movements within the system. It is only important that the model provide a reasonable estimate of operating cost. For example it is not unreasonable to impose historical rain fall data on a model of a proposed system to evaluate its operating cost and effectiveness. Historical data has the important characteristic of periods of drought and flood which a new system will also experience. The difficulty with this approach is that mathematical optimization algorithms have the capacity to look ahead and operate the system to anticipate periods of drought or flood. Thus flows in the system model are directed in ways that do not reflect the operation of a real system in which foreknowledge of the weather is not entirely possible. Estimates of operating costs obtained in this way are considered to be reasonable. Optimization with deterministic inflows is used in the analysis because the consideration of uncertainty in inflows is computationally difficult.

Water distribution systems can conveniently be represented by a network flow model. Thus the arrangements of canals, reservoirs and river reaches can be represented by an abstraction of nodes and arcs which form a network. The network representation allows flow of water in both time and space. Time flows represent storage in reservoirs and space flows represent transfer of water from one part of the system to another. When inflows and outflows to the system are given, algorithms exist to determine the optimum flow in the network. These algorithms are generally very efficient computationally and hence are useful for the solution of very large models. The network flow models and algorithms have been used extensively by the Texas Water Development Board for planning studies.

The primary disadvantage of the network flow models is the limited set of constraints that can be represented. The only constraints that are explicitly represented are conservation of flow at each node and upper and lower bounds on flow for each arc. This report attempts to expand the modeling capabilities of network flow models in two ways; first by allowing flow to be lost or gained as it traverses an arc and second to allow the consideration of uncertainty in inflows and demands for the system.

The report is divided into three main sections. The first section discusses the network model for water systems with losses such as evaporation or seepage. The model with this extension is called the networks with gains model. Here a gain on an arc may be any positive quantity and loss is represented by a fractional gain. This section presents some theoretical results concerning the optimization of the flows for such a model and describes a rudimentary algorithm for optimization.

The second section of the report presents an advanced optimization algorithm for the network with gains model. This algorithm is superior to the algorithm presented in the first section in respect to computational speed and memory requirements. It is of course also more complex. The third section of the report describes an attempt to represent uncertainty of inflows and demands for a water system. This is accomplished by determining a value function for water stored for the future. The value of stored water function is used to determine operating policies at each period. The approach suggested in this section should be useful for both planning studies and for actual determination of operating policies on a real time basis.

OPTIMUM OPERATING POLICIES OF A WATER DISTRIBUTION SYSTEM WITH LOSSES

by

Gora Bhaumik
and
Paul A. Jensen

I. Flow in a Water Distribution System with Losses

I.1 Introduction

In the last few years the Texas Water Development Board (TWDB) has been engaged in a number of studies to determine the policies for optimally managing the water resources of the State of Texas [43, 44, 45]. In analyzing the problem they have determined the total water resources available in the state and have developed projections of water requirements for the next fifty years. To meet the differences between the supply and demand of water in the different regions, the TWDB proposed in 1968 the Texas Water Plan [45]. It consists of a system of reservoirs, rivers, canals, and pumping stations to distribute the water from the surplus areas of East Texas and imported water from Louisiana to the needy parts of the West and Southwest. This is shown schematically in Figure 1 and Figure 2.

The storage and conveyance facilities of the Texas Water System can be conveniently modeled by a network. This network model has been used extensively in a number of TWDB simulation studies [43, 44]. They have analyzed the economic effects of different sizes and locations of reservoirs and canals to meet the demand schedule at the lowest reasonable cost. This is done by determining the optimal storage and shipping policies for a set of reservoirs and canals and iteratively improving the configuration until a solution is reached which minimizes the present worth of construction costs, operating costs, and maintenance costs.

An important assumption of this model is that of conservation of flow at all points within the network. This precludes losses from the system such as spillage, evaporation and seepage. Although, there is provision for forcing a known loss from every reservoir, this loss must be prespecified before the flows can be determined. Therefore, the losses which are a function of the flows in the network are not properly reflected in the present model of the system. It is necessary to incorporate this loss function of the system components in the network model. The economic distribution policy depends not only

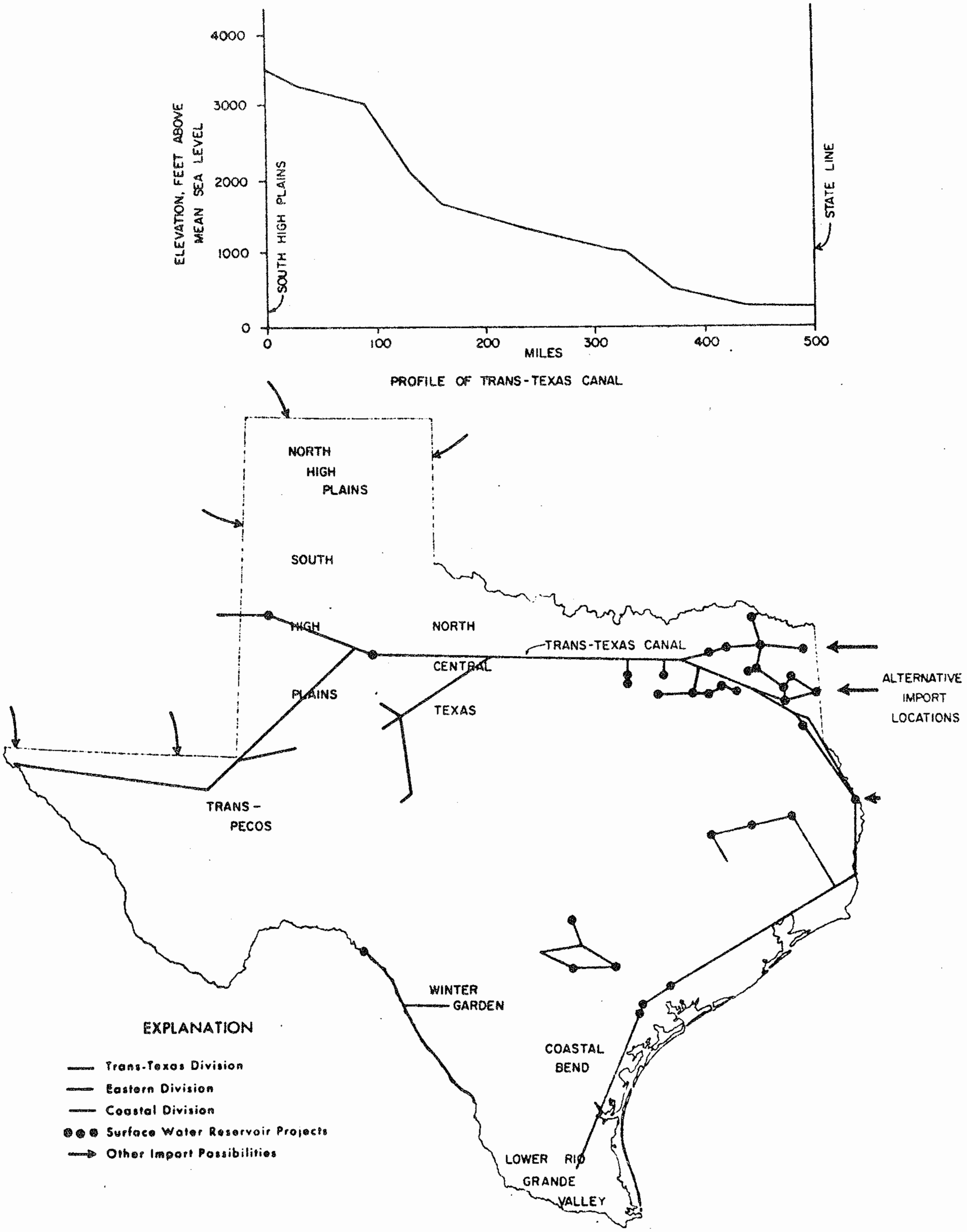


Figure 1

Schematic Diagram of the Texas Water System

(Includes major conveyance facilities and related reservoirs)

(Source: Texas Water Development Board)

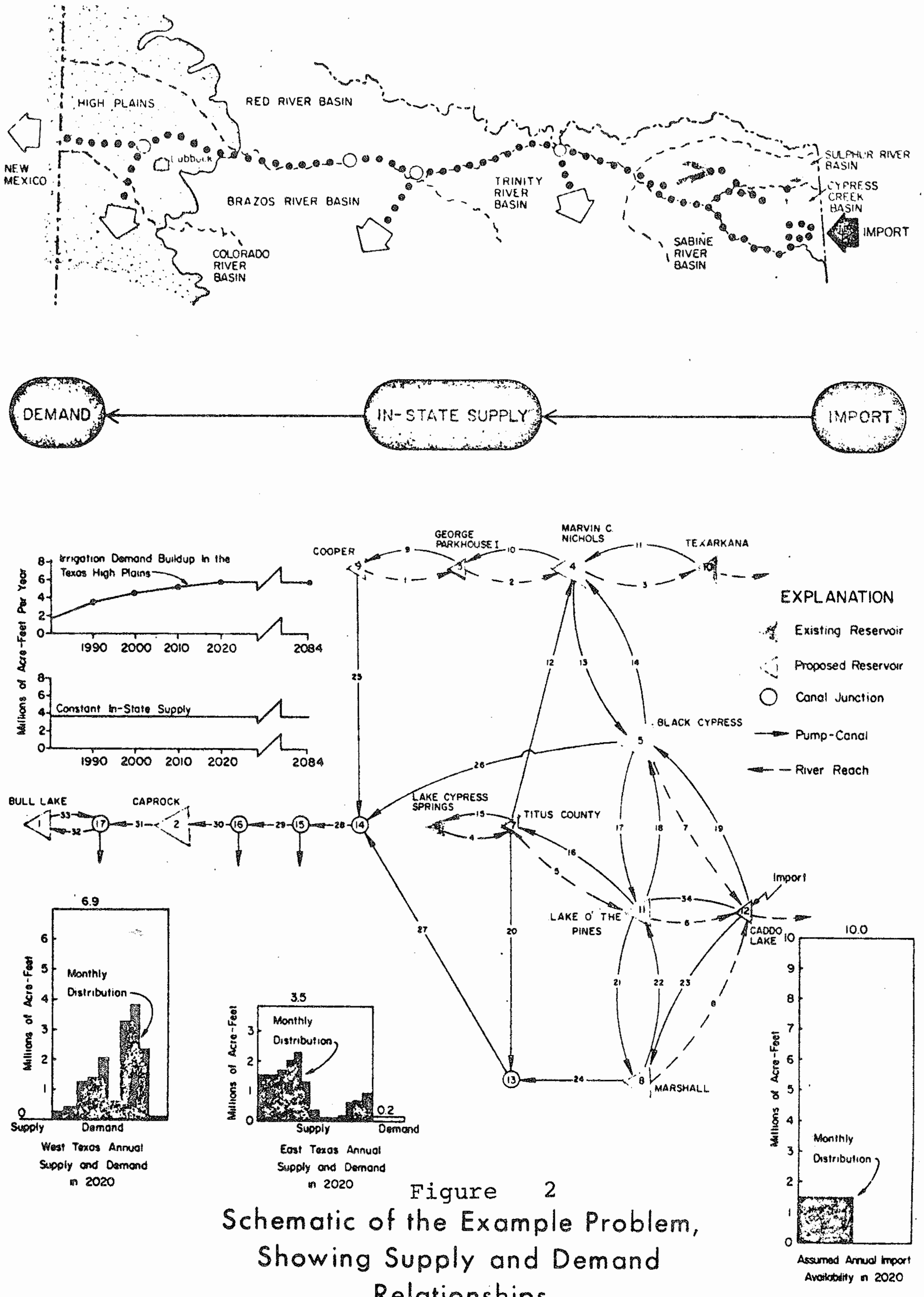


Figure 2

Schematic of the Example Problem, Showing Supply and Demand Relationships

(Source: Texas Water Development Board)

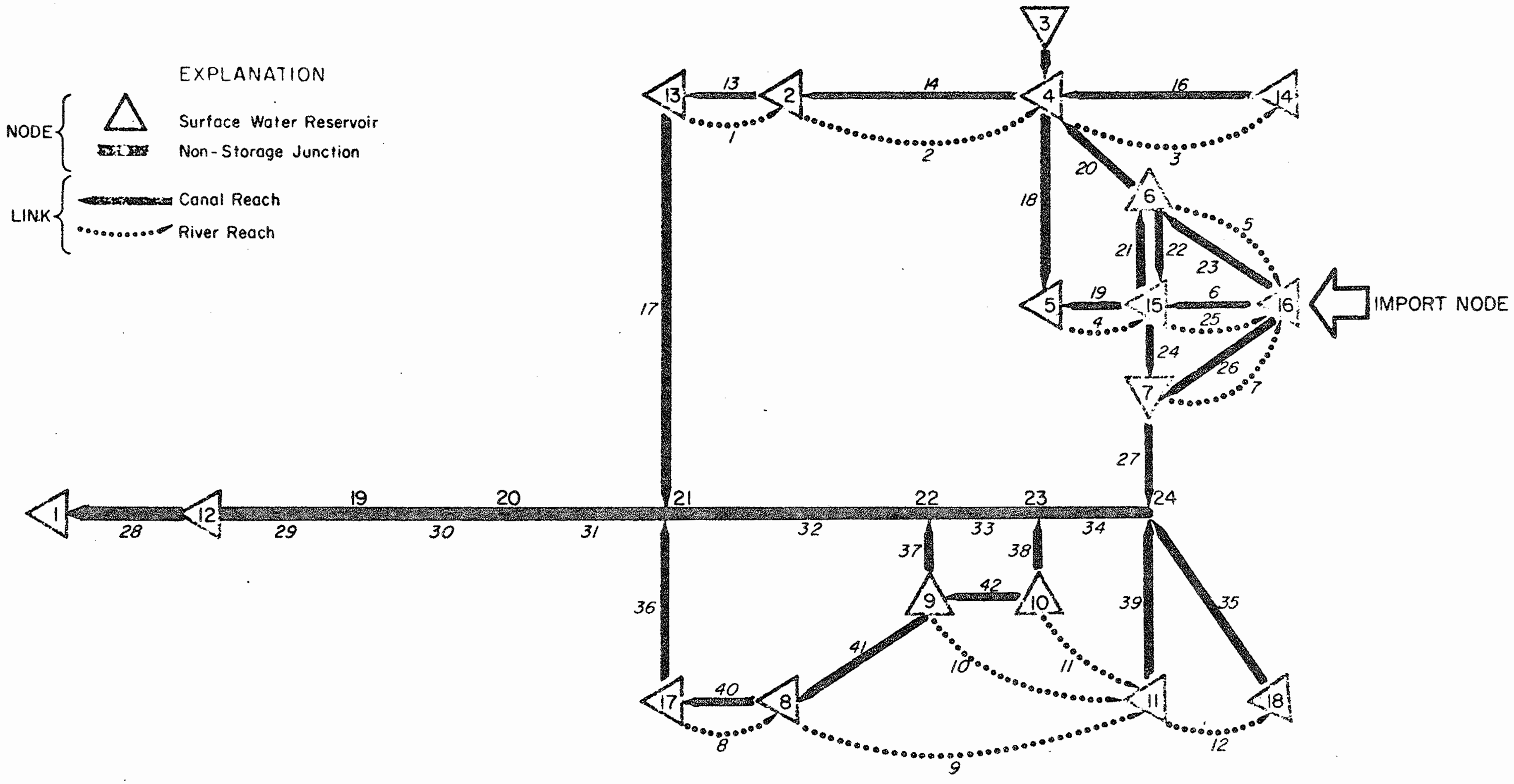


Figure 3

SCHEMATIC REPRESENTATION OF THE TRANS TEXAS RESERVOIR - RIVER - CANAL SYSTEM

(Source: The Texas Water Development Board) on the cost of transportation and storage of water but also upon the loss characteristics of the canals and reservoirs. Hence, evaporation and seepage may play a decisive role in determining the optimal operating policies of a water distribution system with losses.

To generalize the proposed model, the term 'gain' will be used instead of losses. Losses may be defined to be fractional gains. The goal of this study is to develop an algorithm to determine the optimal flows in a network with gains. It should be comparable in speed and efficiency to the algorithms for pure networks or networks without gains. The model for networks with gains should include all the characteristics of the pure network model in addition to the gain parameter. In particular, this model should adequately describe a water distribution system by including the losses associated with storing and transporting water.

The proposed model is similar to models in other fields such as in electrical power transmissions. In power transmission grids part of the power is lost due to electrical resistance of the transmission lines [16]. Still, there are other examples cited by Jewell [26] which can be modeled by a network with gains. These include the machine loading problem [40], warehousing with 'breeding' and 'evaporation' [28], financial budgeting problem [8], aircraft routing problem [11], and the catering problem [39]. In general, these problems can be represented with the help of a network which includes a gain factor with each arc in addition to the other cost and bound parameters. The gain can be any non-negative quantity. If the gain on an arc is less than unity it signifies that part of the flow along that arc is lost and if it is greater than one it indicates that the flow is amplified (as in the machine loading problem [40]) on passing through that arc.

The Network Model with Gains [26] is also known as Generalized Networks [9] or Lossy Networks [16]. These are a class of network problems for which there is a strong need for a fast and simple algorithm. Hence, the proposed algorithm is not only applicable to water distribution systems with losses but also to a large number of similar models which include a gain with each arc.

I.2 The Network Model

The physical facilities comprising The Texas Water System are the canals, storage and regulatory reservoirs, river basins and pumping stations. In the TWDB studies [43, 44, 45] the water distribution system has been represented by a directed network. The canals and river reaches which distribute the water form the arcs and the storage reservoirs and pumping stations are represented by nodes. Figure 3 shows part of this reservoir-river-canal system.

There are three important parameters associated with each of the arcs, the upper and lower bounds on the flow and the cost per unit of flow along the arc. The upper bound on the flows are determined by the capacities of the canals and rivers. The unit cost is given by the cost of transporting one unit of water across the corresponding canal. The operating policies are determined on a monthly basis over a fifty year time span. To represent these six hundred time periods, a similar network is considered for each month. Storage of water is represented by joining the corresponding nodes which represent the same reservoir with an arc. A simplified example of such a system is shown in Figure 4a. This example consists of six nodes (four reservoirs and two link junctions) and eight arcs (seven canal links and one river reach). This node-arc representation is expanded spacially to include four time periods as shown in Figure 4b. The networks for each of the time periods are connected by the storage arcs. Thus, the time-space representation of the problem can be envisioned as a layered network, each layer representing a time period with reservoir storage contents connecting the layers [44]. These carry over storage arcs are also called inventory arcs. Water stored from one month to the next appears as a flow along these arcs. The upper bound on these arcs are the limiting storage capacities of the reservoirs and the unit cost is the cost of storing water over one time period.

This expanded network still does not represent the problem. The system must have some initial reservoir storage contents; inputs to and demands from the system must be made; imports must be allowed to enter the system; and after the last time period, provisions must be made for the final reservoir storage contents. All these are accommodated by adding additional arcs and nodes with the corresponding bounds and costs. The problem, now, is to determine the flows which minimize the cost. A number of algorithms, such as the Out-of-Kilter algorithm of Ford and Fulkerson [12, 14, 17], exist which can solve this problem very efficiently.

To include the evaporation and seepage characteristics of the canals and reservoirs an additional parameter, the gain factor, may be introduced for each of the arcs. The gain factor, or simply gain, is defined to be the fraction of flow that is transmitted through an arc. If the gains for all the arcs are equal to one then the problem is the same as the pure network problem. The pure network algorithms can not solve network problems with gains. A number of algorithms have been proposed for this generalized network problem by Jewell [26], Johnson [27], Charnes and Raike [9], Fujisawa [16], Glover, Klingman and Napier [21] and Maurras [34]. Most of these are either not applicable to the water distribution problem or are too complex to be easily implemented. The algorithm presented in this report is simple enough for hand computations and fast enough on the digital computer to be viable for large network problems with gains.

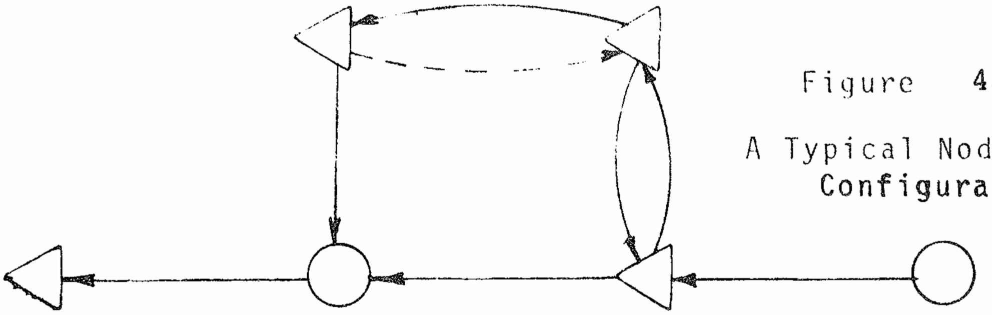


Figure 4a

A Typical Node-Link Configuration

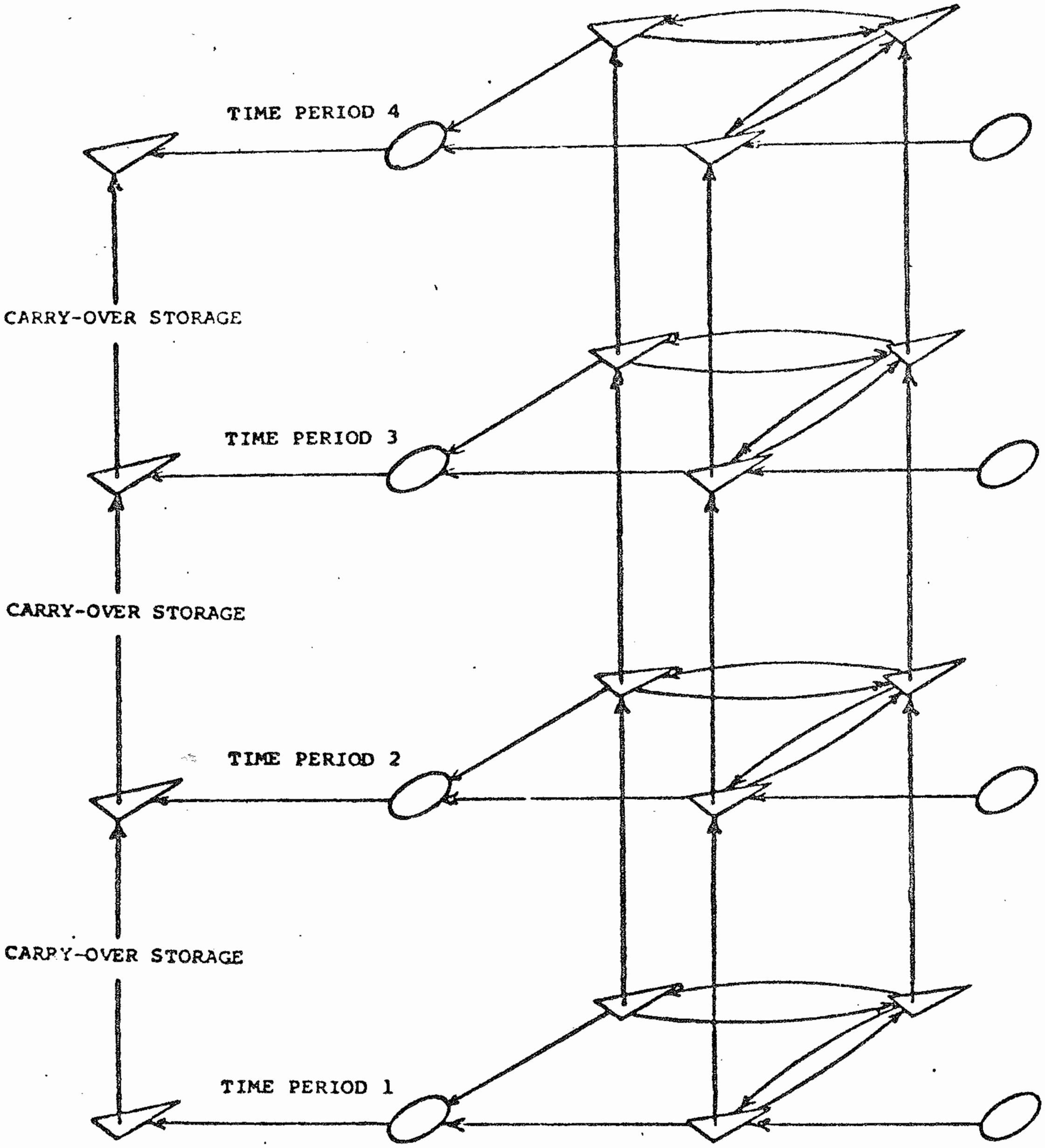


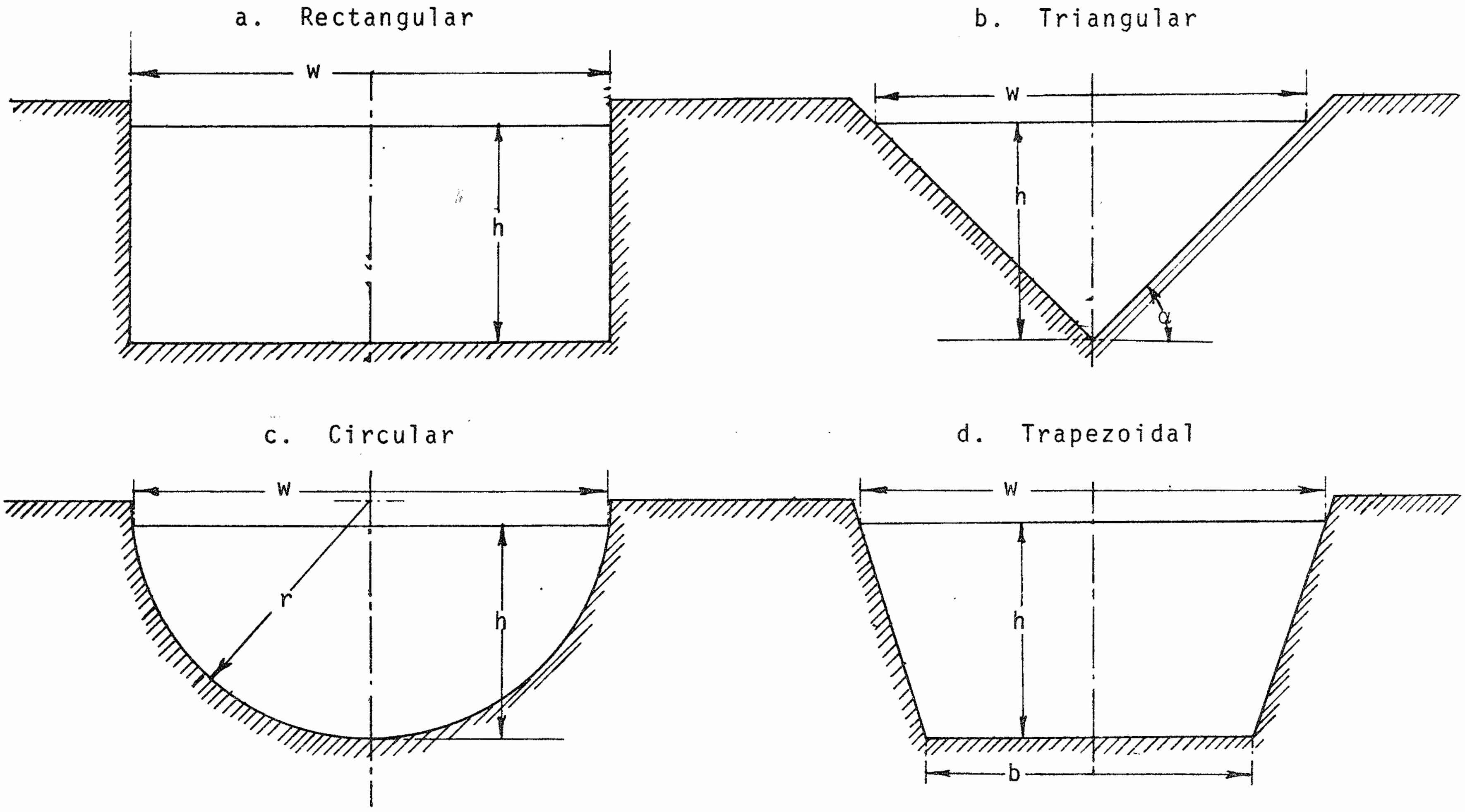
Figure 4b System Network for a Four Time Period Problem
(Source: Texas Water Development Board)

I.3 Evaporation and Seepage

In a water storage and distribution system, such as the Texas water system, evaporation and seepage losses play a major role in determining the optimal operating policies of the system [41, 42]. Anticipated evaporation can be a crucial element in the design of reservoirs in arid regions such as West Texas. Evaporation losses in this area are known to be as high as one-third the annual inflow. Likewise, seepage of water into the soil from unlined canals or from reservoirs located in areas where the ground water level is low can play a decisive role in determining their location and utilization. Therefore, the estimation and accounting for these factors are a very important function of the planning and operation stages of a water distribution system.

On a free water surface there is a continuous exchange of water molecules between the water and atmosphere. This rate of exchange is governed by the temperature and vapor pressure at the surface of the water. If the temperature increases or the vapor pressure decreases, the rate of transfer of water molecules from the water to the atmosphere increases. The evaporation from the surface of a large reservoir is highly dependent upon these two factors. The increase of temperature of a body of water is usually governed by solar radiation. Therefore, evaporation is usually higher during the summer months as compared to the winter months. The vapor pressure of the overlying air is dependent upon the relative humidity and the wind over the surface of the water. Dry air increases evaporation due to lower vapor pressure. On the water surface a layer of vapor is built up due to evaporation. When wind blows over this surface, this partially saturated layer of vapor is blown away, thereby increasing the rate of evaporation. The evaporation rate is greater at the leading edge of the wind because of a wedge of vapor that is formed down-wind. An increase in wind speed increases evaporation. At the Aswan reservoir in Egypt, wind has been credited with significantly increasing the evaporation rate.

The relative effects of controlling meterological factors is difficult to evaluate. However, it can be stated that the rate of evaporation is influenced by solar radiation, air temperature, vapor pressure, wind and possibly atmospheric pressure. Solar radiation is an important factor; evaporation varies with latitude, season, time of day, and sky conditions. Direct measurement of evaporation under field conditions is not feasible, at least not in the sense that one is able to measure river stage, discharge and so on. As a consequence a variety of techniques have been developed for determining or estimating vapor transport from water surface. Reference [31] describes several indirect procedures for estimating the evaporation rate per unit area. The total evaporation rate for a body of water is the product of the evaporation rate and surface area. The gain factor k (fraction of flow transmitted) can be estimated from:



a. Rectangular

b. Triangular

c. Circular

d. Trapezoidal

Figure 5 Lateral Cross Section of Canals and Reservoirs

$$k = 1 - \frac{S \cdot E}{V} \tag{1}$$

where E is the evaporation rate in volume per unit area and V and S are the volume and surface area of the water respectively.

The surface area of a canal or reservoir is determined by its shape and the quantity of water in it. Some simplified lateral cross sections are presented in Fig. 5. The surface areas per unit volume for a unit length of these cross sections can be easily computed from the geometry. For example, the surface area for the vertical sided canal or reservoir is constant and is independent of the volume of water. The surface area of the triangular cross section with respect to the volume is linear and is given by $2/\tan\alpha$. Where α is the slope of the sides. Similarly, relationships between the volume and surface areas of the other cross sections can also be determined. These calculations are quite accurate for canals which have regular geometric shapes such as those shown in Fig. 5. However, it is not always a simple matter to estimate the surface areas of reservoirs from the volume of water stored in it. Reservoirs are usually highly irregular in shape and do not have uniform cross sections along their length. Moreover, reservoirs formed upstream of a dam slope down river as shown in Fig. 6. Therefore, it becomes very difficult to estimate the relationship between the volume and surface area of a reservoir from geometric considerations.

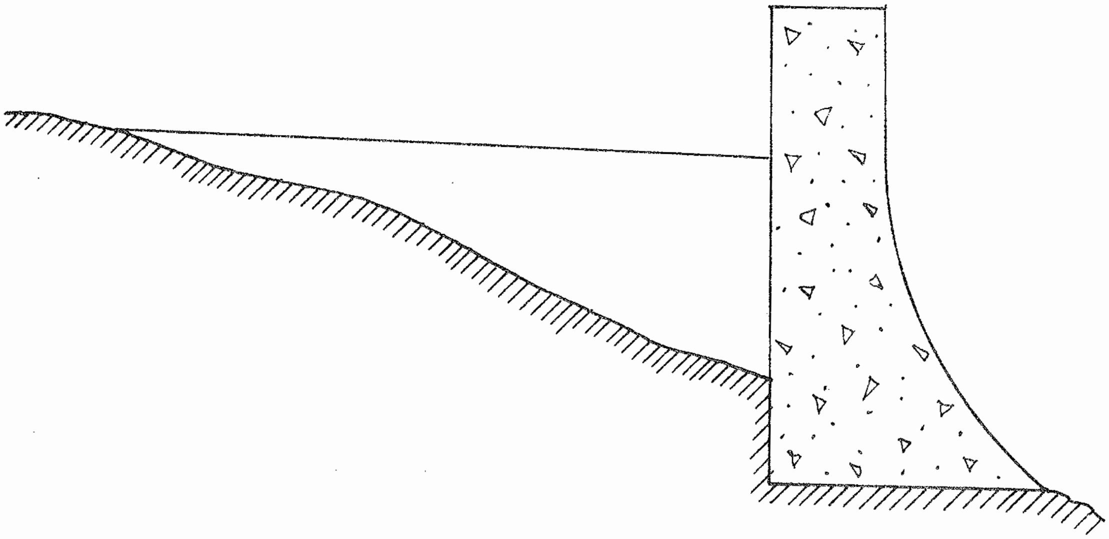


Figure 6 A Reservoir Formed Upstream of a Dam

Although, rough estimates may be obtained for preliminary analysis by considering uniform lateral cross sections of simple geometric shapes, a more accurate measurement of the surface area with respect to the volume can be obtained from the contour maps of the reservoir terrain. Charts showing the surface area vs. depth, depth vs. volume and volume vs. surface area are usually prepared by the hydrologic designers before the construction of a reservoir. An example of such a chart for a typical reservoir is presented in Fig. 7.

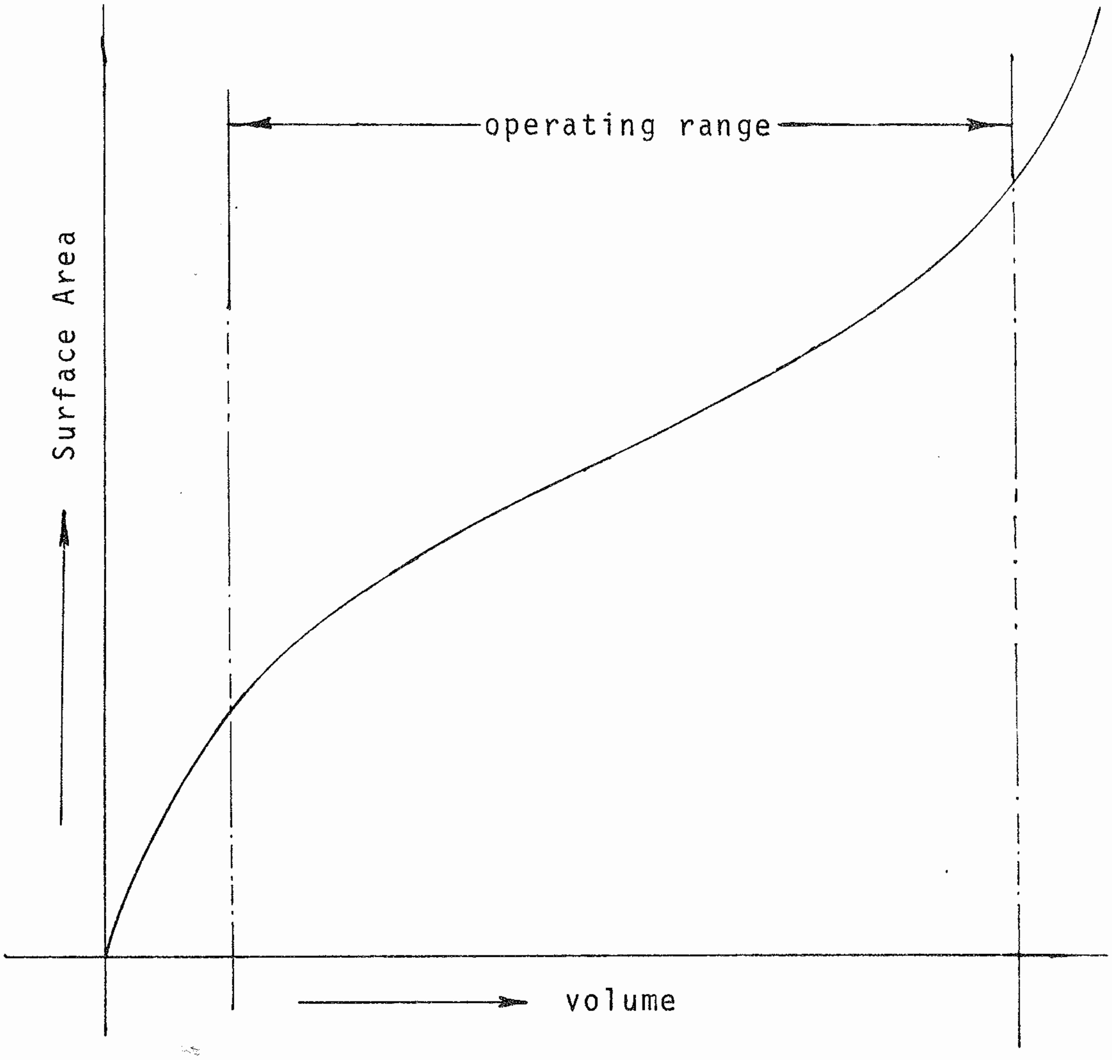


Figure 7 Operating Curve for a Reservoir

It can be seen from the curve, and is usually true, that the relationship between the volume and surface area is linear in the operating range of the reservoir. This linear relationship is important if the gain factor, k, is to remain constant for all flows. Non-linearities in this relationship are discussed in Chapter II.

Once the surface area, S, with respect to the volume and the evaporation rate, E, are known it is a simple matter to estimate the gain factor from equation 1. If the ratio of the surface area and volume are not constant, curves for the gain factor vs. volume can be prepared for the analysis of the problem. In the previous discussion it was assumed that seepage does not play a significant role in water distribution. If, however, this is not true then seepage vs. volume curves have to be prepared. No general statements can be made about this seepage curve because seepage characteristics are very difficult to predict. The composite seepage and evaporation gain factor can now be introduced as an additional parameter in the network model. The operation of the water distribution system can now be optimized by considering the losses together with the cost considerations in determining the optimal flows through the network.

1.4 Overview

In this chapter The Texas Water Plan has been introduced and the important role of losses in the water distribution system has been discussed. Based upon the water requirements of the State of Texas the plan provides for reservoirs, canals and pumping stations for transporting the water to the locations where it is needed. The cost of operating this system depends upon the transportation costs and the loss characteristics of the system. For example, to meet the water demands in West Texas, where evaporation rate is high, it would be economical to store surplus water in East Texas, where the evaporation rate is low, and ship it to West Texas as the demand arises. This policy would not be apparent if the evaporative nature of water storage in West Texas is not reflected in the model.

In the following chapters an algorithm which can determine the optimum water distribution policy in the presence of losses is derived and illustrated. This algorithm has the capability to deal with any network flow with gains problem in which gains are positive, flows are bounded from above and below and costs per unit flow are constant for each arc.

II. Problem Statement and Mathematical Formulation

II.1 Graphs and Networks

Geometrically, a graph may be considered to be a collection of nodes (verticies, points, junction points) in space connected by a system of arcs (edges, lines, links, branches, curves). Or, abstractly, a graph

$$G = [N,A]$$

may be defined to consist of a nonempty set N, a set A (possibly empty).

$$A \subseteq N \& N$$

and a mapping Φ of A into the unordered product N&N, that is

$$\Phi: A \rightarrow N \& N$$

The elements of N and A are called the nodes and arcs of the graph respectively and Φ is called the incidence mapping associated with the graph. Figure 8 shows the geometric representation of a graph which can be described by:

$$N = \{n_1,n_2,n_3,n_4,n_5,n_6\}$$

$$A = \{a_1,a_2,a_3,a_4,a_5,a_6\}$$

$$\Phi = \{a_1 \rightarrow (n_3 \& n_6), a_2 \rightarrow (n_2 \& n_3), a_3 \rightarrow (n_2 \& n_3),$$
$$a_4 \rightarrow (n_1 \& n_1), a_5 \rightarrow (n_1 \& n_4), a_6 \rightarrow (n_2 \& n_6)\}$$

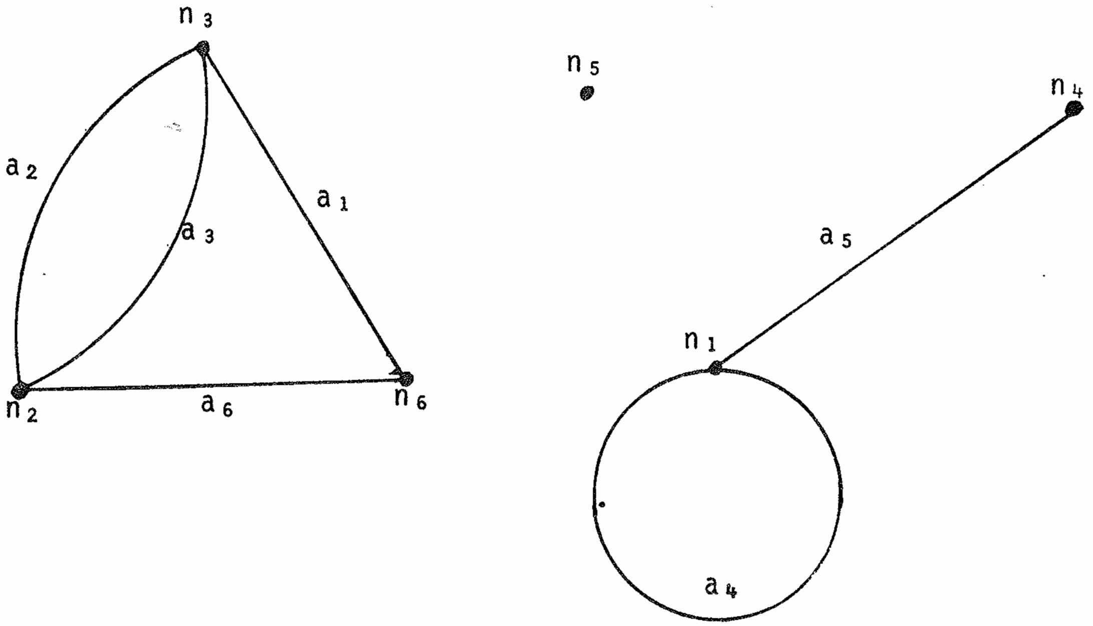


Figure 8 Geometric Representation of a Graph

The term network is frequently used instead of graph, especially when quantitative characteristics are imparted to the verticies and edges, in addition to the purely structural relationship that are the defining characteristics of a graph [6]. In networks it is customary to associate a number of parameters with each arc or node of the graph. These represent the natural limitations and capabilities of the system being modeled. The important characteristics of the system are incorporated into the model as numbers, or weights, on the arcs and nodes of the network. For the water distribution problem, discussed in Chapter I, the arcs represent the canals and river reaches of the Texas Water Distribution System. The nodes represent the reservoirs, pumping stations and canal intersections. Since storage of water and other associated properties of the nodes are represented spacially by the inventory arcs, all the characteristics of the system can be described completely by parameters on the arcs. These are; the capacities of the reservoirs and canals, unit cost of storage and transportation, and leakage and evaporation coefficients. In addition, the arcs may be constrained to transmit flow in one direction only.

Mathematically, a directed network,

$$D = [N,A]$$

can be defined to consist of a finite set N of elements (usually numbers from the natural number system) together with a subset, A, of the ordered pairs of elements taken from the ordered product N⊗N. That is,

$$A \subseteq N \otimes N$$

and a mapping γ of A into the ordered product N⊗N, given by

$$\gamma\colon A \rightarrow N \otimes N$$

The elements of N are called nodes and the elements of A are commonly known as arcs. To represent the parameters associated with each of the arcs, $a \in A$, of the network it is convenient to define a vector valued function which we shall call the flow parameter function, as

$$\Psi(a) = (l_a, u_a, c_a)$$

where l_a - lower bound on flow along the arc, a.

u_a - upper bound on flow along the arc, a.

c_a - unit cost of flow along the arc, a.

A network can be represented graphically by selecting a point corresponding to each node $x \in N$ and directing an arrow from x to y if the ordered pair $(x,y) \in A$. For example, the network shown in Figure 9 consists of four nodes and six arcs.

N = [1,2,3,4]

A = [(1,2),(1,3),(2,3),(2,4),(3,2),(3,4)]

The flow parameter function Ψ is shown directly on the network.

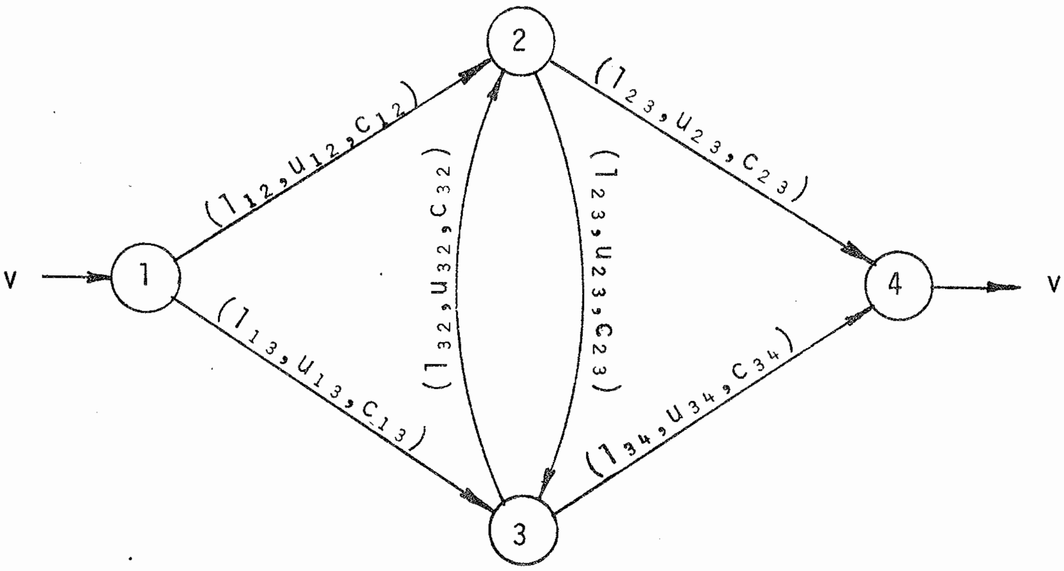


Figure 9 A Directed Network

A few commonly used terms in Network Flow Theory can now be defined for the network D = [N,A]. If $x_1,x_2,---,x_n$ are a set of distinct nodes from N such that (x_i,x_{i+1}) is an arc from the set A for each i = 1,2,---,n-1. Then the sequence of nodes and arcs:

$$x_1,(x_1,x_2),x_2,(x_2,x_3),x_3,---,(x_{n-1},x_n),x_n$$

is called a simple chain leading from x_1 to x_n. If $x_1 = x_n$ then this sequence is called a cycle. Often the term "chain" is used loosely to denote both directed chain or directed cycle. If (x_i,x_{i+1}) or (x_{i+1},x_i) is an arc from the set A for i = 1,2, ---,n-1, then the resulting sequence of nodes and arcs given by the equation is called a path from x_1 to x_n. And, if $x_1 = x_n$ the sequence is called a loop. A path and a loop differs from a directed chain or cycle by allowing an arc to be traversed in a direction opposite to that of its orientation.

II.2 The Pure Network Problem

The pure network problem, as defined to distinguish it from the network with gains or generalized network problem, is one where the flow in each arc is constant over the length of the arc. That is, there is no gain or loss of flow during transmission. Also, there is conservation of flow at each node. The total quantity of flow into each node is equal to the total flow out of the node. Also, the total input into the network is equal to the total output from the network. The set of nodes of the network, N, will be denoted by i = 1,2, ---n and the arcs, A, will be denoted by (i,j) if node i is joined to node j by an arc. Similarly, l_{ij}, u_{ij} and c_{ij} will represent the lower and upper bounds and costs respectively for each of the arcs. For the moment, the arc (i,j) will be considered to be unique. That is, there is only one arc joining any two nodes i and j. For multiple arcs this restriction can be met by introducing an additional node in the middle of some of these arcs.

If f_{ij} is the flow in arc $(i,j)\in A$, then, the conservation of flow constraints for a pure network can be written as:

$$\sum_j f_{ij} - \sum_j f_{ji} = b_i \text{ for all } i \in N \tag{2}$$

$$\sum_i b_i = 0 \tag{3}$$

The b_i's are equal to zero at all nodes except where flows are introduced into the system (source nodes) or taken out of the system(sink nodes). Since the flow into the system equals the flow out, the sum of the b_i's is equal to zero as denoted by Equation 3. For networks with a single source node, s, and a single sink node, t, (e.g. Figure 9) Equation 2 can be written as:

$$\sum_j f_{ij} - \sum_j f_{ji} = \begin{cases} v, & i=s \\ 0, & i\neq s,t \\ -v, & i=t \end{cases}$$
$$\text{for all } i \in N$$

The assumption of single source and sink is really not very restrictive, as multiple sources and sinks can be joined to a super-source or super-sink by appropriate arcs to convert the problem to one of single source and single sink (see [14]).

In a pure network problem, since the total flow into the system equals the total flow out of the system, it is possible to connect the outflow with the inflow and create a circulation problem as shown in Figure 10. The arc parameters of the circulation arc are determined by the objective of the problem. The conservation of flow Equation 2 can now be generalized over the entire network

$$\sum f_{ij} - \sum f_{ji} = 0 \text{ for all } i \in N$$

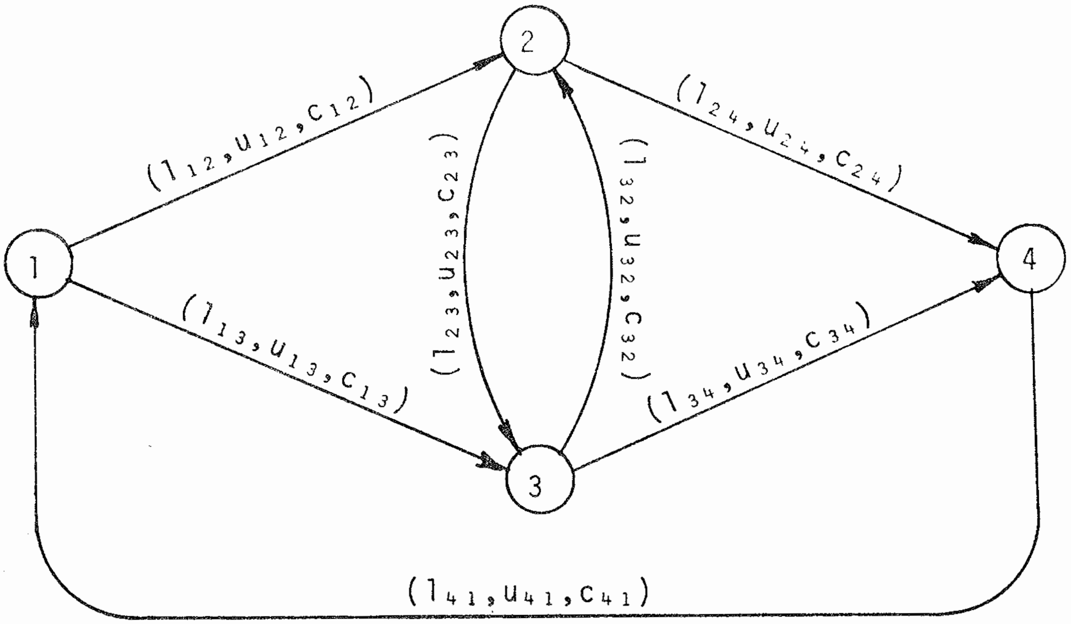


Figure 10 The Circulation Problem

The objective of the pure network problem is to find the arc flows which minimize the total cost of flow:

$$\text{Minimize:} \quad \sum_{(i,j)\in A} c_{ij}f_{ij} \qquad (5)$$

Subject to:

$$\sum_j f_{ij} - \sum_j f_{ji} = 0 \text{ for all } i \in N, \qquad (6)$$

and

$$l_{ij} \leq f_{ij} \leq u_{ij} \text{ for all } (i,j) \in A \qquad (7)$$

$$f_{ij} \geq 0 \text{ for all } (i,j) \in A \qquad (8)$$

This problem fits the Linear Programming Model and can be solved by the simplex algorithm. However, there are several more efficient algorithms which take advantage of the special structure of the problem. These are described in Chapter III.

II.3 The Network with Gains

There are many flow problems for which the conventional pure network model described in the previous section is inadequate. In some practical network problems flow suffers loss during transmission due to leakage and damage while in others flow may be amplified due to conversion or inflows into the system. For such systems an additional parameter, the gain factor, may be introduced into the flow parameter function. The gain factor, k_{ij}, is the amplification or gain in the flow as it passes through the arc (i,j). It denotes the fraction of the flow that is transmitted from node i to node j while passing through the arc (i,j).

If f_{ji} is the flow leaving node j by arc (j,i) then, $k_{ji}f_{ji}$ is the amount of flow that arrives node i through the same arc. If this new flow property is imposed upon the pure network model of Section II.2, the minimum cost flow problem can be stated as:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}f_{ij} \tag{9}$$

$$\text{Subject to:} \quad \sum_j f_{ij} - \sum_j k_{ji}f_{ji} = 0 \quad i\neq s,t \tag{10}$$

$$\sum_j f_{it} = F_t$$

$$\text{and} \quad f_{ij} \leq u_{ij} \text{ for } (i,j)\in A \tag{11}$$

$$f_{ij} \geq 0 \text{ for } (i,j)\in A \tag{12}$$

The introduction of the k_{ji} factor in Equation 10 signifies that any flow to node i from node j along arc (i,j) is amplified by a factor of k_{ji} after it leaves node j and before it arrives at node i. Figure 11 illustrates the notation used for the network with gains model.

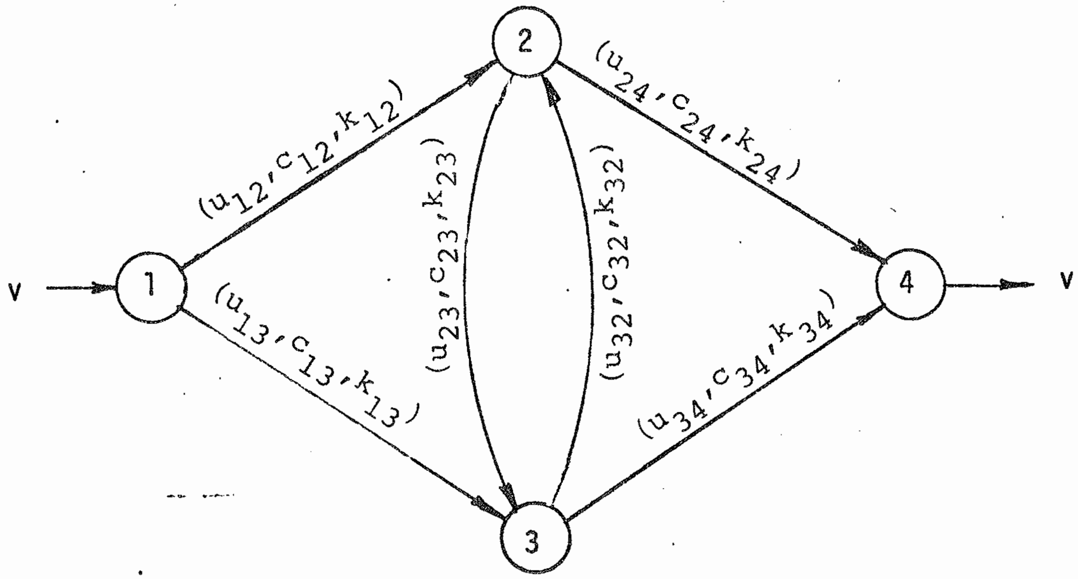


Figure 11 The Network with Gains Model

Notice that the problem as stated here is to obtain a given quantity, F_t, of flow at the sink at minimum cost. This particular form is chosen because most flow problems involve meeting some specified output requirements at minimum cost. Other authors (16,24,25,37) consider alternative formulations for network with gains problems.

The introduction of the gain factors on the arcs of the network with gains introduces some special features in the network. These were first identified by Jewell [25]. First, flow may be destroyed by passing around a directed cycle whose total gain is less than unity as shown in Figure 12. The arrows and the numbers above them denote the flow. The numbers below the arc are the gains. The cycle shown absorbs any amount of flow determined by the upper bounds of the arcs (1,2) and (3,2) and half the upper bound of arc (2,3). Second, flow may be created in an amount limited only by the branch capacities in a cycle whose total gain is greater than one, as shown in Figure 13.

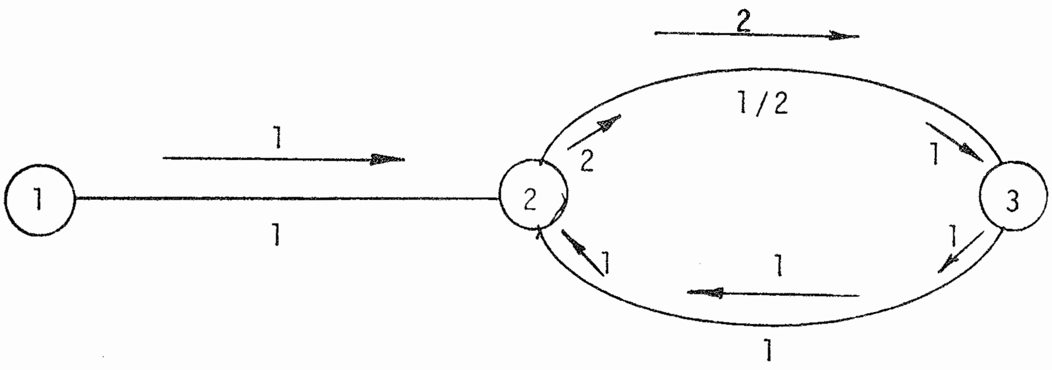


Figure 12 A Flow Absorbing Cycle

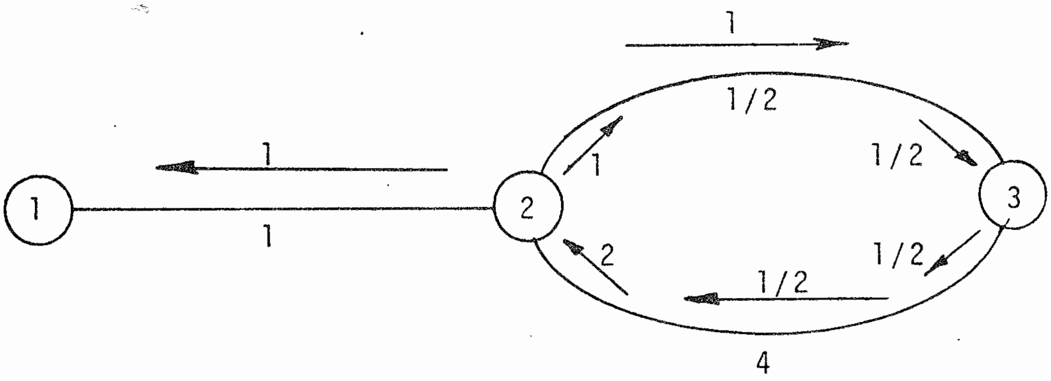


Figure 13 A Flow Generating Cycle

Most network algorithms, including the one presented here, assume lower bounds of zero. In many situations such as discharge along rivers etc. the flow along an arc may be constrained by a positive lower bound. Zero lower bounds on all arcs in a network can be assumed without any loss of generality, because all positive lower bounds can easily be transformed into zero lower bounds by the following transformation of the flow variable.

For any arc (i,j) with a positive lower bound l_{ij} the following substitution can be made for the flow variable f_{ij}:

$$f_{ij} = f'_{ij} + l_{ij}$$

The corresponding upper bound, u_{ij}, should be reduced to $u_{ij} - l_{ij}$.

Physically this transformation is illustrated by Figure 14. The lower bound on the arc (i,j) can be reduced to zero by forcing a flow of l_{ij} from the node i to the sink t, and a flow of l_{ij} from the source s to the node j and reducing the upper bound of the arc (i,j) to $u_{ij} - l_{ij}$. This is achieved by adding two arcs (i,t) and (s,j) with upper bounds of l_{ij} and a very large negative cost -M, as shown in Figure 14b. The large negative costs attempt to induce a flow through these arcs. If a flow can be found which saturates the arcs (i,t) and (s,j) it can be transformed to a feasible solution for the original problem. Arcs with non-unity gain can be transformed in the same manner but with a gain of 1 on arc (i,t) and a gain of k_{ij} on arcs (i,j) and (s,j).

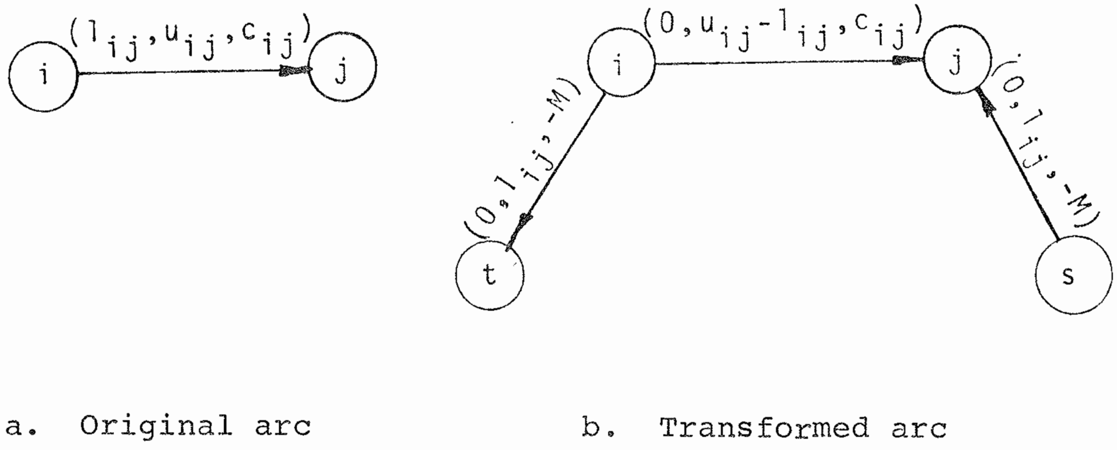
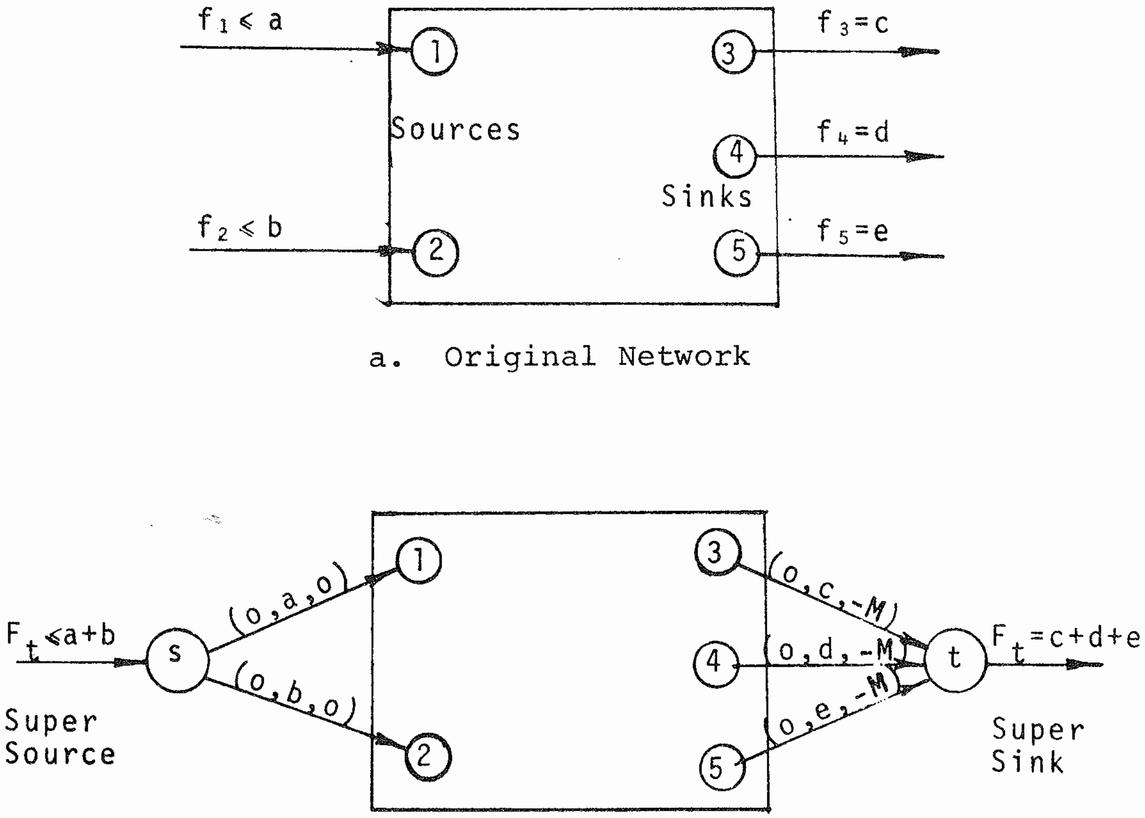


Figure 14 Transformation of Lower Bounds

The algorithm developed in Chapter IV and many other network algorithms assume a single source and a single sink. Multiple sources and sinks like those of Figure 15a can easily be changed into a single source and sink by introducing a super source and a super sink and appropriate arcs joining these to the original sources and sinks as shown in Figure 15b. A large negative cost of -M can be put on the arcs joining the original sinks to the super sink so as to force the required flow out of the network. Hence, it is seen that a single source and a single sink can be assumed for any network without any loss of generality.

Finally, an important requirement of most network algorithms is that there be no negative cycles in the network. A negative cycle is defined to be a directed cycle such that the weighted sum of the costs around it is negative. A negative cycle gives rise to an interesting situation where it is possible to realize revenue or minimize cost by sending flow around the cycle without having any input to or output from the network. Most practical networks are usually free from negative cycles, therefore, this is usually not a major problem.



a. Original Network

b. Transformed Network

Figure 15 Transformation of Multiple

Sources and Sinks into a Single Source and Sink

While estimating the leakage characteristics of the reservoirs and canals, it was seen in Chapter I that the gain factor k_{ij} may not always be linear with respect to the flow through an arc. Whereas, an important assumption of the discussion in Section II.3 has been that the loss is proportional to the flow through an arc. If this property is violated then the mathematical properties discussed in this chapter and in Chapter IV are no longer valid. However, if the curve for the relationship between the flow transmitted and originating flow is concave as shown in Figure 16 it is possible to meet the restriction of linearity for the gain factor by a piecewise linear approximation as shown in Figure 17. (Note: Gain factor is equal to flow transmitted per unit of originating flow). The curve shown in Figure 16 has been approximated by three linear segments AB, BC and CD. If k_1, k_2 and k_3 are the slopes respectively of these three segments and u_1, u_2 and u_3 are the flows corresponding to points B, C and D and if the unit cost of flow is the same over the entire flow range then the flow can be represented by the three arcs shown in Figure 17. It is seen that when flow is to be transmitted between node 1 and 2, arc_1 will be chosen first as it transmits a greater percentage of its flow at the same unit cost as compared to the other arc. After arc_1 is saturated, flow will be sent through arc_2 and so on. This corresponds to the desired flow pattern. However, if the relationship between the originating and transmitted flow was convex, similar piecewise linear approximation could not be used because arc_3 would be chosen first to transmit flow between node 1 and 2 as it would have had the highest gain and would have therefore been unrealistic. Similar piecewise linear approximation can be used to represent flow which has a convex cost function, or a concave revenue function, with respect to the flow (see [14]). If both unit cost and gain vary with flow, it can be shown by similar reasoning that piecewise linear approximation is possible as long as the curve for cost/gain vs. flow is convex. But, if the cost/gain curve is not convex, the procedure developed in this study are not applicable.

Now, considering the various cross sections of Figure 5 and assuming that evaporation is the predominant loss and that it is linearly dependent upon the surface area, the losses from the canals and reservoirs with respect to the flow volume can be represented by the arcs shown in Figure 18, where node n is a sink node. Gains are shown below the arcs.

a) The rectangular cross section gives a uniform loss independent of the volume. It can be represented by a constant leakage arc (2,n), $l_{2n} = u_{2n}$, is the constant leakage.

b) The triangular cross section gives a linear loss with respect to volume and is represented by a constant gain factor k_{12}.

c) The circular cross section is responsible for a convex gain parameter with respect to volume and cannot be represented by piecewise linear approximation.

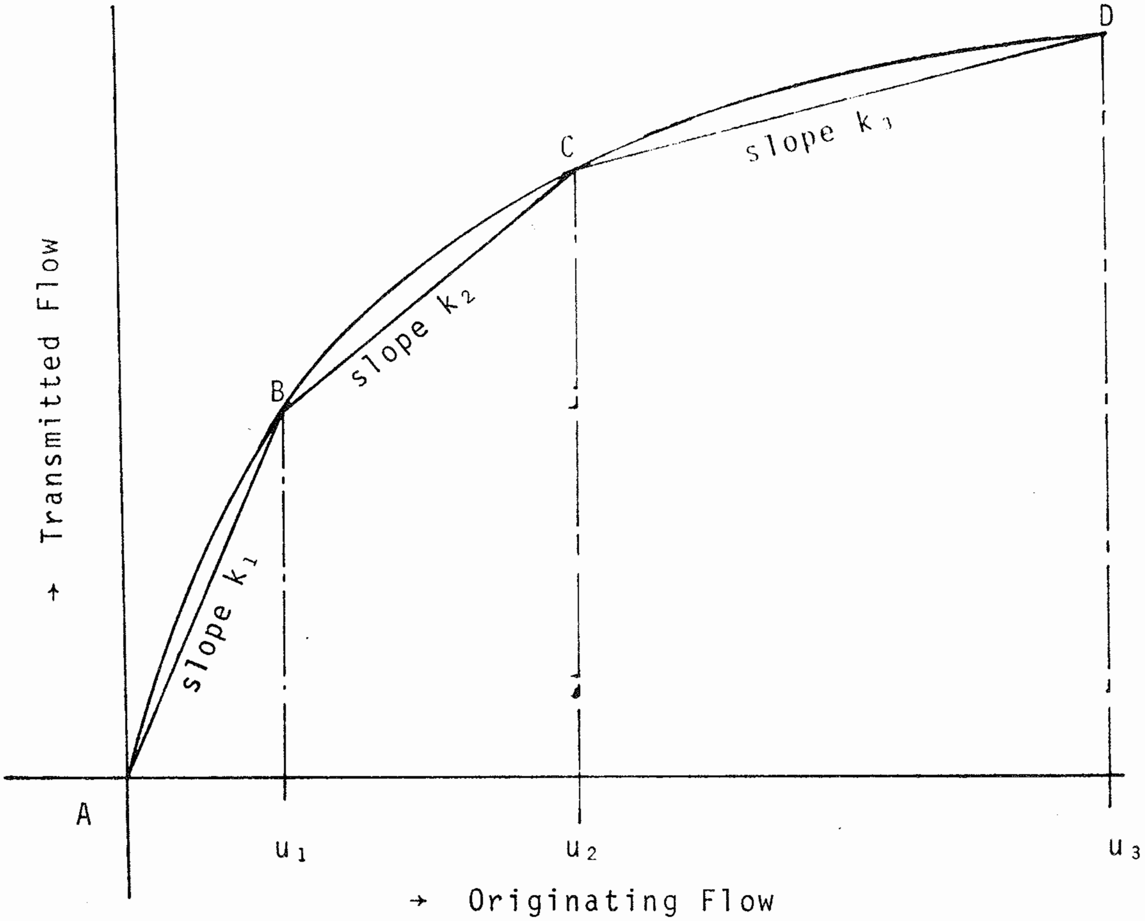


Figure 16 Concave Curve of Flow Transmission

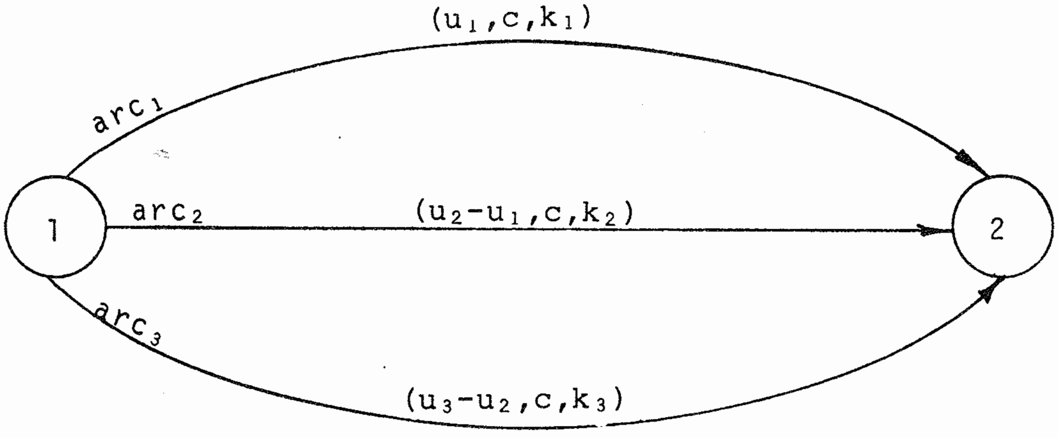


Figure 17 Piecewise Linear Approximation of a Concave Rate of Flow Transmission

d) The trapezoidal cross section gives rise to a loss function which is a combination of a and b, above. It is represented by an arc with a constant gain paramater k_{12} and a constant leakage arc (2,n).

Flow in a reservoir arc represents volume of water stored in a reservoir. The evaporation of water over a storage period is represented by a gain factor on the corresponding reservoir arc. These arcs are treated like any other arc in the network.

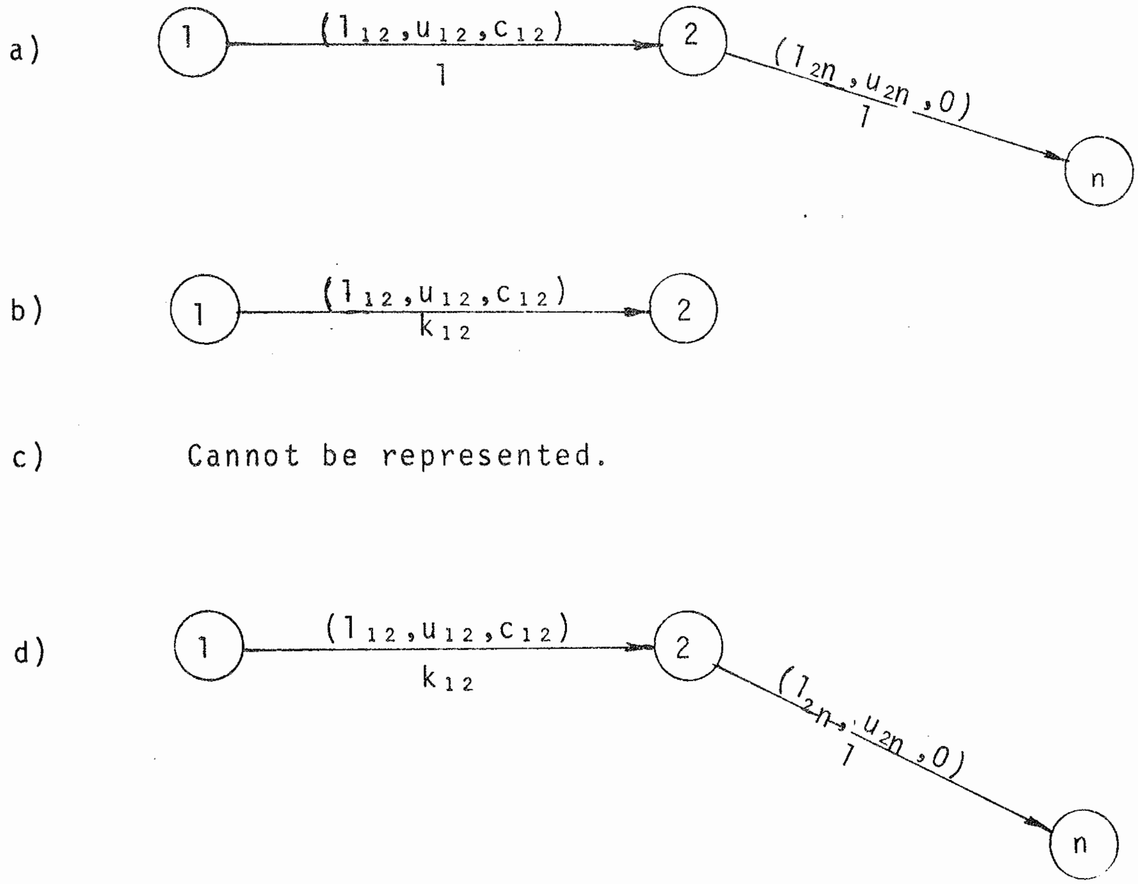


Figure 18 Network Representation of the Cross Section of Figure 5

III. Literature Survey

The minimum cost flow pure network problem formulated in Section II.2 fits the linear programming model and can be solved by any existing linear programming algorithm. However, a number of procedures have been developed which take advantage of the special structure of this problem. Notable among these are the algorithms of Busacher and Gowen [5], Klein [29] and the Out-of-Kilter algorithm of Ford and Fulkerson [14]. The method of Busacher and Gowen starts with all flows equal to zero in the network and incrementally increases the flow at the lowest possible cost until the desired flow is reached. Although the solution of the problem takes place entirely in the primal space, it has been called a dual algorithm by Hu [23], because the primal problem is not feasible until the last iteration. Klein's method starts with a feasible flow through the network and then improves upon the solution iteratively by recirculating the flow through the network to decrease the cost. This method may be classified as a primal method because the problem is always feasible in the primal space from the first to the last step. Ford and Fulkerson's algorithm can be classified as a primal-dual method because the problem starts with any flow through the network and iteratively changes the flow until the primal-dual properties at optimality are satisfied.

Of these solution procedures, the Out-of-Kilter algorithm is the only method that considers lower bounds explicitly and allows multiple sources and sinks. The only requirement is that of circulation of flow as described in Section II.2. In almost all other solution procedures the lower bound on the flows along arcs are assumed to be zero, and only single sources and sinks are permitted. This can always be assumed without any loss of generality because all networks can be transformed to an equivalent network with zero lower bounds having a single source and a single sink as described in Section II.4.

Recently studies have been undertaken by Glover, Karney, Klingman and Napier [18] to compare the computational effectiveness of primal simplex type procedures against the methods mentioned above for transportation problems. The results indicate that the primal simplex type of algorithm is faster than existing linear programming or Out-of-Kilter codes. Klingman, et. al. are trying to develop a similar simplex type of procedure for network problems. Since, not enough information is available at this time about this procedure, it has been omitted from this discussion.

The network problem with gains was first introduced by William S. Jewell [25,26]. His work provides the most complete treatment of flow with gains. In his paper, Jewell describes a generalization of network flow problems to 'process flow networks'. The flow in any branch of the network may be multiplied by an arbitrary constant, called the branch gain, before leaving the branch and flowing into the remainder of the network. This generalization permits the description of networks in which different kinds of flow may be converted one to another without constant returns to scale. Jewell claims that his method is a natural extension of the Ford and Fulkerson technique. Jewell considers any gain factor, positive or negative. Since flow into a network with gains does not have to equal the flow out, some additional arcs are created to account for the constraints on the boundary conditions and to create circulation. These boundary conditions may be either inequalities or equality constraints on the amount of input and output flow.

A closely related problem to the network with gains is the transportation problem with gains or the generalized transportation problem. The generalized transportation problem is the same as the classical (pure) transportation problem except that the flow in an arc (i,j) from origin i to destination j is subject to amplification or attenuation by the factor k_{ij}. It has been shown by Lourie [32], Balas [2], and Eisemann [13] that this generalization of the classical transportation problem has a drastic effect upon the basic structure of such problems. So that, even finding an initial primal feasible basic solution becomes a difficult task. As a consequence of this altered topology existing procedures for solving the transportation problem fail. Several approaches have been suggested for solving this problem by Balas [1], Balas and Ivanescu [2], Eisemann [13], Glover, Klingman and Napier [21] and Louire [32]. Some of these [21] are start procedures which provide an initial feasible solution. They can be used together with Lemke's [30] dual method or the method of Glover, Klingman and Napier [20] to solve certain generalized transportation problems.

It has been shown by Glover, Klingman and Napier [22] that any generalized network can be transformed into a generalized transportation problem. One possible solution technique for networks with gains may be to transform it into a generalized transportation problem and solve it by one of the methods mentioned above. A number of special purpose algorithms have been presented in recent years which have tried to solve a restricted class of the network with gains problem. Two such algorithms are due to Charnes and Raike [9]. Both of these procedures are "one pass" shortest path algorithms. They also point out the possibilities of obtaining a dual feasible solution to the generalized network problem by their method. Glover, Klingman and Napier [21] have characterized the properties of a special class of generalized network problems that permit a dual feasible basic solution to be determined in "one pass" through the network. They suggest using Lemke's [30] dual method or Charnes and Cooper's [7] poly-ω technique to solve the problem from there. Glover and Klingman [19] also propose a method of transforming certain special generalized networks into pure networks by a method of scaling. They have proved that any generalized network incidence matrix that does not have full row rank can be transformed by this procedure. Charnes and Raike [9] have pointed out and proved that the pure network problem is of rank m-1 while the generalized network problem is of rank either m or m-1. Where m is the number of rows in the incidence matrix. Therefore, it is possible to transform some specially structured network problems into pure network problems by the transformation algorithm of Glover and Klingman [19]. Johnson [27] is concerned with solving the problem directly on the network with the help of the simplex procedure. He presents a triple labeling technique to keep the solution basic for pure network problems and suggests an extension of his procedure to the network with gains. Maurras [34] has extended Johnson's idea and developed an algorithm that solves networks with gains problems with the help of the simplex procedure directly on the network structure.

Jarvis and Jezior [24] have recently presented an algorithm for determining the maximal flow through a directed (acyclic) network with positive gains. The proposed method is a primal-dual algorithm comparable to Ford and Fulkerson's [15] max-flow algorithm. Minieka [36] has recently suggested a method for extending Jewell's [25] algorithm to include multiple sinks and negative costs on the arcs. The problem of maximizing flow through a "lossy" network has also been considered by a number of persons in the field of communication. The most comprehensive graph theoretic treatment is due to Onaga [37,38], who introduced the concept of optimal flows in communication networks. Fujisawa [16] has presented topological solution procedures for the maximum flow through a lossy communication net. Mayeda and Van Valkenburg [35] have proved the max-flow min-cut theorem for this problem.

One of the obvious solution procedures for the network with gains problem is with the help of the classical simplex linear programming algorithm. From the mathematical statement of the problem in Chapter II is is seen that the problem can be modeled as a linear programming problem where the objective is to minimize the linear sum of the products of the costs and flows in each of the arcs subject to the conservation of flow and bound constraints, which are also linear. Although this method is quite feasible for small problems, the simplex matrix grows geometrically with the dimension of the network. Therefore, the storage requirement even for a moderate sized problem can be prohibitive. The time required to solve the problem can be very high and degeneracy which is inherent in most networks adds to the complexity and computation time. The network based simplex procedure of Maurras [34] overcomes most of these difficulties. The computation times presented by Maurras look impressive. However, it seems that the problem solved by him is un-capacitated. This is not a serious problem as Dantzig's [10] upper bounding technique could be extended to cover this problem. Details of the algorithm are not yet available and it is too early to tell how useful this algorithm will be to solve typical network with gains problems.

Jewell's work [26] provides the most complete treatment of network with gains. The class of problems that he analyzes is very general. However his solution technique is too complex to make it a useful tool for the solution of large scale real world problems. His absorbing network detection, transfer factor determination and dual variable calculation are all very long and complicated. No computational results are available even for small problems.

The max-flow algorithms of Jarvis and Jezoir [24], Fujisawa [16], Onaga [37,38] and Mayeda and Van Valkenburg [35] are inappropriate for determining the minimum cost flow through a network with gains. The shortest path algorithm of Charnes and Raike [9] is also inapplicable to the problem under consideration. They solve a very restricted minimum cost flow problem, and the network is assumed to be uncapacitated. Capacitated networks are those that have a specified upper bound on flows along the arcs. The transformation algorithm of Glover and Klingman [19] is apparently a very powerful technique if the problem in question is of rank m-1. However, it is not yet known what special structure of a network problem causes it to have one less than row rank. Since it is good only for a restricted class of problems, its general applicability for networks with gains is of limited value.

IV. The Networks with Gains Minimum Cost Flow Algorithm

The algorithm to be described determines the distribution of flow in a network with gains which provides a given amount of flow to the sink at minimum cost. An example network is shown in Figure 19. Each arc has three associated parameters which are upper bound to flow, cost per unit flow and gain. These are shown on each arc with a triplet (u_i, c_i, k_i) for arc i. A lower bound to flow is not included in the parameter list as it is assumed to be zero. Let the required flow at the sink be F_t. An unbounded quantity of flow is available at the source. Let the network D = [N,A] for which the problem is defined be called the original network to distinguish it from the marginal network which will be defined later.

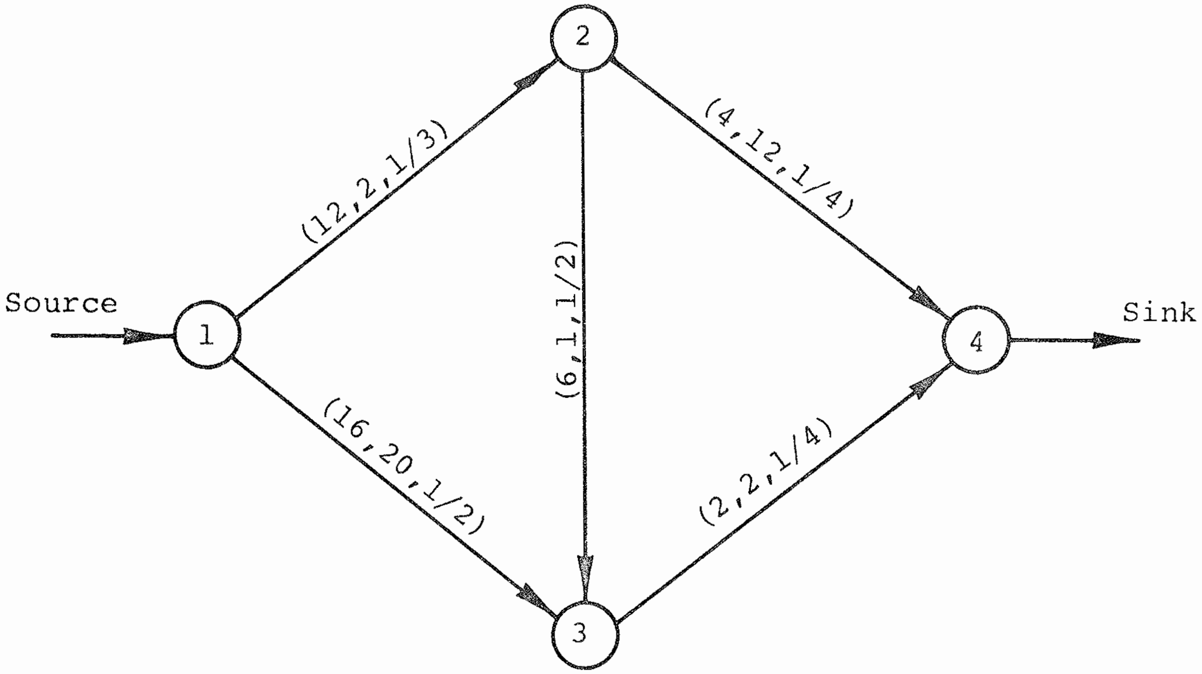


Figure 19 Example Network. The Parameters on each arc are (Upper Bound, Cost, Gain).

The algorithm is based on the Busacher and Gowen algorithm for pure networks [5]. The principal steps of the algorithm are as follows:

- <u>step 0</u> - Start with zero flow on all arcs.
- <u>step 1</u> - Define a marginal cost network with respect to the original network and the current flow through it.
- <u>step 2</u> - Find the mimimum cost flow augmenting chain in the marginal network that can deliver flow at the sink. If there are none, stop--the maximum flow through the network has been found.
- <u>step 3</u> - If such a chain is found, route as much flow as possible through it.

step 4 - Augment the flow in the original network with the flow found for the marginal network. If the desired output flow is reached, stop. The current flow is the minimum cost flow. Otherwise return to step 1.

The flows obtained at each iteration of the algorithm are optimal for the amount of flow obtained at the sink. This is true because the initial flow is optimal (all arc flows are zero for zero flow at the sink) and flow is augmented at each step through the minimum cost chain. The solution is infeasible with respect to the desired output flow until the final step when the flow is both feasible and optimal.

IV.1 The Marginal Network

The marginal network is to reflect the costs of changing flows in the original network. Parameters of this network depend on the flow in the original network. Let F be the flow for the original network. F defines the flow f_{ij} for each arc $(i,j) \in A$. F is feasible in that flow is conserved at each node and arc upper bounds are satisfied.

Call the marginal network $D^* = [N^*,A^*]$. The set of nodes in the marginal network is the same as in the original network, thus $N^* = N$. The arcs in D^* will be of two types called respectively forward and mirror arcs. Forward arcs correspond directly to the arcs in the original network. Thus if $A_1{}^*$ is the set of forward arcs in D^*:

$$(i,j) \in A_1{}^* \text{ if } (i,j) \in A$$

The parameters on the forward arcs are determined from the parameters of the arcs in the original network and the current values of flow in the original network. In the following definitions the starred parameters are for the marginal network and unstarred are from the original network. Thus for each arc $(i,j) \in A_1{}^*$ define:

$$u_{ij}{}^* = u_{ij} - f_{ij}$$

$$c_{ij}{}^* = c_{ij}$$

$$k_{ij}{}^* = k_{ij}$$

Thus the forward arcs in the marginal network are the same as the arcs in the original network except the capacities are reduced by the flows in the original network. If an arc in the original network is saturated (flow equal to capacity) the corresponding arc has zero capacity in the marginal network.

The mirror arcs in the marginal network are the reverse of the arcs in the original network. Let $A_2{}^*$ be the set of mirror arcs. Then

$$(i,j) \in A_2^* \text{ if } (j,i) \in A$$

The parameters of the mirror arcs reflect the effect of removing flow from the arcs in the original network. Thus for each arc $(i,j) \in A_2^*$ define:

$$u_{ij}^* = f_{ji} \cdot k_{ji}$$

$$c_{ij}^* = -c_{ji}/k_{ji}$$

$$k_{ij}^* = 1/k_{ji}$$

In step 4 of the algorithm stated above it is required that flows in the original network be augmented with flows found for the marginal network. This process is now defined. Assume a feasible flow F is defined for the original network. This flow defines a marginal network as above.

The arcs of the marginal network are the collection of forward and mirror arcs. Thus: $A^* = A_1^* \cup A_2^*$. Assume a feasible flow F* has been determined for the marginal network Let K* be the total cost for flow in the marginal network. Thus

$$K^* = \sum_{(i,j) \in A^*} c_{ij}^* \, f_{ij}^*$$

Let the total cost for flow in the original network be K where

$$K = \sum_{(i,j) \in A} c_{ij} \cdot f_{ij}$$

Let a new flow F' defined for the following augmentation rule

$$f_{ij}' = f_{ij} + f_{ij}^* - f_{ji}^*/k_{ij} \qquad \text{for all } (i,j) \in A$$

K', the cost of the new flow in the original network is:

$$\begin{aligned} K' &= \sum_{(i,j) \in A} c_{ij} f_{ij}' \\ &= \sum_{(i,j) \in A} c_{ij} [f_{ij} + f_{ij}^* - f_{ji}^*/k_{ij}] \\ &= \sum_{(i,j) \in A} c_{ij} f_{ij} + \sum_{(i,j) \in A_1^*} c_{ij} f_{ij}^* \\ &\qquad - \sum_{(j,i) \in A_2^*} \frac{c_{ij}}{k_{ij}} f_{ji}^* \end{aligned}$$

Ncting that $c_{ij}* = c_{ij}$ for $(i,j) \in A_1*$ and $c_{ji}* = -c_{ij}/k_{ij}$ it can be observed that:

$$K' = K + K*$$

Thus the cost of the augmented flow in the original network can be obtained by summing the cost of the old flow and the cost of the flow in the marginal network.

It can also be shown that the augmented flow is feasible for the original network.

$$f_{ij}' = f_{ij} + f_{ij}* - f_{ji}*/k_{ij}$$

$$f_{ij}' \leq f_{ij} + f_{ij}*$$

Since $f_{ij}* \leq u_{ij} - f_{ij}$

$$f_{ij}' \leq u_{ij}$$

Also $f_{ij}' \geq f_{ij} - f_{ji}*/k_{ij}$

Since $f_{ji}* \leq f_{ij} \cdot k_{ij}$

$$f_{ij}' \geq 0$$

These results are important to show:

Theorem 1

Let the flow (F) be optimum in the original network for a quantity of flow X delivered at sink. Let the marginal network D* be defined for this flow. Let the flow (F*) in the marginal network be determined which will deliver Y units of flow to the sink of the marginal network at minimum cost. The new flow F' determined by

$$f_{ij}' = f_{ij} + f_{ij}* - f_{ji}*/k_{ij} \text{ for } (i,j) \in A$$

is optimum in the original network for the flow X+Y delivered to the sink.

Proof: For a proof by contradiction assume that the flow F' is not optimum for the flow X+Y at the sink. Then there exists some feasible flow F'' whose cost K'' is lower than the cost, K', of the flows F'.

Define the flow F** for marginal network:

If $f_{ij}'' > f_{ij}$ let $f_{ij}** = f_{ij}'' - f_{ij}$ and $f_{ji}** = 0$

If $f_{ij}'' < f_{ij}$ let $f_{ji}^{**} = (f_{ij} - f_{ij}'')\cdot k_{ij}$ and $f_{ij}^{**}=0$

If $f_{ij}'' = f_{ij}$ let $f_{ij}^{**} = 0$ and $f_{ji}^{**}=0$

It can easily be shown that F** is feasible for the marginal network and that it delivers Y units of flow to the sink. The cost K'' can be determined as

$$K'' = K + K^{**}$$

where K** is the cost of the flow F** in the marginal network.

Since K''<K', K**<K*. But this contradicts the proposition that F* is the minimum cost flow in D* which delivers Y units of flow to the sink. Thus the theorem is proved.

Theorem 1 provides the basis for the algorithm. In step 2 the directed chain is found in the marginal network which can deliver flow to the sink at minimum cost. This is either a simple chain from the source to the sink or on a simple chain that originates at node in a flow generating cycle. In either case the chain will be called the minimum cost flow augmenting chain.

IV.2 Finding the Minimal Cost Flow Augmenting Chain

The minimum cost flow augmenting chain in the marginal network is found using a slightly modified version of the dynamic programming shortest path algorithm of Bellman [4] with adjustments to account for the possibility of cycles.

Before explaining the algorithm it will be helpful to determine several costs associated with flows in the marginal network. Consider a simple chain from the source to the sink:

$$C = \{i_1, b_1, i_2, b_2 \dots i_{m-1}, b_{m-1}, i_m\}$$

where the $i_1, i_2 \dots i_m$ are node indices and $b_1, b_2 \dots b_{m-1}$ are arcs. Note that $b_\ell = \{i_\ell \; i_{\ell+1}\}$ and i_1 is the source and i_k is the sink. Let the parameters of arc b_ℓ be $\{u_\ell, c_\ell, k_\ell\}$.

Assign flows on the arcs of the chain to obtain one unit of flow at the sink. Let the flow on arc b_ℓ be f_ℓ. For simplicity the asterisk has been dropped from the flow variable. Recall however that all flows in this section are in the marginal network. To obtain one unit of flow at the sink requires the following arc flows:

$$f_{m-1} = 1/k_{m-1}$$

$$f_{m-2} = 1/(k_{m-1} \cdot k_{m-2})$$

$$f_2 = 1/ \prod_{i=2}^{m-1} k_i$$

$$f_1 = 1/ \prod_{i=1}^{m-1} k_i$$

The cost of this flow is:

$$K_c = \sum_{\ell=1}^{m-1} \frac{c_\ell}{\prod_{i=\ell}^{m-1} k_i} \tag{13}$$

Let v_i be the cost of obtaining one unit of flow at node i. Then it will be noted that: $v_{i_1} = v_s = 0$ and that: $v_{i_{\ell+1}} = (v_{i_\ell} + c_\ell)/k_\ell$.

It can be shown that a necessary and sufficient condition for C to be the minimum cost flow augmenting chain is:

$$v_j = \min_{(i,j) \in A^*} \{[v_i + c_{ij})/k_{ij}\}$$

for each node j on the chain.

This suggests an algorithm for finding the minimum cost flow augmenting chain. The variable p_i is a pointer which will be used to recover the chain at termination.

1. Let $v_s=0$. Let $v_i=M$ for $i \in N$, $i \neq s$. (Here M is a large number) Let $p_i=0$ for all $i \in N$.
2. For each arc $(i,j) \in A^*$ such that $u_{ij}>0$:

   If $v_j \leq (v_i + c_{ij})/k_{ij}$, take no action.

   If $v_j > (v_i + c_{ij})/k_{ij}$, let $v_j = (v_i + c_{ij})/k_{ij}$

   Let $p_j = i$.
3. If some v_j has been changed in step 2, repeat step 2. If all arcs are inspected without changing any node cost, v_j, then terminate. v_t (t is the sink) is the cost of the minimum cost flow augmenting chain. The chain itself can be recovered from the pointers p_i. If at termination $p_t = 0$ there is no chain to the sink in the marginal network.

Figure 20 illustrates the algorithm applied to the example problem. Since the initial flows are zero, the marginal network is the same as the original network. Note that arcs with zero upper bound are not shown. The labels on the nodes indicate (p_i/v_i). The labels have been crossed out as they are changed. The small circled numbers indicate the order in which the arcs are inspected in step 2. Observe that the flow augmenting chain in the example is found by tracing the pointers backwards from node 4. The chain is:

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

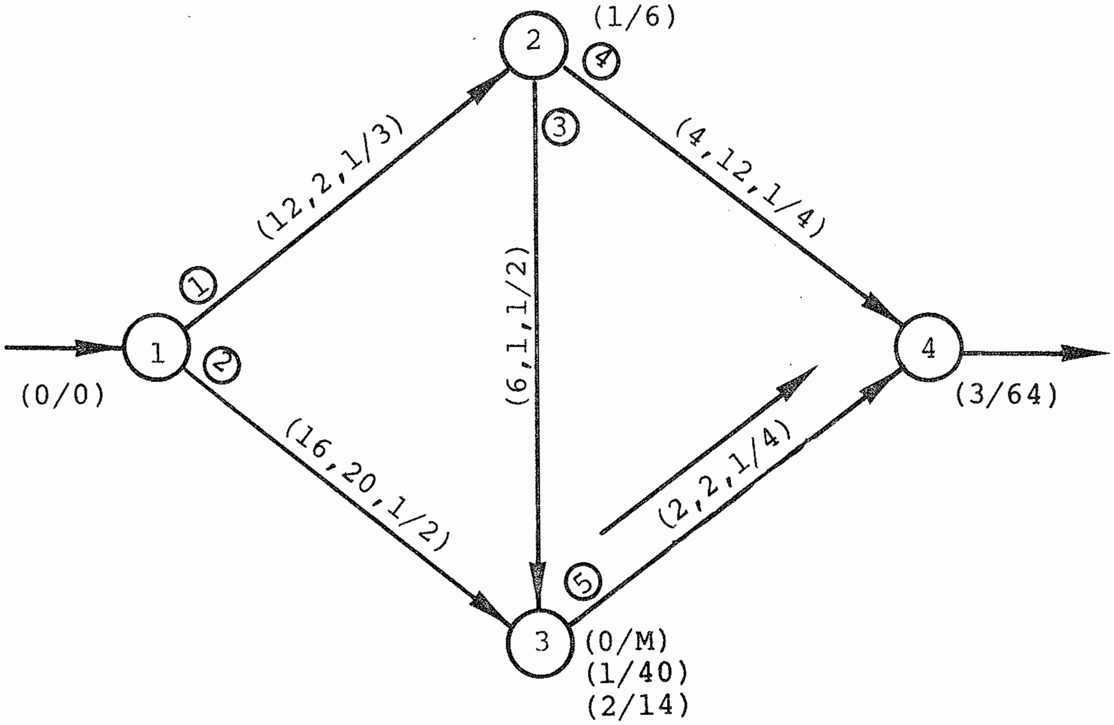


Figure 20 Initial Marginal Network with Minimum Cost Flow Augmenting Chain Algorithm Labels

The algorithm finds the flow augmenting chain to each node in the marginal network. The arcs indicated by the pointers define what is called the flow augmenting tree. One complication that has not been discussed is that flow can be generated from either the source or from a flow generating cycle. If the flow augmenting tree consists of no flow generating cycles the algorithm above terminates in a finite number of steps. If flow generating cycles are in the tree the algorithm will not terminate but values of v_j will converge in the limit to the proper values. The algorithm must be modified to provide finite termination.

Consider a flow augmenting chain which includes a flow generating cycle. The chain is:

$$C = \{i_1, b_1, i_2, b_2 \ldots b_{g-1}, i_g, b_g \ldots b_{m-1}, t\}$$

here $i_1 = i_g$ hence the branches b_1 through b_{g-1} form a cycle. The branches b_g to b_{m-1} form a simple chain from i_g to the sink. Notice that i_g would be the sink if the sink were on the cycle. To calculate the cost of obtaining one unit of flow at the sink through this chain it is first necessary to calculate the cost of generating one unit of flow at node i_g. Flow can be generated at i_g only the gain of cycle

$$\beta = \prod_{i=1}^{g-1} k_i$$

is greater than one. If this is true, in order to obtain $f_g=1$ then it is necessary that:

$$f_g = f_{g-1} \cdot k_{g-1} - f_1$$

$$\text{but } f_{g-1} = f_1 \cdot \prod_{i=1}^{g-2} k_i$$

$$\text{then } f_g = f_1 [\beta-1]$$

$$\text{when } f_g = 1 \text{ then } f_1 = \frac{1}{\beta-1}$$

Thus to obtain one unit of flow in arc b_g the following flows are required: $f_1 = 1/\beta-1$, $f_2 = k_1/\beta-1$, $f_3 = k_1 . k_2/\beta-1$, etc. The cost of obtaining one unit of flow at i_g is therefore

$$v_{i_g} = \sum_{\ell=1}^{g-1} (c_\ell \cdot \prod_{i=1}^{\ell-1} k_i)/(\beta-1)$$

$$= \sum_{\ell=1}^{g-1} (c_\ell / \prod_{i=\ell}^{g-1} k_i) \cdot (\beta/\beta-1) \qquad (14)$$

Equation (14) provides a means for calculating node costs for nodes on a cycle. If equation (14) calculates the node cost for $v_{i_1} = v_{i_g}$ then equation (15) can be used to calculate other node costs on the chain

$$v_{i_{\ell+1}} = (v_{i_\ell} + c_\ell)/k_\ell \text{ for } \ell \geq 1 \qquad (15)$$

The algorithm can now be rewritten incorporating equation (14). The only change will be in step 3. The new step 3 is written

3. If some v_j has been changed in step 2, use the pointers to determine if a cycle has been formed. If so use equation (14) to calculate the node cost for one node on the cycle. Use equation (15) to calculate the node costs for other nodes on the cycle. Then return to step 2.

   If no cycles are found simply return to step 2.

   If all arcs are inspected without changing any node cost, v_j, then terminate. v_t (t is the sink) is the cost of the minimum cost flow augmenting chain. The chain itself can be recovered from the pointers p_i. If at termination $p_t = 0$ there is no chain to the sink in the marginal network.

The modified algorithm will terminate in a finite number of steps even if cycles are part of the flow augmenting tree. Figure 21 illustrates a marginal network in which the minimum cost flow augmenting chain to the sink does include a cycle. The sink lies on the cycle. The labels are shown in the figure. Notice that the cycle can be detected after node 3 is assigned the label (4/11) through arc 5. The final labels on the cycle are calculated using equation 14 and 15.

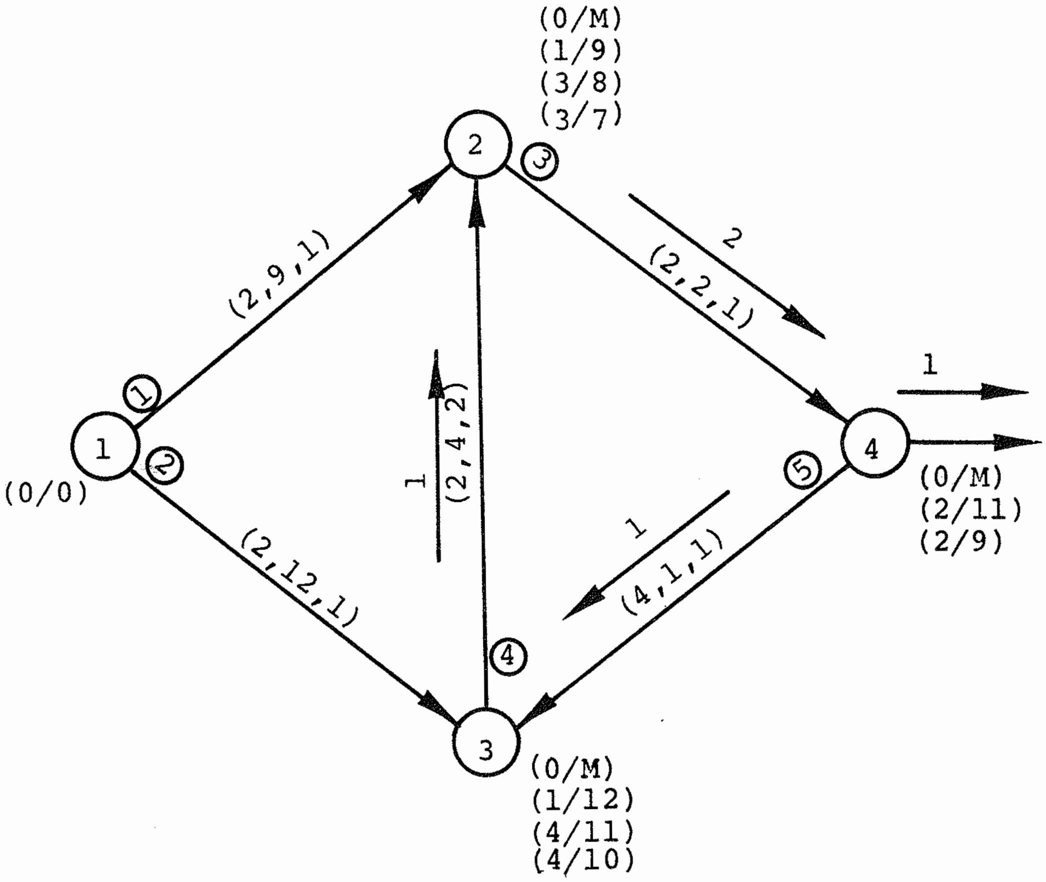


Figure 21 A flow augmenting chain with a cycle

A flow generating cycle can only occur if the cycle gain is greater than one. Thus if all arc gains are less than one (for arcs with capacity greater than zero) the modified algorithm need not be used. Even if the original network has all arc gains less than one, however, the marginal network will have gains greater than one on its mirror arcs. The modified minimum cost flow augmenting chain algorithm is therefore a critical part of the network with gains algorithm.

IV.3 Augmenting the Flow

Once the minimum cost flow augmenting chain is found, it is necessary to increase the flow in the arcs of the chain in order to increase the flow into the sink by as much as possible. We will calculate the flow change in the original network by calculating the maximum flows that can pass through the flow augmenting chain in the marginal network. The amount of the flow change depends on the upper bounds on the arcs of the marginal network.

Let $C = \{i_1, b_1, i_2, b_2 \ldots b_{g-1}, i_g, b_g \ldots i_m\}$.

Consider first the case in which the flow augmenting chain is a simple path from source to sink. Let the flow increase at the sink be ΔF_t. For arc b_ℓ in the chain the flow is:

$$f_\ell^* = \Delta F_t / \mathop{\pi}\limits_{i=\ell}^{m-1} k_i \qquad (16)$$

(the asterisk is to denote flow in the marginal network)

The amount the flow can increase is limited by the upper bound on the flow in the marginal network. Thus

$$f_\ell^* \leq u_\ell^*$$

$$\text{or } \Delta F_t / \mathop{\pi}\limits_{i=\ell}^{m-1} k_i \leq u_\ell^*$$

$$\text{or } \Delta F_t \leq u_\ell^* \mathop{\pi}\limits_{i-\ell}^{m-1} k_i$$

To obtain the maximum flow change

$$\Delta F_t = \mathop{\text{Min}}\limits_{1 \leq \ell \leq m-1} \{u_\ell^* \mathop{\pi}\limits_{i-\ell}^{m-1} k_i\} \qquad (17)$$

Once ΔF_t is known equation (16) is used to calculate the flow change for each arc in the marginal network.

If the flow augmenting chain includes a cycle such that $i_1 = i_g$ a slightly different approach must be taken for the arcs in the cycle. The flow increase in arc b_g is from (16)

$$f_g^* = \Delta F_t / \prod_{i=g}^{m-1} k_i \qquad (18)$$

To generate this amount of flow requires that in arc b_1:

$$f_1^* = f_g^*/\beta-1 \qquad (19)$$

and in arc b_ℓ in the cycle ($\ell \leq g-1$):

$$f_\ell^* = f_1^* \prod_{i-1}^{\ell-1} k_i$$

$$= f_1^*\beta/ \prod_{i=\ell}^{g-1} k_i$$

$$= f_g^*\beta/(\beta-1)(\prod_{i=\ell}^{g-1} k_i)$$

Inserting the value for f_g^* from equation (18):

$$f_\ell^* = \Delta F_t\beta/(\beta-1)(\prod_{i=\ell}^{m-1} k_i) \text{ for } 1\leq\ell\leq g-1 \qquad (20)$$

Thus since $f_\ell^* \leq u_\ell^*$

$$\Delta F_t\beta/\beta-1)\cdot(\prod_{i=\ell}^{m-1} k_i) \leq u_\ell^*$$

$$\text{for } 1 \leq \ell \leq g-1$$

$$\text{or } \Delta F_t \leq (\frac{\beta-1}{\beta})\ u_\ell^*(\prod_{i=\ell}^{m-1} k_i)$$

$$\text{for } 1 \leq \ell \leq g-1$$

Arcs on the chain connecting the cycle to the sink follow the relationships previously given for the chain from source to sink. Thus the maximum flow increase at the sink for a chain including a cycle is:

$$\Delta F_t = \mathrm{Min}\Big\{ \underset{1\leq \ell\leq g-1}{\mathrm{Min}} \Big[\Big(\frac{\beta-1}{\beta}\Big)u^*_\ell \Big(\prod_{i=\ell}^{m-1} k_i\Big)\Big], \qquad (21)$$

$$\underset{g\leq \ell\leq m-1}{\mathrm{Min}} \Big[u^*_\ell \prod_{i=\ell}^{m-1} k_i\Big]\Big\}$$

The flow changes in the arcs of the chain are calculated by equation (16) for arcs not in cycle and equation (20) for arcs in the cycle.

Given the maximum flows in the arcs of the marginal network, the flow changes in the original network are determined by

$$\Delta f_{ij} = f_{ij}{}^* - f_{ji}{}^*/k_{ij} \qquad (22)$$

Notice that although flows in the marginal network are always positive the flow change in the original network is positive or negative depending on whether the flow is in the forward or mirror arc.

IV.4 Example Problem

Figure 22 shows the sequence of marginal networks as the algorithm progresses through the example problem. Notice that the first marginal network is the same as the original network because the initial flows are zero. The flow augmenting chain algorithm yields a simple chain {1,(1,2),2,(2,3),3,(3,4),4}. The minimum per unit cost of flow at the sink is 64. The maximum flow that can be obtained at the sink through in chain is 1/2. With this flow both arcs (1,2) and (3,4) become saturated. The flows in the marginal network are used to adjust the flows according to equation (22) in the original network. The new flows appear in 22b.

These flows are used to construct the marginal network in 22b. The minimum cost flow augmenting chain in this network is a cycle and is {4,(4,3),3,(3,2),2,(2,4),4}. The cost per unit of flow obtained through this chain is 80 and the maximum flow that can be obtained at the sink is 1/2. With this flow arcs (3,2), (4,3) and (2,4) become saturated in the marginal network. The flows in the marginal network are used to adjust the flows in the original network which now appear in 22c. The total flow at the sink is now 1. Notice that saturation of arc (3,2) and (4,3) in the marginal network corresponds to complete removal of the flows in arcs (2,3) and (3,4) respectively in the original network.

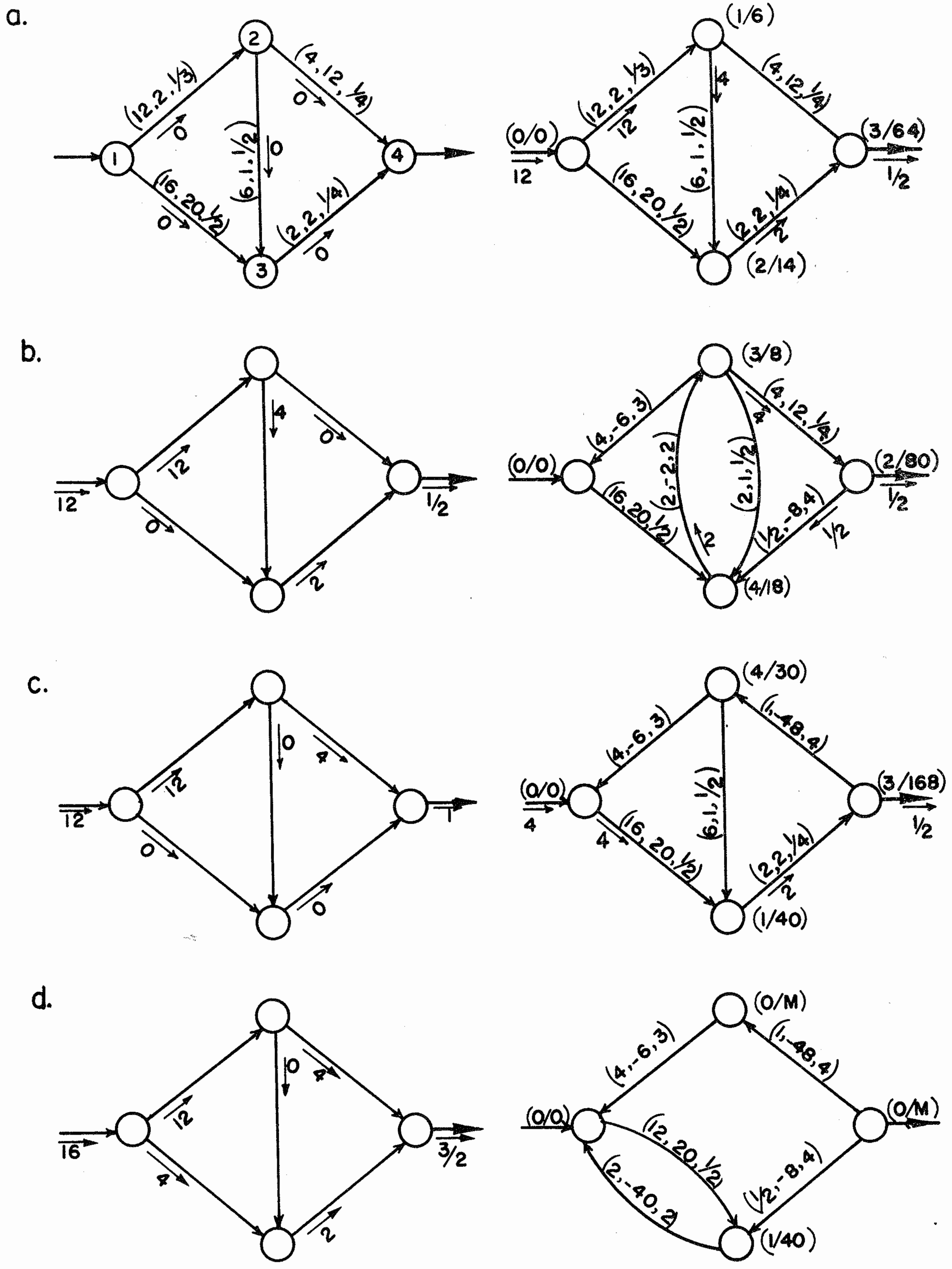


Figure 22 The original and marginal networks for an example problem.

The flows in the original network in 22c are used to construct the marginal network of 22c. The flow augmenting chain yields a cost of 168 per unit of flow at the sink and maximum increase of 1/2. The modified flows in the original network appear in 22d. The marginal network for these flows shown in 22d has no flow augmenting chain to the sink (both arcs (2,4) and (3,4) are saturated in the original network). Thus the maximum flow at the sink has been obtained for the example problem. The flows on the original network of 22d are the minimum cost flows for the sink flow of 3/2. The flows shown on the original networks of 22c and 22b are the minimum cost flows for 1 unit and 1/2 unit of output flow respectively.

V. Example Water Distribution Problem

As an example of the application of these procedures to a water distribution problem consider a hypothetical system for the State of Texas. The system consists of four reservoirs; A, B, C and E and a junction point D, interconnected by a set of five canals and a river as shown in Figure 23. Reservoirs A and B are assumed to be in East and Northeast Texas respectively. Reservoir C is in North-Central Texas and E in West Texas. Demands for water can be made at each of the four reservoirs. The inflows into the system are the rainfall at reservoirs A, B and C and the availability of import water at reservoir A. The monthly capacities of the reservoirs and canals and the maximum import water availability are shown on the arcs and nodes of Figure 23 in thousands of acre-feet. The unit cost of transporting water over the canals and the cost of imported water is also shown in $/1000 acre-feet. The demands and rainfall for a twelve month period are shown in Tables 1 and 2 respectively.

The leakage coefficients or gain factors were computed from the following equations:

$$k_{ij} = 1 \text{ for supply and demand arcs}$$

$$k_{ij} = \ell \cdot m, \text{ otherwise}$$

where ℓ is the location factor. For canal flow,

ℓ = .985 in West Texas (E)

ℓ = .99 for Central Texas (C,D)

ℓ = .995 for East Texas (A,B)

For reservoir storage,

ℓ = .98 in West Texas (E)

ℓ = .985 in Central Texas (C,D)

ℓ = .99 in East Texas (A,B)

and m is the seasonal factor.

m = .995 in Winter from November to February

m = .99 in Spring and Fall. September to October or March to April

m = .985 in Summer from May to August

To determine the optimal operating policies of this water distribution system a similar network was considered for each of twelve months. The storage of water was depicted by joining

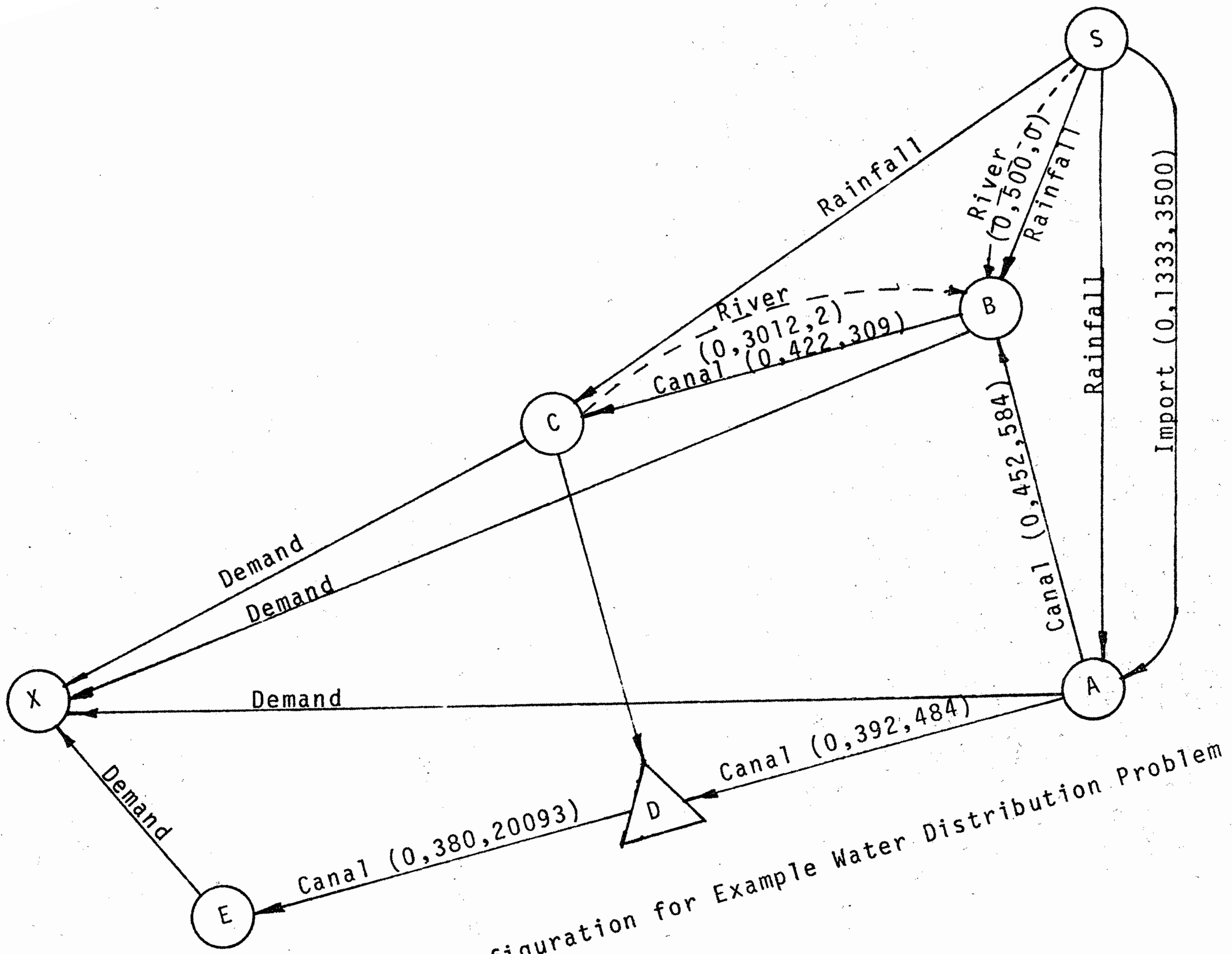


Figure 23 System Configuration for Example Water Distribution Problem

	RESERVOIR			
MONTH	A	B	C	E
JANUARY 1				42.00
FEBRUARY 2				61.00
MARCH 3				212.00
APRIL 4				229.00
MAY 5				375.00
JUNE 6				122.00
JULY 7			12.00	834.00
AUGUST 8	5.00	16.00	26.00	1098.00
SEPTEMBER 9	1.00		7.00	425.00
OCTOBER 10			2.00	20.00
NOVEMBER 11				19.00
DECEMBER 12				22.00

Table 1 Demands for Water for the Example Water Distribution Problem

		RESERVOIR	
MONTH	A	B	C
JANUARY 1	29.00	200.00	37.00
FEBRUARY 2	32.00	196.00	44.00
MARCH 3	33.00	211.00	45.00
APRIL 4	33.00	301.00	52.00
MAY 5	41.00	284.00	64.00
JUNE 6	11.00	173.00	20.00
JULY 7	1.00	20.00	
AUGUST 8			
SEPTEMBER 9		7.00	
OCTOBER 10	3.00	35.00	
NOVEMBER 11	12.00	66.00	22.00
DECEMBER 12	23.00	100.00	33.00

Table 2 Rainfall and Inflows for Example Water Distribution Problem

the nodes representing the same reservoir across the adjacent months. The storage arcs from the month of December were connected back to the respective nodes in the month of January as carry over arcs from the previous year. This in fact simulates the continuous operation of the system from year to year with a deterministic supply and demand pattern. Part of the resulting network is shown in Figure 24.

The optimal solution for this network was determined with computerized implementation of this algorithm. The solution is interpreted in terms of the problem in Tables 3 through 5. Table 3 shows the import water requirements for the six months when import water is available. Table 4 shows the shipping policies along the five canals for the twelve month period and Table 5 provides the guidelines for reservoir storage for the year. From these tables it is seen that the canal A-B of Figure 23 is completely unused and that canal D-E is used to capacity for nine months out of the year. Reservoirs B and E are grossly oversized and reservoir E which has the highest rate of evaporation is used to store water almost over the entire year. This is due to the limited size of the canal D-E, which cannot transmit all the required water to reservoir E during the periods of highest demand in July and August. Therefore, water has to be shipped to West Texas beginning from the month of December and stored in the West Texas reservoir to meet the high summer demands there. Hence, if the canals and reservoirs of this model represent a proposed system it would be beneficial to analyze the effect of increasing the size of canal D-E and eliminating canal A-B and reducing the sizes of reservoirs B and E.

This problem has also been solved as a pure network problem. That is, by ignoring the losses and setting the gain factors, k_{ij}, for all the arcs equal to one. The optimum solution without leakages (76,308,250) was found to be over 10% less than the optimum solution with leakages (85,039,136). This shows that even with the conservative leakages that were considered (maximum 3% per month) the cost difference is significant.

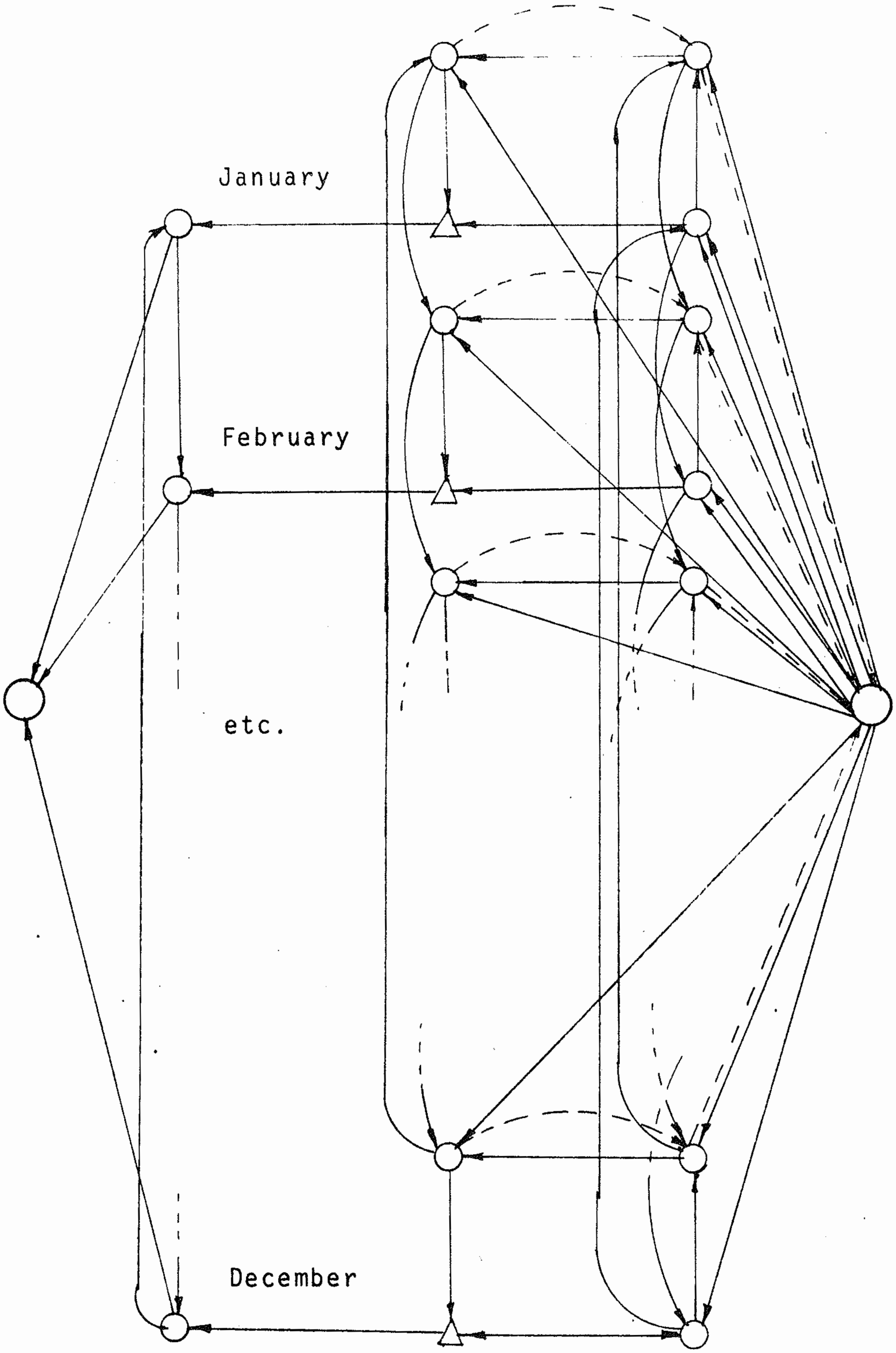


Figure 24 Twelve Month Representation of the Example Water Distribution Problem

MONTH	IMPORT REQUIREMENT
JANUARY 1	122.76
FEBRUARY 2	116.70
MARCH 3	102.91
APRIL 4	31.19
MAY 5	398.68
JUNE 6	1202.68

Table 3 Import Requirements for the Example Water Distribution Problem

			CANAL		
	A-B	A-D	B-C	C-D	D-E
CAPACITY, 1000 ACRE-FEET/MONTH	452	392	422	392	380
$/1,000 ACRE-FEET	584	484	309	346	200.93
MONTH					
JANUARY 1	0.00	151.76	200.00	234.01	380.00
FEBRUARY 2	0.00	148.70	196.00	237.07	380.00
MARCH 3	0.00	135.91	211.00	251.80	380.00
APRIL 4	0.00	64.19	277.04	323.53	380.00
MAY 5	0.00	389.68	0.00	0.00	380.00
JUNE 6	0.00	389.68	0.00	0.00	380.00
JULY 7	0.00	348.83	0.00	40.85	380.00
AUGUST 8	0.00	389.68	0.00	0.00	380.00
SEPTEMBER 9	0.00	47.70	354.07	34.02	380.00
OCTOBER 10	0.00	3.00	20.33	17.93	20.51
NOVEMBER 11	0.00	0.00	0.00	19.68	19.39
DECEMBER 12	0.00	0.00	271.44	302.66	332.43

Table 4 Canal Shipping Policy for the Example Water Distribution Problem

	RESERVOIR			
	A	B	C	E
CAPACITY IN 1000 ACRE-FEET	824	2430	946	3100
MONTH				
JANUARY 1	0.00	0.00	0	626.67
FEBRUARY 2	0.00	0.00	0	922.50
MARCH 3	0.00	0.00	0	1053.56
APRIL 4	0.00	23.96	0	1163.73
MAY 5	0.00	307.36	64.00	1117.03
JUNE 6	824.00	472.73	82.09	1324.95
JULY 7	455.69	480.98	26.80	813.66
AUGUST 8	49.68	453.03	0.00	56.12
SEPTEMBER 9	0.00	96.96	0.00	0.00
OCTOBER 10	0.00	109.69	0.00	0.00
NOVEMBER 11	12.00	174.05	2.32	0.00
DECEMBER 12	0.00	0.00	0.00	303.81

Table 5 Reservoir Storage Policy for the Example Water Distribution Problem

VI. Conclusions

The algorithm developed in this study to solve network problems with gains has proved to be very successful in solving a large number of different types of network problems with various gain parameters. The gain parameter associated with each of the arcs in the network can be any nonzero quantity. The unit costs can be any positive or negative quantity and any directed network cyclic or acyclic can be considered as long as the cost to traverse any cycle is not negative. Multiple arcs between two nodes are permissible. The solution technique is simple in concept and very easy to implement either for hand calculations or on a digital computer. All optimal intermediate solutions are available. Also, the algorithm can be used to find the maximal flow through the network. This maximal flow is at the minimal cost.

With respect to the distribution of water, the proposed algorithm adds a new dimension to the cost considerations that determine the optimal operating policies of the system. The simulation studies of water distribution plans can now be performed to a greater depth of analysis than before by considering the evaporation and seepage characteristics of the reservoirs and canals. The selection of the canal-reservoir system and the optimal policy for managing these can now be achieved with a greater degree of understanding of the system than has been possible before.

References

1. Balas, Egon, "The Dual Method for the Generalized Transportation Problem," Management Science, Vol. 12, pp. 555-568, 1966.

2. Balas, Egon and Hammer (Ivanescu), P. L., "On the Generalized Transportation Problem," Management Science, Vol. 11, pp. 188-202, 1964.

3. Bhaumik, Gora, Optimum Operating Policies of a Water Distribution System with Losses, Unpublished Ph.D. Dissertation, The University of Texas at Austin, 1973.

4. Bellman, Richard I., "On a Routing Program," Quart. Appl. Math., Vol. 16, pp. 87-90, 1958.

5. Busacker, R. G. and Gowen, P. J., "A Procedure for Determining a Family of Minimal Cost Network Flow Patterns," ORO Technical Paper 15, Operation Research Office, The John Hopkins University, 1961.

6. Busacker, R. G. and Saaty, T. L., "Finite Graphs and Networks: An Introduction with Applications," McGraw-Hill, New York, 1965.

7. Charnes, A. and Cooper, W. W., "Management Models and Industrial Applications of Linear Programming," Vols. I and II, Wiley, New York, 1961.

8. Charnes, A., Cooper, W. W. and Miller, M. H., "Application of Linear Programming to Financial Budgeting and Costing of Funds," J. Business University, Chicago, Vol. 32, pp. 20-46, 1959.

9. Charnes, A. and Raike, W. M., "One-Pass Algorithm for Some Generalized Network Problems," Operations Research, Vol. 14, pp. 914-924, 1966.

10. Dantzig, George B., "Linear Programming and Extensions," Princeton University Press, Princeton, New Jersey, 1963.

11. Dantzig, G. B. and Ferduson, A. R. "The Allocation of Aircrafts to Routes - An Example of Linear Programming Under Uncertain Demand," Management Science, Vol. 3, pp. 45-73, 1956.

12. Durbin, E. P. and Kroenke, M., "The Out-of-Kilter Algorithm: A Primer," The RAND Corporation Report No. RM-5472-PR, December 1967.

13. Eisemann, Kurt, "The Generalized Stepping-Stone Method for the Machine Loading Model," Management Science, Vol. 11, No. 1, pp. 154-177, September 1964.

14. Ford, Lester R. and Fulkerson, Delburt R., "Flows in Networks," Princeton University Press, Princeton, New Jersey, 1962.

15. Ford, L. R. and Fulkerson, D. R., "Maximal Flow Through a Network," Canadian Journal of Mathematics, Vol. 8, pp. 399-404, 1956.

16. Fujisawa, T., "Maximal Flow in a Lossy Network," Proceedings of Allerton Conference on Circuit and Systems Theory, Vol. 1, pp. 385-393, 1963.

17. Fulkerson, Delburt R., "An Out-of-Kilter Method for Minimal-Cost Flow Problem," J. Soc. Indust. Appl. Math., Vol. 9, No. 1, March 1961.

18. Glover, Fred, Karney, D., Klingman, D. and Napier, A., "A Computation Studies on the Start Procedures, Basis Change Criteria and Solution Algorithms for Transportation Problems", Research Report CS 93, Center for Cybernetic Studies, The University of Texas, Austin, September 1972.

19. Glover, Fred and Klingman, D., "On the Equivalence of Some Generalized Network Problems to Pure Network Problems," Research Report CS81, Center for Cybernetic Studies, The University of Texas, Austin, Texas, January 1972.

20. Glover, Fred and Klingman, D., "A Note on Computational Simplifications in Solving Generalized Transportation Problems," Research Report CS87, Center for Cybernetic Studies, The University of Texas, Austin, Texas, May 1972.

21. Glover, Fred, Klingman, D. and Napier, A., "Basic Dual Feasible Solutions for a Class of Generalized Networks," Operations Research, Vol. 20, No. 1, January-February 1972.

22. Glover, Fred, Klingman, Darwin, and Napier, Al, "Equivalence of Generalized Network and Generalized Transportation Problems", Research Report CS 125, Center for Cybernetic Studies, The University of Texas, Austin, Texas, January 1973.

23. Hu, T. C., "Integer Programming and Network Flows," Addison Wesley, Reeding, Mass., 1969.

24. Jarvis, John J. and Jezior, Anthony M., "Minimal Flow with Gains Through a Special Network," Operations Research, Vol. 20, No. 3, May-June 1972.

25. Jewell, W. S., "Optimal Flow Through Networks," Interim Technical Report No. 8 on Fundamental Investigations in Methods of Operations Research, Massachusetts Institute of Technology, Cambridge, 1958.

26. Jewell, William S., "Optimal Flow Through Networks With Gains," Operations Research, Vol. 10, No. 4, July-August 1962.

27. Johnson, Ellis L.,"Networks and Basic Solutions," Operations Research, Vol. 14, No. 4, July-August 1966.

28. Johnson, S. M., "Sequential Production Planning Over Time at a Minimum Cost," Management Science, Vol. 3, pp. 435-437, 1957.

29. Klein, Morton, "A Primal Method for Minimal Cost Flows with Application to the Assignment and Transportation Problems," Management Science, Vol. 14, No. 3, November 1967.

30. Lemke, C. E., "The Dual Method of Solving the Linear Programming Problem," Naval Research Logistic Quarterly, Vol. 1, No. 1, pp. 36-47, 1954.

31. Linsley, Ray K., Jr., Kohler, Max A. and Paulhus, Joseph L. H., "Hydrology for Engineers," McGraw-Hill, New York, 1958.

32. Lourie, J. R., "Topology and Computation of the Generalized Transportation Problem," Management Science, Vol. 11, pp. 177-187, 1964.

33. Malek-Zavarei, M. and Aggarwal, J. K., "Optimal Flows in Networks with Gains and Costs," Networks, Vol. 1, No. 4, pp. 355-365, 1972.

34. Maurras, Jean Francois, "Optimization of the Flow through Networks with Gains", Mathematical Programming, 3, 1972.

35. Mayeda, W. and Van Valkenburg, M. E., "Properties of Lossy Communication Nets," IEEE Trans. Circuit Theory, CT-12, pp. 334-388, 1965.

36. Minieka, Edward, "Optimal Flow in a Network with Gains", INFOR, Vol. 10, No. 2, June 1972.

37. Onaga, K., "Dynamic Programming of Optimal Flows in Lossy Communication Nets," IEEE Transactions on Circuit Theory, Vol. CT-12, No. 3, September 1966.

38. Onaga, K., "Optimum Flows in General Communication Networks," *J. Franklin Institute*, Vol. 283, pp. 308-327, 1967.

39. Prager, W., "On the Caterer Problem," *Management Science*, Vol. 3, pp. 15-23, 1956.

40. Salveson, M. E., "A Problem in Optimal Machine Loading," *Management Science,* Vol. 2, pp. 232-260, 1956.

41. Texas Water Development Board, "*A Comparison of Mass-Transfer and Climatic-Index Evaporation Computations from Small Reservoirs in Texas,*" Report No. 141, February 1972.

42. Texas Water Development Board, "*Monthly Evaporation Rates for Texas 1940 through 1965,*" Report No. 64, October 1967.

43. Texas Water Development Board, "*Stochastic Optimization and Simulation Techniques for Management of Regional Water Resource Systems,*" Report No. 131, July 1971.

44. Texas Water Development Board, "*Systems Simulation for Management of a Total Water Resource,*" Report No. 118, May 1970.

45. Texas Water Development Board, "*The Texas Water Plan,*" Report, November 1968.

A Computationally Efficient Algorithm for the Network with Gains Problem

by P. A. Jensen
and Gora Bhaumik

I. Introduction

The networks with gains model is a useful extension of the pure network model. Applications have been suggested in water resources planning (2), electrical power planning (6), and others (9). Several authors have suggested solution approaches (4,7,8,9,10,12). This paper suggests an approach which is simple in concept and very efficient computationally. The approach is based on (3) with the addition of several modifications. Large problems have been solved in times comparable to those required for the most efficient pure network codes.

The network with gains problem treated here is structured on a network with m vertices denoted $v_1,v_2,...,v_m$ and n directed branches denoted $b_1,b_2,...,b_n$. When necessary, a branch will be identified by its end points using the notation $b_k(i,j)$ or simply (i,j) where the initial vertex of the branch is v_i and the terminal vertex is v_j. Associated with each branch b_k will be a flow f_k, a capacity for flow, c_k, a cost per unit flow, h_k and a positive gain, a_k. The branch flows are the decision variables of the problem while the branch costs, capacities and gains are parameters. Flows are bounded from below by zero. A single source vertex is defined and called v_s, and a single sink vertex is defined and called v_t.

The structure of the problem is described by the network $G=(V,\Gamma)$ where V is the set of vertices and Γ is the set of branches. A useful definition related to structure is that the set A_k is the set of branches whose initial vertex is v_k, and B_k is the set of branches whose terminal vertex is v_k.

A path in the network is a sequence of alternating vertices and branches written as:

$$v_{i1}, b_{j1}, v_{i2}, b_{j2}, \ldots, v_{i(k-1)}, b_{j(k-1)}, v_{ik}$$

$$\text{where } b_{j\ell} = (i_\ell, i_{\ell+1}).$$

Here $i_1, i_2, \ldots, i_k$ represent indices of vertices, while $j_1, j_2, \ldots, j_{k-1}$ represent indices of branches. The branches of a path will be distinct but the vertices of a path may be repeated. A path with distinct vertices is called a simple path. Note that a path as defined here is directed. A circuit is a path whose vertices are all distinct except the end points. Thus $v_{i1}=v_{ik}$ for a circuit.

The problem to be solved is to find an assignment of flows to branches such that a given amount of flow is obtained at the sink at a minimum total cost. In addition, the flow must not exceed the capacity for any branch, and flow must be conserved at each vertex except the source and sink. Flow will not be conserved in branches as a flow of f_k which enters a branch $b_k(i,j)$ at v_i will arrive at v_j with a value of $a_k f_k$. A gain is allowed to be any positive number.

This is a linear programming problem and can be written in the usual format:

$$\text{Min} \sum_{k=1}^{n} h_k f_k \tag{1}$$

subject to:

$$\sum_{k \epsilon B_i} a_k f_k \quad - \quad \sum_{k \epsilon A_i} f_k = 0;\ i=1,\ldots,m;\ i \neq s,t \tag{2}$$

$$\sum_{k \epsilon B_t} a_k f_k = F_{s,t} \tag{3}$$

Here $F_{s,t}$ is a given amount of output flow that is to be obtained. The omission of a conservation of flow constraint at v_s implies that an unlimited quantity of flow is available at the source. Most network with gains problems with positive gains can be put into this format with a suitable construction of the network.

An assignment of flows to the branches that satisfies (2) will be called F. An F that satisfies (1) for some value of flow at the sink less than $F_{s,t}$ will be called an intermediate optimum. The flow F which satisfies (1), (2) and (3) is called the optimum.

The general approach taken to obtain a solution will be similar to that of Busacher and Gowen (3) for the pure network problem. The procedure begins with an intermediate optimal F_0 for some value of output flow at v_t. All flows equal to zero are acceptable starting solutions when there are no directed circuits in the network with negative cost. Next an augmenting network is constructed G_A^0 which determines for each vertex the minimum cost per unit of additional flow to the vertex and the path over which that flow may be obtained.

For a pure network this network would take a tree structure, but for the network with gains, the network may consist of one or more components. Each component contains either a flow generating circuit or the source . The output flow is increased in G_A^0 along the minimum cost path defined for the sink until one or more branches in G_A^0 become saturated. This flow augments F_0 to obtain the flow F_1. The flow F_1 thus obtained is an intermediate optimum. A new augmenting network is constructed, G_A^1, and the output flow is again augmented to obtain F_2. This process continues iteratively until the desired output flow or the maximum flow is obtained. At every step F_k

is an intermediate optimum; hence at termination the flow pattern must be the optimum or the maximum flow through the network if $F_{s,t}$ is greater than the maximum flow.

The approach can be made computationally efficient by using a simplex operation to go from one augmenting network to the next. Computerized versions of the algorithm have been tested on large problems (up to 2400 branches) and take only about twice as much time as the most efficient pure network algorithms on problems with the same structure but no gain.

II. Flow Generating Circuits

Flow can be augmented into a vertex from one of two sources. Either the flow can originate at v_s, at which an unlimited amount of flow is available, or it can originate at a flow generating circuit. A flow generating circuit is a set of branches which form a circuit such that the circuit gain (the product of the branch gains) is greater than one. Figure 1 illustrates the concept. In the figure the triplets show (h_k, c_k, a_k). Flows are shown by the arrows above the arcs.

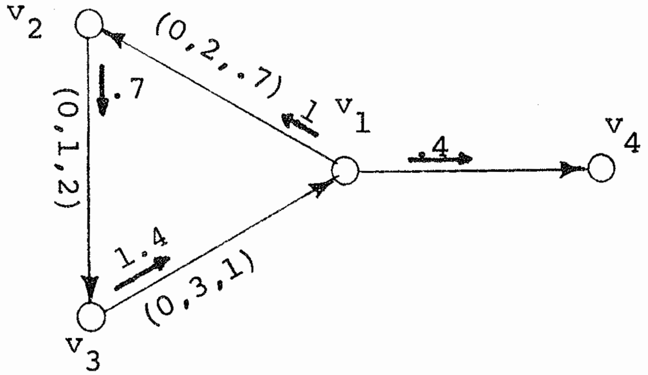


Figure 1: Flow generating circuit

Note that the flow f(1,2) = 1 produces a flow f(2,3) = .7; furthermore, this produces the flow f(3,1) = 1.4. At v_1 conservation of flow requires that f(1,4) = .4. Thus the flow in the circuit with circuit gain greater than 1 has generated flow in b(1,4).

In general, the circuit is defined by a sequence of vertices and branches which can be written as:

$$v_{i1}b_{j1}v_{i2}b_{j2}\cdots\cdot b_{jk}v_{i1}$$

where $b_{j\ell} = b(i_\ell, i_{\ell+1})$ for $\ell = 1,2,\ldots,k-1$

and $b_{jk} = b(i_k, i_1)$.

The gain of the circuit is β where $\beta = \prod_{\ell=1}^{k} a_{j\ell}$. For a flow generating circuit $\beta > 1$. Flow can be generated out of the circuit at any vertex of the circuit. Let τ_{i1} be the flow generated by the circuit and removed at vertex v_{i1}. Let f_{j1} and f_{jk} be the flows in the first and last branches.

By conservation of flow at v_{i1}

$$f_{j1} + \tau_{i1} = a_k f_{jk}$$

But $$f_{jk} = \prod_{\ell=1}^{k-1} a_{j\ell} \cdot f_{j1}$$

$$\therefore \quad f_{j1} + \tau_{i1} = \beta f_{j1}$$

$$\therefore \quad \tau_{i1} = f_{j1}(\beta-1) \tag{4}$$

Thus $\beta-1$ units of flow can be generated at v_{i1} for every unit of flow routed around the circuit.

Let τ_{jk} be the flow entering v_{i1} from b_{jk} in the circuit, thus $\tau_{jk} = a_k f_{jk}$. Also $\tau_{jk} = \beta f_{ji}$. Thus the flow out of circuit can be represented as:

$$\tau_{i1} = \tau_{jk}(1-1/\beta). \tag{5}$$

III. The Augmenting Network

For any intermediate optimal flow,F, the augmenting network $G_A = (V,\Gamma_A)$ is constructed from the original network $G = (V,\Gamma)$ for the purpose of finding the minimum cost path over which flow can be augmented into the sink.

The vertices of G_A are the same as those of G. The branches of G_A depend on the flow F on the branches of G. The branches of G_A are chosen from the set of admissible branches. Define the set of admissible branches Γ_D as follows:

$$b_k(i,j) \epsilon \Gamma_D \text{ if } b_k(i,j) \epsilon \Gamma \text{ and } f_k < c_k$$

$$b_{k+n}(j,i) \epsilon \Gamma_D \text{ if } b_k(i,j) \epsilon \Gamma \text{ and } f_k > 0$$

If the branch b_k appears in G_A, its parameters are derived from the corresponding branch in G. The capacity of b_k will be c_k-f_k, its cost will be h_k and its gain will be a_k. The branch $b_{k+n}(j,i)$ is called the mirror branch of $b_k(i,j)$. Its parameters are $c_{k+n} = f_k \cdot a_k$, $h_{k+n} = -h_k/a_k$ and $a_{k+n} = 1/a_k$. Branch b_k is also called the mirror branch of b_{k+n}.

An augmenting network is a collection of admissible branches which define one and only one augmenting path for each vertex. Augmenting paths are of two types, those that originate at the source and those that originate at a flow generating circuit. An augmenting path originating at the source is a simple path whose initial vertex is the source. An augmenting path originating at a circuit consists of the circuit and possibly a simple path originating at some vertex in the circuit. Figure 2 shows an augmenting network with one and only one augmenting path defined for each vertex. Note the paths for two vertices are not necessarily branch disjoint.

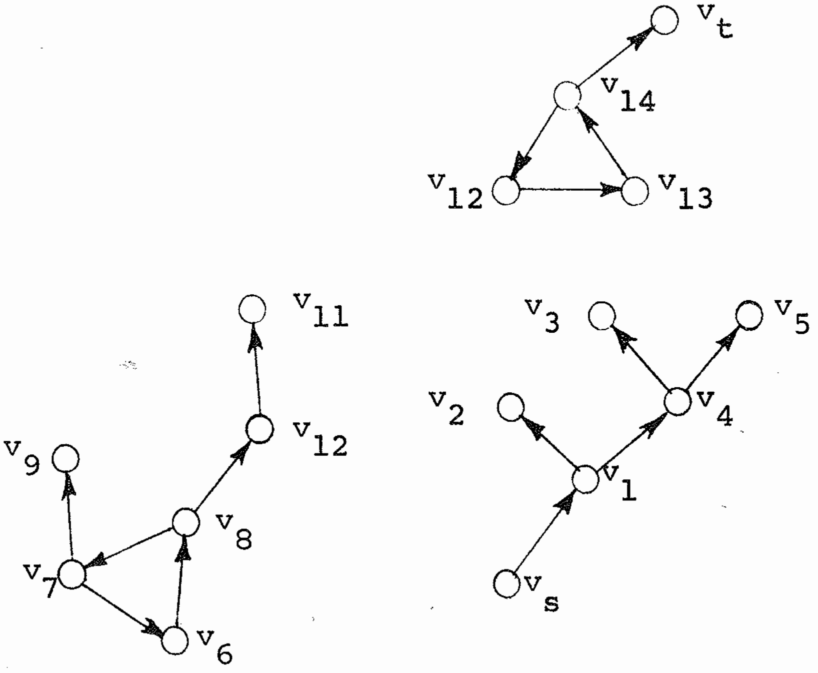


Figure 2: A Flow Augmenting Network

Note that a component of G_A without a circuit, is a tree with k vertices and k-1 branches, while a component of G_A with a circuit connects k vertices with k branches. There is at least one component which is a tree rooted at the source (it may be v_s alone). Thus the augmenting network has m vertices and m-1 branches.

The problem now is to choose Γ_A from the set of admissible branches to form G_A such that for each v_i there exists one and only one augmenting path into v_i. Furthermore, the path thus defined must be the path which can provide flow into v_i at the minimum cost per unit. To calculate the per unit cost, it is necessary to consider the branch gains a_k as well as branch costs h_k, as one unit flow at v_i may require more or less than one unit of flow in each of the branches of the augmenting path.

Define for each vertex, v_i, a potential $\delta(i)$ that is the cost of obtaining one unit of flow at v_i using only the branches in Γ_A. For a tree rooted at the source, the values of $\delta(i)$ are easily assigned by setting $\delta(s) = 0$ and applying recursively the relation $\delta(j) = (\delta(i) + h_k)/a_k$ where $b_k(i,j)\epsilon\Gamma_A$. This relationship follows from:

Theorem 1: If $\delta(i)$ is the cost of obtaining one unit of flow at v_i, then the cost of obtaining one unit of flow at vertex j through the branch $b_k(i,j)$ is:

$$\delta(j) = (\delta(i) + h_k)/a_k. \tag{6}$$

Proof: To obtain one unit of flow at v_j requires that $1/a_k$ units of flow be transmitted through v_i. The cost of bringing this flow to v_i is $\delta(i)/a_k$. The cost of transmitting it through b_k is h_k/a_k. Thus $\delta(j) = (\delta(i) + h_k)/a_k$.

For components of G_A with a circuit the potentials for the vertices on the circuit must first be obtained.

Theorem 2: Given the circuit $v_{i1}, b_{j1}, v_{i2}, b_{j2}, \ldots, v_{ih}, b_{jh}, v_{i1}$ the potential at v_{i1} is:

$$\delta(i_1) = (\beta/\beta-1) \sum_{k=1}^{h} h_{jk} / \prod_{\ell=k}^{h} a_{j\ell}. \tag{7}$$

Proof: First find the cost of routing one unit of flow around the circuit starting at v_{i1}.

The flow in b_{j1} is 1 with cost h_{j1}

The flow in b_{j2} is a_{j1} with cost $a_{j1}h_{j2}$

The flow in b_{j3} is $a_{j1}a_{j2}$ with cost $a_{j1}a_{j2}h_{j3}$

.

.

.

The flow in b_{jh} is $\prod_{i=1}^{h-1} a_{ji}$ with cost $\prod_{\ell=1}^{h-1} a_{j\ell}h_{jn}$.

Thus the total cost to route one unit of flow around the circuit starting at v_{i1} is:

$$h_{j1} + \sum_{k=2}^{h} \left(\prod_{\ell=1}^{k-1} a_{j\ell}\right)h_{jk}.$$

One unit of flow out of circuit at v_{i1} requires $1/(\beta-1)$ units in branch b_{j1}.

Thus the cost per unit of flow at v_{i1} is:

$$\delta(i_1) = \frac{h_{j1} + \sum_{k=2}^{h} (\prod_{\ell=1}^{k-1} a_{j\ell}) h_{jk}}{\beta-1}$$

Multiplying and dividing this expression by β yields:

$$\delta(i_1) = (\beta/\beta-1) \sum_{k=1}^{h} h_{jk} / \prod_{\ell=k}^{h} a_{j\ell}$$

This proves theorem 2.

Given $\delta(i_1)$, potentials can be determined for all other vertices in the flow component using Equation 6.

An optimal augmenting network is G_A^* such that for any other allowable augmenting network G_A

$$\delta^*(i) \leq \delta(i) \text{ for all } v_i,$$

where $\delta^*(i)$ are calculated according to the rules above for the graph G_A^* and $\delta(i)$ are calculated for G_A.

<u>Theorem 3</u>: A necessary and sufficient condition that $G_A^* = (V, \Gamma_A^*)$ is an optimal augmenting network is that:

$$\delta^*(j) \leq (\delta^*(i) + h_k)/a_k$$

for all $b_k(i,j) \epsilon \Gamma_D$.

<u>Proof</u>: To prove necessity, assume that there does exist an admissible branch $b_k(i,j)$ not in G_A^* such that:

$$\delta^*(j) > (\delta^*(i) + h_k)/a_k.$$

Since the branch is admissible, flow can be increased from v_i to v_j. The cost of obtaining a unit of flow at v_j from v_i is $(\delta^*(i) + h_k)/a_k$. Since this cost is less than $\delta^*(j)$, G_A^* cannot be optimum. Therefore, we have a contradiction.

To show sufficiency assume that G_A is not optimum but that: $\delta(j) \leq (\delta(i)+h_k)/a_k$ for all admissible branches. Since G_A is not optimum there must be some v_k such that $\delta(k) > \delta^*(k)$. Then there must be some augmenting path to v_k in G_A^* which is different from that in G_A. Let the path in G_A^* be $(v_{i1}, b_1, v_{i2}, \ldots, v_{ik})$. Consider the vertices in this path in G_A and let $v_{i\ell}$ be the first vertex (in the order they appear in the path) such that $\delta(i_\ell) > \delta^*(i_\ell)$. Thus $\delta(i_{\ell-1}) = \delta^*(i_{\ell-1})$. By the rules for determining vertex potentials

$$\delta^*(i_\ell) = (\delta^*(i_{\ell-1}) + h_{\ell-1})/a_\ell.$$

Thus $\delta(i_\ell) > (\delta(i_{\ell-1}) + h_{\ell-1})/a_\ell$ for the admissible branch $b_{\ell-1}$ and a contradiction has been found. Thus theorem 3 has been proved.

Theorem 3 suggests an algorithm that can be used to construct an optimal augmenting network. This algorithm was used in (2) to find every G_A. In the procedures of this paper, it is used to find only the initial one. In the algorithm P_i is a pointer for v_i which indicates the branch of Γ_D which terminates at v_i in G_A.

Algorithm to find an Optimal Flow Augmenting Network

```
1.  Let $\delta(i) = M$ for all $v_i \varepsilon V$ except $v_s$, where M is a large number.
    Let $\delta(s) = 0$.
    Let $P_i = 0$ for $v_i \varepsilon V$
2.  For each branch $b_k(i,j) \varepsilon \Gamma$
    if $f_k < c_k$ and $\delta(j) > (\delta(i) + h_k)/a_k$,
    set $\delta(j) = (\delta(i) + h_k)/a_k$
    set $P_j = k$
    if $f_k > 0$ and $\delta(i) > (\delta(j) - h_k/a_k)/(1/a_k)$,
    set $\delta(i) = (\delta(j) - h_k/a_k)/(1/a_k)$
    set $P_i = k+n$
    if neither condition occurs, make no change.
3.  If none of the $P_i$ changed in step 2, stop with the $G_A^*$ defined by the pointers. If one or more of the $P_i$ changed, use the $P_i$ to find any flow generating circuits that may have been formed. If so, use equations 6 and 7 to set the potentials for the vertices in the circuit.
    Repeat step 2.
```

IV. The Maximum Flow in the Augmenting Network

The augmenting network is used to determine the change in the branch flows of the original network which will augment the flow into the sink at the minimum cost per unit. This is accomplished in two steps. First the maximum flow to the sink of G_A is determined. This flow is then used to modify the flow of G. In this section branch designations, flows and parameters refer entirely to the graph G_A. $\underset{\sim}{f}_t$ will be the flow out of the sink in G_A.

The flow augmenting path for v_t may be rooted at v_s or at a circuit. First consider the case when the path is rooted at the source. This is a simple path which can be written:

$$v_{i1}, b_{j1}, v_{i2}, \ldots, b_{jk}, v_{i(k+1)}$$

where

$$v_{i1} = v_s \text{ and } v_{i(k+1)} = v_t.$$

Define for each vertex v_{ia} on the path a gain parameter $\gamma(i_a)$ which is the product of the gain on each branch between v_a and v_t. Thus

$$\gamma(i_a) = \prod_{\ell=a}^{k} a_{j\ell} \quad a = 1 \ldots k.$$

This is a useful parameter because it indicates the change in $\underset{\sim}{f}_t$ for a unit change in f_{ja}. Thus for a unit change in f_{ja},

$\underset{\sim}{f}_t$ increases to $\gamma(i_a)$. Alternatively for a unit increase in $\underset{\sim}{f}_t$, the flow f_{ja} must change by $1/\gamma(i_a)$.

Theorem 5: When the augmenting path is rooted at the source, the maximum value of $\underset{\sim}{f}_t$ is:

$$\Delta = \underset{\ell}{\text{Min}} \; c_{j\ell} \cdot \gamma(i_\ell) \qquad (8)$$

where i_ℓ is the index of the ℓth vertex in the path and j_ℓ is the index of the ℓth branch.

Proof: A value of Δ for $\underset{\sim}{f}_t$ results in a flow of $\Delta/\gamma(i_\ell)$ in branch ℓ. For feasibility this must be less than $c_{j\ell}$ or:

$$f_{j\ell} = \Delta/\gamma(i_\ell) \leq c_{j\ell}$$

$$\therefore \Delta \leq c_{\ell k} \cdot \gamma(i_\ell)$$

and theorem 5 is proved.

When the augmenting path is rooted at a circuit, the path consists of two parts, the circuit and a simple path to the sink from some vertex in the circuit. The branches b_{j1} through b_{jh} form the circuit and the branches $b_{j(h+1)}$ through b_{jk} form the simple path. It is possible that the sink is a vertex on the circuit in which case the augmenting path consists only of the circuit.

It follows from theorem 5 that for the portion of the path not in the circuit, the maximum flow increase at v_t is:

$$\Delta_p = \underset{h+1 \leq \ell \leq k}{\text{Min}} \; c_{j\ell} \cdot \gamma(i_\ell) \qquad (9)$$

Theorem 6: For the branches in the flow generating circuit, the maximum flow increase in $\underset{\sim}{f}_t$ is:

$$\Delta_c = (1-1/\beta)\ \mathrm{Min}\left[\underset{2\leq \ell\leq h}{\mathrm{Min}}\ \{c_{j\ell}\cdot\gamma(i_\ell)\}, c_{j1}\cdot\beta\cdot\gamma(i_{h+1})\right] \quad (10)$$

Proof: It has been established that

$$\tau_{i1} = f_{j1}(\beta-1) \text{ or } f_{j1} = \tau_{i1}/(\beta-1)$$

where β is the circuit gain and τ_{i1} is the quantity of flow generated out of the circuit at v_{i1}. It can be shown that the flow on the ℓth branch of the circuit is

$$f_{j\ell} = f_{j1}\cdot\beta\cdot\gamma(i_{h+1})/\gamma(i_\ell) \quad \text{or}$$

$$f_{j\ell} = \tau_{i1}\cdot\beta\cdot\gamma(i_{h+1})/\left[\gamma(i_\ell)\cdot(\beta-1)\right] \quad 2\leq\ell\leq h$$

To obtain an increase of Δ at v_t, τ_{i1} must increase by $\Delta/\gamma(i_{h+1})$. Thus

$$f_{j1} = \Delta/\left[\gamma(i_{h+1})\cdot(\beta-1)\right] \leq c_{j1} \quad \text{or}$$

$$\Delta \leq c_{j1}\cdot\gamma(i_{h+1})\cdot(\beta-1)$$

and from the expression for $f_{j\ell}$:

$$\left[\Delta/\gamma(i_{h+1})\right]\cdot\beta\cdot\gamma(i_{h+1})/\left[\gamma(i_\ell)\cdot(\beta-1)\right] \leq c_{j\ell}$$

Simplifying

$$\Delta \leq \left[(\beta-1)/\beta\right]\cdot c_{j\ell}\cdot\gamma(i_\ell) \quad 2\leq\ell\leq h$$

Combining these results:

$$\Delta \leq 1-1/\beta\ \ \mathrm{Min}\{\underset{2\leq\ell\leq h}{\mathrm{Min}}\left[c_{j\ell}\cdot\gamma(i_\ell)\right], c_{j1}\cdot\beta\cdot\gamma(i_{h+1})\}$$

which proves theorem 6.

Considering all the branches on the flow augmenting path, the maximum increment is:

$$\Delta = \text{Min}\ \{\Delta_p, \Delta_c\} \tag{11}$$

Equations 8 through 11 determine the maximum value of $\underset{\sim}{f}_t$ that can be obtained by increasing flows on branches of the augmenting path. That value is Δ. Δ in turn determines the flows in the branches. Thus for branches not on a circuit

$$f_{j\ell} = \Delta/\gamma(i_\ell) \quad h+1 \le \ell \le k.$$

For the branches on a flow generating circuit

$$f_{j\ell} = \Delta[\beta/\beta-1]/\gamma(i_\ell) \qquad 2 \le \ell \le h$$

and

$$f_{i1} = \Delta/[\gamma(i_{h+1}) \cdot (\beta-1)]$$

With these flows imposed on the augmenting network, at least one branch becomes saturated (flow equals capacity).

V. Augmenting the Flow in the Original Network

The flow on the augmenting network is now used to augment the flow in the original network to find a new intermediate optimum. Let F_K be the flow on G used to define the admissible branches (which in turn determine G_A). Let $f_\ell^K(i,j)$ be the flow on branch $b_\ell(i,j)$ determined by F_K. Let F_{K+1} be the augmented flow on G and $f_\ell^{K+1}(i,j)$ be the branch flow. Let $f_\ell(i,j)$ and $f_{\ell+n}(j,i)$ be flows from G_A as defined in the last section. The augmented flows are determined as follows:

$$f_\ell^{K+1}(i,j) = f_\ell^K(i,j) + f_\ell(i,j) - f_{\ell+n}(j,i)/a_\ell$$

for all $b_\ell \epsilon \Gamma$.

It can easily be shown that the flow F_{K+1} is feasible. The flow at sink for F_{K+1} has been increased by the quantity Δ over its value for F_K. F_{K+1} is an intermediate optimum because F_K was an intermediate optimum, and the augmenting flows provide increased flow at the sink at the minimum cost per unit.

VI. Optimization of the Augmenting Network

With the augmentation of the sink flow, one or more of the branches on the augmenting path becomes saturated and hence inadmissible. The problem now is to find an optimum augmenting network for the new flow. This task is pursued in an iterative manner with each augmenting network derived from the previous one. Therefore identify $G_A^K = (V,\Gamma_A^K)$ as the augmenting network at the Kth iteration, and address the problem of finding $G_A^{K+1} = (V,\Gamma_A^{K+1})$ after the flow has been changed.

Consider first the case in which only one branch becomes saturated. This branch is called the leaving branch, b_L. Deleting the leaving branch from G_A^K forms two graphs $G_{A1} = (X,\Gamma_{A1})$ and $G_{A2} = (\overline{X},\Gamma_{A2})$. The latter graph includes the sink and all other vertices disconnected from the source or a flow generating circuit by the removal of the leaving branch. It can be shown that G_{A2} is a directed tree rooted at the terminal vertex of the leaving branch. G_{A1} consists of the remaining vertices and branches of G_A. G_{A1} includes the source and all flow generating circuits of G_A. It follows that:

$$X \cup \overline{X} = V$$

$$\Gamma_{A1} \cup \Gamma_{A2} = \Gamma^K - b_L.$$

In addition to the branch leaving the admissible set, it is also possible that some branches will enter the admissible set due to the flow change at iteration K. Let M_D be the set of branches which enter the admissible set at this iteration. It is apparent that M_D must consist only of mirror branches on the augmenting path, because flow has been increased only on these branches.

It is convenient to define the network $G'_{A2} = (\overline{X},\Gamma'_{A2})$. The set Γ'_{A2} consists of Γ_{A2} plus the mirror branches of Γ_2. There exists in G'_{A2} a unique path between every pair of vertices of $\overline{X}$. The quantity $\gamma(i)$ is defined for $v_i \epsilon \overline{X}$ as the product of the branch gains on the path from v_i to v_t in G'_{A2}. $\gamma(i)$ can be interpreted as the gain in flow which occurs along this path. Alternatively, $1/\gamma(i)$ is the amount of flow required at v_i to obtain one unit of flow at v_t. The quantity $\gamma(i)$ is called the vertex gain of v_i.

There are now two quantities associated with the vertices of the graph, the vertex potentials $\delta(i)$ determined for each vertex by the structure of G_A^K and the vertex gains $\gamma(i)$ assigned only for the vertices in $\overline{X}$. These two quantities are used extensively in the development to follow and are the key to the efficiency of the algorithm. Two important relationships can be written in terms of vertex potentials and gains. The proofs of these relations are straight forward.

Let v_a and v_b be two vertices in $\bar{X}$. The gain of the unique path in G'_{A2} from v_a to v_b is $\gamma(a)/\gamma(b)$. The gain of the path is the product of the branch gains on the path. Thus one unit of flow at v_b through this path requires $\gamma(b)/\gamma(a)$ units at v_a.

The cost to obtain one unit of flow at v_b through the path from v_a is:

$$\delta(b)-\delta(a) \cdot \gamma(b)/\gamma(a)$$

This quantity might be termed the cost of the path. These relationships are important because characteristics of a path can be calculated using only values associated with the ends of the path.

The new augmenting network will be constructed by one of two approaches. It will later be shown that one of these approaches yields an optimal G_A^{K+1}. For the following discussion, define the following subsets of admissible branches:

$$D_1 = \{b(i,j) \mid b(i,j) \epsilon \Gamma_D, \ v_i \epsilon X, \ v_j \epsilon X\}$$

$$D_2 = \{b(i,j) \mid b(i,j) \epsilon \Gamma_D, \ v_i \epsilon X, \ v_j \epsilon \bar{X}\}$$

$$D_3 = \{b(i,j) \mid b(i,j) \epsilon \Gamma_D, \ v_i \epsilon \bar{X}, \ v_j \epsilon X\}$$

$$D_4 = \{b(i,j) \mid b(i,j) \epsilon \Gamma_D, \ v_i \epsilon \bar{X}, \ v_j \epsilon \bar{X}\}$$

The first approach constructs G_A^{K+1} by adding some branch $b_k(i,j)$ which is a member of D_2 to the graph G_{A1} and the subgraph of G'_{A2} which forms a directed tree rooted at v_j. Thus the sink and the other vertices of $\overline{X}$ are connected to the source or a flow generating circuit of G_{A1} through the branch b_k. This approach is shown in Figure 3a.

The second approach constructs G_A^{K+1} by adding some branch $b_k(i,j)$ which is a member of D_4. Together with branches of G'_{A2} this branch forms a circuit. If the gain of the circuit is greater than one, it is a flow generating circuit. This circuit is source of flow for the vertices of $\overline{X}$. G_A^{K+1} is formed by G_{A1}, the circuit formed by b_k, plus selected branches of G'_{A2} which form a directed tree rooted at the circuit. This approach is shown in Figure 3b.

Both constructions form an augmenting network when all the branches of G'_{A2} are admissible. We assume for the present that this is true and will discuss later the implications when it is not. Note that adding branches from the sets D_1 or D_3 do not yield an augmenting network.

Let $\delta(i)'$ be the vertex potential for v_i in G_A^{K+1}. For the optimum augmenting network $\delta(t)'$ will be at a minimum. Let $\Delta = \delta(t)'-\delta(t)$, or the increase in the sink potential obtained

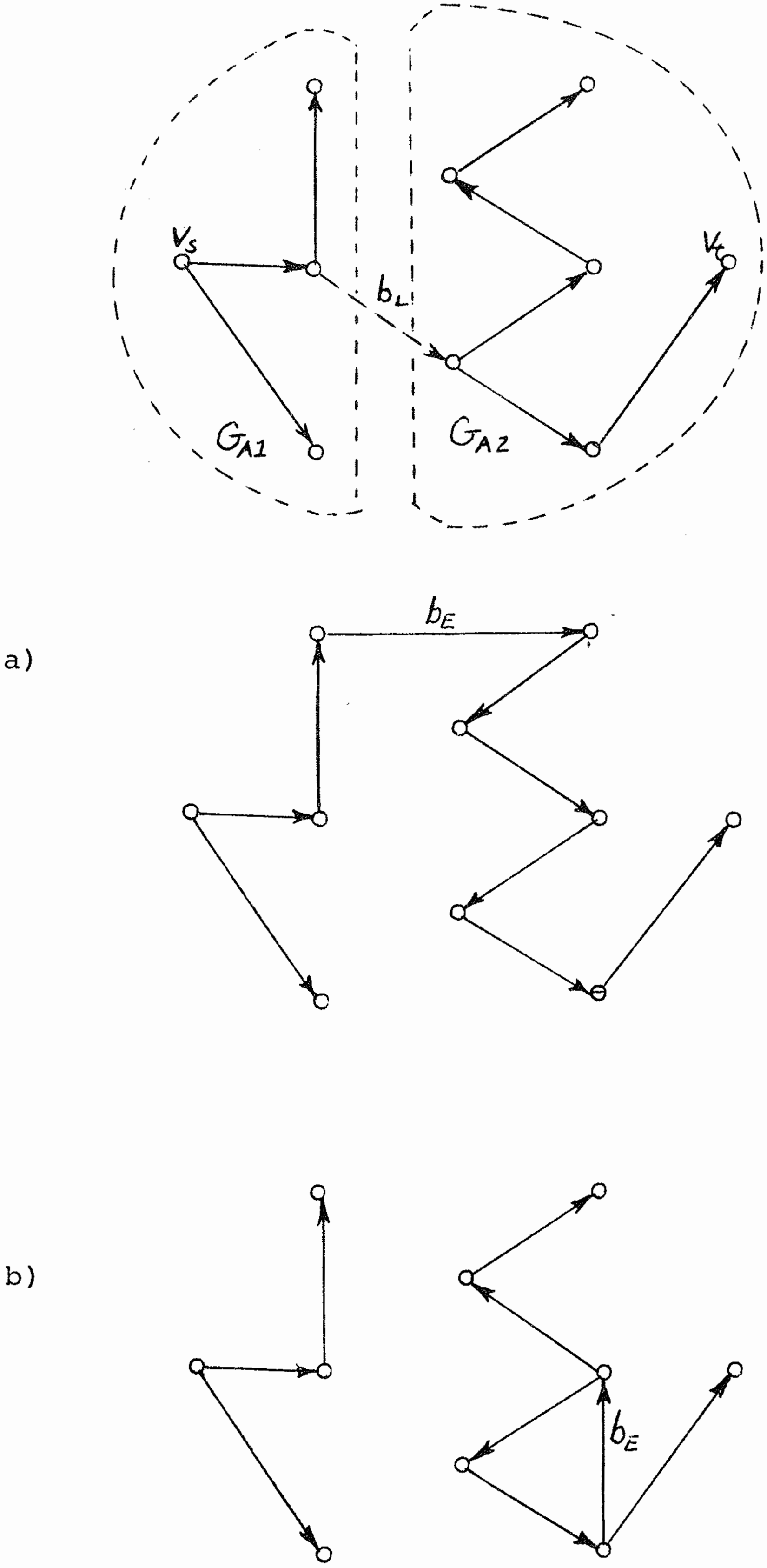


Figure 3 Two Constructions for the augmenting network

for optimum flow augmenting network. Consider the first construction approach.

Theorem 9

$$\Delta \leq \underset{b_k(i,j)\in D_2}{\text{Minimum}} \left[(\delta(i) + h_k)/a_k - \delta(j)\right]/\gamma(j) \qquad (12)$$

Proof: The addition of $b_k(i,j)$ to form an augmenting network, would result in an augmenting path to the sink consisting of three parts: A, the path from v_j to v_t which consists of branches from G'_{A2}; B, the branch b_k; and C, the path from the source (or from a flow generating circuit) to v_i which lies entirely in G_{A1}.

Consider the flows in these three parts required to obtain one unit of flow at v_t. One unit of flow at v_t requires $1/\gamma(j)$ units at v_j and $1/\gamma(j) \cdot a_k$ at v_i. The cost of these flows are computed as follows:

For part A the cost is:

$$\delta(t) - \delta(j)/\gamma(j)$$

For part B the flow through the branch results in a cost of:

$$h_k/\gamma(j) \cdot a_k.$$

For part C, the cost to obtain one unit of flow at v_i is $\delta(i)$. To obtain the required flow the cost is:

$$\delta(i)/\gamma(j) \cdot a_k.$$

Adding the three costs yields:

$$\delta(t)' = \delta(t) + \left[(\delta(i) + h_k)/a_k - \delta(j)\right]/\gamma(j);$$

Thus for any $b_k \varepsilon D_2$

$$\Delta \leq \left[(\delta(i) + h_k)/a_k - \delta(j)\right]/\gamma(j)$$

and theorem 9 is proved.

Now consider the second construction approach.

Theorem 10

The addition of $b_k(i,j) \varepsilon D_4$ to G'_{A2} forms a directed circuit with gain:

$$\beta = \gamma(j) \cdot a_k/\gamma(i) \qquad (13)$$

Proof:

$b_k \varepsilon D_4$ implies that $v_i \varepsilon \overline{X}$ and $v_j \varepsilon \overline{X}$. Thus b_k does form a directed circuit with branches of G'_{A2}. The gain of the path from v_j to v_i in G'_{A2} is $\gamma(j)/\gamma(i)$. Thus the circuit formed by adding b_k has gain $\gamma(j) \cdot a_k/\gamma(i)$.

Theorem 11

$$\Delta \leq \underset{S}{\mathrm{Min}}\left[(\delta(i) + h_k)/a_k - \delta(j)\right]/\left[\gamma(j) \cdot (1-1/\beta)\right] \qquad (14)$$

where

$$S = \{b_k \mid b_k(i,j) \varepsilon D_4,\ a_1 \cdot \gamma(j)/\gamma(i) > 1\}.$$

Proof: When the addition of the branch $b_k(i,j)$ forms a circuit with gain greater than one, the augmenting path to v_t consists of four parts: A, a simple path from some vertex v_ℓ on the circuit to the sink; B, a path from v_j to v_ℓ which forms part of circuit; C, the branch $b_k(i,j)$; and D, the path from v_ℓ to v_i

which completes the circuit. Note that except for b_k, the path lies entirely in G'_{A2}.

In order to calculate the potential at v_t for the augmenting path produced by b_k, consider first the flows required in the four parts to obtain one unit of flow at v_t. Of course, the flow at v_t is to be 1. The flow out of v_ℓ must then be $1/\gamma(\ell)$. The flow into v_ℓ from the circuit must be, according to previous arguments, $1/[(1-1/\beta)\cdot\gamma(\ell)]$.

The flow out of node v_j is to be:

$$\frac{1}{(1-1/\beta)\gamma(\ell)} \cdot \frac{\gamma(\ell)}{\gamma(j)} = \frac{1}{(1-1/\beta)\cdot\gamma(j)}$$

The flow out of node v_i (through b_k) is then:

$$1/[(1-1/\beta) \cdot \gamma(j) \cdot a_k]$$

Now to calculate the cost of this flow in four parts. The cost in part A to obtain one unit of flow at v_t from v_ℓ is:

$$\delta(t) - \delta(\ell)/\gamma(\ell)$$

The cost to obtain one unit of flow at v_ℓ from v_j is:

$$\delta(\ell) - \delta(j) \cdot \gamma(\ell)/\gamma(j)$$

Thus to obtain the required flow the cost in part B is:

$$\begin{aligned} &[\delta(\ell)-\delta(j)\cdot\gamma(\ell)/\gamma(j)][1/(1-1/\beta)\cdot\gamma(\ell)] \\ &= [1/(1-1/\beta)][\delta(\ell)/\gamma(\ell) - \delta(j)/\gamma(j)] \end{aligned}$$

The cost of the required flow in branch b_k (part C) is:

$$[1/(1-1/\beta)][h_k/\gamma(j)\cdot a_k]$$

The cost to obtain one unit of flow at v_i from v_ℓ is:

$$\delta(i) - \delta(\ell) \circ \gamma(i)/\gamma(\ell)$$

Thus the cost in part A is:

$$\left[\delta(i) - \delta(\ell)\cdot\gamma(i)/\gamma(\ell)\right]\left[\, 1/(1-1/\beta)\cdot\gamma(j)\cdot a_k\right]$$

Adding the costs of flow in the four parts and reordering terms, the modified potential at v_t is $\delta(t)'$:

$$\begin{aligned}\delta(t)' = {} & \delta(t) - \delta(\ell)/\gamma(\ell) \\ & + \delta(\ell)\left[1/(1-1/\beta)\right]\left[\, 1/\gamma(\ell) - \gamma(i)/\gamma(\ell)\cdot\gamma(j)\cdot a_k\right] \\ & + \left[1/(1-1/\beta)\cdot\gamma(j)\right]\left[-\delta(j) + h_k/a_k + \delta(i)/a_k\right]\end{aligned}$$

Recognizing that $\gamma(i)/\gamma(j)\cdot a_k = 1/\beta$, the terms involving $\delta(\ell)$ drop out and the change in potential at v_t caused by adding b_k is:

$$\delta(t)' - \delta(t) = \left[(\delta(i) + h_k)/a_k\right]/\left[(1-1/\beta)\cdot\gamma(j)\right] \ .$$

This difference must be at least as great as minimum difference Δ. Thus theorem 11 is proved.

The branch which will enter the augmenting network to form G_A^{K+1} will be the member of D_2 or D_4 which causes the smallest increase in $\delta(t)$ as measured by the equations 12 and 14. Let this be branch b_E. The potentials for the vertices in G_A^{K+1} are determined by the following. First:

$$\delta(t)' = \delta(t) + \Delta.$$

Each $v_i \epsilon \overline{X}$ will be connected to v_t through a path chosen from G'_{A2}. Thus the following relationship is true:

$$\delta(t) - \delta(i)/\gamma(i) = \delta(t)' - \delta(i)'/\gamma(i)$$

Rearranging

$$\delta(i)' - \delta(i) = \gamma(i)\left[\delta(t)' - \delta(t)\right]$$

or $$\delta(i)' = \delta(i) + \gamma(i)\Delta \text{ for } v_i \epsilon \overline{X} \qquad (15)$$

For $v_i \epsilon X$:

$$\delta(i)' = \delta(i) \qquad (16)$$

Thus when the entering branch is determined, the vertex potentials for G_A^{K+1} are easily determined.

Theorem 12

The augmenting network formed by adding the branch from D_2 or D_4 which yields the minimum increase in $\delta(t)$ is optimal.

Proof: According to theorem 3 a necessary and sufficient condition for an optimum augmenting network is that the network defines a unique augmenting path for each vertex and that

$$\delta\ (j)' \leq (\delta(i) + h_k)/a_k$$

$$\text{for all } b_k(i,j) \epsilon \Gamma_D$$

where $\delta\ (i)'$ is the vertex potential determined for G_A^{K+1}. The theorem will be proved for all branches in the sets D_1, D_2, D_3 and D_4. First for $b_k(i,j) \epsilon D_1$ the condition above must be true because $v_i \epsilon X$ and $v_j \epsilon X$, $\delta(\ell)' = \delta(\ell)$ for $v_\ell \epsilon X$, and G_A^K is optimal.

For $b_k(i,j) \epsilon D_2$ the condition is satisfied because $v_i \epsilon X$ and $v_j \epsilon \overline{X}$, $\delta(i)' = \delta(i)$ and $\delta(j)' = \delta(j) + \gamma(j)\Delta$. From the condition for choosing b_E

$$\left[(\delta(i) + h_k)/a_k - \delta(j)\right]/\gamma(j) \geq \Delta;$$

or

$$(\delta(i) + h_k)/a_k \geq \delta(j) + \gamma(j)\Delta$$

or

$$(\delta(i)' + h_k)/a_k \geq \delta(j)'$$

For $b_k(i,j) \epsilon D_3$, $v_i \epsilon \overline{X}$ and $v_j \epsilon X$, $\delta(i)' = \delta(i) + \Delta\gamma(i)$ and $\delta(j)' = \delta(j)$.

Since $$(\delta(i) + h_k)/a_k \geq \delta(j)$$

certainly $$(\delta(i) + \Delta\gamma(i) + h_k)/a_k \geq \delta(j)$$

or $$(\delta(i)' + h_k)/a_k \geq \delta(j)'.$$

For $b_k(i,j) \epsilon D_4$, $v_i \epsilon \overline{X}$, $v_j \epsilon \overline{X}$,

$$\delta(i)' = \delta(i) + \Delta\gamma(i), \; \delta(j)' = \delta(j) + \Delta\gamma(j).$$

From theorem 11,

$$\Delta \leq \frac{(\delta(i) + h_k)/a_k - \delta(j)}{\gamma(j)(1-1/\beta)}$$

where $$\beta = \gamma(j) \cdot a_k/\gamma(i)$$

Substituting for β and manipulating yields:

$$\Delta\gamma(j)(1-\gamma(i)/\gamma(j) \cdot a_k) \leq (\delta(i) + h_k)/a_k - \delta(j)$$

or $$\delta(j) + \Delta\gamma(j) \leq (\delta(i) + \Delta\gamma(i) + h_k)/a_k$$

or $$\delta(j)' \leq (\delta(i)' + h_k)/a_k$$

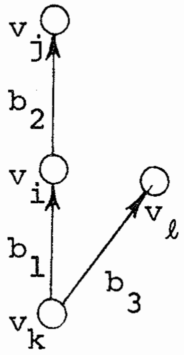
This completes the proof.

Theorems 9, 10, and 11 provide a simple basis for choosing the variable to enter the augmenting network. One must search through each branch in D_2 and calculate the potential change noted in Theorem 9, then for each branch in D_4, one calculates the circuit gain of Theorem 10 and the potential change noted in Theorem 11. The branch to enter is that one which gives the smallest change in $\delta(t)$. Note that all the calculations for each branch require quantities only associated with two terminals of the branch.

The two approaches considered for constructing the new augmenting network both require the selection of branches of G'_{A2} to form a directed tree (or perhaps a directed circuit) in G_A^{K+1}. One difficulty that may arise is that not all the branches of G'_{A2} need be admissible. This is the case when a branch of G_{A2} has zero flow in the original network. In this case the mirror branch is not admissible. Another way this can occur is when more than one branch becomes saturated at a particular iteration. In this case the practice is to choose the branch nearest the source (or flow generating circuit) as the leaving branch. Thus a branch of G_{A2} is not admissible. No provision is made to prevent these occurrances in the algorithm, and indeed inadmissible branches do on occasion enter the augmenting network. This causes the amount of flow to the sink to be augmented by zero for some iterations. This corresponds to degeneracy in a linear programming algorithm. The inefficiencies caused by degeneracy are more than compensated by the efficiencies introduced in the process of generating augmenting networks.

VII. Triple Label Representation of the Augmenting Network

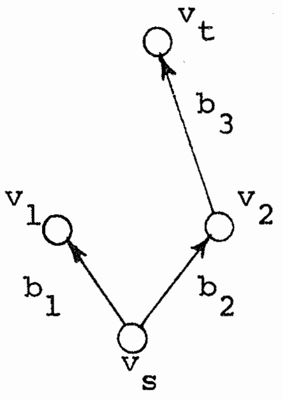
The flow augmenting network $G_A = V,\Gamma_A$ can be described completely using a triple labeling scheme suggested in (10). For this representation each vertex is provided three labels, each indicating a branch of the graph (a member of Γ_A). These labels for v_i are called the back pointer $P_B(i)$, the forward pointer $P_F(i)$ and the right pointer $P_R(i)$. The back pointer of v_i is that branch of the G_A which terminates at v_i. The forward pointer of v_i is a branch of G_A which originates at v_i. The right pointer of v_i is a branch of G_A which originates at v_k, where v_k is the initial vertex of the branch indicated by the back pointer of v_i. Figure 4 illustrates the use of the pointers.



$$P_B(i) = b_1$$
$$P_F(i) = b_2$$
$$P_R(i) = b_3$$

Figure 4: An example of the triple label representation

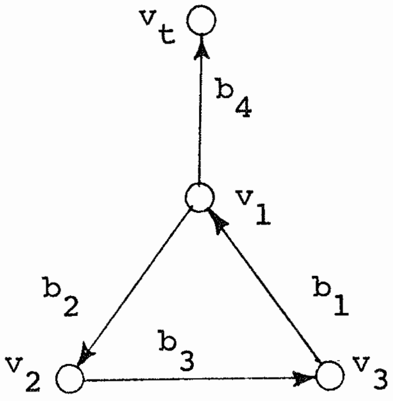
It is characteristic of G_A that each vertex is the terminal vertex of at most one branch. However, each vertex may initiate more than one branch. Thus, for a given G_A, the assignment of back pointers is unique, but the assignment of forward and right pointers may not be unique. Figure 5 shows a directed tree and its representation with vertex pointers. An equivalent representation would replace $P_F(s) = b_2$, $P_R(2) = b_1$, $P_R(1) = 0$ with all other pointers remaining unchanged.



$P_B(s) = 0$	$P_B(2) = b_2$
$P_F(s) = b_1$	$P_F(2) = b_3$
$P_R(s) = 0$	$P_R(2) = 0$
$P_B(1) = b_1$	$P_B(t) = b_3$
$P_F(1) = 0$	$P_F(t) = 0$
$P_R(1) = b_2$	$P_R(t) = 0$

Figure 5: The representation of a tree

Figure 6 shows a component which includes a circuit and its pointer representation. Again this representation is not unique because of the two branches leaving v_1.



$P_B(1) = b_1$	$P_B(3) = b_3$
$P_F(1) = b_2$	$P_F(3) = b_1$
$P_R(1) = 0$	$P_R(3) = 0$
$P_B(2) = b_2$	$P_B(t) = b_4$
$P_F(2) = b_3$	$P_F(t) = 0$
$P_R(2) = b_4$	$P_R(t) = 0$

Figure 6: The representation of a circuit

It may be noted that every vertex in the graph will have a back pointer, except perhaps the source. The source will have a nonzero back pointer only if it is in a directed circuit of the graph. A vertex may or may not have nonzero forward and right pointers. The triple labeling representation is used extensively in the algorithm to calculate vertex potentials and gains and for constructing a new flow augmenting network.

VIII. The Algorithm

The theorems described in the subsequent sections lead to an efficient algorithm to solve the network with gains problem. The algorithm is presented here with references to the theorems which justify the various steps.

1. Set all branch flows in G equal to zero.
2. Use the algorithm described in section III to find the initial augmenting network G_A^O. Let K = 0.
3. Augment the flow into v_t by adjusting flows on the branches of G defined by the augmenting path for v_t in G_A^K. The maximum flow change which causes one of the branches on the path to become inadmissible is defined by Theorems 5 and 6. If the maximum flow change would cause the sink flow to exceed the required flow, increase the flow only to the required amount and terminate the algorithm. Otherwise the branch which becomes inadmissible is b_L. If more than one branch becomes inadmissible, choose the one which appears closest to the source of the augmenting path.
4. Delete b_L from the augmenting network. Use theorems 9, 10, and 11 to determine b_E, the branch which enters the augmenting network and causes the minimum increase in

the sink potential. If there are no admissible branches which can enter, the maximum flow has been found. Stop. Otherwise, go to step 5.

5. Introduce b_E into the augmenting network and form G_A^{K+1} using G_{A1}, b_E, and a suitable selection of the branches in G'_{A2} to obtain an augmenting network.

6. Increase K by one and go to step 3.

For an example of the algorithm consider the network of Figure 7a. This figure shows the branch parameters and vertex indices associated with the network. These are eliminated from subsequent copies for clarity. The required flow at the sink is 10 for this example. Initially the flows are taken to be zero. Figure 7b. shows G_A^0 determined for the initial flows using the algorithm of section III. The numbers in parentheses indicate vertex potentials. The arrows indicate the augmenting path to the sink and the maximum flow that can pass through it. This flow causes branch (3,6) to become saturated; hence, it is removed from the graph. The flows found in 7b. are impressed on the network to obtain the flows in 7c. Figure 7d. shows the G_A^1 found using the results of section VI. Note the entering branch is (2,7). This graph yields a degenerate iteration because the flow augmentation is zero. This occurs because branches (7,5), (5,8) and (8,6) are inadmissible. They entered the augmenting network as a result of the selection of branches in G'_{A2} to obtain a directed tree rooted at vertex 7 when (2,7) entered. The next iteration causes (7,5) to leave and the flow increase is non-zero. Subsequent augmenting networks are shown in Figures 7 f., h., j., l., and n. and the corresponding flows in the original network are shown in Fugures 7 e., g., i., k., and m. Figure 7o. shows the optimum flows. Note in these figures the potential of the sink steadily increases as the algorithm progresses. An interesting augmenting network appears in 7l. This was formed from 7j. by the removal of (1,4) and the addition of (6,3). The resultant network includes a flow generating circuit.

Figure 7: Example Problem

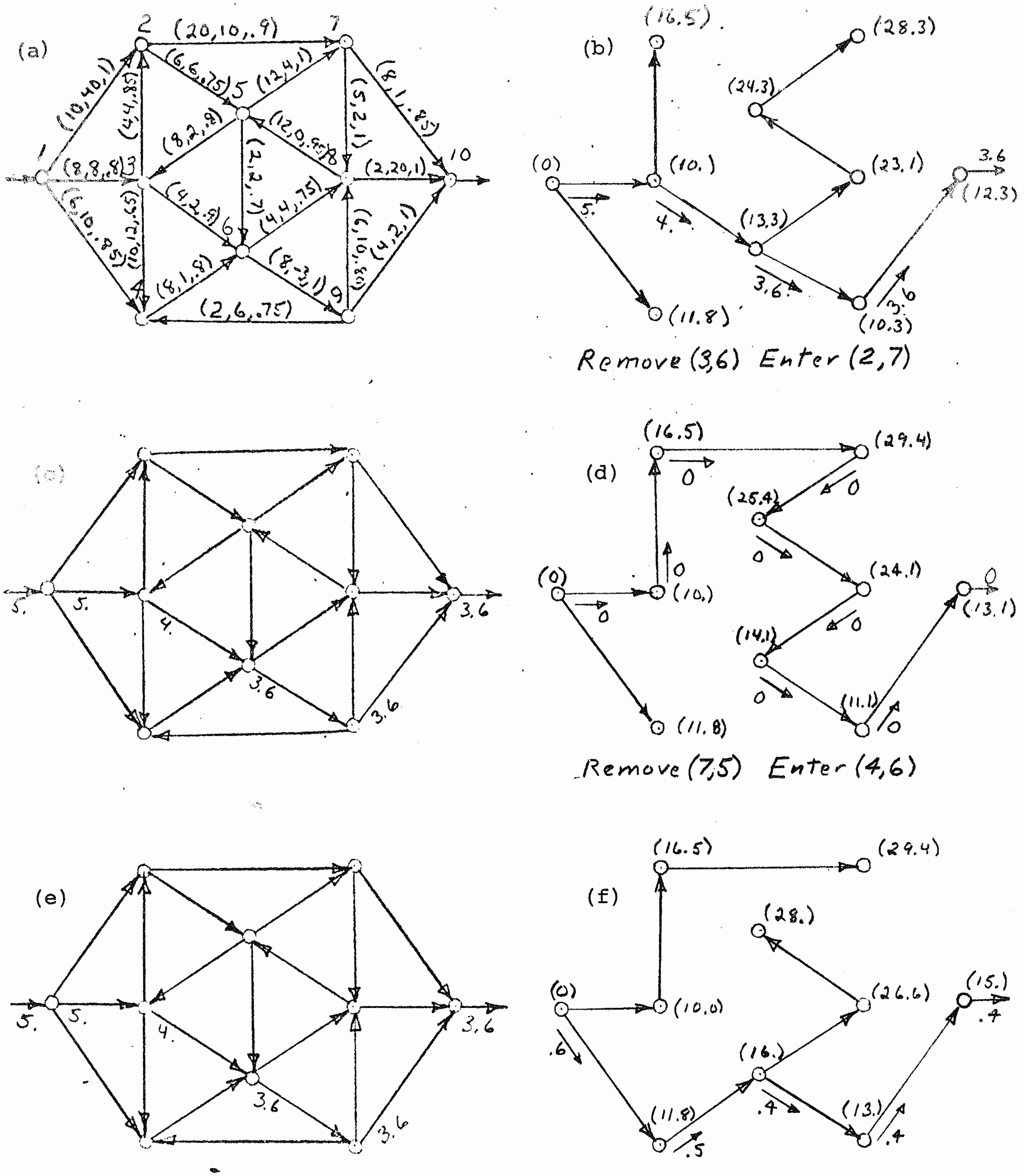


Figure 7 (continued)

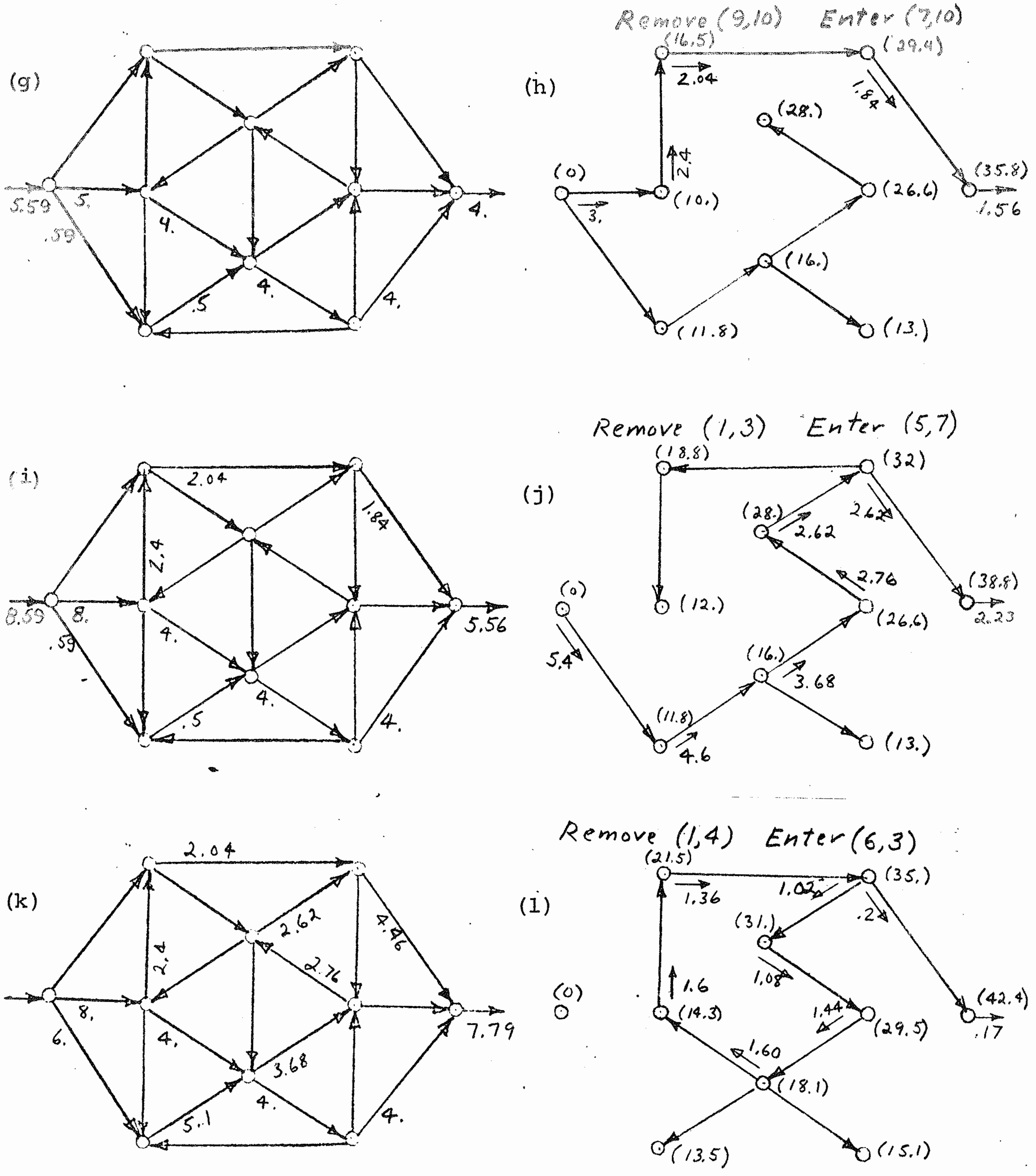
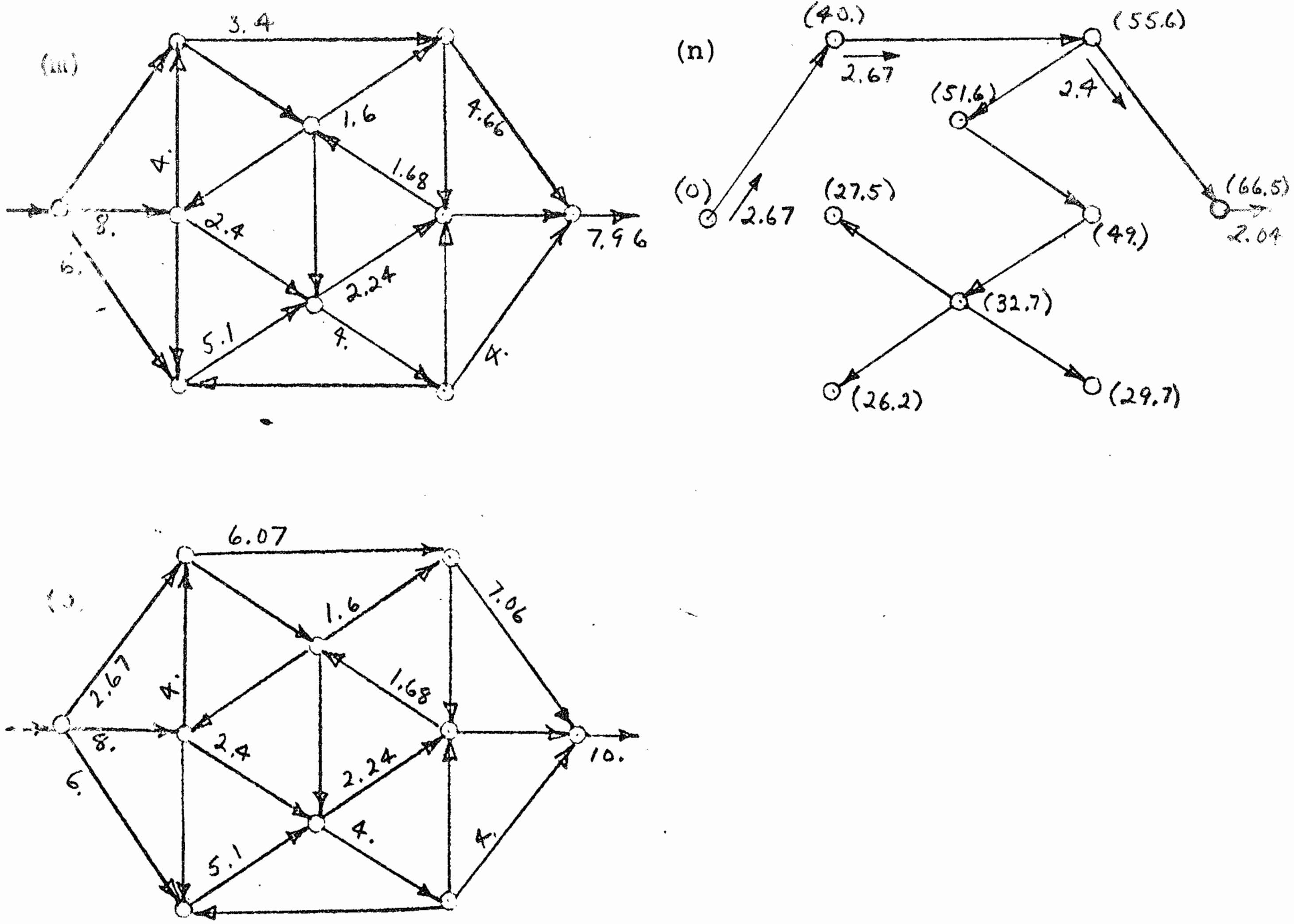


Figure 7 (continued)



IX. Computational Efficiency

To determine the computational efficiency of this approach, four large network problems were solved with: an efficient linear program which operated entirely within the core of the computer (SP6600), an implementation of the algorithm of Bhaumik (2) for the networks with gains problem (LEAKY), and an implementation of the algorithm of this paper (GAIN). The problems solved were large transshipment problems generated by Klingman (11), to test pure network algorithms. The problems were modified by assigning a randomly selected gain factor to each branch from the range .5 to 1.5. The problems were run for various values of output flow up to the maximum flow at the sink. They were run on a CDC 6600 computer using the RUN compiler. The results shown in Table 1 indicate the current algorithm is approximately twenty times faster than the linear program and approximately ten times faster than the Bhaumik algorithm.

The problems shown in Table 1 are in fact pure network problems drawn from the reference (11), modified to incorporate branch gains. Problems 1,2,3, and 4 are respectively problems 16,17,18 and 24 from the reference. The total output flow for each pure problem was 400,000 as compared to the maximum flows

Table 1

Computational Times for Solving Network with Gains Problems
Using Linear Programming and Two
Network with Gains Algorithms

Problem 1

400 vertices, 1306 branches, 8 sources, 60 sinks

	Output Flow		
	114,609	362,198	384,868*
LP6600	181(623)	178(586)	----
LEAKY	20.5(14)	150.2(71)	----
GAIN	1.75(14)	8.52(103)	9.15(113)

Problem 2

400 vertices, 2443 branches, 8 sources, 60 sinks

	Output Flow		
	73,894	394,806	438,559*
LEAKY	21.5(10)	183(64)	----
GAIN	2.1(10)	12.4(107)	15.4(135)

Problem 3

400 vertices, 1306 branches, 8 sources, 60 sinks

	Output Flow		
	152,041	398,380	404,865*
LEAKY	20.5(11)	181.2(67)	----
GAIN	2.09(14)	7.48(91)	7.93(98)

Problem 4

400 vertices, 1382 branches, 4 sources, 12 sinks

	Output Flow	
	191,942	364,614*
LP6600	217.2(732)	178.4(623)
LEAKY	20.3(9)	86.3(34)
GAIN	4.17(39)	7.85(91)

* Maximum flow in network. Entries in table are time in seconds and the number of iterations are shown in parenthesis.

for the problems with gains that appear in the last column of Table 1. Computation times for the pure problems solved on the CDC 6600 with the RUN compiler as obtained from the reference are reproduced in Table 2. The codes shown are PNET, a special purpose simplex network code written by Glover, Harney and Kingman, SUPERK(1), an efficient out-of-kilter algorithm and SHARE (5,13) a generally available out-of-kilter algorithm. Although the computation time for these codes cannot strictly be compared to that of GAIN since the network with gains problem is more complex than the pure network problem, it is apparent from Table 2 that the time for GAIN is of the same order of magnitude as for pure network codes.

Table 2

Comparison of Computational Times (seconds) For
Pure Networks and for Networks with Gains

Problem	Pure Network Solution			Solution with Gains
	PNET	SUPERK	SHARE	GAIN
1	2.02	5.22	21.51	9.15
2	3.23	8.47	32.40	15.4
3	2.38	4.77	20.06	7.93
4	2.68	5.51	23.46	7.85

The computerized algorithm in its present form requires approximately 8m+8n words of core memory where m is the number of vertices and n is the number of branches in the original network.

X. Problem Set Up

The algorithm described here finds the minimum cost flow for a specified required output flow. This may appear to be a restricted class of problems; however, there are several problem set-up techniques that can be used to put other kinds of problems into this form.

Consider the problem of providing some specified quantity of flow at each of a number of sinks from limited amounts of flow at each of a number of sources. This problem is modeled by creating a super sink with a branch from each sink to the super sink with capacity equal to the desired flow. Likewise a super source is created with branches from the super source to each of the sources with capacity equal to the amount of flow available at each source. The super source and super sink are used as the source and sink of the algorithm with the required output flow set equal to the sum of the required output flows at the original sinks.

For some problems it may be necessary to impose lower bounds on the flow in some branches. Thus a particular branch $b_k(i,j)$ might have an additional parameter ℓ_k which is the lower bound on flow. Such a situation would be represented for this algorithm by two parallel branches from i to j. The parameters on one branch will be $(\ell_k, -M, a_k)$ where ℓ_k is upper bound on flow for this branch, $-M$ is large negative cost for the branch and a_k is original branch gain. The second branch has the parameters $(c_k - \ell_k, h_k, a_k)$. Because of the large negative cost, the algorithm will saturate the first branch if possible. This construction is possible only if the negative costs used do not result in a circuit with a negative cost.

Some problems include both positive and negative branch costs (negative costs model revenues). No specific output flow is specified but the goal of the problem is to find the output flow which minimizes total cost. This is equivalent to the flow which maximizes profit. For this situation an additional branch is constructed from the source to the sink with parameters (M, 0, 1) where M is a large number. Thus this branch transmits any amount at zero cost with gain one. The required output flow is set to some large number greater than the optimum flow for the original problem. With this construction the algorithm will increase flow to the sink only if the sink potential is negative. When the flow reaches the point where the minimum cost flow augmenting path in the original network has a positive cost, the added branch becomes the augmenting path. Thus the flow in the original network is the profit maximizing flow.

XI. Conclusion

This paper has described and provided the theoretical foundation for an algorithm for the network with gains problem. The algorithm is similar in spirit to that of Busacker and Gowen for pure networks; however, the incorporation of gains makes for a considerably more complex situation. The algorithm has been coded for computer implementation and has been found to be very efficient. Large problems have been solved with the algorithm in approximately one twentieth of the time required for an efficient linear programming algorithm.

References

1. Barr, R. S., Glover, F., and Klingman, D., "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," C. S. Report 102, Center for Cybernetic Studies, University of Texas, BEB-512, Austin, 1972.

2. Bhaumik, Gora, Optimum Operating Policies of a Water Distribution System with Losses, unpublished Ph.D. Dissertation, The University of Texas at Austin, 1973.

3. Busacker, R. G. and Gowen, P. J., "A Procedure for Determining a Family of Minimal Cost Network Flow Patterns," ORO Technical Paper 15, Operation Research Office, The John Hopkins University, 1961.

4. Charnes, A. and Raike, W. M., "One-Pass Algorithm for Some Generalized Network Problems," Operations Research, Vol. 14, pp. 914-924, 1966.

5. Clasen, R. J., "The Numerical Solution of Network Problems Using the Out-of-Kilter Algorithm," RAND Corporation Memo RM-5456-PR, Santa Monica, California, March, 1968.

6. Fujisawa, T., "Maximal Flow in a Lossy Network," Proceedings of Allerton Conference on Circuit and Systems Theory, Vol. 1, pp. 385-393, 1963.

7. Glover, Fred and Klingman, D., "On the Equivalence of Some Generalized Network Problems to Pure Network Problems," Research Report CS81, Center for Cybernetic Studies, The University of Texas, Austin, January, 1972.

8. Glover, Fred, Klingman, D. and Napier, A., "Basic Dual Feasible Solutions for a Class of Generalized Networks," Operations Research, Vol. 20, No. 1, Jan.-Feb., 1972.

9. Jewell, William S., "Optimal Flow Through Networks With Gains," Operations Research, Vol. 10, No. 4, July-August, 1962.

10. Johnson, Ellis L., "Networks and Basic Solutions," Operations Research, Vol. 14, No. 4, July-August, 1966.

11. Kingman, D., Napier, A and Stutz, J., "NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," Management Science, Vol. 20, No. 5, pp. 814-821, January, 1974.

12. Maurras, Jean Francois, "Optimization of the Flow through Networks with Gains," Mathematical Programming, 3, 1972.

13. "Out-of-Kilter Routine," SHARE Distribution 3536, SHARE Distribution Agency, Hawthorne, New York, 1967.

Appendix

The Network with Gains Computer Program

This program solves the generalized network minimum cost flow problem. The user must specify the arcs of the network to be solved and four parameters for each arc. The parameters are cost per unit flow, lower bound on flow, upper bound on flow, and the gain for flow passing through the arc. The user must also specify an input node which is assumed to be an infinite source of flow and the output node which is to be the sink for flow. The user will specify a desired output flow. A requirement on the network for this program is that zero flows on all arcs must be otpimum for zero output flow. The program will determine the flows in the arcs of the network which will provide the desried output flow at minimum cost. These flows are printed by the program.

Input Data:

Card 1 TITLE

FORMAT (80A1)

The title is an arbitrary alpha-numeric identifier of the run.

Card 2 SOURCE, SINK, NODES, NPRINT, IPRINT

Format 5I10

SOURCE is the node number of the infinite source of flow.

SINK is the node number of the sink of flow.

NODES is the number of nodes in the network.

NPRINT determines the number of arcs for which input and output data will be printed. The input data for the first NPRINT arcs are printed and output results are printed for the first and last NPRINT arcs. This option is useful for large problems.

IPRINT controls the intermediate printout. IPRINT = 0 results in only input and output printout. IPRINT = 1 shows in addition the output flow, cost, entering arc, leaving arc and change in output node potential at each iteration. IPRINT = 2 shows in addition the triple label representation of the tree, all node potentials and all node gains at each iteration.

Card 3 OUTFLO

Format F10.2

OUTFLO is the desired amount of flow that is to be delivered to the SINK at minimum cost.

Cards 4 and following IARC(I), JARC(I), LOWER(I), UPPER(I), COST(I), AMP(I)

Format (2I10, 4F10.2)

Read one card for each arc of the network. Place a blank card after the set of arc cards.

IARC(I) is the node which originates arc I.

JARC(I) is the node which terminates arc I.

LOWER(I) is the lower bound on the flow entering arc I.

UPPER(I) is the upper bound on the flow entering arc I. (Note: because of the gain factor a different amount of flow leaves an arc than enters it. This should be a positive quantity.)

COST(I) is the cost per unit of flow entering arc I. This cost may be positive or negative.

AMP(I) is the gain factor for arc I. The flow leaving arc I is equal to the flow entering the arc multiplied by the quantity AMP(I). This should be a positive quantity greater than zero.

Program Listing of Network with Gains Algorithm

```
      PROGRAM GAIN(INPUT,OUTPUT)
C     PROGRAM TO SOLVE THE NETWORK WITH GAINS PROBLEM FOR A GIVEN QUANTI
C     TY OF OUTPUT FLOW.
C     PROGRAM BY P. JENSEN AND GORA BHAUMIK, UNIVERSITY OF TEXAS, 1973.
C     READ DATA AS FOLLOWS---
C      CARD 1   TITLE  FORMAT(80A1)
C      CARD 2  SOURCE NODE,SINK NODE,NUMBER OF NODES,NUMBER OF ARCS TO
C     BE LISTED IN PRINTOUT, PRINT OPTION (0 FOR SHORT PRINT, 1 FOR LONG
C     PRINT, 2 FOR EXTRA LONG PRINT).
C     FORMAT(5I10)
C      CARD 3  DESIRED OUTPUT FLOW.
C         FORMAT(F10.2)
C      CARD 4 AND FOLLOWING FOR EACH ARC IN THE NETWORK.
C     ORIGIN NODE OF ARC, TERMINAL NODE OF ARC, LOWER BOUND ON FLOW,
C       UPPER BOUND ON FLOW, COST PER UNIT FLOW, GAIN FACTOR FOR ARC.
C        FORMAT (2I10,4F10.2)
C      THE LIST OF ARCS IS TERMINATED WITH A BLANK CARD.
      COMMON /1/ IARC(5000)/2/JARC(5000)/3/COST(2500)/4/AMP(2500)/5/
     1      FLOW(2500)/6/UPPER(2500)/7/LOWER(2500)
      COMMON /8/ V(500)/9/BARC(500)/10/RARC(500)/11/FARC(500)/12/
     1      DISSET(500)/13/GAN(500)/14/ICHK(500)/15/LIST(500)
      COMMON /16/ TITLE(80)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
      NDEG=0
      NLOP=0
      ITER=0
      FLONET=0.0
      TOTCST=0.0
      IFS=0
      NPRIT=40
      EPS=1.E-6
      BIG=1.E6
      READ 70, (TITLE(I),I=1,80)
      NARC=10000
      PRINT 80, (TITLE(I),I=1,80)
      READ 90, SOURCE,SINK,NODES,NPRINT,IPRINT
      PRINT 100, SOURCE,SINK
      READ 110, OUTFLO
      PRINT 120, OUTFLO
      PRINT 130
      PRINT 140
      PRINT 150
      I=0
 10   CONTINUE
      I=I+1
      READ 160, IARC(I),JARC(I),LOWER(I),UPPER(I),COST(I),AMP(I)
         IF (IARC(I).EQ.0) GO TO 20
         IF ((I.GT.NPRIT).AND.(I.LT.(NARC-NPRIT))) GO TO 10
      PRINT 170, I,IARC(I),JARC(I),LOWER(I),UPPER(I),COST(I),AMP(I)
         GO TO 10
 20   CONTINUE
      NARC=I-1
C     CREATE A DUMMY ARC TO PROVIDE FEASIBLE OUTPUT FLOW IN CASE THE
C     DESIRED OUTPUT IS NOT FEASIBLE.
```

```
      NARC=NARC+1
      IARC(NARC)=SOURCE
      JARC(NARC)=SINK
      LOWER(NARC)=0
      UPPER(NARC)=BIG
      COST(NARC)=BIG/10
      AMP(NARC)=1.
C     INITIALIZE FLOWS AND CREATE MIRROR ARCS.
      DO 30 I=1,NARC
      NN=NARC+I
      FLOW(I)=0.0
      IARC(NN)=JARC(I)
      JARC(NN)=IARC(I)
 30   CONTINUE
      NARCS=NARC*2
 40   CONTINUE
      CALL SHORT (IENTER,ILEAV)
         IF (IENTER.EQ.0) GO TO 60
      CALL MAXFLO
      TOTCST=TOTCST+CSTNOW
      TOTCSP=TOTCST-COST(NARC)*FLOW(NARC)
      FLONEP=FLONET-FLOW(NARC)
      ITER=ITER+1
         IF (IPRINT.EQ.0) GO TO 50
      PRINT 180, ITER,FLONEP,TOTCSP
      PRINT 190, ILEAV,IENTER,SICH
 50   CONTINUE
         IF (ABS(FLONET-OUTFLO).LE.0.000001) GO TO 60
         GO TO 40
 60   CONTINUE
      CALL PROUT
 70   FORMAT (80A1)
 80   FORMAT (1H1,///80A1///)
 90   FORMAT (5I10)
 100  FORMAT (* SOURCE NODE=*,I5,5X,*SINK NODE =*,I5,///)
 110  FORMAT (F10.2)
 120  FORMAT (* OUTPUT FLOW REQUIREMENT*,F10.2,///)
 130  FORMAT (21H *****INPUT DATA*****,//)
 140  FORMAT (*         ARC     START       END      LOWER      UPPER
     1 COST     AMPLIFICATION*)
 150  FORMAT (*         NO.       NODE       NODE       BOUND      BOUND*,//)
 160  FORMAT (2I10,4F10.2)
 170  FORMAT (3I10,6F11.2)
 180  FORMAT (* ITERATION *,I5,5X,* FLOW *,F10.2,5X,* COST *,F20.2)
 190  FORMAT (* REMOVE*I5,5X*ENTER*I5,5X*DELTA*F20.5)
      END
      SUBROUTINE SHORT (IENTER,ILEAV)
C     SUBROUTINE TO FIND THE INITIAL AND SUBSEQUENT FLOW AUGMENTING TREE
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
         IF (IFS.NE.0) GO TO 150
C     SET UP POINTERS TO FIND INITIAL TREE.
      DO 10 I=1,NODES
      BARC(I)=0
      FARC(I)=0
      RARC(I)=0
      DISSET(I)=1
      GAN(I)=1
      V(I)=999999.
   10 CONTINUE
      V(SOURCE)=0
      ICHANGE=0
      IENTER=1
      ITF=0
      IFS=1
C     SET UP SHORTEST PATH TREE FOR FIRST ITERATION.
   20 CONTINUE
      DO 70 K=1,NARC
         IF ((UPPER(K)-FLOW(K)).GT.EPS) GO TO 30
         IF ((FLOW(K)-LOWER(K)).LT.EPS) GO TO 70
      II=JARC(K)
      JJ=IARC(K)
      POT=V(II)*AMP(K)-COST(K)
      I=K+NARC
         GO TO 50
   30 CONTINUE
      I=K
      JJ=JARC(I)
      II=IARC(I)
         IF ((LOWER(I)-FLOW(I)).GT.EPS) GO TO 40
      POT=(V(II)+COST(I))/AMP(I)
         GO TO 50
   40 CONTINUE
      POT=(V(II)-BIG)/AMP(I)
   50 CONTINUE
         IF ((V(JJ)-POT).LT.0.001) GO TO 70
      V(JJ)=POT
      BARC(JJ)=I
         IF (ITF.EQ.0) GO TO 60
      CALL LOOP (I,JJ)
   60 CONTINUE
      ICHANGE=1
   70 CONTINUE
         IF (ICHANGE.EQ.0) GO TO 80
      ICHANGE=0
      ITF=1
         GO TO 20
   80 CONTINUE
```

```
C     CALCULATE FORWARD POINTERS FOR FIRST ITERATION.
      DO 120 I=1,NODES
         IF (DISSET(I).EQ.0) GO TO 120
      KK=BARC(I)
         IF (KK.EQ.0) GO TO 120
      LL=IARC(KK)
         IF (FARC(LL).NE.0) GO TO 90
      FARC(LL)=KK
         GO TO 120
 90   CONTINUE
      MM=FARC(LL)
 100  CONTINUE
      MN=JARC(MM)
         IF (RARC(MN).NE.0) GO TO 110
      RARC(MN)=KK
         GO TO 120
 110  CONTINUE
      MM=RARC(MN)
         GO TO 100
 120  CONTINUE
 130  CONTINUE
         IF (IPRINT.LT.2) GO TO 140
      PRINT 320, (BARC(I),I=1,NODES)
      PRINT 330, (FARC(I),I=1,NODES)
      PRINT 340, (RARC(I),I=1,NODES)
      PRINT 300, (DISSET(I),I=1,NODES)
      PRINT 290, (V(I),I=1,NODES)
      PRINT 310, (GAN(I),I=1,NODES)
 140  CONTINUE
      RETURN
C     FIND NEW FLOW AUGMENTING TREE AFTER THE FIRST ITERATION.
 150  CONTINUE
      DO 160 I=1,NODES
      DISSET(I)=0
 160  CONTINUE
C     DELETE BRANCH FROM BASIS AFTER THE FIRST ITERATION.
      II=IROOT
      I=BARC(II)
      ILEAV=I
      IA=IARC(I)
      CALL DESUB (I,IA)
      RARC(II)=0
      BARC(II)=0
C     SET NEW NODE GAINS ON DISCONNECTED NOEDS.
 170  CONTINUE
      I=FARC(II)
         IF (I.EQ.0) GO TO 180
      JJ=JARC(I)
      IF (I.GT.NARC) GAN(JJ)=GAN(II)*AMP(I-NARC)
      IF (I.LE.NARC) GAN(JJ)=GAN(II)/AMP(I)
      II=JJ
         IF (II.EQ.IROOT) GO TO 180
         GO TO 170
 180  CONTINUE
      J=RARC(II)
      DISSET(II)=1
         IF (J.EQ.0) GO TO 190
      II=JARC(J)
      K=BARC(II)
      JJ=IARC(K)
      IF (K.GT.NARC) GAN(II)=GAN(JJ)*AMP(K-NARC)
      IF (K.LE.NARC) GAN(II)=GAN(JJ)/AMP(K)
         GO TO 170
 190  CONTINUE
      K=BARC(II)
         IF (II.EQ.IROOT) GO TO 200
      II=IARC(K)
         GO TO 180
 200  CONTINUE
C     DETERMINE THE NEW BRANCH TO ENTER THE BASIS.
      SICH=1.E+10
      IENTER=0
      DO 250 K=1,NARC
         IF ((UPPER(K)-FLOW(K)).GT.EPS) GO TO 210
C     LOOKING AT A BACKWARD BRANCH
         IF ((FLOW(K)-LOWER(K)).LT.EPS) GO TO 250
      JJ=IARC(K)
         IF (DISSET(JJ).EQ.0) GO TO 250
      II=JARC(K)
      I=K+NARC
         IF (DISSET(II).EQ.0) GO TO 240
C     NEW BRANCH FORMS A LOOP
         GO TO 220
 210  CONTINUE
      I=K
      JJ=JARC(I)
         IF (DISSET(JJ).EQ.0) GO TO 250
      II=IARC(I)
         IF (DISSET(II).EQ.0) GO TO 240
C     NEW BRANCH FORMS A LOOP.
 220  CONTINUE
      GALPIV=GAN(II)/(GAN(JJ)*AMPF(I))
         IF (GALPIV.GE..999999) GO TO 250
      POTCH=(((V(II)+COSTF(I))/AMPF(I))-V(JJ))/((1-GALPIV)*GAN(JJ))
 230  CONTINUE
         IF (POTCH.GE.SICH) GO TO 250
      SICH=POTCH
      IENTER=I
         GO TO 250
C     NEW BRANCH DOES NOT FORM A LOOP.
 240  CONTINUE
      POTCH=(((V(II)+COSTF(I))/AMPF(I))-V(JJ))/GAN(JJ)
         GO TO 230
 250  CONTINUE
         IF (IENTER.EQ.0) GO TO 270
C     CHANGE NODE LABELS AND POINTERS TO REFLECT ENTERING BRANCH.
C     CHANGE POINTERS.
      CALL TRECHG (IENTER,ILEAV)
C     CHANGE NODE POTENTIALS.
      DO 260 II=1,NODES
         IF (DISSET(II).EQ.0) GO TO 260
      V(II)=SICH*GAN(II)+V(II)
 260  CONTINUE
         GO TO 130
 270  CONTINUE
      PRINT 280, FLONET
      II=IARC(ILEAV)
      CALL ADSUB (ILEAV,II)
      JJ=JARC(ILEAV)
      BARC(JJ)=ILEAV
         GO TO 130
280   FORMAT (* MAXIMUM FLOW FOUND *,F20.10)
290   FORMAT (*  LAB      *,(21F5.2))
300   FORMAT (*  DISSET   *,(21I5))
310   FORMAT (*  GAIN     *,(21F5.3))
320   FORMAT (*  BARC     *,(21I5))
330   FORMAT (*  FARC     *,(21I5))
340   FORMAT (*  RARC     *,(21I5))
      END
      SUBROUTINE TRECHG (IENTER,ILEAV)
C     SUBROUTINE TO FORM THE FLOW AUGMENTING TREE BY DELETING ONE ARC
C     FROM THE TREE AND INSETING A NEW ARC INTO THE TREE.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
C     PRINT 10,IENTER,ILEAV
      IROOT=JARC(ILEAV)
      NLIS=0
      JJ=JARC(IENTER)
C     DELETE PATH FROM JARC(IENTER) TO IROOT
 10   CONTINUE
      JB=BARC(JJ)
         IF (JJ.EQ.IROOT) GO TO 20
      NLIS=NLIS+1
      II=IARC(JB)
      CALL DESUB (JB,II)
      IF (JB.LE.NARC) I=JB+NARC
      IF (JB.GT.NARC) I=JB-NARC
      ICHK(NLIS)=I
      BARC(JJ)=0
      RARC(JJ)=0
      JJ=II
         GO TO 10
 20   CONTINUE
C     ADD IN THE REVERSE OF THE PATH JUST DELETED
         IF (NLIS.EQ.0) GO TO 30
      I=ICHK(NLIS)
      NLIS=NLIS-1
      II=IARC(I)
      JJ=JARC(I)
      CALL ADSUB (I,II)
      BARC(JJ)=I
         GO TO 20
 30   CONTINUE
      II=IARC(IENTER)
      CALL ADSUB (IENTER,II)
      JJ=JARC(IENTER)
      BARC(JJ)=IENTER
      RARC(JJ)=0
      RETURN
      END
      SUBROUTINE LOOP (I,JJ)
C      SUBROUTINE TO DETERMINE IF THE FLOW AUGMENTING TREE INCLUDES A
C     FLOW GENERATING CYCLE.  IF SO NODE POTENTIALS ARE ADJUSTED
C     ACCORDINGLY. USED ONLY IN THE FIRST ITERATION.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHT(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
      DIMENSION ICHK(100)
C     DETERMINE IF POINTERS INDICATE A LOOP.
      DO 10 K=1,NODES
      ICHK(K)=0
 10   CONTINUE
      ICHK(JJ)=1
      IJ=JJ
 20   CONTINUE
      IJK=BARC(IJ)
      IF (IJK.EQ.0) RETURN
      IA=IARC(IJK)
         IF (IA.EQ.JJ) GO TO 30
      IF (ICHK(IA).EQ.1) RETURN
      ICHK(IA)=1
      IJ=IA
         GO TO 20
 30   CONTINUE
      TEMP=0
      GN=1
      IJ=JJ
C     CALCULATE THE COST TO OBTAIN ONE UNIT OF FLOW INTO JJ
 40   CONTINUE
      IJK=BARC(IJ)
      GN=GN*AMPF(IJK)
      TEMP=TEMP+COSTF(IJK)/GN
      IA=IARC(IJK)
         IF (IA.EQ.JJ) GO TO 50
      IJ=IA
         GO TO 40
 50   CONTINUE
C     CALCULATE COST TO OBTAIN ONE UNIT OF FLOW OUT OF LOOP AT JJ
      V(JJ)=TEMP/(1-1/GN)
 60   CONTINUE
      IJ=IA
      IJK=BARC(IJ)
      IA=IARC(IJK)
      IF (IA.EQ.JJ) RETURN
      V(IA)=V(IJ)*AMPF(IJK)-COSTF(IJK)
         GO TO 60
      END
      SUBROUTINE MAXFLO
C     SUBROUTINE TO CALCULATE THE MAXIMUM FLOW INCREASE INTO THE SINK.
C     ARC TO LEAVE THE TREE IS ALSO DETERMINED.  THE FLOW IS CHANGED IN
C     THE AUGMENTING PATH.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
C     FIND OUT IF THERE IS A LOOP. IF SO JJ IS THE JUNCTION OF THE LOOP.
      DO 10 I=1,NODES
      ICHK(I)=0
 10   CONTINUE
      JJ=SOURCE
      I=SINK
 20   CONTINUE
      ICHK(I)=1
         IF (I.EQ.SOURCE) GO TO 40
      II=BARC(I)
         IF (II.EQ.0) GO TO 130
      I=IARC(II)
         IF (ICHK(I).EQ.1) GO TO 30
         GO TO 20
 30   CONTINUE
      JJ=I
      NLOP=NLOP+1
C     FIND MAXIMUM FLOW CHANGE POSSIBLE.
 40   CONTINUE
      FLMX=999999.
      GN=1
      I=SINK
      GAN(I)=1.
 50   CONTINUE
         IF (I.EQ.JJ) GO TO 60
      KK=BARC(I)
      GN=GN*AMPF(KK)
      FLMXT=FLMXC(KK,1.)*GN
      I=IARC(KK)
      GAN(I)=GN
         IF (FLMXT.GT.FLMX) GO TO 50
      FLMX=FLMXT
      IROOT=JARC(KK)
         GO TO 50
 60   CONTINUE
         IF (JJ.EQ.SOURCE) GO TO 70
      CALL FLOP (JJ,FLMAX,GLOOP,IROOTL)
      FLMXT=FLMAX*GN
         IF (FLMXT.GT.FLMX) GO TO 70
      FLMX=FLMXT
      IROOT=IROOTL
 70   CONTINUE
C     INCREASE TOTAL FLOW BY THE MAXIMUM FLOW CHANGE.
      FLO=OUTFLO-FLONET
      IF (FLO.GT.FLMX) FLO=FLMX
      IF (FLO.LT.1.E-6) NDEG=NDEG+1
      CSTNOW=0.0
      FLONET=FLONET+FLO
         IF (FLO.LT.EPS) GO TO 120
C     CALCULATE FLOW CHANGE ON EACH ARC.
      I=SINK
 80   CONTINUE
         IF (I.EQ.JJ) GO TO 90
      II=BARC(I)
         IF (II.EQ.0) GO TO 130
      I=IARC(II)
      FLONOW=FLO/GAN(I)
      CALL FLCHG (II,FLONOW)
         GO TO 80
 90   CONTINUE
         IF (JJ.EQ.SOURCE) GO TO 120
      FLOOP=FLO/(GAN(I)*(1-(1/GLOOP)))
      I=JJ
      FLGA=FLOOP*GAN(JJ)
 100  CONTINUE
      II=BARC(I)
      I=IARC(II)
      FLONOW=FLGA/GAN(I)
         IF (I.NE.JJ) GO TO 110
      J=JARC(II)
      FLONOW=FLGA/(GAN(J)*AMPF(II))
 110  CONTINUE
      CALL FLCHG (II,FLONOW)
         IF (I.EQ.JJ) GO TO 120
         GO TO 100
 120  CONTINUE
      RETURN
 130  CONTINUE
      PRINT 140, FLONET,TOTCST
      CALL EXIT
 140  FORMAT (///,* PROBLEM IS INFEASIBLE. THE MAXIMUM FLOW IS   *,F10.2,
     1*   AT A TOTAL COST OF  *,F10.2)
      END
      SUBROUTINE FLOP (JJ,FLMAX,GN,IROOTL)
C     SUBROUTINE TO DETERMINE THE GAIN, MAXIMUM FLOW CHANGE, AND ROOT OF
C     A FLOW GENERATING CYCLE IN THE FLOW AUGMENTING TREE.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
      FLMAX=9999999999.0
      GN=1
      IJ=JJ
 10   CONTINUE
      IJK=BARC(IJ)
      GN=GN*AMPF(IJK)
      FLMXT=FLMXC(IJK,1.)*GN
         IF (FLMXT.GT.FLMAX) GO TO 20
      FLMAX=FLMXT
      IROOTL=JARC(IJK)
 20   CONTINUE
         IF (IARC(IJK).EQ.JJ) GO TO 30
      IJ=IARC(IJK)
      GAN(IJ)=GAN(JJ)*GN
         GO TO 10
 30   CONTINUE
      FLMAX=FLMAX*(1-1/GN)
      RETURN
      END
```

```
      SUBROUTINE FLCHG (II,FLONOW)
C     SUBROUTINE TO INCREASE THE FLOW IN AN ARC BY A GIVEN AMOUNT.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
         IF (II.GT.NARC) GO TO 10
C     CHANGE FLOW IN A FORWARD ARC.
      FLOW(II)=FLOW(II)+FLONOW
      CSTNOW=CSTNOW+FLONOW*COST(II)
         GO TO 20
C     CHANGE FLOW IN A MIRROR ARC.
 10   CONTINUE
      KK=II-NARC
      FLONOW=FLONOW/AMP(KK)
      FLOW(KK)=FLOW(KK)-FLONOW
      CSTNOW=CSTNOW-FLONOW*COST(KK)
 20   CONTINUE
      RETURN
      END
      SUBROUTINE ADSUB (I,II)
C     SUBROUTINE TO ADD AN ARC TO THE TRIPLE LABEL REPRESENTATION OF THE
C     FLOW AUGMENTING TREE.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
C     ADDS ARC I TO THE LIST OF SUBSEQUENT ARCS TO NODE II.
C     PRINT 50,I,II,FARC(II)
         IF (FARC(II).NE.0) GO TO 10
      FARC(II)=I
         GO TO 40
 10   CONTINUE
      MM=FARC(II)
 20   CONTINUE
      MN=JARC(MM)
         IF (RARC(MN).NE.0) GO TO 30
      RARC(MN)=I
         GO TO 40
 30   CONTINUE
      MM=RARC(MN)
         GO TO 20
 40   CONTINUE
      RETURN
      END


      SUBROUTINE DESUB (I,II)
C     SUBROUTINE TO DELETE AN ARC FROM THE TRIPLE LABEL REPRESENTATION
C     OF THE FLOW AUGMENTING TREE.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
C     DELETES ARC I FROM THE LIST OF SUBSEQUENT ARCS TO NODE II.
C     PRINT 50,I,II,FARC(II)
      JJ=JARC(I)
         IF (FARC(II).NE.I) GO TO 10
      FARC(II)=RARC(JJ)
      RETURN
 10   CONTINUE
      MM=FARC(II)
 20   CONTINUE
      MN=JARC(MM)
         IF (RARC(MN).NE.I) GO TO 30
      RARC(MN)=RARC(JJ)
      RETURN
 30   CONTINUE
      MM=RARC(MN)
         GO TO 20
      END
      SUBROUTINE PROUT
C     SUBROUTINE TO PRINT OUT THE OPTIMAL SOLUTION.
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1       UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2       FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /16/ TITLE(80)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1       IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL FLMXC,AMPF,COSTF
 10   CONTINUE
      PRINT 30, (TITLE(I),I=1,80)
      PRINT 40
      DO 20 I=1,NARC
         IF ((I.GT.NPRIT).AND.(I.LT.(NARC-NPRIT))) GO TO 20
      ACOST=COST(I)*FLOW(I)
      PRINT 50, I,IARC(I),JARC(I),LOWER(I),UPPER(I),COST(I),AMP(I),
     1       FLOW(I),ACOST
 20   CONTINUE
      TOTCSP=TOTCST-FLOW(NARC)*COST(NARC)
      PRINT 60, TOTCSP
      PRINT 70, ITER,NDEG,NLOP
      RETURN
 30   FORMAT (1H1//10X,80A1//31H*****OPTIMAL FLOW PATTERN*****,///)
     1    FLOW*,//)
 40   FORMAT (*   ARC START  END      LOWER      UPPER       COST        GAIN
     1       FLOW    ARC COST*)
 50   FORMAT (I5,2X,2I5, 5F10.2,F15.2)
 60   FORMAT (///,21H *****TOTAL COST*****,F20.4)
 70   FORMAT (* NUMBER OF ITERATIONS *,I10/* NUMBER OF DEGENERATE ITERAT
     1IONS *,I10/* NUMBER OF LOOP ITERATIONS *,I10/)
      END
      FUNCTION FLMXC (I,S)
C     FUNCTION TO DETERMINE MAXIMUM FLOW CHANGE IN AN ARC
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
      EXTERNAL COSTF,AMPF
      II=IARC(I)
      JJ=JARC(I)
         IF (S) 10,10,50
C     FLOW IS TO BE DECREASED
 10   CONTINUE
         IF (I.GT.NARC) GO TO 30
         IF (FLOW(I).GT.UPPER(I)) GO TO 20
      FLMXC=FLOW(I)-LOWER(I)
      RETURN
 20   CONTINUE
      FLMXC=FLOW(I)-UPPER(I)
      RETURN
 30   CONTINUE
      K=I-NARC
         IF (FLOW(I).LT.LOWER(I)) GO TO 40
      FLMXC=(UPPER(K)-FLOW(K))*AMP(K)
      RETURN
 40   CONTINUE
      FLMXC=(LOWER(K)-FLOW(K))*AMP(K)
      RETURN
C     FLOW IS TO BE INCREASED
 50   CONTINUE
         IF (I.GT.NARC) GO TO 80
         IF (FLOW(I).GE.UPPER(I)) GO TO 110
         IF (FLOW(I).LT.LOWER(I)) GO TO 70
         IF ((((V(II)+COSTF(I))/AMP(I))-EPS).GT.V(JJ)) GO TO 60
      FLMXC=UPPER(I)-FLOW(I)
      RETURN
 60   CONTINUE
 70   CONTINUE
      FLMXC=LOWER(I)-FLOW(I)
      RETURN
 80   CONTINUE
      K=I-NARC
         IF (FLOW(K).GT.UPPER(K)) GO TO 100
         IF (FLOW(K).LE.LOWER(K)) GO TO 110
         IF ((((V(II)+COSTF(I))/AMPF(I))-EPS).GT.V(JJ)) GO TO 90
      FLMXC=(FLOW(K)-LOWER(K))*AMP(K)
      RETURN
 90   CONTINUE
 100  CONTINUE
      FLMXC=(FLOW(K)-UPPER(K))*AMP(K)
      RETURN
 110  CONTINUE
      FLMXC=0.0
      RETURN
      END
      FUNCTION COSTF  (I)
C     FUNCTION TO CALCULATE THE COSTS ON AN ARC CONSIDERING UPPER AND
C     LOWER BOUNDS
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
         IF (I.GT.NARC) GO TO 50
         IF (FLOW(I)-LOWER(I)) 10,30,30
 10   CONTINUE
      COSTF=-BIG
      RETURN
 20   CONTINUE
      COSTF=COST(I)
      RETURN
 30   CONTINUE
         IF (UPPER(I)-FLOW(I)) 40,40,20
 40   CONTINUE
      COSTF=BIG
      RETURN
 50   CONTINUE
      K=I-NARC
         IF (FLOW(K)-LOWER(K)) 80,80,70
 60   CONTINUE
      COSTF=-COST(K)/AMP(K)
      RETURN
 70   CONTINUE
         IF (UPPER(K)-FLOW(K)) 90,60,60
 80   CONTINUE
      COSTF=BIG/AMP(K)
      RETURN
 90   CONTINUE
      COSTF=-BIG/AMP(K)
      RETURN
      END
```

```
      FUNCTION AMPF  (I)
      COMMON /1/ IARC(1)/2/JARC(1)/3/COST(1)/4/AMP(1)/5/FLOW(1)/6/
     1      UPPER(1)/7/LOWER(1)/8/V(1)/9/BARC(1)/10/RARC(1)/11/
     2      FARC(1)/12/DISSET(1)/13/GAN(1)/14/ICHK(1)/15/LIST(1)
      COMMON /V/ SOURCE,SINK,NARC,OUTFLO,FLONET,CSTNOW,TOTCST,NODES,IFS,
     1      IROOT,EPS,BIG,NDEG,NLOP,SICH,ITER,NPRIT,TIMAX,TIME,IPRINT
      INTEGER SOURCE,SINK,BARC,FARC,RARC,DISSET
      REAL LOWER
         IF (I.GT.NARC) GO TO 10
      AMPF=AMP(I)
      RETURN
 10   CONTINUE
      AMPF=1/AMP(I-NARC)
      RETURN
      END
```

Input Data for the Example Problem
The iterations of this problem are illustrated in Figure 7.

```
EXAMPLE PROBLEM
       1          10          10          30          2
   10.
       9           8       0.          6.         10.          .80
       9          10       0.          4.          2.          1.
       1           2       0.          10.        40.          1.
       1           3       0.          8.          8.          .8
       1           4       0.          6.         10.          .85
       2           5       0.          6.          6.          .75
       2           7       0.          20.        10.          .90
       3           2       0.          4.          4.          .85
       3           4       0.          10.        12.          .65
       3           6       0.          4.          2.          .90
       4           6       0.          8.          1.          .80
       5           3       0.          8.          2.          .80
       5           6       0.          2.          2.          .70
       5           7       0.          12.         4.          1.
       6           8       0.          4.          4.          .75
       6           9       0.          8.         -3.          1.
       7           8       0.          5.          2.          1.
       7          10       0.          8.          1.          .85
       8           5       0.          12.         0.          .95
       8          10       0.          2.         20.          1.
       9           4       0.          2.          6.          .75
       0
```

Output for the Example Problem (NPRINT = 0)

```
   EXAMPLE PROBLEM


SOURCE NODE=     1      SINK NODE =    10


OUTPUT FLOW REQUIREMENT      10.00


*****INPUT DATA*****

          ARC       START       END       LOWER      UPPER       COST     AMPLIFICATION
          NO.        NODE       NODE      BOUND      BOUND


           1          9          8        0.00       6.00      10.00           .80
           2          9         10        0.00       4.00       2.00          1.00
           3          1          2        0.00      10.00      40.00          1.00
           4          1          3        0.00       8.00       8.00           .80
           5          1          4        0.00       6.00      10.00           .85
           6          2          5        0.00       6.00       6.00           .75
           7          2          7        0.00      20.00      10.00           .90
           8          3          2        0.00       4.00       4.00           .85
           9          3          4        0.00      10.00      12.00           .65
          10          3          6        0.00       4.00       2.00           .90
          11          4          6        0.00       8.00       1.00           .80
          12          5          3        0.00       8.00       2.00           .80
          13          5          6        0.00       2.00       2.00           .70
          14          5          7        0.00      12.00       4.00          1.00
          15          6          8        0.00       4.00       4.00           .75
          16          6          9        0.00       8.00      -3.00          1.00
          17          7          8        0.00       5.00       2.00          1.00
          18          7         10        0.00       8.00       1.00           .85
          19          8          5        0.00      12.00       0.00           .95
          20          8         10        0.00       2.00      20.00          1.00
          21          9          4        0.00       2.00       6.00           .75
```

Output for the Example Problem (NPRINT = 0)

```
           EXAMPLE PROBLEM

****OPTIMAL FLOW PATTERN*****,


 ARC START  END      LOWER      UPPER       COST        GAIN       FLOW   ARC COST
  1      9    8       0.00       6.00      10.00         .80       0.00        0.00
  2      9   10       0.00       4.00       2.00        1.00       4.00        8.00
  3      1    2       0.00      10.00      40.00        1.00       2.67      106.79
  4      1    3       0.00       8.00       8.00         .80       8.00       64.00
  5      1    4       0.00       6.00      10.00         .85       6.00       60.00
  6      2    5       0.00       6.00       6.00         .75       0.00        0.00
  7      2    7       0.00      20.00      10.00         .90       6.07       60.70
  8      3    2       0.00       4.00       4.00         .85       4.00       16.00
  9      3    4       0.00      10.00      12.00         .65       0.00        0.00
 10      3    6       0.00       4.00       2.00         .90       2.40        4.80
 11      4    6       0.00       8.00       1.00         .80       5.10        5.10
 12      5    3       0.00       8.00       2.00         .80       0.00        0.00
 13      5    6       0.00       2.00       2.00         .70       0.00        0.00
 14      5    7       0.00      12.00       4.00        1.00       1.60        6.38
 15      6    8       0.00       4.00       4.00         .75       2.24        8.96
 16      6    9       0.00       8.00      -3.00        1.00       4.00      -12.00
 17      7    8       0.00       5.00       2.00        1.00       0.00        0.00
 18      7   10       0.00       8.00       1.00         .85       7.06        7.06
 19      8    5       0.00      12.00       0.00         .95       1.68        0.00
 20      8   10       0.00       2.00      20.00        1.00       0.00        0.00
 21      9    4       0.00       2.00       6.00         .75       0.00        0.00
 22      1   10       0.001000000.00 100000.00        1.00       0.00        0.00


*****TOTAL COST*****             335.7930
NUMBER OF ITERATIONS            7
NUMBER OF DEGENERATE ITERATIONS           1
NUMBER OF LOOP ITERATIONS            1
```

Intermediate Output for NPRINT = 1.

```
FLOW    is the output flow for the iteration.
COST    is the total cost of the flow.
REMOVE  is the arc which leaves the tree at
        the current iteration.
ENTER   is the arc which enters the tree.
DELTA   is the increase in the node potential at
        the sink.
```

```
ITERATION     1       FLOW       3.60      COST                        44.40
REMOVE     8    ENTER      1    DELTA              24.14736
ITERATION     2       FLOW       3.60      COST                        44.40
REMOVE    10    ENTER      7    DELTA                .77255
ITERATION     3       FLOW       4.00      COST                        50.38
REMOVE    36    ENTER     11    DELTA               1.85000
ITERATION     4       FLOW       5.56      COST                       106.22
REMOVE     2    ENTER     18    DELTA              20.82266
ITERATION     5       FLOW       7.79      COST                       192.77
REMOVE     4    ENTER     14    DELTA               3.05470
ITERATION     6       FLOW       7.96      COST                       199.90
REMOVE     5    ENTER     32    DELTA               3.55535
ITERATION     7       FLOW      10.00      COST                       335.79
REMOVE     8    ENTER      3    DELTA              24.14736
```

Intermediate Output for NPRINT = 2.

BARC   is the backward pointer in the three label
        tree representation.
FARC   is the forward pointer.
RARC   is the right pointer.
DISSET indicates the nodes in G_{A2}.
LAB   shows the node potentials.
GAIN   shows the node gains.

```
 BARC        0    8    4    5   19   10   14   15   16    2
 FARC        4    0    8    0   14   15    0   19    2    0
 RARC        0   10    5    0    0    0    0   16    0    0
 DISSET      1    1    1    1    1    1    1    1    1    1
 LAB      0.0016.4710.0011.7624.3313.3328.3323.1110.3312.33
 GAIN    1.0001.0001.0001.0001.0001.0001.0001.0001.0001.000
ITERATION     1      FLOW       3.60      COST                44.40
REMOVE    0     ENTER    1     DELTA             0.00000
 BARC        0    8    4    5   36   37    7   41   16    2
 FARC        4    7    8    0   41   16   36   37    2    0
 RARC        0    0    5    0    0    0    0    0    0    0
 DISSET      0    0    0    0    1    1    1    1    1    1
 LAB      0.0016.4710.0011.7625.4114.1129.4124.1411.1113.11
 GAIN     .7201.000 .9001.0001.4041.0001.4041.3331.0001.000
ITERATION     2      FLOW       3.60      COST                44.40
REMOVE   10     ENTER    7     DELTA              .77255
 BARC        0    8    4    5   19   11    7   15   16    2
 FARC        4    7    8   11    0   16    0   19    2    0
 RARC        0    0    5    0    0    0    0    0   15    0
 DISSET      0    0    0    0    1    1    0    1    1    1
 LAB      0.0016.4710.0011.7628.0115.9629.4126.6112.9614.96
 GAIN     .8591.2631.0741.0001.4041.0001.4041.3331.0001.000
ITERATION     3      FLOW       4.00      COST                50.38
REMOVE   36     ENTER   11     DELTA             1.85000
 BARC        0    8    4    5   19   11    7   15   16   18
 FARC        4    7    8   11    0   16   18   19    0    0
 RARC        0    0    5    0    0    0    0    0   15    0
 DISSET      0    0    0    0    0    0    0    0    0    1
 LAB      0.0016.4710.0011.7628.0115.9629.4126.6112.9635.78
 GAIN     .6801.2631.074 .8001.4041.0001.4041.3331.0001.000
ITERATION     4      FLOW       5.56      COST               106.22
REMOVE    2     ENTER   18     DELTA            20.82266
```

```
 BARC          0    29    30     5    19    11    14    15    16    18
 FARC          5    30     0    11    14    16    18    19     0     0
 RARC          0     0     0     0     0     0     0     0    15    29
 DISSET        0     1     1     0     0     0     1     0     0     1
 LAB        0.0018.8111.9911.7628.0115.9632.0126.6112.9638.83
 GAIN       .520 .765 .650 .8001.4041.000 .8501.3331.0001.000
ITERATION      5        FLOW         7.79        COST                    192.77
REMOVE      4       ENTER   14       DELTA                 3.05470
 BARC          0     8    32    33    36    37     7    41    16    18
 FARC          0     7     8     0    41    16    18    37     0     0
 RARC          0     0     0    32     0     0     0     0    33    36
 DISSET        0     1     1     1     1     1     1     1     1     1
 LAB        0.0021.5314.3013.4931.0318.1135.0329.4815.1142.39
 GAIN       .412 .765 .650 .484 .850 .606 .850 .807 .6061.000
ITERATION      6        FLOW         7.96        COST                    199.90
REMOVE      5       ENTER   32       DELTA                 3.55535
 BARC          0     3    32    33    36    37     7    41    16    18
 FARC          3     7     0     0    41    16    18    37     0     0
 RARC          0     0     0    32     0     0     0     0    33    36
 DISSET        0     1     1     1     1     1     1     1     1     1
 LAB        0.0040.0027.4625.1951.5632.7355.5648.9829.7366.54
 GAIN       .412 .765 .545 .484 .850 .606 .850 .807 .6061.000
ITERATION      7        FLOW        10.00        COST                    335.79
REMOVE      8       ENTER    3       DELTA                24.14736
```

MULTIRESERVOIR OPERATING POLICIES CONSIDERING UNCERTAINTY

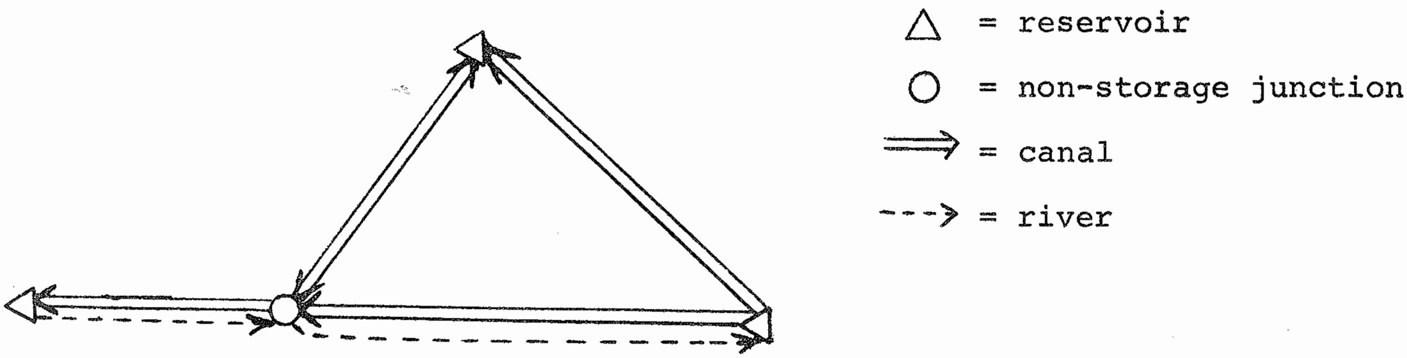
by

W. D. Driscoll

I. Introduction

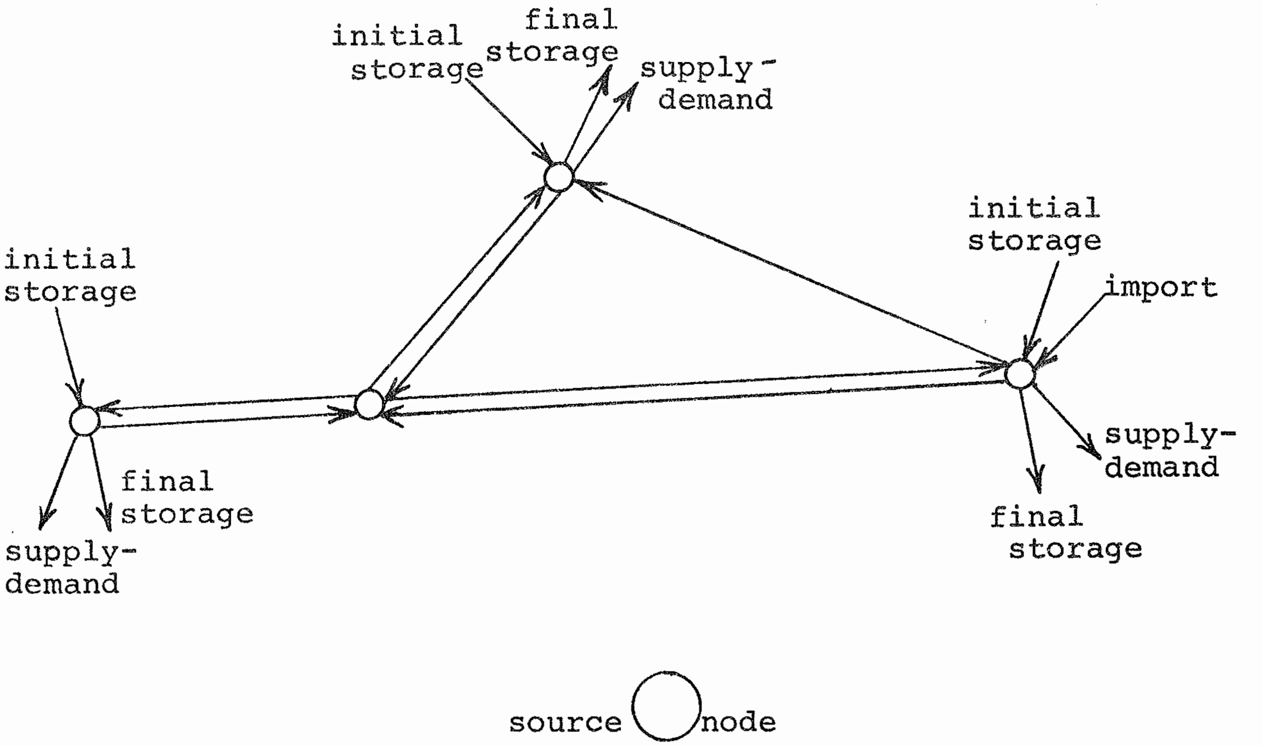
Between 1968 and 1970 the Texas Water Development Board engaged in extensive development of planning techniques for a large multireservoir water distribution system. The out-of-kilter algorithm for determining a minimum cost flow pattern in a network was an important part of these planning techniques. The out-of-kilter algorithm operates on a model consisting of nodes and arcs arranged in any desired pattern. For each arc three parameters are specified: a cost per unit flow, a lower bound on flow, and an upper bound on flow. For a canal these parameters might represent pumping cost per acre-foot per month, minimum throughput per month, and maximum throughput per month. The out-of-kilter algorithm will find the minimum cost circulation for such a network, that is an assignment of flows to arcs which minimizes the total cost while requiring that the flow on each arc remains between the bounds for that arc and that flow be conserved at each node. Negative flows are permissable if they satisfy these requirements.

To illustrate a network consider the following hypothetical multireservoir system.



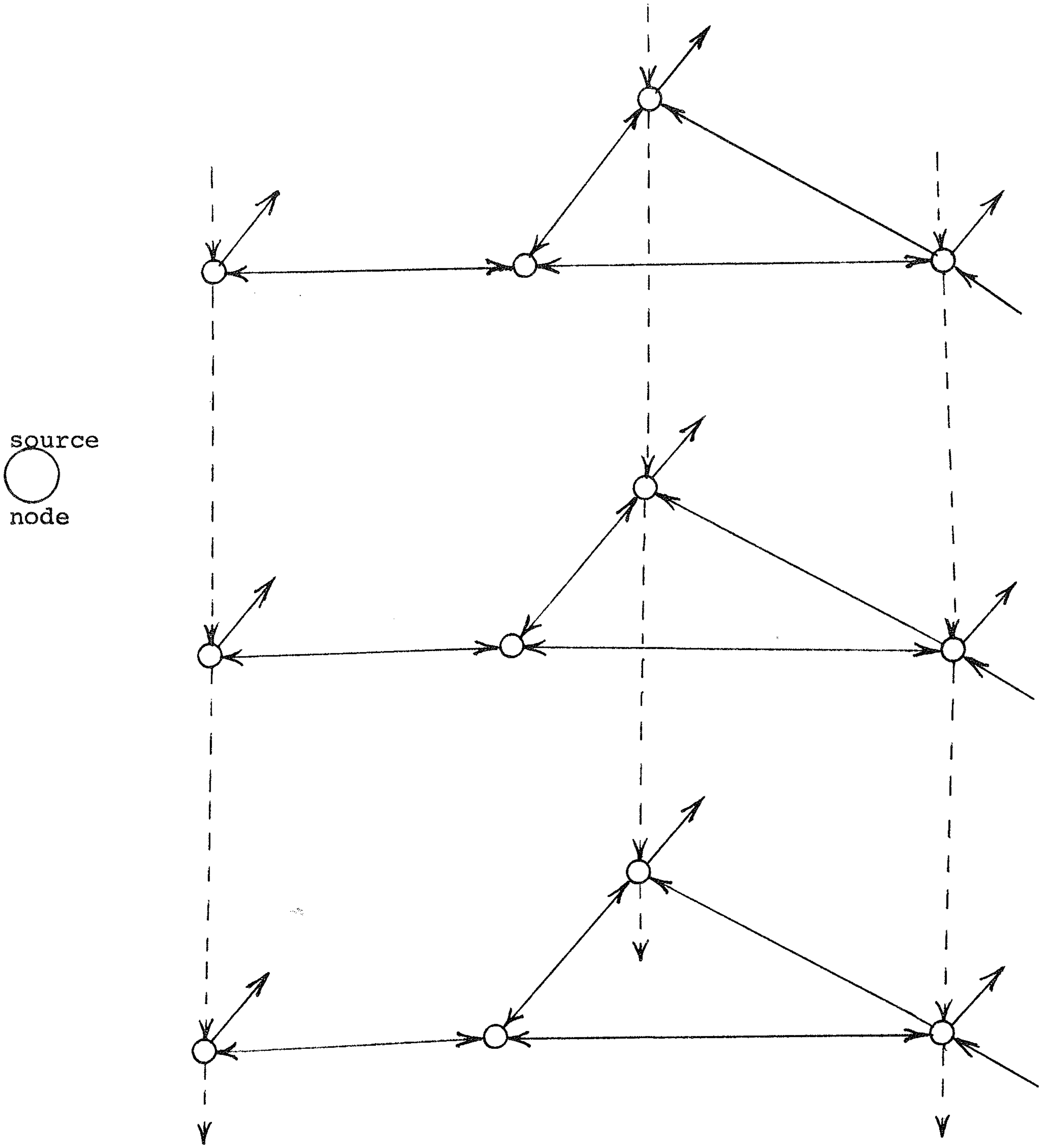
Associated with each canal there is a pumping cost per unit of water per month, a minimum throughput per month, and a maximum throughput per month. Associated with each reservoir there is an initial storage level, a minimum final storage level, a maximum final storage level, and a cost of storage per unit of water per month (which may be negative to represent a benefit). There are also a supply and a demand at each reservoir. At some reservoirs water can be imported from outside the system.

The following diagram illustrates the network which when solved by the out-of-kilter algorithm would yield the optimum water distribution policy for a single month.



All unattached arc ends should connect to the source node. Of course each arc would have the three parameters mentioned earlier. At each reservoir the supply-demand arc would have flow equal to the net of demand (water used or sold) and supply (rainfall and runoff). Hence flow will be negative if supply exceeds demand. In order to allow for the possibility that a demand cannot be satisfied (in the case where demand exceeds supply) the upper bound on the demand arc can be set equal to the net demand, the lower bound set equal to zero, and a per unit cost of -M where M is a large positive number can be assigned to the arc. The physical characteristics of the multireservoir system determine the costs and bounds on the arcs representing canals, rivers, reservoir storages, or imports.

To determine optimal reservoir operating rules over a longer time span, a multiperiod model is required. The following network is a three-month model for determining an optimal reservoir operating policy for the multireservoir system depicted earlier.

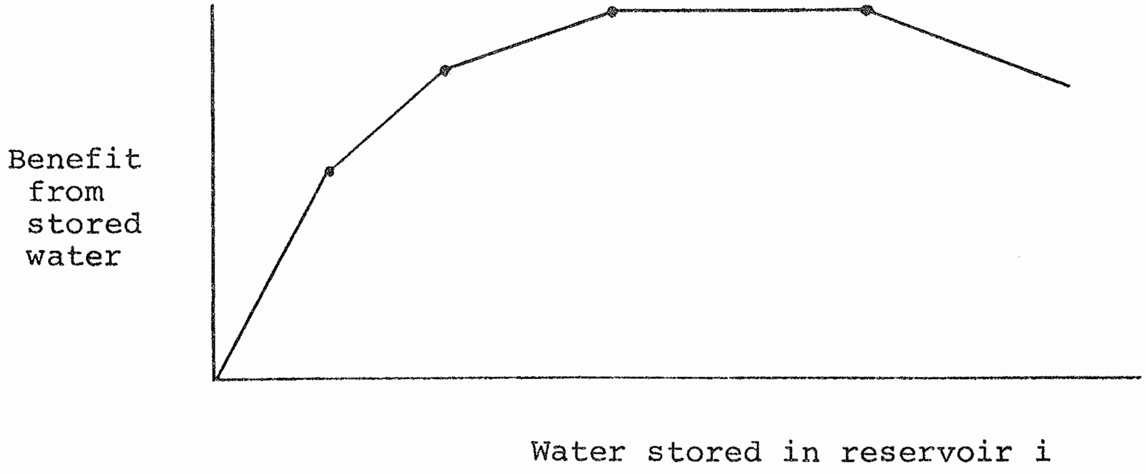

source
node

As before all unconnected arc ends should connect to the source node. To make the diagram more easily readable initial storage, final storage, and intermonth storage arcs are depicted with dashed lines. A multiperiod network of this type consisting of up to 36 months was used by the Texas Water Development Board. However there are inherent difficulties in using a multiperiod model of this type.

1. The necessity for perfect information. The assumption is made that inflows and demands are known exactly for the entire time horizon.
2. The large size of the network. Computation time for the out-of-kilter algorithm increases rapidly as the number of arcs in the network increases.
3. Economic value of final reservoir storage is ignored.
4. Wasted computation. Since inflows and demands will not exactly duplicate their expected values, the solution obtained may no longer be optimal or even feasible in later months. The multiperiod network is needed to allow planning for the future, but the operating policy determined for later months will probably not be useful.

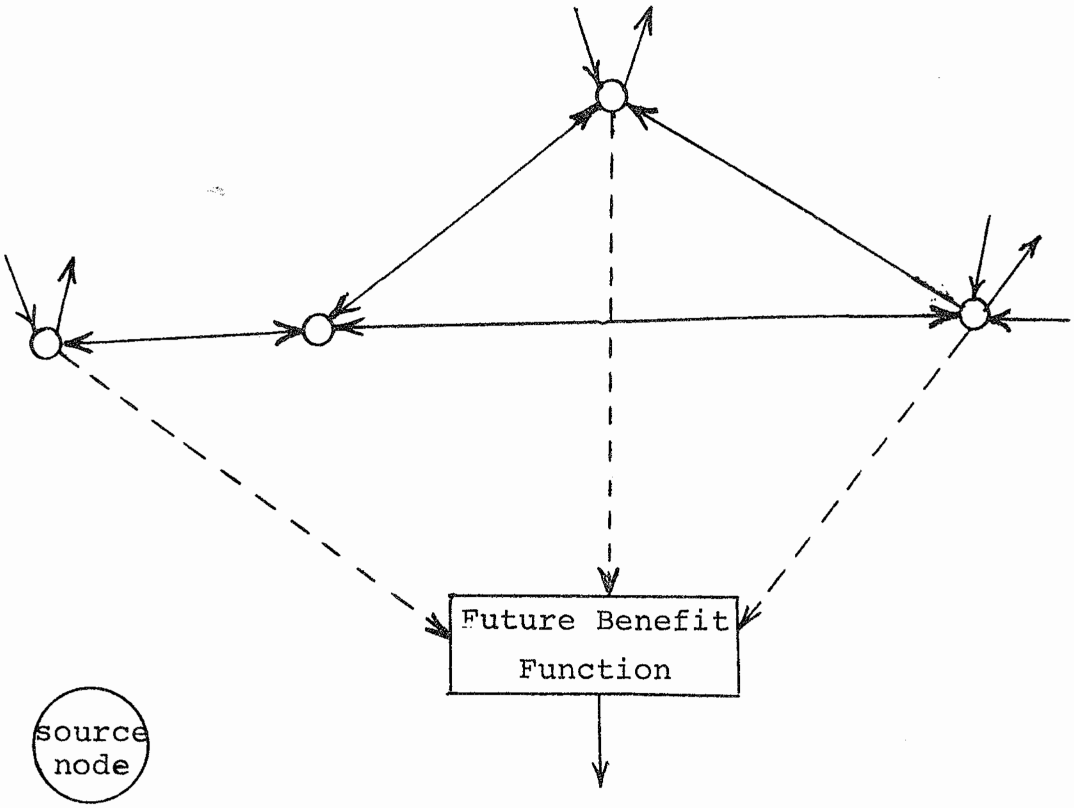
It would be desirable to work exclusively with one-month networks. However, if this is done, some means must be provided to encourage the algorithm to store water in the reservoirs in appropriate amounts for use in subsequent months. One method of accomplishing this is to assign a benefit to water stored for the future so that water will be stored if it is valuable to do so. The Texas Water Development Board took this approach, but encountered difficulties as to what benefits to assign to water stored for the future in the reservoirs. They found that solutions were very sensitive to these values.

For given values of storage in other reservoirs, a curve can be determined that will express the value of the water stored in a single reservoir as a function of the amount of water stored in the reservoir. For the multireservoir system as we have modeled it, this benefit curve will be continuous piecewise-linear and concave.



A curve such as this is compatible with the out-of-kilter algorithm but requires multiple final storage arcs for each reservoir. The non-linear out-of-kilter algorithm developed by Jensen and Reeder can handle a benefit curve such as this with a single arc. In fact any continuous concave benefit function (or convex cost function) would be allowable. The benefit function could be found by changing the amount of water parametrically from 0 to the maximum capacity of the reservoir, and using the out-of-kilter algorithm to find the effect on cost over some time horizon. The out-of-kilter algorithm is well sutied to parametric analysis of this sort. The difficulty with this approach is that it neglects the fact that the value of water stored in a particular reservoir of a multi-reservoir system is effected by the amount of water stored in the other reservoirs. Due to these interactions, separate benefit functions cannot be expressed for each reservoir.

The purpose of this research has been to develop a method whereby reservoir operating policies can be determined using iterative out-of-kilter solutions of simple one-month networks. This requires the development of benefit functions that express the future value of any configuration of reservoir storages at the end of each month. The benefit functions and the out-of-kilter algorithm are made compatible, so that a good solution can be found rapidly. The method is able to take into account uncertainties in supply and demand. The following diagram illustrates this approach to the multireservoir problem.



II. Literature Review

Most of the research involving mathematical analysis of multireservoir systems has occurred during the last decade. There has been a great deal of variation in both the scope of the problem investigated and in the mathematical techniques employed. Roefs [20], Buras [1], and Hall and Dracup [11] discuss the mathematical techniques used and those variations of the problem for which each technique is most suitable. Butcher [2] notes that a mathematical tool useful for one water resources problem may be unsuitable for other seemingly similar problems. Multireservoir models can be roughly classified in the following three categories depending on their emphasis and scope.

1. Design models. These models make decisions concerning the construction of the reservoir system. They are sometimes called capacity expansion models. Decisions are made concerning the size, location, and time of construction of reservoirs and canals in addition to determining water allocation.
2. Water-use models. In these models the reservoirs are considered to be multipurpose; that is, several possible uses are available for water at each reservoir. Decisions are made concerning such things as the timing and extent of irrigation for various crops. These models most often concern a single reservoir.
3. Time planning models. The main objective with these models is to determine the use and storage of water in several interconnected reservoirs in such a way as to be prepared for future storages or excesses. This research concerns itself with a model of this type.

Just as there is great variation in the scope of the model for multireservoir problems, the mathematical methods employed are diverse. Included are the following:

1. Linear programming
2. Dynamic programming, both deterministic and stochastic
3. Chance-constrained programming
4. Decomposition approaches
5. Simulation
6. Markov chains
7. Network methods

We will try to give references to papers employing these methods, the strengths and weaknesses of the methods with regard to the multireservoir problem, and the type of multireservoir models to which they are most suitable.

Network methods have already been mentioned. They include the techniques used by the Texas Water Development Board and those proposed in this paper. The out-of-kilter algorithm has been used in both time-planning and design models. Its strengths are speed of computation and ease in diagramming the problem. Its greatest weakness is the lack of flexibility of the network form.

Linear programming has been applied to all types of multireservoir models. Roefs [20, p. 43-52] gives an example of a four-reservoir model of the time planning variety. This model is similar in nature to the network model of the Texas Water Development Board. The linear programming method is considerably slower than the linear out-of-kilter algorithm, but has more flexibility. For instance monthly evaporation can be included in the model as a percentage of reservoir storage. For a multiperiod model, linear programming has the same shortcomings as previously mentioned for the linear out-of-kilter algorithm--the necessity for perfect information, the large size of the problem, failure to make use of final reservoir storage, wasted computation, and of course the restriction to linearity. As stated by Buras [1, p. 112], ". . . linear programming yields only point solutions in the policy space, no matter how many dimensions this space may have. Most situations in which the state of the system changes (in time or in space) and in which decisions have to be taken successivley are clearly outside the grasp of linear programming."

Windsor and Chow [25] present a linear programming model that is a combination design and water-use model. The linear programming method appears to be more suitable for models of these types where the number of time periods can be kept to a minimum. They consider their model a practical one computationally, but admit its weakness in not considering the stochastic nature of the problem. Other linear programming models that take stochastics into account are discussed later, under the topic of chance-constrained programming. Salcedo [22] employs linear programming as one of his tools in a water-use model.

Dynamic programming is the most theoretically appealing approach to multiperiod reservoir models of all types since these problems invlove sequential decision-making processes. Also the outcome of each decision (or set of decisions in a time period) appears as a function rather than as a point solution. That is, the optimal decision to be made is determined for any state of the system. This allows suboptimal policies to be examined, a desirable feature due to the inherent uncertainty in multireservoir problems. This feature makes dynamic programming especially useful for real-time system operation. The limitation on the usefulness of dynamic programming is the so-called "curse of dimensionality." Each reservoir gives rise to a new state variable (usually the final reservoir storage). If there are n reservoirs and each reservoir has k possible levels there are k^n possible state combinations per time period. For this reason most of the dynamic programming models have been for systems with either one or two reservoirs. Buras [1, p. 125] fixes four as the maximum number of reservoirs that can be handled computationally by dynamic programming. Dynamic programming also fails to take into account the stochastic nature of the multireservoir problems. This can be rectified by using stochastic dynamic programming as will be discussed soon.

Dynamic programming formulations can be found in Roefs [20, p. 53-58], Hall and Dracup [11, p. 172-176], and Buras [1, p. 123-125] for time-planning multireservoir models. Fults and Hancock [10] present a state-increment dynamic programming algorithm they applied to a two-reservoir system in California. Dynamic programming has also been applied to one-reservoir water-use models, but this has been largely the stochastic variety. Buras [1] presents a dynamic programming formulation for a design-type model [p. 143-149] and for a water-use model [p. 155-171]. Young [26] combines dynamic programming with a simulation approach. Rood [21] presents a dynamic programming model of the time-planning variety that is especially designed for serially linked reservoirs. This reduces the state space dimensionality problem.

Butcher [2] defines stochastic dynamic programming to be ". . . those formulations of dynamic programming in which the value of one of the state variables is related in a probabilistic way to the value of that same variable in adjacent time periods." In terms of multireservoir problems, stochastic dynamic programming allows the demand (or net demand, i.e. demand minus inflow) at a reservoir in one time period to be dependent probabilistically on the demand at that reservoir in the previous time period. The optimal policy developed is that which minimizes expected costs (or maximizes expected returns) for the system. As with traditional dynamic programming the optimal policy is in a form that can readily be used for real time system operation. However the dimensionality problem is compounded due to the additional state variables which are the demands at the various reservoirs in the previous period. Hence Butcher states [2, p. 6] "Any attempt to use stochastic dynamic programming for a two-reservoir formulation involves so many possible states that the problem is too highly dimensional for feasible computation."

For this reason stochastic dynamic programming models have been applied mainly to water-use models, rather than the other models that tend to have a multiplicity of reservoirs. Butcher [3] presents such a model for one multipurpose reservoir. Loucks [16, p. VI-8 through VI-13] presents three stochastic dynamic programming models that he used to define operating policies for several of the Finger Lakes in New York. These are also one-reservoir models. Instead of economic objectives Loucks minimized the sum of squares of the departures of releases from a set of target releases specified by the state. Roefs [20, p. 72-77] presents stochastic dynamic programming formulation for one- and two- reservoir systems.

Chance-constrained linear programming is another tool that has been applied to multireservoir problems in an attempt to account for stochastic variation. Like deterministic linear programming, this method is more suitable for design or water-use models than it is for time-planning models. The fact that point solutions are found, causes chance-constrained linear programming to be less applicable to real-time system operation over a long time span. Loucks [17] proposes a one-reservoir water-use model. Roefs [20, p. 70] extends this model to two reservoirs but expresses doubts about its computational feasibility. Curry and Helm [5] present a chance-constrained model for a single multi-purpose reservoir and then [6] extend this to a system of linked multi-purpose reservoirs. They allow the unregulated inflow into each reservoir at each time period to be stochastic with known probability distribution. There is independence between reservoirs and for the same reservoir in different time periods. They show how their formulations reduce to a deterministic linear programming problem. In addition to these water-use models Curry and Helm [12] present a design model for a system of reservoirs. This formulation employs a mixed continuous and integer linear programming form that is solved by Benders' decomposition technique.

The remaining mathematical methods have been less used and are not easily classified. Parikh [18] and [19] presents a linear decomposition method designed for use in a Northern California system. Bodin and Roefs [4] present a non-linear decomposition approach that employs simulation and regression analysis. Young [26] combines dynamic programming with Monte Carlo simulation of stochastic inflows. Su [23] presents a markov chain approach for serially connected reservoirs. He solves the markov system by a method of successive approximations rather than be dynamic programming. It is interestnng to note that Butcher [2] foresees more extensive use of simulation methods as multireservoir models grow in complexity.

III. Proposed Solution Method

A. An Overview of the Proposed Method and its Objectives

The method to be presented here for determining an optimal operating policy for a multireservoir system combines the basic ideas of several of the approaches that have been previously proposed. We make use of a network formulation and the out-of-kilter algorithm. The idea of decomposition is present in that we break up the larger problem of determining operating policy for a number of years into a sequence of one-month optimizations. Communication between one-month problems is provided by a benefit function which assigns a numerical value or benefit to any configuration of reservoir storage levels. The idea of dynamic programming is employed in iteratively computing these benefit functions.

Our basic objective is to model the reservoir operating problem with simple one-month networks. Future needs will be taken into account through benefit functions that express the future value of stored water at the end of each month. The model will be adaptable to uncertainties in net demand. Flexibility and ease of use will be important considerations.

The method of presentation will be to first consider a multireservoir system with deterministic monthly demands at each reservoir and a fixed time horizon. Shortcomings of this model will be discussed and the changes will be made to make the model more usable. In the next section the idea of benefit functions will be introduced. Properties of the benefit functions will be shown and a method of decomposing a long-term problem into a sequence of monthly problems will be presented. This will entail a few additional assumptions which require a further reformulation of the deterministic problem. The final section of this chapter will adapt the benefit function concept to stochastic demands. Properties and use of the resulting stochastic benefit functions will be shown.

B. Definition of the Deterministic Problem

Let us consider a multireservoir system with deterministic monthly demands at each reservoir and a fixed time horizon.

Parameters of the System

R = Number of reservoirs (treat junctions as reservoirs with zero storage capacity)

r = Reservoir number, an integer index with range $1 \le r \le R$

C = Number of canals

c = Canal number, an integer index with range $1 \le c \le C$

A = Beginning month $\qquad$ B = ending month

t = Time period, an integer index with range $A \leq t \leq B$

$s(r)$ = Maximum storage capacity of reservoir, $r, \forall r$

$u(c)$ = Maximum pumping capacity per month in canal $c, \forall c$

$d(r,t)$ = Demand at reservoir r in month $t, \forall r$ and t

$i(r,t)$ = Unregulated inflow (runoff and rainfall) at reservoir r in month $t, \forall r$ and t

$m(r,t)$ = The maximum amount of water that may be imported at reservoir r in month $t, \forall r$ and t

$k(c)$ = Cost for pumping one unit of water through canal c, $\forall c$

$p(r,t)$ = Cost for importing one unit of water at reservoir r in month $t, \forall r$ and t

$F(r)$ = The set of canals branching away from reservoir $r, \forall r$

$T(r)$ = The set of canals branching into reservoir $r, \forall r$

$\ell(r,0)$ = The initial storage of reservoir $r, \forall r$ subject to the restriction $0 \leq \ell(r,0) \leq s(r) \forall r$

R, r, C, c, A, B, and t are positive integers

$s(r)$, $u(c)$, $d(r,t)$, $i(r,t)$, $m(r,t)$, and $\ell(r,0)$ are non-negative numbers.

Variables of the System

$\ell(r,t)$ = The storage level of reservoir r at the end of month $t, \forall r$ and t

$f(c,t)$ = The amount of flow in canal c in month $t, \forall c$ and t

$w(r,t)$ = The amount of water imported at reservoir r in month $t, \forall r,t$.

$\forall t$ let $\ell(t)$ = The Vector whose r^{th} element is $\ell(r,t) \forall r$

Deterministic Problem #1

Given fixed values for the parameters, find values for the variables that satisfy the following problem:

$$\min \sum_{t=A}^{B} \left\{ \sum_{c=1}^{C} k(c)\ f(c,t) + \sum_{r=1}^{R} p(r,t)\ w(r,t) \right\}$$

s.t.

$$w(r,t) + i(r,t) - d(r,t) + \ell(r,t-1) - \ell(r,t)$$

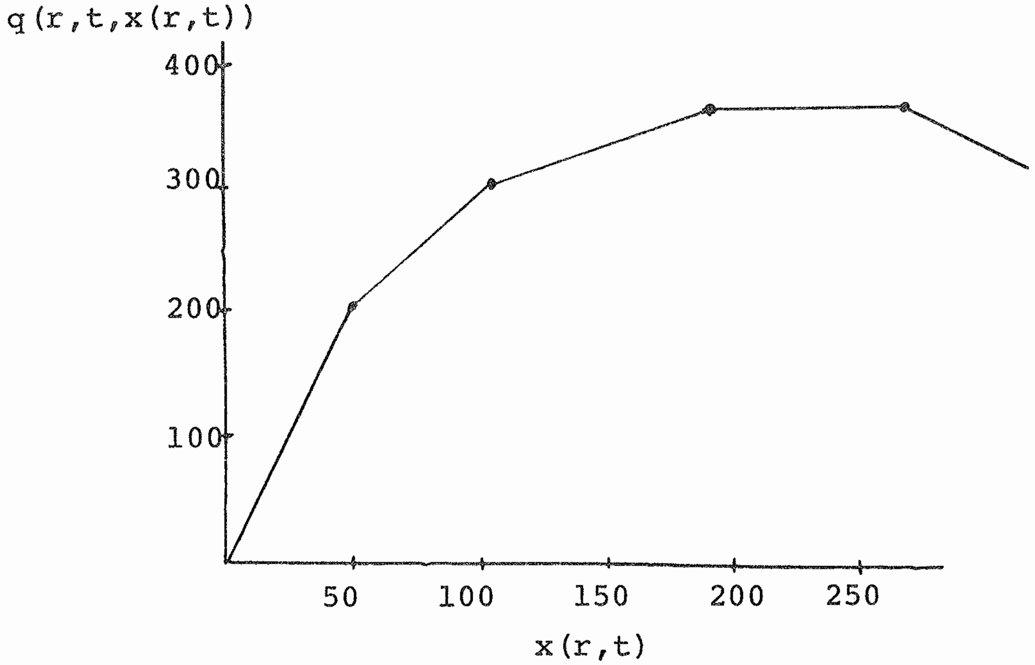
$$+ \sum_{C \in T(r)} f(c,t) - \sum_{C \in F(r)} f(c,t) = 0 \quad \forall r,t$$

$$0 \leq \ell(r,t) \leq s(r) \quad \forall r,t$$

$$0 \leq f(c,t) \leq u(c) \quad \forall c,t$$

$$0 \leq w(r,t) \leq m(r,t) \ \forall r,t$$

Problem one is a linear problem that may be directly translated into network form and solved by the out-of-kilter algorithm previously described. However, it may be the case that the above problem has no solution, that is, that the existing water system cannot meet all the specified future demands. In these cases a policy is needed that will make "best" use of the available resources. One possible method is to reward the releasing for use of water at a reservoir. Let x(r,t) be a new system variable denoting the amount of water released for consumptive use at reservoir r in month t for all r,t and subject to the constraint $0 \le x(r,t) \le d(r,t)$. Let $q\big(r,t,x(r,t)\big)$ denote the return or reward for providing x(r,t) units of water at reservoir r in month t for all r,t. Thus the return functions $q\big(r,t,x(r,t)\big)$ are new system parameters. In order to remain within the confines of the linear network formulation the $q\big(r,t,x(r,t)\big)$ must be piecewise linear and concave in x(r,t). That is the graph of $q\big(r,t,x(r,t)\big)$ must consist of linear segments with successively decreasing slope. An example of an allowable piecewise-linear and concave form for $q\big(r,t,x(r,t)\big)$ consisting of four linear pieces is as follows.



Return functions of this sort may be used to insure that as much of the demands will be satisfied as possible. In order to accomplish this a large positive per unit return is assigned to each unit of demand satisfied. Let M be a large positive number and define $q\big(r,t,x(r,t)\big) = M\ x(r,t)$ where $0 \le x(r,t) \le d(r,t)$. If M is chosen large enough, it will always be profitable to supply unfullfilled demands if possible. Feasibility of the problem may also be affected by large surpluses of water, hence it is also necessary to allow dumping or spillage of water at each reservoir if we are to insure feasibility. Since dumping consists of releasing more water than the demand, the upper bound on $x(r,t)$ is removed for all r and t. The penalty or cost of dumping may be incorporated into the $q\big(r,t,x(r,t)\big)$ with an additional linear segment having negative slope. Hence $q\big(r,t,x(r,t)\big)$ would take on a form of the type

$$q(r,t,x(r,t)) = \begin{cases} M \cdot x(r,t) & \text{for } 0 \leq x(r,t) \leq d(r,t) \\ -\,P[x(r,t) - d(r,t)] + M\,d(r,t) & \text{for } x(r,t) \geq d(r,t) \end{cases}$$

where $0 \leq x(r,t)$ and P is the per-unit penalty for dumping. Of course, other forms are possible for the return functions. For instance, there might be a stepwise decreasing marginal return for water at a reservoir. A graph of a return function of this sort is shown on the previous page.

Incorporating the additional system variables $x(r,t)\ \forall r,t$ and the additional system parameters $q\big(r,t,x(r,t)\big)\forall r,t$ the problem can be reformulated as follows:

Deterministic Problem #2

Given fixed values for the parameters, find values for the variables that satisfy the following problem.

$$\min \sum_{t=A}^{B} \left\{ \sum_{c=1}^{C} k(c)\ f(c,t) + \sum_{r=1}^{R} p(r,t)w(r,t) - \sum_{r=1}^{R} q\big(r,t,x(r,t)\big) \right\}$$

s.t.

$$w(r,t) + i(r,t) - x(r,t) + \ell(r,t-1) - \ell(r,t) + \sum_{c \in T(r)} f(c,t) - \sum_{c \in F(r)} f(c,t) = 0 \quad \forall r,t$$

$$0 \leq \ell(r,t) \leq s(r) \quad \forall r,t$$

$$0 \leq f(c,t) \leq u(c) \quad \forall c,t$$

$$0 \leq w(r,t) \leq m(r,t) \quad \forall r,t$$

$$0 \leq x(r,t) \quad \forall r,t$$

Note that the $d(r,t)$ no longer explicitly appear in the formulation. They may be present in the $q\big(r,t,x(r,t)\big)$ or there may be no explicit $d(r,t)$.

It is evident that problem 2 has a feasible solution consisting of the following:

$$\begin{aligned} \ell(r,t) &= \ell(0,t) \quad \forall\, r,t \\ f(c,t) &= \quad 0 \quad\;\; \forall\, c,t \\ w(r,t) &= \quad 0 \quad\;\; \forall\, r,t \\ x(r,t) &= i(r,t) \quad \forall\, r,t \end{aligned}$$

If the $q(r,t,x(r,t))$ are piecewise linear and concave, then problem 2 can be solved as a linear network problem. It may be characterized as a deterministic problem with perfect information since the $q(r,t,x(r,t))$ are deterministic and known for all r,t. This is equivalent to having perfect knowledge of all inflows and demands.

C. Definition and Use of Benefit Functions

In practice, problem 2 has several deficiencies as a model for the operation of a multireservoir system. Most important of these is the need for perfect information. This usually involves the assumption that the past record of rainfall and demands can be used to exactly predict the future rainfall and demands. Another deficiency is the large size of the optimization problem when a number of months are considered simultaneously. Problem 2 assumes perfect information of not just one month, but for months A through B. Although we initially obtain a solution for months A through B, deviations in rainfall and demands from the assumed values will often yield our solution suboptimal and perhaps even infeasible after a few months. This means another optimization needs to be performed and the results from the previous optimization might as well be thrown out. Quite naturally the question arises, why not optimize over one month at a time, since the solution for most later months will be updated anyway. The response, of course, is that some account must be taken of the future water requirements. To overcome the need for including a large number of months in the optimization, the idea arises of using a benefit function to express the future value of stored water.

The value of the objective function in problem 2 will be referred to as the cost of operating the multireservoir system for months A through B. For all t let $b(t,\ell(t))$ be a function with the following property: $b(t,\ell(t))$ equals the optimal cost of operating the multireservoir system for months $t + 1$ through B with initial storages in month $t + 1$ equal to zero, minus the optimal cost of operating the water system for the same interval of time with initial storages $\ell(t)$ in month $t + 1$. That is $b(t,\ell(t))$ is the amount of future cost that may be saved by having initial storages $\ell(t)$ in month $t + 1$ instead of having all reservoirs dry. The $b(t,\ell(t))$ will be more precisely defined subsequently.

By use of the function $b(A,\ell(A))$ an optimal policy for months A through B can be determined by optimizing over month A and then optimizing over months A + 1 through B. The important feature here is that in a one-month optimization a policy can be found for month A that will be part of an optimal policy for months A through B. The following development shows that this can be done.

Let 2b designate the multireservoir problem 2, but for months A + 1 through B (instead of A through B). Let $K(2b,z)$ denote the optimal cost for 2b given initial storages $\ell(A) = z$ in month A + 1. Define the function $b(A,z)$ by $b(A,z) = K(2b,0) - K(2b,z)$ for all z.

Let 2a designate the multireservoir problem for month A (i.e. for months A through A instead of A through B) with the alteration that the term $-b(A,\ell(A))$ is added to the objective function (except where otherwise indicated). Let $K(2a)$ denote the optimal cost for 2a. Let $K(2a,z)$ denote the optimal cost for 2a when $\ell(A)$ is set equal to z, a vector. Hence $K(2a) = \min_z K(2a,z)$. Let 2a' denote 2a with the alteration that the term $-b(A,\ell(A))$ is omitted from the objective function. Let $K(2a')$ denote the optimal cost for 2a' and $K(2a',z)$ denote the optimal cost for 2a' when $\ell(A)$, the final storage in month A, is set equal to z.

Recall that 2 designates the multireservoir problem for months A through B. Let $K(2)$ denote the optimal cost for 2. Let $K(2,z)$ denote the optimal cost for 2 when $\ell(A)$ is set equal to z. Then $K(2) = \min_z K(2,z)$. The solutions to 2a and 2b can be combined to form a solution to 2 where $\ell(A)$ is determined in 2a. Let $K(2d)$ denote the cost of this solution to 2.

The definitions just given can be summarized on the following table:

Problem	2a	2a'	2b	2
Months	A	A	A + 1 thru B	A thru B
Optimal Cost	$K(2a)$	$K(2a')$		$K(2)$
Optimal Cost given $\ell(A)$, the final storage in Month A equals Z.	$K(2a,z)$	$K(2a',z)$	$K(2b,z)$	$K(2,z)$

Several properties of the above defined quantities will now be presented. The objective is to show that $K(2d) = K(2)$, that is an optimal solution to 2 can be found by sequentially solving 2a and 2b.

Properties:

i. $b(A,0) = 0$. This follows directly from the definition, $b(A,0) = K(2b,0) - K(2b,0) = 0$.

ii. $K(2,z) = K(2a',z) + K(2b,z)$. This is apparent since fixing $\ell(A)$ at z decomposes 2 into two problems, one being problem 2a' with $\ell(A)$ fixed at z and the other being problem 2b with $\ell(A)$ fixed at z.

iii. $K(2) = \min_z K(2,z) = \min_z \; K(2a',z) + K(2b,z)$ from ii.

iv. $K(2d) = K(2a) + K(2b,z') + b(A,z')$ where z' is the value of $\ell(A)$ from the optimization K(2a). This follows directly from the definition of K(2d). The term $b(A,z')$ is present to offset the term $-b(A,z')$ in K(2a). This is necessary since there is no $-b(A,z)$ term in the objective function for 2.

v. $K(2a,z) = K(2a',z) - b(A,z)$. This follows since $b(A,z)$ is a constant for each z.

vi. $K(2) = K(2d)$. The proof is as follows:

$K(2d) = K(2a) + K(2b,z') + b(A,z')$ where z' is the value of $\ell(A)$ from the optimization K(2a). This is property iv. above.

$K(2d) = K(2a) + K(2b,z') + K(2b,0) - K(2b,z')$, using the definitions of $b(A,z')$

$K(2d) = K(2a) + K(2b,0)$

$K(2d) = \min_z\{K(2a,z)\} + K(2b,0)$

$K(2d) = \min_z \{K(2a,z) + K(2b,0)\}$

$K(2d) = \min_z \{K(2a',z) - b(A,z) + K(2b,0)\}$ by property v

$K(2d) = \min_z \{K(2a',z) - K(2b,0) + K(2b,z) + K(2b,0)\}$ by definition of $b(A,z)$

$K(2d) = \min_z \{K(2a',z) + K(2b,z)\}$

$K(2d) = \min_z K(2,z)$ property ii

$K(2d) = K(2)$.

Hence a solution to problem 2a gives a policy for month A that is part of an optimal policy for problem 2. The benefit function $b(A,z)$ can be shown to be concave in z. (See Appendix)

The concavity of $b(A,z)$ is an important property when considering a method for solving problem 2a. It should be noted here that problem 2a, unlike problems 2b and 2, is not a linear network problem even when the $q(r,t,x(r,t))$ are piecewise linear and concave. The distinction is that $b(A,z)$ is not separable into terms each involving only one variable. Hence problem 2a is not directly amendable to an out-of-kilter algorithm solution. An out-of-kilter method of approximating a solution to 2a will be discussed later.

D. Computation of Benefit Functions

A method for computing the benefit functions will now be discussed. This method is based on a periodic trend in rainfall and demands, and the resulting periodic benefit functions. By definition $b(A,z) = K(2b,0) - K(2b,z)$, hence $b(A,z)$ could be found for any value of z by solving problem 2b with $\ell(A) = 0$ and then with $\ell(A) = z$. Since problem 2b encompasses a number of months this would be a cumbersome process. Instead we will propose computing a sequence of benefit functions using one-month networks at each stage. This sequence will converge to the desired benefit functions. In order to justify this procedure a few changes in the formulation of the problem will be necessary.

We will now assume that the return functions $q\langle r,t,x(r,t)\rangle$ are periodic in nature and examine the effect of this periodicity on the other elements in the water distribution system. This assumption is not necessary to the solution method, but it does reduce the amount of computation involved. For some positive integer p, assume $q(r,t,f) = q(r,t + p,f)$ for all r,t,f. It follows that optimal policies are, in a sense, periodic. Suppose we have an optimal policy for months A through A + k with initial storages z in month A. Call this policy #1. An optimal policy for months A + p through A + p + k with initial storages z in month A + p can be found by following the decisions of month A + i of policy #1 in month A + p + i for $0 \leq i \leq k$. This is evident since the optimization problems in question would be identical except for the variable names.

Two further assumptions will be made in order to establish the periodicity of the benefit functions. Let us assume the following.

i. There is a positive integer M such that maximum benefit can always be obtained from stored water within M months. It is not necessary for M to be known in the case where the return functions are periodic. The purpose of this assumption is to allow consideration of only a finite number of months when investigating a particular benefit function. The explicit value of M becomes important when the assumption of periodic return functions is abandoned. This will be discussed later.

ii. The time horizon is infinite. Alternatively we may assume that the horizon is finite with T months and only consider those $b(t,z)$ where $t \leq T - M$.

In light of the above assumptions we let B, the final month in problems 2b and 2, equal A + M. Consider M as fixed and A as a parameter. For any value of A, the problem 2b determines a benefit function $b(A,f)$. We will now show that the $b(A,z)$ are periodic with respect to A, i.e. that $b(A,z) = b(A + p,z)$ for all A,f where p is the period of the return functions. Let $K(2e,z)$ denote the optimal cost for months A + p + 1 through A + p + M given initial storages $\ell(A + p) = z$ in month A + p + 1. Suppose we have a policy for months A + 1 through A + M with initial storages z in month A + 1. This policy will have cost $K(2b,z)$. By applying the decisions in month A + i of this policy to month A + p + i for $1 \leq i \leq M$ we obtain an optimal policy for months A + p + 1 through A + p + M. Recall optimal policies were previously shown to be periodic in this respect. This policy will still have cost K(2b,z) ; hence K(2e,z) = K(2b,z). For z = 0 this yields K(2e,0) = K(2b,0). It follows that K(2b,0) - K(2b,z) = K(2e,0) - K(2e,z) and therefore b(A,z) = b(A + p,z). Hence b(A,z) is periodic.

We will now show that given a benefit function the benefit function for the previous month can be found by parametrically solving a one-month optimization problem. Begin with a known benefit function b(A,z). Consider the one-month optimization problem.

$$c\big(\ell(A-1)\big) = \min \left\{ \sum_{c=1}^{C} k(c)\ f(c,A) + \sum_{r=1}^{R} p(r,A)\ w(r,A) - \sum_{r=1}^{R} q\big(r,A,x(r,A)\big) - b\big(A,\ell(A)\big) \right.$$

s.t.

$$w(r,A) + i(r,A) - x(r,A) + \ell(r,A-1) - \ell(r,A) + \sum_{c \in T(r)} f(c,A) - \sum_{c \in F(r)} f(c,A) = 0 \quad \forall r$$

$$0 \leq \ell(r,A) \leq s(r) \quad \forall r$$

$$0 \leq f(c,A) \leq u(c) \quad \forall c$$

$$0 \leq w(r,A) \leq m(r,A) \quad \forall r$$

$$0 \leq x(r,A) \quad \forall r$$

The optimal cost $c\big(\ell(A-1)\big)$ can be expressed as $c\big(\ell(A-1)\big) = c\big(A,\ell(A-1)\big) - b\big(A,\ell(A)\big)$ where $c\big(A,\ell(A-1)\big)$ is the cost incurred during month A and $b\big(A,\ell(A)\big)$ is the cost savings realizable over the next M months from final storages $\ell(A)$ as compared to zero final storage. Let us consider the difference c(0) - c(z). Let z(1) denote the optimal value of $\ell(A)$ in c(0) and z(2) denote the optimal value of $\ell(A)$ in c(z). Then

$$\begin{aligned} c(0) - c(z) &= c(A,0) - b\big(A,z(1)\big) - c(A,z) + b\big(A,z(2)\big) \\ &= c(A,0) - c(A,z) + b\big(A,z(2)\big) - b\big(A,z(1)\big). \end{aligned}$$

C(A,0) - c(A,z) represents the cost savings during month A due to initial storages z. $b\big(A,z(2)\big) - b\big(A,z(1)\big)$ is the cost savings realized over months A + 1 through A + M for initial storages in month A + 1 equal to z(2) as opposed to z(1). Hence c(0) - c(z) is the cost savings over months A through A + M due to initial storages z. By assumption this same cost savings can be realized in months A through A + M -1 and equals, by definition, b(A - 1,z). Hence b(A - 1,z) = c(0) - c(z) and for any z,b(A - 1,z) can be found by solving two one-month optimization problems once b(A,z) is known. We can fully define b(A - 1,z)by parametrically varying the initial reservoir levels and solving a one-month optimization problem for each possible r-tuple of initial storages in month A.

Since the benefit functions are periodic, there are only p benefit functions to be found. Once one of these is known, the other p - 1 may be found iteratively using parametive variations in p - 1 one-month optimizations as described in the previous paragraph. In order to find a benefit function that will serve as a starting point for calculating the remaining benefit functions, we make use of the assumption that maximum benefit can be derived from stored water within M months. For some integer T, where T > M, define b(T,z) = 0 for all z. Then it is possible to iteratively find b(t,z) for t = T - 1, T - 2, T - 3 . . . By our assumption when t ≤ T - M, b(t - p,z) = b(t,z) for all z. We can terminate the iterative process with all b(t,z) determined when t = T - M - p.

It may not be necessary to perform M + p iterations to determine all the b(t,z). The iterative process of finding b(t,z) for t = T - 1, T - 2, T - 3,. . . will be said to converge when for some A, b(A,z) = b(A - p,z) for all z. Once this occurs the process can be terminated since b(t,z) = b(t + p,z) for all z and t ≤ A. This can be shown as follows:

1. for t = A-1, the one-month optimization problem that finds b(A - 1,z) knowing b(A,z) is identical with the problem that found b(A + p -1,z) knowing b(A + p,z). Hence b(A - 1,z) = b(A + p - 1,z) for all z.

2. let t* be an integer, t* ≤ A and assume b(t,z) = b(t + p,z) for all z for t* ≤ t ≤ A. The one-month optimization problem that finds b(t* - 1,z) knowing b(t*,z) is identical with the problem that found b(t* + p - 1,z) knowing b(t* + p,z). Hence b(t* - 1,z) = b(t* + p - 1,z) for all p.

3. Hence by induction b(t,z) = b(t + p,z) for all z and t ≤ A.

As we have seen we are guaranteed convergence in the iterative computation of the b(t,z) as described above given the assumption that maximum benefit can be derived from stored water within M months. It may be possible to speed up the convergence by making a judicious choice for b(T,z) instead of b(T,z) = 0. No guarantee of improved convergence can be made, but at most M + p benefit function would have to be computed as before. In practice, the computational savings may be considerable.

When the net demands are not periodic in nature, the return functions q(r,t,x(r,t)) will also not be periodic. Hence the benefit functions b(t,z) will not be periodic. In this case the explicit value of M becomes important because it determines how far into the future we must look in order to determine the benefit functions. If we begin with some month T, and an assumption for $b(T,z)$ then $b(t,z)$ for $t = T - 1, T - 2, T - 3, \ldots$ can be iteratively calculated as previously described. For t such that $t \leq T - M$, $b(T,z)$ will be accurate. By making a judicious choice for $b(T,z)$ instead of $b(T,z) = 0$ it would be possible to reduce the number of iterations required before the $b(t,z)$ are accurate.

E. Benefit Functions for Stochastic Demands

In the previous developments the return functions $q\big(r,t,x(r,t)\big)$ were deterministic in nature. We will now examine how the procedures can be altered to accommodate a form of stochastic return functions. For each time period t, let there be N possible sets of return functions designated by $q\big(r,t,x(r,t),k\big)$ for $k = 1, 2 \ldots, N$. For each k let $P(t,k)$ be the probability of return function $q\big(r,t,x(r,t),k\big)$ $\forall r,t$ thus $P(t,k) \geq 0$ and $\sum_{k=1}^{N} P(t,k) = 1$. The assumption of periodic return functions is retained; for some positive integer p assume $q(r,t,f,k) = q(r,t + p,f,k)$ for all r,t,f,k and $P(t,k) = P(p + t,k)$ for all t,k. We also retain the assumption that the return functions are concave.

In solving a multireservoir problem it will be assumed that when the time arrives to make decisions for a particular month, the return functions for that month will be known, however those for succeeding months will be known only in stochastic form. Thus, in finding a one-month policy that will be part of an optimal policy for the future, the one-month optimization problem will be deterministic in nature, but will involve a benefit function that takes into account the stochastic knowledge of future return functions.

Define $b(A,z,k)$ to be the amount of future cost that may be saved in months $A + 1$ through $A + M$ by having initial storages z in month $A + 1$ instead of initial storages zero assuming the following:

i. the return function in month $A + 1$ is $q\big(r,A + 1, x(r,A + 1),k\big)$

ii. the return functions for months $A + 2$ through $A + M$ are stochastic in nature as described earlier

iii. in future months we will have knowledge of the return functions for a month just prior to making the decisions for that month.

We will then define $b(A,z)$ by $b(A,z) = \sum_{k=1}^{N} P(A + 1,k)\, b(A,z,k)$. Thus $b(A,z)$ is the expected cost savings in months $A + 1$ through $A + M$ due to having initial storages z (instead of zero) in month $A + 1$, subject to assumptions

i. the return functions for months A + 1 through A + M are stochastic in nature

ii. in all months we will have knowledge of the return functions for a month just prior to making the decisions for that month.

Next we will develop a recursive algorithm for computing the stochastic benefit function b(A,z). It will be very similar to the algorithm for finding the deterministic benefit functions. The first step will be to show, as before, that given a benefit function, the benefit function for the previous month can be found by parametrically solving a one-month optimization problem. Begin with a known stochastic benefit function b(A,z). Consider the one-month optimization problem

$$c\Big(\ell(A-1),k\Big) = \min\left\{ \sum_{c=1} k(c)\; f(c,A) + \sum_{r=1}^{R} p(r,A)\; w(r,A) - \sum_{r=1}^{R} q\Big(r,A,x(r,A),k\Big) - b\Big(A,\ell(A)\Big)\right\}$$

s.t.

$$w(r,A) + i(r,A) - x(r,A) + \ell(r,A-1) - \ell(r,A) \sum_{c \in T(r)} f(c,A) - \sum_{c \in F(r)} f(c,A) = 0 \quad \forall r$$

$$0 \le \ell(r,A) \le s(r) \quad \forall r$$

$$0 \le f(c,A) \le u(c) \quad \forall c$$

$$0 \le w(r,A) \le m(r,A) \quad \forall r$$

$$0 \le x(r,A) \quad \forall r$$

In the same manner as was done for the deterministic case c(0,k) − c(0,z) can be shown to be the cost savings realizable over months A through A + M − 1 due to having initial storages z rather than initial storages subject to assumptions i, ii, and iii on page . Hence b(A,z,k) = c(0,k) − c(0,z) for any z and k. Thus knowing b(A,z) we can fully define b(A − 1,z,k) for k = 1, 2, . . . , N. Then $b(A-1,z) = \sum_{k=1}^{N} p(A,k)\; b(A-1,z,k)$.

As in the deterministic case we begin with an integer T when T > M, and define b(T,z) = 0. Then we iteratively find b(t,z) for t = T − 1, T − 2, T − 3, ... Using the assumption that maximum benefit can be obtained from stored water within M months, we conclude that when t ≤ T − M, b(t − p,z) = b(t,z) for all z. Hence, we can terminate the process with all b(t,z) determined when t = T − M − p as before. As before the periodicity assumption is not a necessity in using this method. It just reduces the number of benefit functions to be found.

Note that the one-month optimization problem c(z,k) minimizes the sum of the cost incurred in month A minus $b(A,\ell(A))$, that is, it minimizes the cost incurred in month A minus the expected cost savings in months A + 1 through A + M due to having initial storages $\ell(A)$ in month A + 1 instead of initial storages zero. Since the expected cost of operating the system optimally in months A + 1 through A + M starting with initial storages zero in month A + 1 is a constant, it may be added to the objective function c(z,k) without changing the optimal policy. Once this is done the objective function is now the sum of the cost incurred in month A plus the expected cost to be incurred in months A + 1 through A + M. This is obviously a quantity it is desirable to minimize.

IV. A Revised Out-of-Kilter Algorithm

The multireservoir problem has been reduced to a large number of one-period problems containing nonseparable benefit functions. To handle problems of this sort, the out-of-kilter algorithm must be revised.

A. Optimality Conditions for the Network Formulation of the Multireservoir Problem

Let N be the number of nodes

A be the number of arcs

n be the index on nodes with range $1 \leq n \leq N$

k be an index on arcs with range $1 \leq k \leq A$

A_n be the set of all arcs directed from node n, for all n

B_n be the set of all arcs directed into node n, for all n

arc k be directed from node i_k to node j_k, for all k

l_k denote the lower bound on arc k, for all k

u_k denote the upper bound on arc k, for all k

f_k denote the flow on arc k, for all k

$F = \{f_k | \text{for all } k\}$

f_k^* denote a flow on arc k, for all k, such that $F^* = \{f_k^* | \text{for all } k\}$ is an optimal set of flows

$C(F)$ denote a convex function of F

$\pi = \{\pi_n | \text{for all } n\}$ be a set of unrestricted Lagrange multipliers, also referred to as node numbers.

$\delta = \{\delta_k | \text{for all } k\}$ be a set of nonnegative Lagrange multipliers

$\gamma = \{\gamma_k | \text{for all } k\}$ be a set of nonnegative Lagrange multipliers

F^*, π^*, δ^*, γ^* are used to denote an optimal set of flows and Lagrange multipliers

F', π', δ', γ' are used to denote a fixed set of flows and Lagrange multipliers

Δ_k denote a vector of length A with its kth element equal to Δ and all other elements zero.

The problem to be solved is as follows:

Find F to minimize C(F) subject to:

$$k_n(F) = \sum_{k\varepsilon A_n} f_k - \sum_{k\varepsilon B_n} f_k = 0 \quad \text{for all } n$$

$$g_k(F) = -f_k + \ell_k \leq 0 \quad \text{for all } k$$

$$m_k(F) = f_k - u_k \leq 0 \quad \text{for all } k$$

Since these constraints are linear, the constraint qualification is satisfied [Zangwill, p. 56].

Theorem 1.

A sufficient condition that a vector of flows F* be optimal for the stated problem is that F* satisfy the constraints and that there exists a vector of node numbers π^* such that

$$-\pi^*_{j_k} + \pi^*_{i_k} + \frac{\partial C(F^*)}{\partial f^{*-}_k} > 0 \Rightarrow f^*_k = \ell_k \text{ for all } k$$

and $$-\pi^*_{j_k} + \pi^*_{i_k} + \frac{\partial C(F^*)}{\partial f^{*+}_k} < 0 \Rightarrow f^*_k = u_k \text{ for all } k$$

Another theorem and several lemmas will be stated as preliminaries to proving this theorem.

Theorem 2.

A necessary and sufficient condition that F* be optimal for the problem is that F* satisfy the constraints and that there exist unrestricted Lagrange multipliers π^* and nonnegative Lagrange multipliers δ^* and γ^* such that $(F^*,\pi^*,\delta^*,\gamma^*)$ is a saddle point of the Lagrangian [Kungi and Krelle p. 63-70].

$$L(F,\pi,\delta,\gamma) = C(F) + \sum_{n=1}^{N} \pi_n \left(\sum_{k\varepsilon A_n} f_k - \sum_{k\varepsilon B_n} f_k \right)$$

$$+ \sum_{k=1}^{A} [\delta_k(\ell_k - f_k) + \gamma_k(f_k - u_k)].$$

Since each f_k appears twice in the conservation of flow constraints the Lagrangian may be rewritten as

$$L(F,\pi,\delta,\gamma) = C(F) + \sum_{k=1}^{A} [f_k(\pi_{i_k} - \pi_{j_k}) + \delta_k(\ell_k - f_k) + \gamma_k(f_k - u_k)]$$

The conditions for a saddle point are that

$$L(F^*,\pi^*,\delta^*,\gamma^*) \leq L(F,\pi^*,\delta^*,\gamma^*) \text{ for all feasible } F$$

and $L(F^*,\pi^*,\delta^*,\gamma^*) \geq L(F^*,\pi,\delta,\gamma)$ for all π, $\delta \geq 0$, $\gamma \geq 0$

Prior to stating and proving three lemmas, some additional notation needs to be developed. For all k define

$$s_k^+(F) = \frac{\partial C(F)}{\partial f_k^+} \quad \text{and} \quad s_k^-(F) = \frac{\partial C(F)}{\partial f_k^-}$$

and let $\delta_k = \max\{0,-\pi_{j_k}+\pi_{i_k}+s_k^-(F)\}$

and $\gamma_k = \max\{0,\pi_{j_k}-\pi_{i_k}-s_k^+(F)\}$

<u>Lemma 1:</u> $\delta_k>0 \Rightarrow \gamma_k = 0$. Also $\gamma_k>0 \Rightarrow \delta_k = 0$

Proof:

Suppose $\delta_k>0$ then $-\pi_{j_k} + \pi_{i_k} + s_k^-(F)>0$

Due to the convexity of $C(F)$, $s_k^+(F)\geq s_k^-(F)$ hence

$$-\pi_{j_k} + \pi_{i_k} + s_k^+(F) > 0$$

$$\pi_{j_k} - \pi_{i_k} - s_k^+(F) < 0$$

$$\text{hence} \quad \gamma_k = 0$$

A similar proof applies to the other part of Lemma 1.

<u>Lemma 2:</u> $-\delta_k + \gamma_k \geq \pi_{j_k} - \pi_{i_k} - s_k^+(F)$ and

$$-\delta_k + \gamma_k \leq \pi_{j_k} - \pi_{i_k} - s_k^-(F)$$

Proof:

Using Lemma 1 $-\delta_k + \gamma_k = \begin{cases} 0 \text{ if } \delta_k = \gamma_k = 0 \\ -\delta_k = \pi_{j_k}-\pi_{i_k}-s_k^-(F) \text{ if } \delta_k > 0 \\ \gamma_k = \pi_{j_k}-\pi_{i_k}-s_k^+(F) \text{ if } \gamma_k < 0 \end{cases}$

Case 1 $\delta_k = \gamma_k = 0$

Since $\gamma_k = 0, \ \pi_{j_k} - \pi_{i_k} - s_k^+(F) \leq 0$

hence $\delta_k - \gamma_k \geq \pi_{j_k} - \pi_{i_k} - s_k^+(F)$

Case 2 $\delta_k > 0$

$$-\delta_k + \gamma_k = -\delta_k = \pi_{j_k} - \pi_{i_k} - s_k^-(F)$$

but $s_k^-(F) \leq s_k^+(F)$ hence

$$-\delta_k + \gamma_k \geq \pi_{j_k} - \pi_{i_k} - s_k^+(F)$$

Case 3 $\gamma_k > 0$

$$-\delta_k + \gamma_k = \gamma_k = \pi_{j_k} - \pi_{i_k} - s_k^+(F)$$

A similar proof applies to the other part of Lemma 2.

Lemma 3: Let F' be a set of feasible flows, π' be a set of node numbers, δ' be defined by $\delta'_k = \max\{0, -\pi'_{j_k} + \pi'_{i_k} + s_k^-(F')\}$ for all k and γ' be defined by $\gamma'_k = \max\{0, \pi'_{j_k} - \pi'_{i_k} - s_k^+(F')\}$ for all k. Then for any set F of feasible flows

$$L(F', \pi', \delta', \gamma') \leq L(F, \pi', \delta', \gamma')$$

Proof:

Define S(F',F) to be a vector whose kth element is $s_k(F,F')$ where

$$s_k^+(F,F') = \begin{cases} s_k(F') & \text{if} \quad f_k > f'_k \\ s_k^-(F') & \text{if} \quad f_k < f'_k \\ 0 & \text{if} \quad f_k = f'_k \end{cases}$$

Due to the convexity of C(F)

$$C(F) \geq C(F') + S^T(F,F')(F-F') = C(F') + \sum_{k=1}^{A} s_k(F,F')(f_k - f'_k).$$

Now $L(F,\pi',\delta',\gamma') = C(F)$

$$+ \sum_{k=1}^{A} [f_k(\pi'_{i_k} - \pi'_{j_k}) + \delta'_k(\ell_k - f_k) + \gamma'_k(f_k - u_k)]$$

$$\geq C(F') + \sum_{k=1}^{A} s_k(F,F')(f_k - f'_k)$$

$$+ \sum_{k=1}^{A} [f_k(\pi'_{i_k} - \pi'_{j_k}) + \delta'_k(\ell_k - f_k) + \gamma'_k(f_k - u_k)]$$

$$\geq C(F') + \sum_{k=1}^{A} [f'_k(\pi'_{i_k} - \pi'_{j_k}) + \delta'_k(\ell_k - f'_k) + \gamma'_k(f'_k - u_k)]$$

$$+ \sum_{k=1}^{A} (f_k - f'_k)(s_k(F,F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k)$$

$$L(F,\pi',\delta',\gamma') \geq L(F',\pi',\delta',\gamma')$$

$$+ \sum_{k=1}^{A} (f_k - f'_k)(s_k(F,F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k)$$

Hence $L(F,\pi',\delta',\gamma') \geq L(F',\pi',\delta',\gamma')$ provided

$$\sum_{k=1}^{A} (f_k - f'_k)(s_k(F,F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k) \geq 0$$

This is satisfied if the following two conditions hold:

$$f_k - f'_k > 0 \Rightarrow s_k(F,F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k \geq 0$$

and $$f_k - f'_k < 0 \Rightarrow s_k(F,F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k \leq 0$$

Due to the definition of $s_k(F,F')$ these conditions may be restated as

$$s^+_k(F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k \geq 0$$

and $$s^-_k(F') - \pi'_{j_k} + \pi'_{i_k} - \delta'_k + \gamma'_k \leq 0$$

or $$-\delta'_k + \gamma'_k \geq \pi'_{j_k} - \pi'_{i_k} - s^+_k(F')$$

and $$-\delta'_k + \gamma'_k \leq \pi'_{j_k} - \pi'_{i_k} - s^-_k(F')$$

These conditions were verified in Lemma 2.

Hence $L(F',\pi',\delta',\gamma') \leq L(F,\pi',\delta',\gamma')$

Proof of theorem 1:

It will be shown that if F* is feasible, and F*, π* satisfy the conditions

$$-\pi^*_{j_k} + \pi^*_{i_k} + s^-_k(F^*) > 0 \Rightarrow f^*_k = \ell_k \quad \text{for all } k$$

and $$-\pi^*_{j_k} + \pi^*_{i_k} + s^+_k(F^*) < 0 \Rightarrow f^*_k = u_k \quad \text{for all } k$$

then $L(F^*,\pi,\delta,\gamma) \leq L(F^*,\pi^*,\delta^*,\gamma^*) \leq L(F,\pi^*,\delta^*,\gamma^*)$

where δ^* and γ^* are defined in the previous manner.

Part 1 $\quad L(F^*,\pi^*,\delta^*,\gamma^*) \leq L(F,\pi^*,\delta^*,\gamma^*)$ by lemma 3.

Part 2

Using the form of the Lagrangian

$$L(F,\pi,\delta,\gamma) = C(F) + \sum_{n=1}^{N} \pi_n \Big(\sum_{k\varepsilon A_n} f_k - \sum_{k\varepsilon B_n} f_k \Big)$$

$$+ \sum_{k=1}^{A} [\delta_k(\ell_k - f_k) + \gamma_k(f_k - u_k)]$$

$$L(F^*,\pi^*,\delta^*,\gamma^*) - L(F^*,\pi,\delta,\gamma)$$

$$= \sum_{n=1}^{N} (\pi^*_n - \pi_n) \Big(\sum_{k\varepsilon A_n} f^*_k - \sum_{k\varepsilon B_n} f^*_k \Big)$$

$$+ \sum_{k=1}^{A} [(\delta^*_k - \delta_k)(\ell_k - f^*_k) + (\gamma^*_k - \gamma_k)(f^*_k - u_k)]$$

Since the constraints are satisfied

$$\sum_{k\varepsilon A_n} f^*_k - \sum_{k\varepsilon B_n} f^*_k = 0 \quad \text{for all } n.$$

Also $\ell_k - f^*_k \leq 0$ and $\delta_k \geq 0$ hence $-\delta_k(\ell_k - f^*_k) \geq 0$.

Also $f^*_k - u_k \leq 0$ and $\gamma_k \geq 0$ hence $-\gamma_k(f^*_k - u_k) \geq 0$.

Hence for $L(F^*,\pi^*,\delta^*,\gamma^*) - L(F^*,\pi,\delta,\gamma) \geq 0$ it is sufficient that

$$\delta^*_k > 0 \Rightarrow f^*_k = \ell_k \quad \text{for all } k$$

and $$\gamma^*_k > 0 \Rightarrow f^*_k = u_k \quad \text{for all } k.$$

These conditions may be expressed as

$$-\pi^*_{j_k} + \pi^*_{i_k} + s^-_k(F^*) > 0 \Longrightarrow f^*_k = \ell_k \qquad \text{for all } k$$

and

$$-\pi^*_{j_k} + \pi^*_{i_k} + s^+_k(F^*) < 0 \Longrightarrow f^*_k = u_k \qquad \text{for all } k.$$

These were the conditions given in the theorem, hence theorem 1 is proved.

B. The Δ Problem and a Revised Out-of-Kilter Algorithm

The Δ problem is the minimization problem where the flow in each arc must be an integer multiple of some number Δ. That is $f_k = \Delta m_k$ where m_k is an integer for all k. It is tempting to modify the sufficient conditions for the continuous problem by replacing

$$\frac{\partial C(F^*)}{\partial f^{*+}_k} \quad \text{with} \quad \frac{C(F^*+\Delta_k) - C(F^*)}{\Delta}$$

and

$$\frac{\partial C(F^*)}{\partial f^{*-}_k} \quad \text{with} \quad \frac{C(F^*) - C(F^*-\Delta_k)}{\Delta}$$

where Δ_k is the vector of dimension A with all elements zero except for the kth element which is Δ. The difficulty here is that a flow change of size Δ on arc k may greatly change

$$\frac{\partial C(F)}{\partial f_j}$$

for $j \neq k$. Thus a flow change may be desirable (i.e. cost-saving) even though the modified conditions are satisfied. Similarly the modified conditions may indicate a flow change is desirable when actually, due to changing partial derivatives, it is undesirable. These difficulties dictate changes in the out-of-kilter algorithm. Fortunately the multireservoir problem can be modelled by a very special form of network to which algorithmic changes can be applied more easily than in the general case.

Recall that in the formulation of the multireservoir problem the objective function C(F) consists of separable terms to which a nonseparable benefit function is added. The only arcs whose flows are involved in the benefit function are the arcs representing final storage in the reservoirs. There is one such arc for each reservoir and all these arcs terminate at the source node. Arcs whose flows enter into the objective function in a separable manner will be referred to as separable arcs. Arcs whose flows represent final reservoir storage and thus are present in the objective function in a nonseparable manner will be referred to as interactive arcs. In addition it will be desirable at times to add to the network arcs whose purpose is to transfer final storage from one reservoir to another. These will be referred to as transfer arcs. This algorithm is based on the revised out-of-kilter algorithm for separable convex network problems [Jensen, 13].

Let $d_k^+(F) = C(F+\Delta_k) - C(F) - \pi_{j_k} + \pi_{i_k}$

$d_k^-(F) = C(F) - C(F-\Delta_k) - \pi_{j_k} + \pi_{i_k}$

for all F and all separable or interactive arcs.

The $d_k^+(F)$ and $d_k^-(F)$ will be called modified costs on arc k. The optimality conditions for   problem will be

$$d_k^+(F^*) < 0 \Rightarrow f_k^* = u_k$$

and $$d_k^-(F^*) > 0 \Rightarrow f_k^* = \ell_k \quad \text{for all } k.$$

However as mentioned previously even when these conditions are satisfied a flow change may be desirable. An arc k is said to be "in-kilter" if $\ell_k \leq f_k \leq u_k$ and both the above conditions hold. An arc that is not in-kilter is said to be "out-of-kilter." Also note that due to convexity $d_k^-(F) \leq d_k^+(F)$. If an arc is out-of-kilter since $d_k^+(F) < 0$ but $f_k < u_k$, the two possible remedies are to increase f_k or to change the node numbers so that $\pi_{j_k} - \pi_{i_k}$ is decreased. If an arc is out-of-kilter since $d_k^-(F) > 0$ but $f_k > \ell_k$, the two possible remedies are to decrease f_k or to change the node numbers so that $\pi_{j_k} - \pi_{i_k}$ is increased.

The transfer arcs are handled in a similar manner except that $d^+(F)$ and $d^-(F)$ are defined as follows. Suppose nodes n_1 and n_2 represent reservoirs. Let k_1 represent the final storage arc for reservoir n_1 and k_2 represent the final storage arc for reservoir n_2. Define arc A+1 in the following manner

$$i_{A+1} = n_2$$

$$j_{A+1} = n_1$$

$$\ell_{A+1} = \max\{-f_{k_1}, f_{k_2} - u_{k_2}\}$$

$$u_{A+1} = \min\{u_{k_1} - f_{k_1}, f_{k_2}\}$$

$$f_{A+1} = 0$$

$$d^+_{A+1}(F) = C(F + \Delta_{k_1} - \Delta_{k_2}) - C(F) - \pi_{j_{A+1}} + \pi_{i_{A+1}}$$

$$d^-_{A+1}(F) = C(F) - C(F - \Delta_{k_1} + \Delta_{k_2}) - \pi_{j_{A+1}} + \pi_{i_{A+1}}$$

If arc A+1 is out-of-kilter since $d^+_{A+1}(F) < 0$ but $f_{A+1} < u_{A+1}$ the two possible remedies are to decrease f_{k_2} while increasing f_{k_1} or to change the node numbers so that $\pi_{j_{A+1}} - \pi_{i_{A+1}}$ is decreased. If arc A+1 is out-of-kilter since $d^-_{A+1}(F) > 0$ but $f_{A+1} > \ell_{A+1}$ the two possible remedies are to decrease f_{k_1} while increasing f_{k_2} or to change the node numbers so that $\pi_{j_{A+1}} - \pi_{i_{A+1}}$ is increased.

If there are R reservoirs, there are

$$\binom{R}{2} = \frac{R!}{2!(R-2)!}$$

possible transfer arcs. For convenience these should be numbered from one to $\binom{R}{2}$ in an arbitrary manner.

Algorithm for the Δ Problem

1. Start with an arbitrary assignment of node numbers and an arbitrary set of flows. The flows need not be feasible, but must satisfy conservation of flow at each node. No transfer arcs have been added to the network yet.

2. Calculate $d^+_k(F)$ and $d^-_k(F)$ for each arc. Set $\alpha = 1$.

3. Check whether arc α is in kilter. If it is and $\alpha = A$, set $\alpha = 1$ and go to step 7. If arc α is in kilter and $\alpha < A$ increase α by one and repeat this step. If arc α is out-of-kilter determine whether f_α should be increased or decreased. If it is to be increased let s be node j_α and t be node i_α. If it is to be decreased, let s be node i_α and t be node j_α. If α is an interactive arc, recalculate the modified costs as follows. Let Δ_α represent a vector of length A with element number α equal to Δ and all other elements zero. If the flow on arc α is to be increased calculate $d^+_k(F-\Delta_\alpha)$ and $d^-_k(F+\Delta_\alpha)$. If the flow on arc α is to be decreased calculate $d^+_K(F-\Delta_\alpha)$ and $d^-_k(F-\Delta_\alpha)$. In the labeling algorithm described later these will be referred to as "the appropriate d^+_k and d^-_k". The argument F, $F+\Delta_\alpha$, or $F-\Delta_\alpha$ will be omitted for simplicity. Go on to the next step.

4. Using the labeling algorithm search for a path from s to t in which the flow may be increased. If such a path is found go to step 5. If no such path is found go to step 6.

5. The labeling algorithm has found a path from node s to node t along which flow may be increased. This path together with the out-of-kilter arc forms a cycle. Increase the flow by Δ in those arcs which appear in the forward direction in the cycle. Decrease the flow by Δ in those arcs which appear in the reverse direction in the cycle. Delete all node labels from the labeling algorithm and return to step 2.

6. The labeling algorithm could not find a path from node s to node t. Call the set of labeled nodes X and the set of unlabeled nodes $\bar{X}$. Change the node numbers with the node number change algorithm. Without deleting the labels from the labeling algorithm, recalculate the appropriate d_k^+ and d_k^- for those arcs where changes would occur. If arc α is now in kilter and $\alpha < A$ increase α by one and go to step 3. If arc α is now in kilter and $\alpha = A$, set $\alpha = 1$ and go to step 7. If arc α is still out-of-kilter go to step 4.

7. Form transfer arc α , thus giving the network A+1 arcs. If transfer arc α is in kilter and $\alpha = \binom{R}{2}$, set $\alpha = 1$ and go to step 11. If transfer arc α is in kilter and $\alpha < \binom{R}{2}$, delete transfer arc α, increase α by one, and repeat this step. If transfer arc α is out-of-kilter determine whether f_{A+1} whould be increased or decreased. If it is to be increased let s be node i_{A+1} and t be node j_{A+1}. If it to be decreased let s be node j_{A+1} and t be node i_{A+1}. No further recalculations of the modified costs are needed at this point since interactive arcs other than arcs k_1 and k_2 will not be allowed in the flow augmenting cycle.

8. Temporarily delete all interactive arcs from the network. Using the labeling algorithm search for a path from node s to node t in which the flow may be increased. If such a path is found go to step 9. If no such path is found go to step 10.

9. The labeling algorithm has found a path from node s to node t along which flow may be increased. This path together with the transfer arc forms a cycle. Increase the flow by Δ in those arcs which appear in the forward direction in the cycle. Decrease the flow by Δ in those arcs which appear in the reverse direction in the cycle. Delete all node labels from the labeling algorithm. Restore interactive arcs to the network.
If $f_{A+1} = -\Delta$ decrease f_{k_1} by Δ and increase f_{k_2} by Δ.
If $f_{A+1} = \Delta$ increase f_{k_1} by Δ and decrease f_{k_2} by Δ.
Set $f_{A+1} = 0$ and return to step 7.

10. The labeling algorithm could not find a path from node s to node t. Call the set of labeled nodes X and the

set of unlabeled nodes $\bar{X}$. Attempt to change the node numbers with the node number change algorithm. If this is not possible delete the transfer arc, restore the interactive arcs to the network, set $\alpha = 1$, and go to step 2. If the node number change algorithm succeeds, change the node numbers. Without deleting the labels from the labeling algorithm, recalculate the appropriate d_k^+ and d_k^- for those arcs where changes would occur. If transfer arc α is now in kilter and $\alpha < \binom{R}{2}$ increase α by one and go to step 7. If transfer arc α is in kilter and $\alpha = \binom{R}{2}$ go to step 11. If transfer arc α is still out-of-kilter to to step 8.

11. If all arcs (separable, interactive, and transfer) were in kilter without aid of flow or node number changes since step 2 was last encountered, go to step 12. If any flow changes have occurred since step 2 was last encountered, set $\alpha = 1$ and go to step 2. If only node number changes have occurred since step 2 was last encountered, compare the present node numbers with the node numbers the previous time step 2 was encountered. If the two sets of node numbers are equal go to step 12, otherwise set $\alpha = 1$ and go to step 2.

12. Terminate the algorithm. The present solution cannot be improved by this algorithm.

Labeling Algorithm

This algorithm seeks to find a path from node s to node t along which the flow can be increased by Δ. This is accomplished by increasing the flow by Δ. In the process no arcs in the path that are in-kilter are allowed to become out-of-kilter. Also no arcs in the path that are out-of-kilter are allowed to become more out-of-kilter. That is, if an arc k has the appropriate $d_k^+ < 0$ but $f_k < u_k$, then it is not allowable to decrease f_k. Similarly if an arc k has the appropriate $d_k^- > 0$ but $f_k > \ell_k$, then it is not allowable to increase f_k. Recall that the appropriate d_k^+ and d_k^- refers to the fact that d_k^+ and d_k^- for interactive arcs may have been updated in anticipation of flow changes in other interactive arcs. In the course of the labeling algorithm, further updating of the modified costs for interactive arcs may be necessary.

In the algorithm a node will be labeled when a path has been found from node s to that node along which flow can be increased by Δ. Once a node has been labeled it can be scanned. A labeled node is scanned once all branches which enter or leave the labeled node have been investigated for possible extension of the path from s. The labeling algorithm will now be described.

1. Label node s.

2. Find a labeled but unscanned node, say node i. If there are no such nodes, the algorithm terminates without finding a path from s to t. This is called nonbreakthrough.

   For each arc k with $i_k = i$

   a. If node j_k is already labeled, go on to the next arc.

   b. Node j_k is not labeled. Check the following conditions

      i. $f_k + \Delta > u_k$

      ii. $f_k \geq \ell_k$ and $d_k^+ > 0$.

   If either condition holds, go on to the next arc. If neither condition holds, then the flow on arc k can be increased by Δ. Assign node j_k the label i^+. If arc k is not an interactive arc or if the modified costs have already been updated due to another interactive arc, go on to the next arc. Otherwise for each interactive arc β, redefine d^+ and d^- in the following manner:

   $$d_\beta^+ = C(F+\Delta_\beta+\Delta_k) - C(F+\Delta_k) + \pi_{j_\beta} - \pi_{i_\beta}$$

   $$d_\beta^- = C)F+\Delta_k) - C(F-\Delta_\beta+\Delta_k) + \pi_{j_\beta} - \pi_{i_\beta}$$

   Then go on to the next arc.

   For each arc k with $j_k = i$

   a. If node i_k is already labeled, go on to the next arc.

   b. Node i_k is not labeled. Check the following conditions.

      i. $f_k - \Delta < \ell_k$

      ii. $f_k \leq u_k$ and $d_k^- > 0$.

   If either condition holds, go on to the next arc. If neither condition holds, then the flow on arc k can be decreased by Δ. Assign node i_k the label i^-. If arc k is not an interactive arc or if the modified costs have already been updated due to another interactive arc go on to the next arc. Otherwise for each interactive arc β, redefine d_β^+ and d_β^- in the following manner:

$$d_\beta^+ = C(F+\Delta_\beta-\Delta_k) - C(F-\Delta_k) + \pi_{j_\beta} - \pi_{i_\beta}$$

$$d_\beta^- = C(F-\Delta_k) - C(F-\Delta_\beta - \Delta_k) + \pi_{j_\beta} - \pi_{i_\beta}$$

Then go on to the next arc.
When all arcs incident to node i have been examined, node i is scanned. Go to step 3.

3. If node t has been labeled, breakthrough has occurred and the algorithm terminates. If node t is not labeled, go to step 2.

Node Number Change Algorithm

A reference node should be defined for the network. Its node number will always be zero. Let γ denote the out-of-kilter arc; of course γ = A+1 if the out-of-kilter arc is a transfer arc. Determine the following two sets of arcs:

$$A_1 = \{\text{arcs } k | k \neq \gamma,\ i_k \varepsilon X,\ j_k \varepsilon \bar{X},\ d_k^+ > 0, \text{ and } f_k \leq u_k - \Delta\}$$

$$A_2 = \{\text{arcs } k | k \neq \gamma,\ i_k \varepsilon \bar{X},\ j_k \varepsilon X,\ d_k^- < 0, \text{ and } f_k \geq \ell_k + \Delta\}$$

Either A_1 or A_2 may be empty

Define $$\delta_1 = \min_{A_1}\{d_k^+\}$$

$$\delta_2 = \min_{A_2}\{-d_k^-\}$$

If either of the sets A_i is empty define the corresponding δ_i to be ∞. Let

$$\delta_3 = \begin{cases} -d_\gamma^+ & \text{if arc } \gamma \text{ is out-of-kilter due to } d_\gamma^+ < 0 \text{ but} \\ & f_\gamma \leq u_\gamma - \Delta \\ d_\gamma^- & \text{if arc } \gamma \text{ is out-of-kilter due to } d_\gamma^- > 0 \text{ but} \\ & f_\gamma \geq \ell_\gamma + \Delta \end{cases}$$

Let $\delta = \min\{\delta_1, \delta_2, \delta_3\}$

If the reference node is in the set $\bar{X}$ subtract δ from all node numbers in the set X. If the reference node is in the set X, add δ to all node numbers in the set $\bar{X}$.

C. Comments on the Algorithm

The most important observation concerning the revised out-of-kilter algorithm is that the algorithm may terminate without finding the optimal solution. The algorithm takes into account cost interactions between the flows on two arcs, but does not directly concern itself with 3-way or higher interactions. However, the algorithm will not make any flow changes that increase or leave unchanged the cost of a feasible solution. Once the algorithm terminates it is possible to continue searching for a better solution. This can be done by forcing certain flows on the interactive arcs. For instance if the algorithm terminates with a flow of $k\Delta$ in one of the interactive arcs, the bounds on that arc can be altered to force a flow of $(k+1)\Delta$ (or $(k-1)\Delta$) and a second attempt can be made with the algorithm. This procedure can be repeated if it seems desirable.

Some explanation may be needed concerning various parts of the algorithm. The necessity of the transfer arcs is a result of the interactions in the cost function. For instance when transfer arcs were omitted from the algorithm, termination occurred with reservoir number 6 containing 100 units final storage and reservoir number 7 containing 60 units final storage. Let F be the vector of flows at termination. Let $-\Delta_{R6}$ denote a decrease in reservoir 6 final storage by Δ (twenty) units. Let Δ_{R7} denote an increase in reservoir 7 final storage by Δ. At termination we had the following costs (modified by subtracting a constant

$$C(F) = 0$$

$$C(F-\Delta_{R6}) = 5300$$

$$C(F+\Delta_{R7}) = -4625$$

$$C(F-\Delta_{R7}) = 5425$$

$$C(F+\Delta_{R7}-\Delta_{R6}) = 0$$

The node numbers were as follows:

$$\pi(R_6) = 5300$$

$$\pi(R_7) = 5200$$

$$\pi(\text{source node}) = 0$$

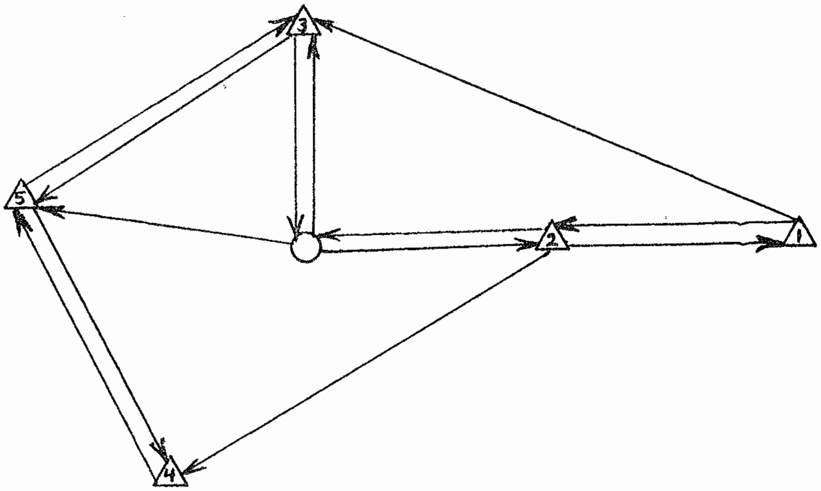
Hence when examining reservoir 6 we have $d^-_{R6} = 0$ (d^+_{R6} not given since flow was equal to upper bound). Also $d^+_{R7} = 575$ and $d^-_{R7} = -225$. Hence both interactive arcs are in kilter. However, cost can be reduced by 100 units by moving Δ units of flow from reservoir 6 to reservoir 7. If we form the transfer arc from reservoir 6 to reservoir 7, this arc has $d^+ = -100$, which shows that flow should be increased in this arc.

An aspect of step 11 requires some explanation. Note that it is necessary for all arcs to be in kilter without flow changes for the algorithm to be declared done. This is because it is possible for one of the interactive arcs to be driven out-of-kilter by flow changes in another (or two other) interactive arcs.

Another point to be made concerns the concavity of the benefit function that results from solving a sequence of problems with parametric variation in the initial reservoir levels. This function was shown to be concave in the continuous case. If higher interactions cause inaccuracies in the solutions, the resulting benefit function may not be concave. These inaccuracies may be partially corrected by linearizing the nonconcave segments. Not only does this improve the solutions, but it may be necessary to avoid cycling in the next stage.

V. Example Problem

This method is being tested on a five-reservoir problem. The configuration of the reservoirs is as follows:



Each year is divided into six periods. For each period, four possible sets of net demands are specified. These correspond to an average, wet, dry, or irregular period. Equal probabilities are assigned to each possible demand set. It is assumed that this data applies in each year. The actual net demand sets used are as follows

		Period					
	average	1	2	3	4	5	6
reservoir	1	-60	20	40	40	20	-80
	2	-80	20	40	20	20	-60
	3	80	40	-60	-40	20	40
	4	20	60	100	120	60	40
	5	20	40	80	120	60	20

		Period					
	wet	1	2	3	4	5	6
reservoir	1	-60	20	40	20	-40	-100
	2	-80	-20	20	20	-20	-60
	3	60	20	-100	-80	-40	40
	4	-20	40	80	100	60	20
	5	40	20	60	100	40	-40

		Period					
	dry	1	2	3	4	5	6
reservoir	1	-20	40	60	40	20	-40
	2	-60	20	60	60	20	-40
	3	100	80	-40	-20	40	60
	4	20	80	120	140	100	40
	5	40	40	100	140	80	20

		Period					
	irregular	1	2	3	4	5	6
reservoir	1	-60	20	40	60	20	-120
	2	-40	20	-40	60	40	-20
	3	100	80	-40	-100	-20	60
	4	40	20	80	120	60	40
	5	20	80	100	80	80	40

Imports are permitted at reservoir #1 according to the following schedule:

period	1	2	3	4	5	6
maximum import	300	120	60	60	120	300
per-unit cost	100	200	300	300	200	100

Again it is assumed that this data applies in each year. The reservoir storage capacities are as follows:

reservoir	1	2	3	4	5
storage capacity	200	100	100	100	100

Flows are permitted in step sizes of 20. For each period, the benefit function is stored in an array of size 6^5. In iteratively computing the benefit functions, initial storages in reservoirs 2, 3, 4, and 5 are varied in increments of 20. Reservoir 1 has its initial storage varied in increments of 40, in between values are found by interpolation.

In order to begin the iterative process an initial approximation for one of the benefit functions can be obtained as follows. A six-period network is set up using the "average" set of net demands. However, the network is set up in a closed loop form, that is, final storage at the end of period 6 corresponds to initial storage at the beginning of period 1. In this way a steady-state solution can be obtained using the linear separable out-of-kilter algorithm. For any period in the network and any reservoir, the difference between the node number of the node representing that reservoir is that period and the node number of the source node gives an indication of the marginal value.per unit of water. These values can then be used to form a linear separable benefit function for storage at the beginning of the period employed. This procedure was followed to give the following initial approximation for the benefit function for storage at the end of period 6.

reservoir	per-unit value of storage
1	100
2	320
3	345
4	380
5	375

Thus the iterative process could now begin with period 6.

For each period 6^5 = 7776 runs were made for each of the four sets of net demands. The resulting benefit functions were averaged to form the benefit function for the end of the previous period.

Several methods were used to speed up the computation. The most important of these is that each computer run starts from the optimal results of a previous run. In each case the only difference between the two runs was an increase of the initial storage in some reservoir by twenty units (forty units if reservoir number 2). This necessitated the storing of four sets of flows and node numbers. Another speed saving device is to force the added initial storage directly into final storage for that reservoir if this is feasible. This may give the optimal solution immediately, or just give a feasible solution from which to procede.

These methods helped reduce the average computation time to between .03 and .04 seconds per run on the CDC 6600 computer. The range of the times per run was from too small for the computer clock to distinguish from 0, up to about .065 seconds. Usually about 300 seconds were required per data set, thus about 1200 seconds per period. The iterative process can be continued as long as is desirable condidering the tradeoff between increased computation time and gain in convergence. In the example problem it appeared that 12 periods would be more than sufficient and 9 would probably be acceptable. Once the benefit functions have been found, and the program is being used operationally, the computation time required would be insignificant.

The benefit functions were stored on permanent files, either disk or magnetic tape. Benefits, node numbers, increase costs, and decrease costs were stored as real variables while flows, upper bounds, and lower bounds were integer varibles. A listing of the computer code used may be found in Driscoll [27]. Also more detailed computational results can be found in the previous reference if desired.

VI. Conclusion

The proposed algorithm gives a method for solving a multi-reservoir problem of five or six reservoirs without using an unreasonable amount of computation time. The solutions obtained may not be optimal, but should be within a small tolerance of optimality. This should be acceptable to the practitioner since the parameters of the problem are not exact either. The model employed is easily represented pictorially and should be readily understandable to the user. Extensive knowledge of the algorithm is not required of the user, expecially once the operational stage is reached.

There is scope for further improvement of this method. A more efficient computer code could be devised by those well-versed in such work. Further research may lead to more exact methods of coping with higher interactions. Short cuts in iteratively finding the benefit functions may become apparent once greater experience is obtained.

APPENDIX

Theorem 1. The deterministic benefit function $b(A,z) = K(2b,0) - K(2b,z)$ is concave in z.

Proof:

Since $b(A,z) = K(2b,0) - K(2b,z)$ where $K(2b,0)$ is a constant, it is equivalent to show that $K(2b,z)$ is convex. By definition

$$K(2b,z) = \min \sum_{t=A+1}^{T} \quad \sum_{c=1}^{C} K(c)\ f(c,t) + \sum_{r=1}^{R} p(r,t)\ w(r,t)$$

$$- \sum_{r=1}^{R} q\big(r,t,x(r,t)\big)$$

subject to:

$$w(r,t) + i(r,t) - x(r,t) + \ell(r,t-1) - \ell(r,t)$$

$$+ \sum_{c \in T(r)} f(c,t) - \sum_{c \in F(r)} f(c,t) = 0 \quad \forall r \text{ and } A+1 \le t \le T$$

$$z = \ell(A)$$

$$0 \le \ell(r,t) \le s(r) \quad \forall r \text{ and } A+1 \le t \le T$$

$$0 \le f(c,t) \le u(c) \quad \forall c \text{ and } A+1 \le t \le T$$

$$0 \le w(r,t) \le m(r,t) \quad \forall r \text{ and } A+1 \le t \le T$$

$$0 \le x(r,t) \quad \forall r \text{ and } A+1 \le t \le T\ .$$

Suppose $z(1)$ and $z(2)$ are two possible values of z and $0 < \Theta < 1$. Let $z(3) = \Theta z(1) + (1-\Theta)\ z(2)$. $K(2b,z)$ is convex if and only if $K\big(2b,z(3)\big) \le K\big(2b,z(1)\big) + (1-\Theta)\ K\big(2b,z(2)\big)$.

Let the values of the variables $\ell(r,t)$, $f(c,t)$, $w(r,t)$ and $x(r,t)$ in $K\big(2b,z(i)\big)$ be denoted $\ell(r,t,i)$, $f(c,t,i)$, $w(r,t,i)$ and $x(r,t,i)$ respectively for all r and c and appropriate t and for $i = 1$ or 2. Let

$$\ell(r,t,3) = \Theta\ell(r,t,1) + (1-\Theta)\ \ell(r,t,2)$$

$$f(c,t,3) = \Theta f(c,t,1) + (1-\Theta)\ f(c,t,2)$$

$$w(r,t,3) = \Theta w(r,t,1) + (1-\Theta)\ w(r,t,2)$$

$$x(r,t,3) = \Theta x(r,t,1) + (1-\Theta)\ x(r,t,2)$$

for all r and c, and appropriately t.

Multiply each constraint in the problem finding $K(2b,z(1))$ by Θ and each constraint in the problem finding $K(2b,z(2))$ by $1-\Theta$, then addition of like constraints yields

$$w(r,t,3) + i(r,t) - x(r,t,3) + \ell(r,t-1,3) - \ell(r,t,3) + \sum_{c \in T(r)} f(c,t,3) - \sum_{c \in F(r)} f(c,t,3) = 0 \;\; \forall r \text{ and } A+1 \leq t \leq T.$$

$z(3) = \ell(A,3)$ where $\ell(A,3)$ is the vector whose rth element is $\ell(r,A,3)$ for all r.

$0 \leq \ell(r,t,3) \leq s(r) \quad \forall r$ and $A+1 \leq t \leq T$

$0 \leq f(c,t,3) \leq u(c) \quad \forall c$ and $A+1 \leq t \leq T$

$0 \leq w(r,t,3) \leq m(r,t) \quad \forall r$ and $A+1 \leq t \leq T$

$0 \leq x(r,t,3) \quad \forall r$ and $A+1 \leq t \leq T.$

Hence $\ell(r,t,3)$, $f(c,t,3)$, $w(r,t,3)$, and $x(r,t,3)$ are feasible values for the variables $\ell(r,t)$, $f(c,t)$, $w(r,t)$, and $x(r,t)$ respectively in the constraints of the problem finding $K(2b,z(3))$. Hence

$$K(2b,z(3)) \leq \sum_{t=A+1}^{T} \left\{ \sum_{c=1}^{C} k(c)\, f(c,t,3) + \sum_{r=1}^{R} p(r,t)\, w(r,t,3) - \sum_{r=1}^{R} q\bigl(r,t,x(r,t,3)\bigr) \right\} .$$

Since $K(2b,z(3))$ is defined to be the value of the optimal solution for problem 3. Now

$$\begin{aligned} \Theta K(2b,z(1)) + (1-\Theta)\, K(2b,z(2)) = {} & \sum_{t=A+1} \left\{ \sum_{c=1} k(c)\, \Theta f(c,t,1) \right. \\ & \left. + \sum_{r=1}^{R} p(r,t)\, \Theta w(r,t,1) - \sum_{r=1}^{R} \Theta q\bigl(r,t,x(r,t,1)\bigr) \right\} \\ & + \sum_{t=A+1}^{T} \left\{ \sum_{c=1}^{C} k(c)(1-\Theta)\, f(c,t,2) + \sum_{r=1}^{R} p(r,t)(1-\Theta)\, w(r,t,2) \right. \\ & \left. - \sum_{r=1}^{R} (1-\Theta)\, q\bigl(r,t,x(r,t,2)\bigr) \right\} = \sum_{t=A+1}^{T} \left\{ \sum_{c=1}^{C} k(c)\, f(c,t,3) \right. \\ & + \sum_{r=1}^{R} p(r,t)\, w(r,t,3) - \sum_{r=1}^{R} \Bigl(\Theta q\bigl(r,t,x(r,t,1)\bigr) \\ & \left. + (1-\Theta)\, q\bigl(r,t,x(r,t,2)\bigr) \Bigr) \right\} \end{aligned}$$

Since $q(r,t,x)$ is concave in x,

$$\Theta q\Big(r,t,x(r,t,1)\Big) + (1 - \Theta)\, q\Big(r,t,x(r,t,2)\Big) \le q\Big(r,t,x(r,t,3)\Big) .$$

Hence $\Theta K\Big(2b,z(1)\Big) + (1 - \Theta)\, K\Big(2b,z(2)\Big) \ge$

$$\sum_{t=A+1}^{T} \left\{ \sum_{c=1}^{C} k(c)\, f(c,t,3) + \sum_{r=1}^{R} p(r,t)\, w(r,t,3) \right.$$

$$\left. - \sum_{r=1}^{R} q\Big(r,t,x(r,t,3)\Big) \right. .$$

Combining with (4) yields

$$\Theta K\Big(2b,z(1)\Big) + (1 - \Theta)\, K\Big(2b,z(2)\Big) \ge K\Big(2b,z(3)\Big) .$$

Hence $K(2b,z)$ is convex in z and $b(A,z)$ is concave in z.

<u>Theorem 2</u>. The stochastic benefit function $b(A,z) = \sum_{k=1}^{N} P(A + 1,k)\, b(A,f,k)$ is concave in z.

Proof:

Since $b(A,f) = \sum_{k=1}^{N} P(A + 1,k)\, b(A,f,k)$ where $P(A + 1,k) \ge 0$, it is sufficient to show that each $b(A,f,k)$ is concave. This is due to the facts that a non-negative constant times a concave function is a concave function and the sum of a finite number of concave functions is a concave function. Since $b(A,f,k) = C(0,k) - C(f,k)$ it suffices to show that $C(f,k)$ is convex in f. This is because the negative of a convex function is a concave function and the sum of a concave function and a constant is a concave function. $C(0,k)$ is, of course, a constant for any k.

Suppose $f(1)$ and $f(2)$ are two possible values of f and $0 < \Theta < 1$. Let $f(3) = \Theta f(1) + (1 - \Theta)\, f(2)$. $C(f,k)$ is convex if and only if $C\Big(f(3),k\Big) \le \Theta C\Big(f(1),k\Big) + (1 - \Theta)\, C\Big(f(2),k\Big)$.

Let the values of the variables $\ell(r, A - 1)$, $\ell(r,A)$, $f(c,A)$, $w(r,A)$, and $x(r,A)$ in $C\Big(f(c),k\Big)$ be denoted by $\ell(r,A - 1,i)$, $\ell(r,A,i)$, $f(c,A,i)$, $w(r,A,i)$, and $x(r,A,i)$ respectively for all r and i = 1 or 2. Let

$$\ell(r,A - 1,3) = \Theta 1(r,A - 1,1) + (1 - \Theta)\, \ell(r,A - 1,2)$$

$$\ell(r,A,3) = \Theta \ell(r,A,1) + (1 - \Theta)\, 1(r,A,2)$$

$$f(c,A,3) = \Theta f(c,A,1) + (1 - \Theta)\, f(c,A,2)$$

$$w(r,A,3) = \Theta w(r,A,1) + (1 - \Theta)\, w(r,A,2)$$

$$x(r,A,3) = \Theta x(r,A,1) + (1 - \Theta)\, x(r,A,2)$$

for all r and c.

Multiply each constraint in the problem $C\big(f(1),k\big)$ by Θ and each constraint in the problem $C\big(f(2),k\big)$ by $(1 - \Theta)$, then addition of like constraints yields

$f(3) = \ell(A - 1,3)$ where $\ell(A - 1,3)$ is the vector whose rth element is $\ell(r,A - 1,3)$ for all r

$$w(r,3) + i(r,A) - x(r,3) + \ell(r,A - 1,3) - \ell(r,A,3)$$

$$+ \sum_{c \in T(r)} f(c,A,3) - \sum_{c \in F(r)} f(c,A,3) = 0 \quad \forall r$$

$$0 \le \ell(r,A,3) \le s(r) \quad \forall r$$

$$0 \le f(c,A,3) \le u(c) \quad \forall c$$

$$0 \le w(r,A,3) \le m(r,A) \quad \forall r$$

$$0 \le x(r,A,3) \quad \forall r.$$

Hence $\ell(r,A - 1,3)$, $\ell(r,A,3)$, $f(c,A,3)$, $w(r,A,3)$, and $x(r,A,3)$ are feasible values for the variables $\ell(r,A - 1)$, $\ell(r,A)$, $f(c,A)$, $w(r,A)$, and $x(r,A)$ respectively in the constraints of the problem $C\big(f(3),k\big)$. Hence

1. $$C\big(f(3),k\big) \le \sum_{c=1}^{C} k(c)\ f(c,A,3) + \sum_{r=1}^{R} p(r,A)\ w(r,A,3)$$

$$- \sum_{r=1}^{R} q\big(r,A,x(r,A,3),k\big) - b\big(A,\ell(A,3)\big).$$

Now $\Theta C\big(f(1),k\big) + (1 - \Theta)\ C\big(f(2),k\big)$

$$= \sum_{c=1}^{C} k(c)\Theta f(c,A,1) + \sum_{r=1}^{R} p(r,A)\Theta w(r,A,1)$$

$$- \sum_{r=1}^{R} \Theta q\big(r,A,x(r,A,1),k\big) - \Theta b\big(A,\ell(A,1)\big)$$

$$+ \sum_{c=1}^{C} k(c)\ (1 - \Theta)\ f(c,A,2) + \sum_{r=1}^{R} p(r,A)\ (1 - \Theta)\ w(r,A,2)$$

$$- \sum_{r=1}^{R} (1 - \Theta)\ q\big(r,A,x(r,A,2),k\big) - (1 - \Theta)\ b\big(A,\ell(A,2)\big)$$

$$= \sum_{c=1}^{C} k(c)\ f(c,A,3) + \sum_{r=1}^{R} p(r,A)\ w(r,A,3)$$

$$- \sum_{r=1}^{R} q\big(r,A,x(r,A,1),k\big) + (1 - \Theta)\ q\big(r,A,x(r,A,2),k\big)$$

$$- \Big(\Theta b\big(A,\ell(A,1)\big) + (1 - \Theta)\ b\big(A,\ell(A,2)\big)\Big) \quad .$$

Since q(r,A,x,k) is concave in x

$$\Theta q\big(r,A,x(r,A,1),k\big) + (1-\Theta)\, q\big(r,A,x(r,A,2),k\big) \leq q\big(r,A,x(r,A,3),k\big) \text{ for all } r.$$

Similarly since b(A,z) is concave in z

$$\Theta b\big(A,\ell(A,1)\big) + (1-\Theta)\, b\big(A,\ell(A,2)\big) \leq b\big(A,\ell(A,3)\big) \ .$$

Incorporating these results into the above equation yields

$$\Theta C\big(f(1),k\big) + (1-\Theta)\, C\big(f(2),k\big) \geq \sum_{c=1}^{C} k(c)\, f(c,A,3) + \sum_{r=1}^{R} p(r,A)\, w(r,A,3) - \sum_{r=1}^{R} q\big(r,A,x(r,A,3),k\big) - b\big(A,\ell(A,3)\big) \ .$$

Combining with 1. above yields

$$\Theta C\big(f(1),k\big) + (1-\Theta)\, C\big(f(2),k\big) \geq C\big(f(3),k\big) \ .$$

Hence C(f,k) is convex in f and b(A,f) is concave in f.

REFERENCES

1. Buras, Nathan, Scientific Allocation of Water Resources, American Elsevier Co., New York, N. Y., 1972.

2. Butcher, William S., "Stochastic Pyramic Programming and Water Resources Management", Presented to Operations Research Society of America in San Diego, California, November 14, 1973.

3. Butcher, William S., "Stochastic Dynamic Programming For Optimal Reservoir Operation", Water Resources Bulletin, Vol. 7, No. 1, February 1971, p. 115-123.

4. Bodin, L. D. and Roefs, T. G., "A Decomposition Approach To Non-Linear Programs As Applied To Reservoir Systems", Networks, Vol. 1, No. 1, November 1971, p. 59-73.

5. Curry, Guy L. and Helm, James C., "A Chance-Constrained Model For A Single Multipurpose Reservoir", Texas A&M University, 1972.

6. Curry, Guy L., Helm, James C., and Clark, Robert A., "A Chance-Constrained Programming Model For A Corrected System Of Multipurpose Reservoirs ", Texas A&M University, 1972.

7. Dudley, Norman J. and Burt, Oscar R., "Stochastic Reservoir Management and System Design For Irrigation", Water Reservoir Research, Vol. 9, No. 3, June 1973, p. 507-572.

8. Eastman, John and ReVelle, Charles, "Linear Decision Rule In Reservoir Management and Design", Water Resources Research Vol. 9, No.1, February 1973.

9. Evenson, D. E. and Moseley, J. C., "Simulation/Optimization Techniques For Multibasin Water Resources Planning", Water Resources Bulletin, Vol. 6, No. 5, p. 725-736.

10. Fults, Dan M. and Hancock, Lawrence F., "Water Resources Optimum Operations Model", U.S. Dept. of the Interior.

11. Hall, Warren A., and Dracup, John A., Water Resources Systems Engineering, McGraw Hill Book Co., New York, N. Y., 1970.

12. Helm, James C., Curry, Guy L., and Hasan, Sayeed, "A Capacity Expansion Model For A System Of Linked Multipurpose Reservoirs With Stochastic Inflows", Texas A&M University, 1972.

13. Jensen, Paul A. and Reeder, Hugh A., "Solution of Convex Network Problems With a Revised Out-of-Kilter Algorithm", University of Texas, 1974.

14. Kerr, J. A., "Multireservoir Analysis Techniques In Water Quantity Studies", Water Resources Bulletin, Vol. 8, No. 5, October 1972, p. 871-880.

15. Lane, Morton, "Conditional Chance-Constrained Model For Reservoir Control", Water Resources Research, Vol. 9, No. 4, August 1973.

16. Loucks, Daniel P., "Stochastic Methods For Analyzing River Basin Systems", Cornell University Water Resources Center, 1969, Technical Report, No. 16.

17. Loucks, Daniel P., "Computer Model For Reservoir Regulation", Proceedings, American Society of Civil Engineers, 94(SA4), 1968, p. 657-669.

18. Parikh, Shailendra C., "Linear Dynamic Decomposition Multi-purpose Reservoir System", Operations Research Center, University of California, Berkeley, California, September 1966.

19. Parikh, Shailendra C. and Shepherd, R. W., "Linear Dynamic Decomposition Programming Approach To Long Range Optimization of Northern California Water Resources System", Operations Research Center Report 67-30, University of California, Berkeley, California, August 1967.

20. Roefs, Theodore G., "Reservoir Management: The State Of The Art", I.B.M. Washington Scientific Center, July 15, 1968.

21. Rood, Omar E., Dessouky, Mohaded I., and Meredith, Dale P., "Optimal Dynamic Operation Of M Serially Linked Reservoirs For N Time Periods ", Presented at the 44th National ORSA Meeting,November 1973.

22. Salcedo, Daniel E., "Dynamic Economic Simulation Of An Irrigation System", University of Texas At Austin, August 1972.

23. Su, Shraw Y. and Deininger, Rolf A., "Optimal Operation Policies For Multi-Purpose Multi-Reservoir System", 40th National ORSA meeting, October 25-29, 1971, Anaheim, California.

24. Weirs, A. O. and Beard, L. R., "A Multibasin Planning Strategy", Texas Water Development Board, 1971, Water Resources Bulletin, Vol. 7, No. 4, August 1971, p. 750-764.

25. Windsor, James S. and Chow, Ven Te, "Multireservoir Optimization Model", Journal of Hydraulic Division, Proceeding of American Society of Civil Engineers, October 1972, p. 1827-45.

26. Young, G. K., "Finding Reservoir Operating Rules", Proceedings, American Society of Civil Engineers 98 (HY3), p. 297-321, 1967.

27. Driscoll, W. D., "Multireservoir Operating Policies Considering Uncertainty", Unpublished Dissertation, University of Texas, Austin, Texas, September, 1974.