New Methods in Mathematical Programming—Optimal Flow Through Networks with Gains

William S. Jewell,

OPTIMAL FLOW THROUGH NETWORKS WITH GAINS

William S. Jewell

University of California, Berkeley, California

(Received August, 1960)

The special class of linear programs described as network flow problems includes many of the special structure optimization problems in such areas as assignment, transportation, catering, warehousing, production scheduling, etc Besides the physical motivation of the flow description, this class of problems possesses conceptually simple, and computationally elegant, solution techniques originated by DANTZIG, FORD, FULKERSON, ET AL This paper describes a generalization of this class of problems to 'process-flow networks,' or 'flow with gains,' in which flow in any branches 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 ' Among other interesting problems, this model includes the metal-processing problem, the machine-loading problem, financial budgeting, aircraft routing, warehousing with 'breeding' or 'evaporation,' the two-equation capacitated linear program, etc The solution method here described and proved is a natural extension of Ford and Fulkerson's technique (or specialization of the primal-dual method) and retains much of the physical motivation and easy computational aspect of their technique, a direct starting solution is usually available, and optimization of the imbedded linear program (the restricted primal problem) can be accomplished without the use of the simplex technique Conceptually, the solution procedure consists of the following steps

(1) Find the incrementally cheapest loop in the network that will absorb flow
(2) Establish as much additional flow into the network along this route as possible
(3) Repeat Steps (1) and (2) until the desired flow is established, or until infeasibility is discovered

The solution technique includes a maximal-flow procedure and produces a 'min-cut equals max-flow' theorem for networks with gains Additional features, such as piece-wise linear convex costs, parametric studies, network 'tearing,' mixed boundary conditions, etc , may be easily incorporated in the algorithm

THERE ARE several reasons for examining special structures in mathematical programming A simple hand computation method may be needed for small problems, or, general-purpose digital computer programs may be inefficient or may lack sufficient capacity to handle a large, sparse constraint matrix

Also, the investigation of problems with special structures provides deeper insight into the problem itself, as computational simplifications are revealed, and the solution algorithm achieves a more compact form Often, extensions can be made to related optimization problems, and, occasionally, the special solution technique will suggest desirable variations of a more general algorithm

This paper will consider a special class of linear programs that can be described as the optimization of flow through a network with 'gains,' or 'multipliers' This class of problems possesses an efficient solution algorithm, which is based on the network flow techniques of Ford and Fulkerson for networks without gains, and upon the primal-dual algorithm of Dantzig, Ford, and Fulkerson for general linear programs

+1	+1	+1	-1			-1			-1		
-1			+1	+1	+1		-1			-1	
	-1			-1		+1	+1	+1			-1
		-1			-1			-1	+1	+1	+1
+1											
	+1										
		+1									
			+1								
				+1							
					+1						
						+1					
							+1				
								+1			
									+1		
										+1	
											+1

Fig. 1. Constraint matrix for simple networks

FLOW PROBLEMS

THE SPECIAL class of linear programs described as optimal flow through networks was developed primarily by Dantzig, Ford, and Fulkerson of the Rand Corporation Briefly, the simple network optimal flow problem is

A network is constructed of oriented branches that can pass a limited amount of flow from one node to another, flow conservation laws are assumed to hold at all nodes, except where some required amount of flow is supplied to and withdrawn from the network Given that the cost (or disutility) of flow in each branch is directly proportional to the amount of flow, what flow pattern will satisfy the restrictions at minimum total cost?

Mathematical statements of this problem, solution techniques, and problem applications may be found in the literature,[8 10,11,12,14,15] as well as in a forthcoming book by Ford and Fulkerson [13]

The constraint matrix for these problems is just the branch-node incidence matrix of the network, augmented by a unit matrix (Fig 1) Although this matrix consists of only plus or minus unity in very special locations, the optimal flow model describes many important operational situations the assignment and transportation problems, strategic production and warehouse scheduling problems, catering and overhaul problems, etc Because of the simplicity of its formulation, and the computational elegance of its solution algorithms, the flow model has become probably the most successful special-purpose linear program

The generalization of this paper has the constraint matrix shown in Fig 2 Although no new nonzero positions are occupied, the coefficients of the incidence matrix are now arbitrary in sign and magnitude This new feature is obtained by associating a constant gain or multiplier with

$$\left[\begin{array}{cccccccccccc}
+k_{12} & +k_{13} & +k_{14} & -\ell_{21} & & & -\ell_{31} & & & -\ell_{41} & & \\
-\ell_{12} & & & +k_{21} & +k_{23} & +k_{24} & & -\ell_{32} & & & -\ell_{42} & \\
& -\ell_{13} & & & -\ell_{23} & & +k_{31} & +k_{32} & +k_{34} & & & -\ell_{43} \\
& & -\ell_{14} & & & -\ell_{24} & & & -\ell_{34} & +k_{41} & +k_{42} & +k_{43} \\
\hline
+1 & & & & & & & & & & & \\
& +1 & & & & & & & & & & \\
& & +1 & & & & & & & & & \\
& & & +1 & & & & & & & & \\
& & & & +1 & & & & & & & \\
& & & & & +1 & & & & & & \\
& & & & & & +1 & & & & & \\
& & & & & & & +1 & & & & \\
& & & & & & & & +1 & & & \\
& & & & & & & & & +1 & & \\
& & & & & & & & & & +1 & \\
& & & & & & & & & & & +1
\end{array}\right]$$

Fig 2 Constraint matrix for networks with gains

each branch, which multiplies the flow as it passes through the branch Thus, flow problems can be described in which 'conversion' from one type of flow to another type takes place, without 'constant returns to scale'

It will be seen that this extension does not add appreciably to the labor of finding a solution, since many of the special features of the Ford-Fulkerson technique are retained In particular, the restricted primal problem (a 'max-flow' problem) can be solved by a labelling procedure instead of using the simplex technique, and the simple structure of the dual makes the computation of new feasible duals an easy task Finding the solution directly 'on the network' is easier for hand computations than a formal tableau manipulation, and it is more efficient for machine computations because of the compactness of problem statement

With this new flow model, one can formulate and solve such additional problems as the machine-loading or metal-processing problem, the aircraft route allocation problem, financial budgeting, warehousing with 'breeding' or 'evaporation,' catering problems with losses, the two-equation capacitated linear program, etc Several typical network formulations will be displayed in a later section

OPTIMAL FLOW WITH GAINS

By REDEFINING flow variables, it is always possible to depict a typical branch with gains as in Fig 3 Each oriented branch (i,j) has a non-negative flow X_{ij}, possibly limited by a branch capacity, M_{ij} The branch flow is measured in the units of flow at the input node i, but is 'converted' to the units of flow at node j by multiplying by a nonzero gain, K_{ij}, before leaving the branch Each branch also has associated with it a unit cost, C_{ij}, which measures the cost of establishing one unit of flow $V_i(V_j)$ and U_{ij} are the dual variables associated with the nodes and branches of the network The object will be to route some desired flow through a network of these branches at least total cost

The topology of the network depends upon the problem of interest For convenience in the algorithm to be described, it is assumed that there are $N+1$ nodes in the network, indexed by i or $j=0, 1, 2, \ldots, N$, exactly

Fig 3. A typical branch (i,j) with gain K_{ij}

two branches, (i,j) and (j,i) connect any two nodes, and, there are no branches (i,i) These assumptions are not restrictive, since parallel or looping branches can be incorporated into the model by adding new nodes and branches, and nonexistent branches can be effectively removed from the algorithm by setting their unit cost at some arbitrarily large value

At almost every node in the network, conservation of flow will be assumed—that is, flow cannot be created or destroyed at the node The exception occurs at those nodes that satisfy the flow boundary conditions of the problem, the 'input' and 'output' nodes

There are several interesting features of networks with flow multipliers that contrast with the behavior of the simpler networks considered by Ford and Fulkerson The first feature is that flow may be 'destroyed' by passing around a loop whose total gain is less than unity (Fig 4a) Secondly, flow may be 'created' (in an amount limited only by the branch capacities) in a loop structure whose total gain is greater than unity (Fig 4b) A third situation is detected in the algorithm when flow is 'cancelled' by passing through two parallel branches with gains of opposite signs (Fig 4c), of course, on an incremental basis, this is identical with the first situation It is these features that make the construction of a special-purpose algorithm an interesting problem

The boundary conditions for a network with gains may be either inequalities or equality constraints on the amount of input and output flow, since, unlike the simple flow networks, there is no over-all requirement that 'flow in equals flow out' Figure 5 indicates how additional branches can be added to the network to reduce typical boundary conditions to the following canonical input requirement total flow into the network at the source (node 0) must equal an amount $\hat{Q}$ By examining the flow conservation equations at the nodes of the enlarged network, we see that at optimality the four boundary conditions of Fig 5(a) may

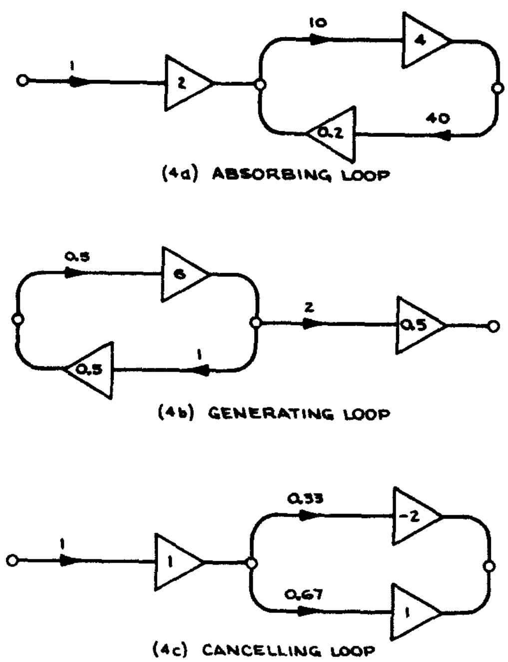


Fig. 4 Special structures in networks with gains

be replaced by the single boundary condition of Fig 5(b) For simplicity, only this boundary condition will be used in the algorithm

The mathematical statement of the primal and dual problems of optimal flow through a network with gains is

Primal Problem

Minimize

$$\mathcal{C} = \sum_{i,j} C_{ij} X_{ij}, \tag{1}$$

$$\sum_j (X_{ij} - K_{ji} X_{ji}) = \begin{cases} Q & (i=0) \\ 0, & (i \neq 0) \end{cases} \tag{2a}$$

$$0 \leqq X_{ij} \leqq M_{ij} \tag{2b}$$

Dual Problem

Maximize

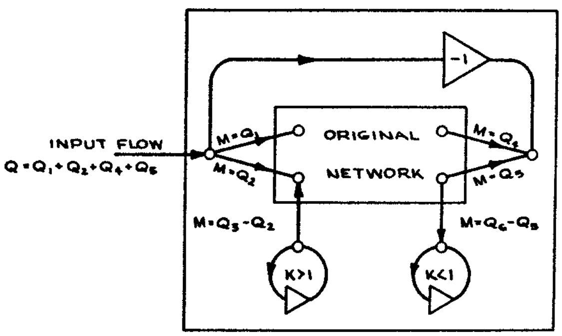
$$e = QV_0 - \sum_{i,j} M_{ij} U_{ij}, \tag{3}$$

$$V_i - K_{ij} V_j - U_{ij} \leqq C_{ij}, \tag{4a}$$

$$U_{ij} \geqq 0, \tag{4b}$$

$$V_i \text{ unrestricted}, \tag{4c}$$

(5a) ORIGINAL NETWORK WITH MIXED INPUT-OUTPUT CONDITIONS



(5b) ENLARGED NETWORK WITH SINGLE INPUT CONDITION

Fig. 5 Reformulation of network problem in canonical form

when $i,j=0, 1, 2, \ldots, N$, C_{ij}, M_{ij}, and Q are given nonnegative constants, and the K_{ij} are arbitrary nonzero constants The constraint matrix of equation (2) is similar to that of Fig 2, except that some unity coefficients have reappeared, because of the renormalization of the flow variables The dual problem has a simple constraint for each branch (i,j), in terms of dual variables for each node, V_i and V_j, and a dual variable for each branch, U_{ij}

The relation between the primal and dual variables *at optimality* is of importance, from the weak theorem of complementary slackness,[17] if there is an optimal solution to problems (1) (2) and (3) (4), it must be that

$$\text{If } V_i - K_{ij} V_j < C_{ij}, \quad (U_{ij}=0) \quad \text{then} \quad X_{ij}=0, \tag{5a}$$

$$\text{If } 0<X_{ij}<M_{ij}, \quad \text{then} \quad V_i-K_{ij}V_j=C_{ij} \quad \text{and} \quad U_{ij}=0, \tag{5b}$$

$$\text{If } U_{ij}>0, \quad (V_i-K_{ij}V_j=C_{ij}+U_{ij}) \quad \text{then} \quad X_{ij}=M_{ij} \tag{5c}$$

U_{ij} is a slack variable which allows $V_i-K_{ij}V_j$ to be greater than C_{ij} when the branch is *saturated* ($X_{ij}=M_{ij}$) if a branch is *empty* ($X_{ij}=0$) or has any feasible flow not equal to the capacity, then the optimal U_{ij} must be zero

Thus, according to the dual variables, there are three mutually exclusive 'states' of a branch, defined by

$$\text{A branch } (i,j) \text{ is } \begin{cases} \textit{inactive} \\ \textit{active} \\ \textit{hyperactive} \end{cases} \quad \text{if } V_i-K_{ij}V_j \begin{cases} <C_{ij} \\ =C_{ij} \\ >C_{ij} \end{cases}$$

Since feasibility is maintained in the dual throughout the algorithm, these dual states of a branch are also collectively exhaustive

Using these definitions, the complementary slackness conditions (5) state that, at optimality

1. If a branch is inactive, it must be empty
2. If a branch is nonempty and nonsaturated, it must be active
3. If a branch is hyperactive, it must be saturated

These conditions will be used in the optimal-flow algorithm to assure that the optimal solutions to the primal and the dual are attained simultaneously Also, the duality theorem of linear programming[17] states that, if there exist feasible solutions to both the primal and dual problems, at optimality $\mathcal{C}=\mathfrak{c}$

SOLUTION TECHNIQUES

THE OPTIMAL-FLOW problem just stated is a linear program, and could be solved directly using the simplex technique of G Dantzig Because of the sparseness of the constraint matrix, this method might be quite inefficient, unless a digital computer routine were available to take advantage of this sparseness, as the referee has pointed out, the usual limitation in solving network problems with a simplex routine is the maximum number of constraints that can be handled

Another possibility would be to use a special adaptation of the simplex method for this problem Roughly speaking, such a technique would proceed as follows

Phase I

1. Establish any routing of flow in the network that obeys conservation and capacity restrictions, and absorbs the desired input flow

Phase II

2. Attempt to assign dual variables to each node and branch, satisfying the complementary slackness conditions If this can be done in a consistent manner, the optimal solution has been reached

3. Otherwise, select some branch for which the conditions are not satisfied, and reroute the flow in an appropriate manner to decrease dual infeasibility Repeat Step **2**

Algorithms of this type exist for transportation problems with gains (the machine-loading problem),[2 7 23] and they can probably be extended to network problems by using the device of transshipment [25]

The difficulty with such algorithms is that much effort may be expended on Phase I, 'getting feasible ' If the network is tightly constrained, so that the optimal solution is close to all feasible solutions, this may nonetheless be an efficient procedure In the author's experience, however, most network-type problems are loosely constrained, and it is desirable to have an algorithm that takes account of the costs as the flow is established, such a solution technique is the primal-dual method of Dantzig, Ford, and Fulkerson [7]

This technique was the generalization of Ford and Fulkerson's network flow methods to the solution of general linear programs In contrast with the simplex method, the primal-dual method maintains a feasible dual, and always satisfies the complementary slackness relations, in each cycle of the algorithm, an imbedded linear-program must be solved to decrease the infeasibility of the primal Unfortunately, in the general case, this imbedded linear program must itself be solved by the simplex process! For further details, *see* references 7 and 21

The algorithm to be presented for optimal-flow through networks with gains may also be thought of as a generalization of Ford and Fulkerson's techniques for simple networks, however, it retains much of the physical motivation and easy computational aspect of their method, and solves the imbedded linear program (a maximal-flow problem) by means of a labelling technique, and not the simplex technique

Conceptually, this primal-dual procedure consists of the following steps

1. Find the incrementally cheapest loop in the network which will absorb flow from the source

2. Establish as much additional flow into the network as possible, satisfying the conservation and flow capacity restrictions

3. Repeat Steps **1** and **2** until the desired input flow is established, or until infeasibility is discovered

The first description of this procedure was given by the author in 1958 [19]

THE MAXIMAL-FLOW SUBROUTINE

As in the simple networks of Ford and Fulkerson, the imbedded linear program that decreases the infeasibility of the primal in our algorithm is allowed to have only one unsatisfied constraint, the input flow equation. Since the requirement of complementary slackness will keep all inactive branches empty, and all hyperactive branches saturated, the primal-dual subroutine is an incremental maximal-flow procedure for a restricted network of active branches.

Appendix A describes the labelling procedure that can be used to solve the problem of maximal flow into a network with gains. Although this subroutine is closely related to the labelling technique of Ford and Fulkerson, additional complexities were introduced by the structures in Fig. 4, for this reason, it was expedient to split the procedure into three parts.

1. A labelling procedure is used to detect network structures that can absorb flow into the network.

2. A *transfer factor* calculation determines how much flow can be absorbed into this structure, and calculates the changes in branch flows, and in the input flow.

3. A dual calculation procedure uses the results of Steps **1** and **2** to determine the variables dual to the restricted primal.

It is primarily the simplicity of this maximal-flow procedure that makes this special-purpose primal-dual method so efficient in hand computations. Usually, only one or two new active branches are added to the restricted network in each cycle of the algorithm, and it is a simple matter to determine if additional flow can be absorbed into a structure including these new branches.

It is not yet known if this procedure will remain as effective in automatic digital computations as, say, a general-purpose simplex routine, experience with optimal-flow routines for networks with unity gains is encouraging.

THE OPTIMAL-FLOW ALGORITHM FOR NETWORKS WITH GAINS

The algorithm to be described has the following program.

Begin → **1.** Select an initial feasible solution to the dual of the optimal-flow problem.

Cycle

→ **2.** Using the principle of complementary slackness, define a restricted primal problem, and identify the primal variables that can or cannot be changed in the next step.

3. Maximize flow into the restricted primal network by using the maximal-flow subroutine, thus solving the restricted primal and its dual.

Optimal solution ← **4.** If the flow input equals Q, the optimal solution has been reached. Terminate the algorithm.

5. Otherwise, use the solution to the dual restricted problem to define possible changes in the optimal-flow dual variables, thus obtaining a new feasible dual. Repeat Step 2.

Problem infeasible ⟵ 6. However, if no such changes can be made in the dual, the optimal-flow problem is infeasible. Terminate the algorithm.

Step 1

To begin, select a feasible starting solution to the dual of the optimal-flow problem. If all of the unit costs are nonnegative, a simple choice is to set all dual variables equal to zero.

Step 2

From Step 1, or from the output of the previous cycle (Step 5), define the following mutually exclusive and collectively exhaustive states for each branch.

A branch is *hyperactive* if $U_{ij}>0$ (6a)

A branch is *active* if $V_i-K_{ij}V_j-U_{ij}=C_{ij}$ and $U_{ij}=0$ (6b)

A branch is *inactive* if $V_i-K_{ij}V_j-U_{ij}<C_{ij}$ and $U_{ij}=0$ (6c)

Define the *restricted primal problem*

$$\text{Maximize } F_0 \tag{7}$$

Subject to

$$\textstyle\sum_j (X_{ij}-K_{ji}X_{ji})=\begin{cases}F_0 & (i=0) \quad (8a)\\ 0, & (i\neq 0) \quad (8b)\end{cases}$$

$$0\leqq X_{ij}\leqq M_{ij} \text{ for all active branches,} \tag{8c}$$

$$0=X_{ij} \quad \text{for all inactive branches,} \tag{8d}$$

$$X_{ij}=M_{ij} \quad \text{for all hyperactive branches} \tag{8e}$$

The *dual restricted problem* is

$$\text{Minimize } \textstyle\sum_{i,j} M_{ij}\sigma_{ij} \tag{9}$$

subject to

$$\sigma_i-K_{ij}\sigma_j-\sigma_{ij}\leqq 0 \text{ for all active or hyper-active branches,} \tag{10a}$$

$$\sigma_i=\begin{cases}+1 & (i=0) \quad (10b)\\ \text{unrestricted,} & (i\neq 0) \quad (10c)\end{cases}$$

$$\sigma_{ij}\geqq 0 \text{ for all active branches} \tag{10d}$$

Step 3

Solve the restricted primal and its dual by using the maximal-flow subroutine described in Appendix A, an efficient starting solution is the X_{ij} determined on the last cycle of the algorithm. The output of the subroutine will be X_{ij} that satisfy restrictions (8), and σ_i and σ_{ij} that satisfy restrictions (10), complementary slackness will be maintained

Step 4

If F_0 attained the desired value Q in Step **3**, the algorithm is terminated, with the X_{ij} just found as the optimal solution to the optimal-flow problem, (1) (2) The dual feasible solution at the beginning of this cycle is the optimal solution to the dual optimal-flow problem, (3) (4)

Step 5

Otherwise, find new feasible variables, V_i' and U_{ij}', to the dual optimal-flow problem

$$V_i' = V_i + \vartheta\sigma_i, \qquad (i,j=0,1,2,\quad,N) \tag{11a}$$

$$U_{ij}' = U_{ij} + \vartheta\sigma_{ij}, \tag{11b}$$

with

$$\vartheta = \min\{\min[C_{ij} - (V_i - K_{ij}V_j - U_{ij})]/(\sigma_i - K_{ij}\sigma_j - \sigma_{ij}), \min U_{ij}/-\sigma_{ij}\} \tag{11c}$$

for all branches such that the denominators in (11c) are positive, if none of the denominators are positive for any of the branches in the network, set $\vartheta = +\infty$, and go to Step **6** Otherwise, ϑ is positive and finite, begin a new cycle of the algorithm at Step **2**, using the new dual variables

Step 6

If $\vartheta = +\infty$, terminate the algorithm, since no feasible solution to the optimal-flow restrictions exists, and the dual functional is unbounded The maximal flow into the network is $F_0 < Q$

COMPUTATIONAL EXTENSIONS

THE ONLY reason that the branch unit costs were considered to be non-negative was to get started in Step **1** of the optimal-flow algorithm If a problem arises in which some of the unit costs are negative, one should first try to find a feasible dual with the U_{ij} all equal to zero (so that one may use an initial restricted primal in which all flows are zero) If this still can not be done, then one must find any feasible dual, and a restricted primal flow that obeys the complementary slackness relations, various devices are available to do this with more or less difficulty [7] Of course, if *all* costs are nonpositive, the initial solutions are again easily found after a change of variable

Costs need not be linear, a convex cost function may be approximated to any desired degree of accuracy by a piecewise-linear function [5] In network problems, this results in several parallel branches with increasing unit costs, and one can easily modify the algorithm to suit An important application of this approximation occurs in risk situations, when certain output flows ('demands') will be samples from a known probability distribution The a priori risk that one should assume to maximize return over many trials will be given by the linear programming solution, if one computes the total cost as a function of different policies, and makes a piecewise-linear approximation for the demand branches [4]

Lower bounds on flow in a branch (i,j) may be incorporated in the algorithm by making the minimal flow from node i and into node j separate

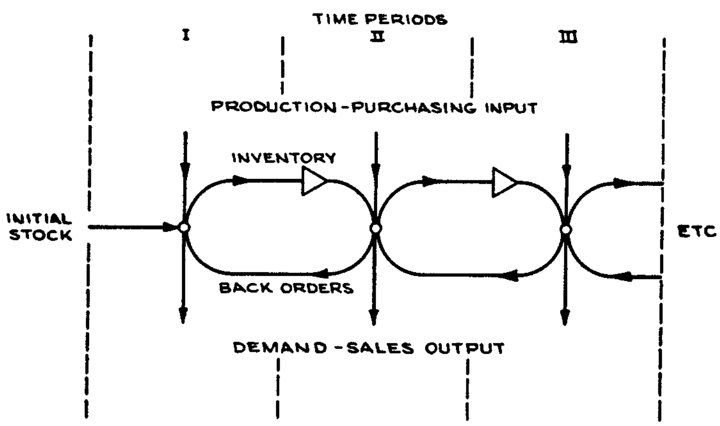


Fig. 6. Production and warehousing problem with evaporation

flow boundary conditions, and letting flow in the branch represent incremental flow above the lower bound Appropriate changes in capacity must be made

Parametric studies with flow as a parameter may be done directly, since each stage of the algorithm furnishes the optimal solution for some input flow between zero and Q Then, any optimal solution may serve as a starting solution for a new computation in which different flow conditions are applied at other nodes

A modification of the algorithm presented here will also handle cost and capacity-parametric studies, because of the inherent relation between parametric programming and the primal-dual method [21 27] The basic idea involves the 'extraction' of the branch of interest from the rest of the network, and the variation of both nodes' dual variables so that flow from the network can cancel or augment current flow in the branch

Dynamic flow in networks may be approximated by duplicating the network for each sampling period, flow in each replica is assumed to occur instantaneously, transition branches between the smaller networks represent storage from one time period to another [9,12]

If there are only a few connecting branches between subnetworks, it may be more efficient to 'tear' the network apart and solve the smaller problems separately The pieces are then fitted together by using a cost-parametric procedure for the connecting branches

The author hopes to describe some of these special-purpose computational devices in greater detail in the near future For some idea of the possible variety of special features that can be incorporated into network

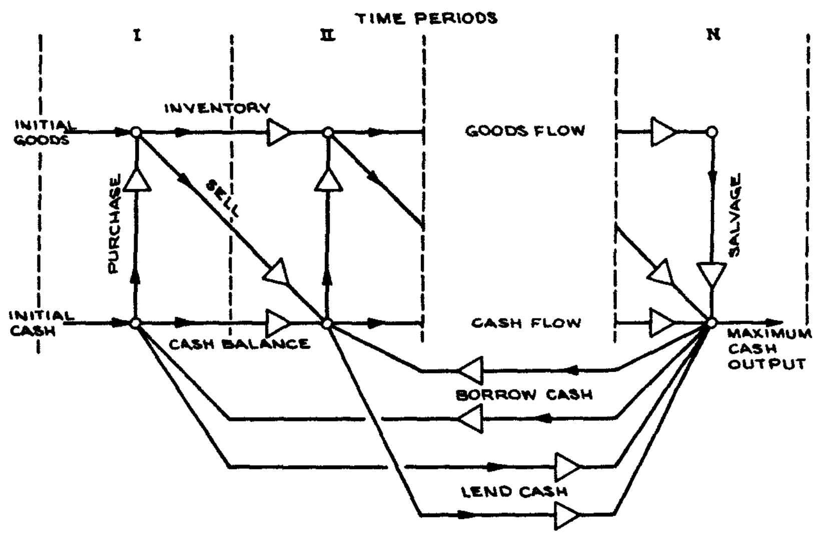


Fig. 7. Financial budgeting in a warehouse operation

flow models, the reader is referred to the references and the forthcoming book by Ford and Fulkerson [13]

PROBLEM EXAMPLES

To ILLUSTRATE the types of problems that can be formulated with the help of networks with gains, we consider the networks of several typical operational problems

Figure 6 illustrates the structure of a simple production and warehousing problem,[20] where 'breeding' or 'evaporation' occurs from one time period to the next Boundary conditions are not explicitly shown, since they may be inequalities or equalities, depending upon the problem For clarity, no flow, dual, cost, or capacity parameters are shown

Certain kinds of capital budgeting problems[3] may be expressed as

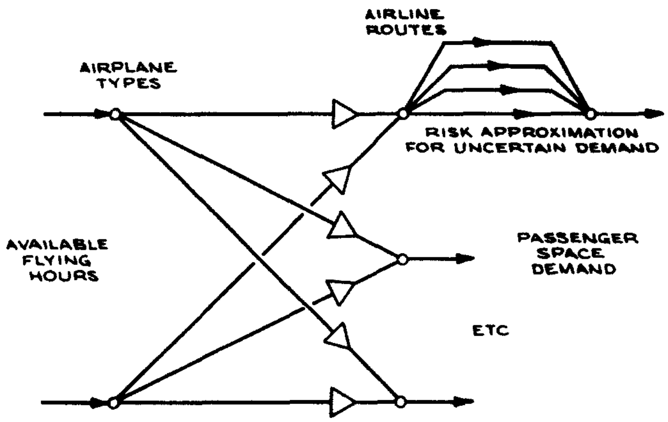


Fig. 8. Aircraft routing (machine loading) problem with demand risk

flow with gains (Fig 7) In the example shown, one is interested in the strategy of managing a warehousing operation and an initial amount of assets Given the various cash investment and loan possibilities, together with the conversion rates between goods and cash (and vice-versa) for the coming planning period, one attempts to maximize the amount of cash available at the end of N periods Various features such as policy limits on maximum and minimum cash and goods flows, alternative investments, etc , may similarly be added

Figure 8 illustrates the network encountered in Dantzig's study of aircraft routing under uncertainty (actually, a risk situation) [6] The solution to this problem tells what aircraft to assign to which routes, and

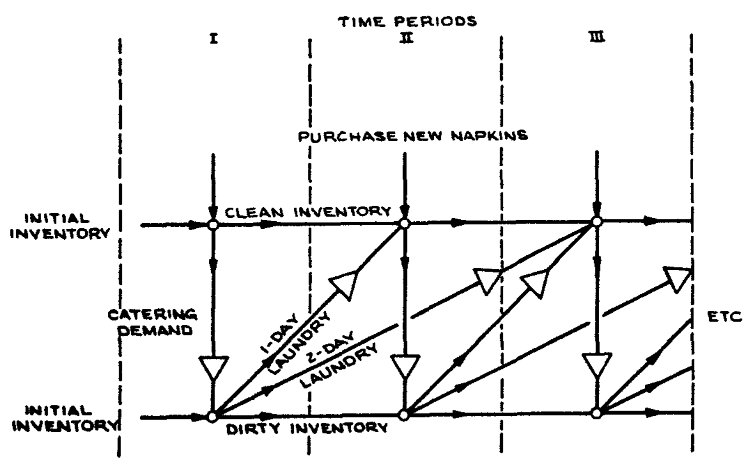
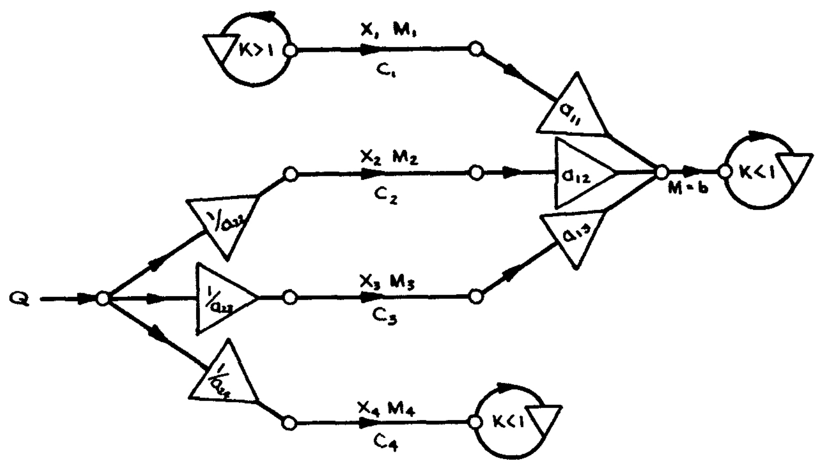


Fig 9 Catering (overhaul) problem with losses

what a priori passenger demand to assume for each route This same network is also encountered in the machine loading problem,[1, 2, 23, 28] in which several machines can be used to produce different parts at varying conversion efficiencies (parts per hour) and costs

The catering, or overhaul, problem has the structure shown in Fig 9, when gains are added to take care of proportional losses in catering, or in laundering For the statement and assumption of this model, the



$$\text{MIN} \quad C_1X_1 + C_2X_2 + C_3X_3 + C_4X_4$$

SUBJECT TO

$$a_{11}X_1 + a_{12}X_2 + a_{13}X_3 \leq b$$

$$a_{22}X_2 + a_{23}X_3 + a_{24}X_4 = Q$$

$$0 \leq X_i \leq M_i \quad (i = 1,2,3,4)$$

Fig 10 The two-equation capacitated linear program

reader is referred to references 16, 18, and 26, usually the flow in the vertical 'demand' branches is constrained by an equality

An idea of the inherent limitations of the flow with gains formulation may be obtained by considering Fig 10, which shows the network obtained for the two-equation capacitated linear program [24] Conceptually, one may imagine that the variables X_1, $\ldots$, X_4 'flow' from one restriction to the other, since a branch of a network only has two ends, it can be seen that there would be difficulties in trying to solve a more general linear program by using the network formulation

Nevertheless, the author feels that the special-purpose algorithm presented in this paper will prove to be quite valuable in the analysis of operational problems with flow-like processes It is hoped that the approach taken here will stimulate more investigation of special structures in mathematical programming

APPENDIX A· THE MAXIMAL-FLOW SUBROUTINE FOR NETWORKS WITH GAINS

THIS SECTION describes the maximal-flow subroutine, as used in the optimal-flow algorithm For use separately as a maximal-flow procedure, assume all branches in the network are active, eliminate the determination of the dual variables in Step **11**, and set $Q=+\infty$, in order to obtain as much flow as possible

Absorbing Network Detection

1. Beginning with any set of flow variables X_{ij} satisfying the restricted primal (8) (an efficient set is the set of flow variables established as maximal-flow during the previous cycle of the algorithm), set the label of the source to

$$L_0=(-|+1)$$

2. Starting at any node i which has previously been labelled with the form $L_i=(h|K_i)$, consider all other labelled or unlabelled nodes j that have not previously been examined in this pass through this Step, and that are connected to node i by an active branch (i,j) or (j,i)

a If there are no such branches, go to Step **10** of this subroutine Otherwise,

b If

i branch (i,j) is active and empty and $K_i<0$,

ii or if (i,j) is active and saturated and $K_i>0$,

iii or if (j,i) is active and empty and $K_i/K_{ji}>0$,

iv or if (j,i) is active and saturated and $K_i/K_{ji}<0$, then *no* labelling of node j is possible Repeat Step **2** for some other active branch (i,j) or (j,i)

c Otherwise, node j *is* a labelling possibility Calculate this tentative label for node j as

$$L_j=(i|K_j=K_i\,K_{ij})\ \text{if}\ (i,j)\ \text{is active,}$$

or

$$L_j=(i|K_j=K_i/K_{ji})\ \text{if}\ (j,i)\ \text{is active}$$

We now test to see if this tentative label should be assigned

3. If node j does not already have a label, assign the label just found in Step **2c**, and repeat Step **2** for some other active branch

4. Otherwise, node j already has a label, which we denote $L_j^{(1)}$, call the new tentative label $L_j^{(2)}$

a If $K_j^{(1)}$ and $K_j^{(2)}$ are of opposite sign, go to Step **5** of this subroutine

b Otherwise, $K_j^{(1)}$ and $K_j^{(2)}$ are of the same sign, since neither of them can be zero Compare their absolute magnitudes, then, consider the effect of erasing the label that has the largest absolute value of K_j, realizing that if a label is erased, all labels that proceeded from it in Step **2** must also be erased

i If erasing the label with larger absolute value of K_j also causes the other labelling candidate to be erased (i e , the newly found alternative label of Step 2c is a 'later' member of a labelling sequence from the source that has the original label as an 'earlier' member), then proceed to Step 5 of this subroutine

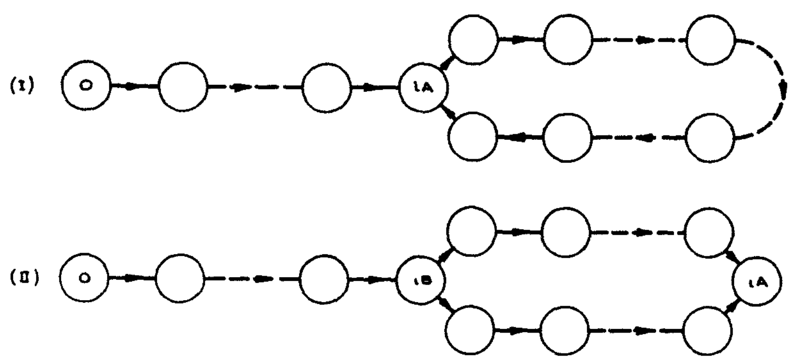
ii If erasing the label with larger absolute value of K_j does *not* cause the other labelling candidate to be erased, then perform that erasure, placing the label with smaller absolute value of K_j upon node j Erase all labels that proceeded from the erased label, if any, by following the indices in the labels Repeat Step 2 for some other active branch

iii If the two possible K_j are identical, the choice of label is immaterial, for simplification, leave the original label on the node Repeat Step 2 of this subroutine for some other active branch

(Notice that many alternate labels for a given node may be tested during the sub-cycle from Steps 2 through 4, in fact, different labels may be transmitted from any one node to another as the labels are changed elsewhere in the network)

Absorbing Network Calculation

5. An increase in flow into the network is possible through the structure detected in Steps 4a or 4bi above One node has two alternate labels, call this node i_A Identify the absorbing structure by considering those branches with both nodes labelled, and where the branch is the path by which the output node is labelled from the input node, or vice-versa Beginning at the source, follow the labels forward along such branches until one of the following two sequences of nodes is found



In these diagrams, the circles refer to the nodes of the network, and the arrows indicate the direction of labelling, not flow Notice that the general form for an absorbing network is a simple chain of branches (possibly of zero length) plus a loop-like structure in which either *

---

* Structure II is, of course, a special case of Structure I, since the product of incremental gain around either loop is less than unity However, it is advantageous to recognize possible absorption structures as soon as possible, rather than having to modify labels all the way to i_B from i_A

I Node i_A is a 'feedback' point for a one-way loop,

II Or, node i_A is the common terminal point for two parallel 'feed-forward' paths which began at some common branching node, i_B

If this Step is reached from Step 4bi of this subroutine, then only structure I will be found, if this Step is reached from Step 4a, then either structure may occur Note that for each node of a branch in the simple chain, there is only one path to the source that does not pass again through that node, however, for all nodes whose branches are in the loop-like portions (excluding the feedback or feedforward branching nodes), there are exactly two such paths to the origin

6. Having identified the absorbing structure, compute a *transfer factor*, T_{ij}, for every branch in the structure, by first calculating two preliminary numbers, T_i and T_j

a For every branch (i,j), begin at the input node i If all paths from node i to the source (via the absorbing structure) contain the branch (i,j), set $T_i = 0$ Otherwise, starting with the factor unity, proceed from node i to the source [not via (i,j)], dividing the current value of the factor by the gain of each branch traversed in the reverse sense, and multiplying by the gain of each branch traversed in the forward sense The resulting product of incremental gains upon reaching the source is defined as T_i for that branch (i,j)

b Begin at the output node j of the same branch, and proceed as in Step **6a** to trace out the path [not via (i,j)] in the absorbing structure that leads to the source If there is no such path, then $T_j = 0$ Otherwise, T_j is the product of all incremental gains encountered 'en route' from node j to the source

c Finally, for the branch under consideration, calculate the transfer factor as

$$T_{ij} = T_i - K_{ij}\, T_j$$

This transfer factor can be interpreted as an 'incremental reciprocal-transmission ratio', it may be either positive or negative (There are also ways to compute the T_{ij} from the K_i found in the labelling procedure, details are left to the reader)

d Repeat Steps **6a** through **6c** for all branches in the absorbing structure

7. Compute the maximum possible increase into the absorbing structure

$$f_0 = \min\{(Q - F_0), \min_{T_{ij}>0}[T_{ij}(M_{ij} - X_{ij})], \min_{T_{ij}<0}[(-T_{ij}\, X_{ij})]\},$$

where the minimization is taken over all (i,j) in the absorbing structure

8. Increase the flow X_{ij} in each branch (i,j) of the absorbing structure by the positive or negative increment x_{ij}, found to be

$$x_{ij} = f_0(T_{ij})^{-1},$$

and increase the total input flow F_0 by an amount f_0 This change of flow will do at least one of the following things

a Establish the desired input flow,

b Saturate at least one active branch in the absorbing structure,

c Or empty at least one active branch in the absorbing structure

9. If the flow into the network is now Q, the optimal flow is established, go to Step **4** of the main algorithm If the desired flow is not yet obtained, erase all labels except that of the source, and begin a new cycle of the subroutine at Step **2**, in order to search for new paths that will absorb flow (It is possible to save some of the labels from this cycle for use in the next, details are left to the reader)

10. No new nodes can be labelled in Step **2** via any active branch Those nodes with alternate labels either have all of the K_j equal, or they have the label with the smallest possible absolute value of K_j The optimal solution has not been reached, since the desired flow has not been achieved, a new cycle of the main algorithm is needed, and we must compute the variables of the dual to the max-flow problem just completed (A separate maximal-flow procedure terminates here, with F_0 as the maximum possible flow into the network)

Calculation of Variables Dual to Maximal-Flow Problem

11. At the end of the maximal-flow procedure in Step **10**, certain nodes have labels, which progressed from the source to those nodes via active branches If there are alternate labels for a node, they either have all K_j equal, or the node has the label affixed with the smallest absolute value of K_j Call the set of indices of the labelled nodes I, we shall now construct the variables σ_i and σ_{ij}, which solve the dual to the restricted primal, (9) and (10)

a Set $\sigma_0 = +1$

b For all active branches with $ij\varepsilon$II, where the label on one node is constructed from the label on the other node via (i,j) or (j,i), set

$$\sigma_i = (K_i)^{-1}, \qquad \sigma_j = (K_j)^{-1}, \quad \text{and} \quad \sigma_{ij} = 0$$

c For all active branches with $ij\varepsilon$II, where one node is not labelled from the other node via (i,j) or (j,i), but via some other path, σ_i and σ_j are already determined in Steps **11***a* and **11***b*

i If the branch is empty, set $\sigma_{ij} = 0$ (It can be shown that $\sigma_i - K_{ij}\,\sigma_j - \sigma_{ij} \leqq 0$ for these branches)*

ii If the branch is saturated, set $\sigma_{ij} = \sigma_i - K_{ij}\,\sigma_j$ (It can be shown that $\sigma_{ij} \geqq 0$ for these branches)*

iii It can be shown that there are no nonempty, nonsaturated branches in this category *

d For all active branches with $ij\varepsilon\mathrm{I}\bar{\mathrm{I}}$, σ_i is already determined

i Set $\sigma_j = 0$

ii If the branch is empty, set $\sigma_{ij} = 0$ (From Step **2***bi*, $\sigma_i < 0$, otherwise j would be labelled Therefore $\sigma_i - K_{ij}\,\sigma_j - \sigma_{ij} < 0$)

iii If the branch is saturated, set $\sigma_{ij} = \sigma_i - K_{ij}\,\sigma_j$ (From Step **2***bii*, $\sigma_i > 0$, otherwise j would be labelled Therefore $\sigma_{ij} > 0$)

iv There are no nonempty, nonsaturated branches in this category, otherwise j would be labelled

---

* Proof of the starred statements is by exhaustive examination of all possible combinations of labels and gains, and amounts of flow, the results indicated follow by contradiction, since alternate labellings would occur in the maximal-flow procedure While straightforward, space limitations prohibit reproducing all of these combinations here

e For all active branches with $ij\varepsilon,\bar{\mathrm{I}}\mathrm{I}$ the variable σ_j is already determined
  i Set $\sigma_i=0$
  ii If the branch is empty, set $\sigma_{ij}=0$ (From Step **2***biii*, $K_{ij}\,\sigma_j>0$, otherwise *i* would be labelled Therefore, $\sigma_i-K_{ij}\,\sigma_j-\sigma_{ij}<0$)
  iii If the branch is saturated, set $\sigma_{ij}=\sigma_i-K_{ij}\,\sigma_j$ (From Step **2***biv*, $K_{ij}\,\sigma_j<0$, otherwise *i* would be labelled Therefore, $\sigma_{ij}>0$)
  iv There are no nonempty, nonsaturated branches in this category, otherwise *i* would be labelled

f For all active branches with $ij\varepsilon\overline{\mathrm{II}}$, not all dual variables are determined in previous Steps Set $\sigma_i=\sigma_j=\sigma_{ij}=0$

g All hyperactive branches (i,j) are kept saturated during the maximal-flow subroutine Set any previously undetermined σ_i or $\sigma_j=0$, and set $\sigma_{ij}=\sigma_i-K_{ij}\,\sigma_j$ Note that σ_{ij} may be negative, positive, or zero

h All inactive branches (i,j) are kept empty during the maximal-flow subroutine Set any previously undetermined σ_i or $\sigma_j=0$, and set $\sigma_{ij}=0$ Note that $\sigma_i-K_{ij}\,\sigma_j-\sigma_{ij}$ may be negative, positive, or zero

i All dual variables are now defined, since all branches are either active, hyperactive, or inactive Return to Step 4 of the optimal-flow algorithm

APPENDIX B · PROOF OF THE MAXIMAL-FLOW SUBROUTINE AND THE OPTIMAL-FLOW ALGORITHM

An induction argument will be used to show that the restricted primal constraints and the optimal-flow dual constraints are always satisfied Then, by demonstrating that the functional of the dual always increases in each cycle of the algorithm (until the desired flow is established, or infeasibility is discovered), it will be shown that the algorithm provides the optimal solution in a finite number of cycles

Proof that the subroutine maximizes flow into the restricted network at each cycle of the algorithm will consist of showing that a feasible solution to the dual restricted problem also exists, and that complementary slackness conditions are satisfied between the restricted primal and its dual

Proof

At the beginning of the algorithm, Step 1 selects a feasible solution to the dual of the optimal-flow problem If there are no hyperactive branches, a feasible starting solution for the restricted primal is to set all flows equal to zero, otherwise, a starting solution satisfying complementary slackness must be provided Later it will be seen that the maximal flow solution of one cycle provides a feasible starting solution for the restricted primal of the succeeding cycle

Now proceed with the induction argument, using unprimed symbols for *this* (arbitrary) cycle of the algorithm, and using primed symbols for the *next succeeding* cycle

For simplicity, define

$$\beta_{ij}=\sigma_i-K_{ij}\,\sigma_j-\sigma_{ij}$$

Lemma 1 *If a branch is active, then* $\sigma_{ij}\geqq 0$

From the maximal-flow subroutine, the only time $\sigma_{ij}<0$ is for hyperactive branches

LEMMA 2 *If a branch is not saturated, then* $\sigma_{ij}=0$

From the subroutine, no σ_{ij} is nonzero unless the branch is saturated

LEMMA 3 *If a branch is active or hyperactive, then* $\beta_{ij}\leqq 0$

From the subroutine, the only time $\beta_{ij}>0$ is for inactive branches

LEMMA 4 *If a branch is not empty, then* $\beta_{ij}=0$

From the subroutine, no β_{ij} is nonzero unless the branch is empty

LEMMA 5 *The maximal-flow subroutine solves the restricted primal problem and its dual*

The restrictions (10) of the dual restricted problem are satisfied since $\beta_{ij}\leqq 0$ for all active and hyperactive branches, by Lemma 3, $\sigma_0=+1$ by Step **11***a* of the subroutine, and $\sigma_{ij}\geqq 0$ for all active branches, by Lemma 1 The restricted primal constraints (8) are assumed satisfied at the beginning of this cycle, and Lemma 10 will show that they remain satisfied at the end of the subroutine, hence the maximal-flow of this cycle may serve as an initial feasible flow for the next cycle

Also, complementary slackness conditions hold between the restricted primal and its dual, for if a branch is empty, $\beta_{ij}=0$ by Lemma 4, if a branch is not saturated, then $\sigma_{ij}=0$ by Lemma 2, and, the subroutine keeps hyperactive branches saturated, and inactive branches empty

Therefore, by the theorem of complementary slackness, optimal solutions to the restricted primal problem and its dual have been found

LEMMA 6 $F_0=\sum_{i,j} M_{ij}\,\sigma_{ij}$

Furthermore, by the fundamental theorem of linear programming, the two functionals (7) and (9) must be equal when the optimal solutions are reached

In the special case where all of the gains are unity (except at a special output 'sink'), all of the nonzero σ_{ij} are ± 1, according to their orientation across a 'cut set' which separates the source from the rest of the network This Lemma then specializes to the famous 'min-cut equals max-flow' Theorem [8]

Therefore, this Lemma provides a similar theorem for branches with gains, the 'cut set' is just the set of saturated branches, with the value of the cut set being a complicated weighted sum of the branch capacities in the set

LEMMA 7 *If* V_i *and* U_{ij} *satisfy the optimal-flow dual restrictions* (4) *in one cycle of the algorithm, then* V'_i *and* U'_{ij} *also satisfy these restrictions in the next cycle of the algorithm*

The new dual variables for the next cycle of the algorithm are given by

$$V'_i=V_i+\vartheta\sigma_i, \quad \text{and} \quad U'_{ij}=U_{ij}+\vartheta\sigma_{ij}$$

Dual inequality (4a) states that $V_i-K_{ij}V_j-U_{ij}\leqq C_{ij}$ Making the change of variables indicated, we find that the left-hand-side of (4a) increases by $\vartheta\beta_{ij}$ Now, if a branch was active or hyperactive in this cycle, then (4a) was an equality, by definition But by Lemma 3, $\beta_{ij}\leqq 0$ for these branches, so that inequality (4a) will hold in the next cycle for any nonnegative ϑ On the other hand, if the branch was inactive in this cycle, then (4a) was a strict inequality, by definition Therefore the inequality will still hold for $0<\vartheta\leqq\vartheta_1$, where

$$\vartheta_1=\min_{\beta_{ij}>0}\{[C_{ij}-(V_i-K_{ij}V_j-U_{ij})]/\beta_{ij}\}$$

or equals $+\infty$ if all $\beta_{ij}\leqq 0$

Dual inequality (4b) states that $U_{ij} \geqq 0$ In the new cycle, the left-hand-side increases by $\vartheta\sigma_{ij}$ Now if a branch is inactive or active, then $U_{ij}=0$ by definition But by Lemma 1 and Step 11*h* of the subroutine, $\sigma_{ij} \geqq 0$ for these branches, and inequality (4b) will hold in the new cycle for any nonnegative ϑ On the other hand, for branches that are hyperactive, $U_{ij}>0$ by definition Therefore the inequality will still hold in the next cycle for $0<\vartheta \leqq \vartheta_2$, where

$$\vartheta_2 = \min_{\sigma_{ij}<0}[U_{ij}/-\sigma_{ij}]$$

or equals $+\infty$ if all $\sigma_{ij} \geqq 0$

Since one starts with a feasible dual in Step 1 of the algorithm, and since Step 5 picks $\vartheta = \min(\vartheta_1, \vartheta_2)$, the dual inequalities (4) will remain satisfied during every cycle of the algorithm

Notice that if ϑ is finite, at least one change of state will occur at the end of the cycle, with either some inactive state becoming active, or some hyperactive state becoming active

LEMMA 8 *If a branch is hyperactive in the next cycle, it is saturated*

If the branch is hyperactive in this cycle, the subroutine leaves it saturated Otherwise it was active with $\sigma_{ij}>0$, which can only occur for saturated branches

LEMMA 9 *If a branch is inactive in the next cycle, it is empty*

If the branch is inactive in this cycle, the subroutine leaves it empty Otherwise it was active with $\beta_{ij}<0$, which only occurs for empty branches

LEMMA 10 *The maximal-flow solution of the previous cycle may be used as an initial solution in the new cycle*

Constraints (8a),(8b), and (8c) are always satisfied Lemmas 8 and 9 show that (8d) and (8e) are feasible for the next cycle

LEMMA 11 *The optimal solution to the restricted primal problem with a maximal flow F_0 provides a new feasible solution to the optimal-flow dual, with a strict increase in the functional (3) $=(Q-F_0)\vartheta$*

Lemma 7 demonstrates the feasibility of the new dual The increase in the dual functional is calculated to be $\vartheta(Q\sigma_0 - \Sigma\, M_{ij}\, \sigma_{ij})$ By Lemma 6 and (10b), the increase is just $\vartheta(Q-F_0)$, which is strictly positive

LEMMA 12 *If $\vartheta=+\infty$ in any cycle, the optimal-flow problem is infeasible, if $\vartheta<\infty$ in each cycle, F_0 will attain Q in a finite number of cycles, if F_0 attains Q, the algorithm terminates with the optimal solution to (1)(2) and (3)(4)*

The first result follows from the duality theorem, since the dual functional is unbounded The second result follows from the fact that only a finite number of different restricted primal problems is possible, since the dual functional (3) is strictly increasing in each cycle, and is bounded by its optimal value Using the complementary slackness relations we calculate that the two functionals (1) and (3) are equal, and optimality follows from the fundamental theorem of linear programming, when $F_0=Q$

ACKNOWLEDGMENTS

MOST OF the research on the theory of flow with gains was supported at the Massachusetts Institute of Technology by the Union Carbide Corporation through a Grant-in-Aid, and by the Operations Research Center, through the Army Office of Ordnance Research Contract DA-19-020-ORD-2684

Preparation and reproduction of this paper was supported by Broadview Research Corporation Travel assistance by the Council of the Operations Research Society of America is gratefully acknowledged

The author would also like to express his appreciation to the referees who helped to improve the content of the paper

REFERENCES

1 A Charnes, W W Cooper, and D Farr, and Staff, "Linear Programming and Profit Preference Scheduling for a Manufacturing Firm," *Opns Res* **1,** 114–129 (1953)
2 ——— and ———, "Management Models and Industrial Applications of Linear Programming," *Management Sci* **4,** 38–91 (1957)
3 ———, ———, and M H Miller, "Application of Linear Programming to Financial Budgeting and the Costing of Funds," *J Business Univ Chicago* **32,** 20–46 (1959)
4 G B Dantzig, "Linear Programming Under Uncertainty," *Management Sci* **1,** 197–206 (1955)
5 ———, "Recent Advances in Linear Programming," *Management Sci* **2,** 131–144 (1956)
6 ——— and A R Ferguson, "The Allocation of Aircraft to Routes—An Example of Linear Programming Under Uncertain Demand," *Management Sci* **3,** 45–73 (1956)
7 ———, L R Ford, and D R Fulkerson, "A Primal-Dual Algorithm for Linear Programs," *Linear Inequalities and Related Systems*, Princeton University Press, Princeton, 1956
8 ——— and D R Fulkerson, "On the Min-Cut Max-Flow Theorem of Networks," *Linear Inequalities and Related Systems*, Princeton University Press, Princeton, 1956
9 ———, "On the Status of Multistage Linear Programming Problems," *Management Sci* **6,** 53–72 (1959)
10 L R Ford and D R Fulkerson, "Solving the Transportation Problem," *Management Sci* **3,** 24–32 (1956)
11 ——— and ———, "A Primal Dual Algorithm for the Capacitated Hitchcock Problem," *Naval Res Log Quart* **4,** 47–54 (1957)
12 ——— and ———, "Constructing Maximal Dynamic Flow from Static Flows," *Opns Res* **6,** 419–433 (1958)
13 ——— and ———, *Network Flow Theory*, to appear
14 D R Fulkerson, "Increasing the Capacity of a Network, the Parametric Budget Problem," *Management Sci* **5,** 472–483 (1959)
15 ——— and G B Dantzig, "Computation of Maximal Flows in Networks," *Naval Res Log Quart* **2,** 277–283 (1955)
16 J W Gaddum, A J Hoffman, and D Sokolowsky, "On the Solution of the Caterer Problem," *Naval Res Log Quart* **1,** 223–229 (1954)
17 A J Goldman and A W Tucker, "Theory of Linear Programming," *Linear Inequalities and Related Systems*, Princeton University Press, Princeton, 1956
18 W Jacobs, "The Caterer Problem", *Naval Res Log Quart* **1,** 154–165

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

20 S M JOHNSON, "Sequential Production Planning Over Time at Minimum Cost," *Management Sci* **3,** 435–437 (1957)

21 J E KELLEY, JR , "Parametric Programming and the Primal-Dual Algorithm," *Opns Res* **7,** 327–334 (1959)

22 H W KUHN, "The Hungarian Method for the Assignment Problem," *Naval Res Log Quart* **2,** 83–97 (1955)

23 H MARKOWITZ, "Concepts and Computing Procedures for Certain X_{ij} Programming Problems," *Proceedings of the Second Symposium in Linear Programming 1* and *2*, Directorate of Management Analysis, DCS/Comptroller, Headquarters, U S Air Force, Washington, D C

24 L G MITTEN, "An Algorithm for a Restricted Class of Linear Programming Problems," talk presented at the Fifth Annual Meeting of ORSA, 9–10 May, 1957, Philadelphia

25 A ORDEN, "The Transshipment Problem," *Management Sci* **2,** 276–285 (1956)

26 W PRAGER, "On the Caterer Problem," *Management Sci* **3,** 15–23 (1956)

27 T L SAATY AND S I GASS, "The Parametric Objective Function," *Opns Res* **3,** 395–405 (1955)

28 M E SALVESON, "A Problem in Optimal Machine Loading," *Managemen Sci* **2,** 232–260 (1956)