J. Soc. Indust. Appl. Math.
Vol. 10, No. 2, June, 1962
Printed in U.S.A.

AN APPLICATION OF GENERALIZED LINEAR PROGRAMMING TO NETWORK FLOWS*

R. E. GOMORY† AND T. C. HU†

1. Introduction. As in [1] we consider a communication net or a network of nodes or terminals connected by links. Associated with the link from node i to node j is a branch capacity y_{ij}. Using the usual definition of the maximal flow from node i to node j (see [2, 3]) in such a network we denote the value of this maximal flow by f_{ij}.

There are several problems associated with the connection between the y_{ij} and the f_{ij} of an n-node network. One problem, which we call analysis is this: given the y_{ij}, find the f_{ij}. An economical way of doing this is given in [1]. The reverse problem which we call synthesis and which is explained below, is treated by special methods applicable only to a special case in [1] and more generally here by techniques from linear programming, similar to those of [4, 5, 6].

The problem we consider is: given nonnegative numbers r_{ij} and c_{ij}, $i = 1, \cdots, n$, $j = 1, \cdots, n$, called the flow requirements and the costs, respectively, construct a network such that the maximum flow values f_{ij} satisfy

$$f_{ij} \geqq r_{ij} \qquad \text{(all } i, j\text{)}$$

and such that the cost M

$$M = \sum_{i \neq j} c_{ij} y_{ij}$$

is minimized.

We will assume throughout that $y_{ij} = y_{ji}$, $r_{ij} = r_{ji}$, $c_{ij} = c_{ji}$. The numbers c_{ii}, y_{ii}, r_{ii} play no role and need not be given.

We start with the notion of cut.

Let N be the set of n nodes in a given network. Following [2], we shall define a cut in a network separating nodes i and j as a set of links connecting the sets A and $\bar{A}$, where $\bar{A}$ is the complement of A with respect to N, and such that $i \in A$ and $j \in \bar{A}$. The capacity of a cut is equal to the sum of the branch capacities of these links. A link is considered to connect sets A and $\bar{A}$ if it connects two nodes, one in A and the other in $\bar{A}$.

It is shown by Ford and Fulkerson [2] that the value f_{ij} in a network is equal to the minimum among the capacity of all cuts which separate nodes i and j. A formulation of the problem of synthesis using linear programming

* Received by the editors February 2, 1961 and in revised form September 5, 1961.
† IBM Research Center, Yorktown Heights, N. Y.

is to minimize $\sum c_{ij}y_{ij}$ subjected to the constraints that the capacity of all cuts separating nodes i and j must not be less than r_{ij}, and that this must be true for all pairs of nodes. See for example [7]. This results in an enormous linear programming problem ($2^{n-1} - 1$ constraints) which we now proceed to simplify.

Two algorithms are developed which avoid treating all inequalities simultaneously. One algorithm is essentially the dual simplex method but treats only $n - 1$ inequalities at a time, plus the usual requirements that the variables be nonnegative. The other algorithm is related to the primal simplex method. These methods are described in §3 and §4.

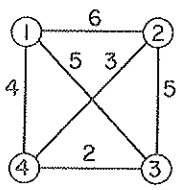
A first step toward a feasible computational procedure is to show that there are $n - 1$ dominating flow requirements, the fulfillment of which would imply the fulfillment of all other flow requirements. This is shown in the next section and in [1].

2. Dominating requirement tree. Let a set of n points connected by links be given together with a positive real number assigned to every link. Two points without a link connecting them directly can be considered as having a link with the value zero. A spanning tree of the n points is defined as a tree of $n - 1$ links connecting all n points. A maximum spanning tree is such a tree with the sum of the values on the links in the tree maximum. Let v_{ij} be the value assigned to the link connecting points i and j. And let a sequence of links in a maximum spanning tree connecting the points a, b, c, d, e, have values $v_{ab}, v_{bc}, v_{cd}, v_{de}$. Then it follows from the definition of a maximum spanning tree that

$$v_{ae} \leqq \min (v_{ab}, v_{bc}, v_{cd}, v_{de}), \tag{2.1}$$

i.e., the value of a link connecting points i and j not in a maximum spanning tree is less than or equal to the smallest value of the links in the maximum spanning tree on a path connecting i and j. For suppose (2.1) is not true and $v_{ae} > v_{bc}$ say; then we can omit the link connecting points b and c and put the link connecting points a and e into the maximum spanning tree and the result will be a spanning tree with a sum greater than before. This contradicts the assumption that the original tree is a maximum spanning tree. Efficient algorithms for constructing the maximum spanning tree are given by Kruskal [8] and Prim [9]. Essentially these algorithms first select the link with the largest value, then of those remaining, the links with the largest value, etc. At each stage the links eligible for selection are those (a) not already chosen and (b) not forming a loop when added to those that have been chosen. After $n - 1$ choices the maximal tree is obtained.

Now, nodes (or terminals) of a communication network can be considered as points and the flow requirement r_{ij} of two nodes can be considered as the



	①	②	③	④
①	d	6	5	4
②	6	d	5	3
③	5	5	d	2
④	4	3	2	d

(c)

FIG. 1. *Flow requirements.*

value assigned to the link connecting the two points in the above discussion. For example the maximum spanning tree for the set of terminals shown in Fig. 1(a) is drawn in Fig. 1(b) in unbroken lines.

A maximum spanning tree for terminals, with the r_{ij} as link values is called a *dominating requirement tree*. It is shown in [1] that it is necessary and sufficient for the synthesis problem to consider only the $n - 1$ flow requirements in the dominating requirement tree. We recapitulate briefly the reasons.

As we show in [1] the maximum flow numbers of a network always satisfy the inequality

$$f_{ae} \geqq \min (f_{ab}, f_{bc}, f_{cd}, f_{de}) \tag{2.2}$$

where the subscripts $a, b, \cdots, c, e$ involved on the right in (2.2) are any sequence starting with a and ending with e.

Because of this we have the following. Suppose a network has flows f_{ij} satisfying the requirements for all cases where the requirement belongs to the dominant requirement tree T, i.e.,

$$f_{ij} \geqq r_{ij} \qquad (\text{all } r_{ij} \in T).$$

Then consider any $r_{ae} \notin T$ and a path $a, b, c, \cdots, e$ within T from a to e. We have from (2.1)

$$r_{ae} \leqq \min (r_{ab}, r_{bc}, \cdots, r_{de}) \tag{2.3}$$

and from (2.2)

$$f_{ae} \geqq \min (f_{ab}, f_{bc}, \cdots, f_{de});$$

so if the requirements within T are satisfied, i.e.,

$$f_{ab} \geqq r_{ab}, \qquad f_{bc} \geqq r_{bc}, \qquad \ldots \text{ etc.},$$

it follows that

$$f_{ae} \geqq r_{ae};$$

so the requirements not in the tree are satisfied as well.

Since it is of course necessary to satisfy the $n - 1$ requirements in the dominant tree, we have now shown it to be both necessary and sufficient, and in the remainder of this paper we will, in synthesizing our networks, pay attention only to the $n - 1$ dominating requirements

$$f_{ij} \geqq r_{ij}\,, \qquad (r_{ij} \in T).$$

It is worth noting that whenever in (2.3) we have strict inequality

$$r_{ac} < \min\,(r_{ab}\,, r_{bc}\,, \cdots\,, r_{de}) = \bar{r}_{ac}\,,$$

the process of satisfying the dominant requirements will produce a flow $f_{ac} \geqq \bar{r}_{ac}$. Consequently such r_{ac} can be replaced by the larger requirements $\bar{r}_{ac}$ throughout with no increase in cost. Once this replacement is made, there are, due to tie values among the new requirements (and the values among the old requirements), several different dominant trees which could be used. This fact is occasionally useful.

Figure 1(b) shows the new requirements $\bar{r}_{ac}$ in parentheses. An alternate dominating tree consists of the links 1-2, 2-3, and 3-4.

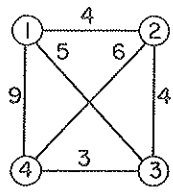
For example in Fig. 1 (b) we will only consider the requirements given by the unbroken lines.

There is one further reduction which can be made, sometimes, before starting the algorithm. We define a term "unreasonable cost" which may help to reduce the number of variables involved in the computation. A cost c_{ij} is called unreasonable if there is a path in the network from i to j such that

$$c_{ij} \geqq c_{ia} + c_{ab} + \cdots + c_{dj}\,. \tag{2.4}$$

If the corresponding y_{ij} is nonzero in the optimal solution, it can be replaced by a series of variables $y_{ia}\,, \cdots\,, y_{dj}$ corresponding to the right-hand side in (2.4) with no increase in cost. Consequently the variable y_{ij} can be eliminated from the problem.

In Fig. 2(a) for example, c_{14} is an unreasonable cost as $c_{14} > c_{13} + c_{34}$.



(a)

	1	2	3	4
1	d	4	5	9
2	4	d	4	6
3	5	4	d	3
4	9	6	3	d

(b)

FIG. 2. *Unit cost matrix.*

3. Algorithm I. The method used here is basically identical with Wolfe's notion of generalized linear programming.

Since the notation of the simplex method differs in various texts, the particular notation used here will be explained briefly; see [10] for more detail. We shall illustrate this first by a numerical example, and then follow it with the general statement. Suppose we want to minimize

$$4y_{12} + 5y_{13} + 4y_{23} + 6y_{24} + 3y_{34} \tag{3.1}$$

subject to

$$\begin{aligned} y_{12} + y_{13} &\geqq 6 \\ y_{12} + y_{23} + y_{24} &\geqq 5 \\ y_{24} + y_{34} &\geqq 4 \end{aligned} \tag{3.2}$$

and $y_{ij} \geqq 0$.

Then in our notation

$$z = -4y_{12} - 5y_{13} - 4y_{23} - 6y_{24} - 3y_{34} \tag{3.3}$$

will be the objective function that is to be maximized. We also intruduce slack variables

$$\begin{aligned} s_{12} &= -6 + y_{12} + y_{13} \\ s_{23} &= -5 + y_{12} + y_{23} + y_{24} \\ s_{14} &= -4 + y_{24} + y_{34}\,. \end{aligned} \tag{3.4}$$

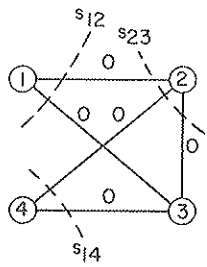
These slack variables, just as the y_{ij}, are required to be nonnegative. In addition to (3.3) and (3.4), we shall introduce trivial relations

$$y_{ij} = -(-y_{ij}). \tag{3.5}$$

The relations (3.3), (3.4) and (3.5) are tabulated in Table A1.

TABLE A1

	1	$-y_{12}$	$-y_{13}$	$-y_{23}$	$-y_{24}$	$-y_{34}$
z	0	4	5	4	6	3
y_{12}	0	−1	0	0	0	0
y_{13}	0	0	−1	0	0	0
y_{23}	0	0	0	−1	0	0
y_{24}	0	0	0	0	−1	0
y_{34}	0	0	0	0	0	−1
s_{12}	−6	−1*	−1	0	0	0
s_{23}	−5	−1	0	−1	−1	0
s_{14}	−4	0	0	0	−1	−1



In Table A1, y_{ij} in the top row are independent (or nonbasic) variables, and all variables appearing in the first column are expressed in terms of these independent variables.

This tabulation is, in matrix form,

$$X = A^0 Y^0 \tag{3.6}$$

where X is the column vector with z as first component and y_{ij}, s_{ij} the other components. A^0 is the rectangular matrix with all the constants in Table A1 as elements and Y^0 is a column vector with the negatives of independent variables as components.

The fundamental operation used in the simplex method is pivoting or Gaussian elimination on rows. Here this means that a constant a_{ij} in the ith row and jth column is selected as "pivot" element. Then the ith component of the vector X will now be used as a new independent variable, instead of the jth component of Y^0. This change of variable is accomplished by a Gaussian elimination on the pivot element. The simplex method consists of a series of pivot steps (or equivalently, choosing different sets of nonbasic variables). This will produce a series of new equations like (3.6)

$$X = A^k Y^k \tag{3.7}$$

in which Y^k represent the negative of variables that are independent after the kth pivot step, and A^k is the new matrix of constants. The optimum solution, as is well known, is obtained when A^k has the following properties

$$a_{0j} \geqq 0 \qquad (\text{all } j \neq 0) \tag{3.8}$$

$$a_{i0} \geqq 0 \qquad (\text{all } i \neq 0) \tag{3.9}$$

the solution being $X = (a_{00}, a_{10}, a_{20}, \cdots)$.

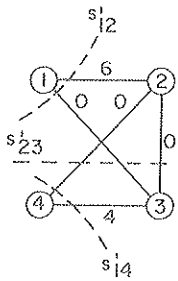
During the series of pivot steps, a trial solution exists for each A^k. This trial solution is obtained by setting the current independent variables equal to zero, and letting the components of X equal the constants in the first column of A^k. The matrix and the trial solution are said to be "dual feasible" if (3.8) is true. They are feasible (or primal feasible) if (3.9) is true. If a solution is both dual feasible and primal feasible, then it is an optimal solution. The standard rule of pivoting for the dual simplex method is to choose a negative constant in the first column of A^k (usually the most negative one) say a_{i0}. Then find nonzero constants in the ith row of A^k for which a_{0j}/a_{ij} is negative and select from these that column for which the ratio is least negative. Then appropriate multipliers of column j are added to other columns to make all elements in the ith row zero except the jth element which should be -1. In Table A1, the pivot element is indicated by a "$*$" and the result of pivoting in Table A2. After another pivot step

TABLE A2
Pivoting step 1

	1	$-s_{12}$	$-y_{13}$	$-y_{23}$	$-y_{24}$	$-y_{34}$
z	-24	4	1	4	6	3
y_{12}	6	-1	1	0	0	0
y_{13}	0	0	-1	0	0	0
y_{23}	0	0	0	-1	0	0
y_{24}	0	0	0	0	-1	0
y_{34}	0	0	0	0	0	-1
s_{12}	0	-1	0	0	0	0
s_{23}	1	-1	1	-1	-1	0
s_{14}	-4	0	0	0	-1	-1^*

TABLE A3
Pivoting step 2

	1	$-s_{12}$	$-y_{13}$	$-y_{23}$	$-y_{24}$	$-s_{14}$
z	-36	4	1	4	3	3
y_{12}	6	-1	1	0	0	0
y_{13}	0	0	-1	0	0	0
y_{23}	0	0	0	-1	0	0
y_{24}	0	0	0	0	-1	0
y_{34}	4	0	0	0	1	-1
s_{12}	0	-1	0	0	0	0
s_{23}	1	-1	1	-1	-1	0
s_{14}	0	0	0	0	0	-1



in Table A2, the result is shown in Table A3. Since in Table A3, the solution is both dual feasible and primal feasible, the optimum solution is obtained. The solutions are $y_{12} = 6$, $y_{34} = 4$, and $y_{13} = y_{23} = y_{24} = 0$. The negatives of the independent variables are used so that the criteria for the choice of pivot element agree with the usual notation.

For the problem of synthesis of a communication net, we consider the linear programming formulation as

$$\max z = \sum_{i \neq j} c_{ij}(-y_{ij}) \tag{3.10}$$

with

$$s_{ij} = C_{ij}^{\alpha} - r_{ij} \tag{3.11}$$

where C_{ij}^{α} represent the sum of y_{ij} which form any cut α separating nodes

i and j, r_{ij} is the flow requirement, and s_{ij} are slack variables required to be nonnegative.

Our algorithm can be described by the following steps.

0. *Setting up the Matrix.* This is the matrix which expresses the objective function z and all y_{ij} in terms of independent variables. This can be done, for example, as in Table A1. If one has m variables y_{ij}, then the matrix will be a $(m + 1) \times (m + 1)$ matrix. At the start, we have relations like (3.5) for all y_{ij}. In subsequent steps, independent variables will be changed as the result of row elimination. We shall call the values of y_{ij} obtained by putting all independent variables equal to zero, the current values of y_{ij}. If one solves this problem by using the ordinary simplex method, this would require all the $2^{n-1} - 1$ inequalities in (3.11) to be added to the matrix for row eliminations. The purpose of this algorithm is to avoid carrying all the inequalities at all times but to choose among all inequalities, those that are not satisfied by the current y_{ij}. This is done in Step 1.

1. *Selecting Inequalities.* Using the current values of y_{ij} as network capacities, we do $n - 1$ flow problems in the network to find the maximal flows corresponding to the $n - 1$ dominating requirements.

The flow problems can be done by the labeling process of [2]. Doing each of these problems automatically locates a minimal cut C, one whose capacity (measured in terms of the current values of the y_{ij}) is equal to the current maximal flow. We can then impose the linear inequality

$$\sum_{a_{ij} \in C} y_{ij} \geqq r_{ij} .$$

If all $n - 1$ of these inequalities are satisfied by the current y_{ij}, all flow requirements have been met and the optimal solution is obtained. If not, the $n - 1$ inequalities are updated and added to the bottom of the matrix. The updating, which is merely expressing the inequalities in terms of the current independent set, is explained later. Since there are $n - 1$ dominating requirements one always adds $n - 1$ inequalities.

2. *Satisfying the Inequalities.* We do a dual linear programming problem for these inequalities. This will result in another set of current y_{ij} which will be used for the flow problem in Step 1. Step 1 and Step 2 are repeated until the matrix is both primal and dual feasible. The iterations of Step 1 and Step 2 are essentially a short cut for doing the ordinary dual simplex method with $2^{n-1} - 1$ constraints. Step 1 selects certain inequalities among the $2^{n-1} - 1$ inequalities which are not satisfied by the current y_{ij}. Step 2 improves the current y_{ij} by linear programming subjected to these inequalities. In order to carry on the linear programming, it is only necessary to keep the $(m + 1) \times (m + 1)$ matrix. Old inequalities are disregarded as soon as a new set of y_{ij} has been obtained in Step 2, so there is no accumulation of inequalities.

For a network of m arcs, a reasonable estimate of the number P of pivot steps is $2m$ and the number of flow problems to be solved certainly does not exceed $P(n-1)$. Each pivot step is carried out on a $(m+n) \times (m+1)$ matrix.

We shall illustrate the procedure by an example. Consider the flow requirements shown in Fig. 1(a) (or equivalently in Fig. 1(c)) with the costs of construction shown in Fig. 2(a) (or equivalently in Fig. 2(b)). The dominating requirement tree is shown in Fig. 1(b) by unbroken lines. In the cost matrix c_{ij}, we note that c_{14} is unreasonable, therefore y_{14} will be zero and will not enter into the calculation. Applying Step 0, we set up the matrix as shown in Table A1 above the double line. Step 1, now the current values of y_{ij} are all zero and any cut will be a minimum cut. The three cuts used as inequalities are drawn beside Table A1 for illustrative purposes. These are added to the bottom of the double line in Table A1. Step 2, the row eliminations in Table A1 have already been done in explaining the particular notation of the simplex method used here. The result of Step 2 is then shown in Table A3. Now the current values of y_{ij} are $y_{12} = 6$, $y_{34} = 4$, $y_{13} = y_{23} = y_{24} = 0$, as indicated by the constants in the first column of the matrix. For Step 1, based on these current values, we shall do three flow problems to find f_{12}, f_{23} and f_{14}. It is found that $f_{12} = r_{12}$, $f_{23} = 0$, $f_{34} = r_{34}$ and we also get the minimum cuts which are drawn beside Table A3. These minimum cuts are used as inequalities which are adjoined to the matrix forming Table A4. Since all minimum cuts found from the flow problems are in terms of y_{ij}, they must be expressed in terms of current independent variables in adjoining them to the matrix. This process is called updating and can be easily done as follows. The top row of the current matrix is replaced by (1, 0, 0, 0, 0, 0) and all other elements by their

TABLE A4

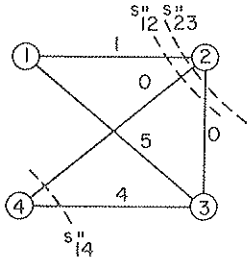
Three flow problem

	1	$-s_{12}$	$-y_{13}$	$-y_{23}$	$-y_{24}$	$-s_{14}$
z	−36	4	1	4	3	3
y_{12}	6	−1	1	0	0	0
y_{13}	0	0	−1	0	0	0
y_{23}	0	0	0	−1	0	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	1	−1
s'_{12}	0	−1	0	0	0	0
s'_{13}	−5	0	−1*	−1	−1	0
s'_{14}	0	0	0	0	0	−1

TABLE A5

Pivoting step 3

	1	$-s_{12}$	$-s'_{23}$	$-y_{23}$	$-y_{21}$	$-s_{14}$
z	-41	4	1	3	2	3
y_{12}	1	-1	1	-1	-1	0
y_{13}	5	0	-1	1	1	0
y_{23}	0	0	0	-1	0	0
y_{24}	0	0	0	0	-1	0
y_{34}	4	0	0	0	-1	-1
s'_{12}	0	-1	0	0	0	0
s'_{23}	0	0	-1	0	0	0
s'_{14}	0	0	0	0	0	-1



negatives. The inequalities generated by minimum cuts multiplied by this modified matrix then express all inequalities in terms of new independent variables. For example

$$\begin{pmatrix} -6 & -1 & -1 & 0 & 0 & 0 \\ -5 & 0 & -1 & -1 & -1 & 0 \\ -4 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -6 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -4 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ -5 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

The result of successive pivot steps is shown in Table A5. Tables A6–A9 are two more cycles of Step 1 and Step 2. The figure beside Table A9 is the optimum solution. This can be verified by doing three flow problems for f_{12}, f_{23}, and f_{14}. As none of the maximal flows are less than corresponding flow requirements, this means the solution is primal feasible. But the matrix has been dual feasible at all times, therefore, the solution is optimum. Here we have done five pivot steps and 12 flow problems. The finiteness of the iterations of Step 1 and Step 2 can be seen just as in the simplex method. Among finite numbers of variables, no set of independent variables will be chosen twice, and the pivot step makes the objective function monotonically increasing.

Since this is a dual simplex method, at each stage of the calculation, the solution is dual feasible but not primal feasible. At the end of the cal-

TABLE A6

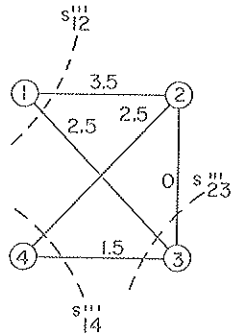
Three flow problems

	1	$-s_{12}$	$-s'_{23}$	$-y_{23}$	$-y_{24}$	$-s_{14}$
z	−41	4	1	3	2	3
y_{12}	1	−1	1	−1	−1	0
y_{13}	5	0	−1	1	1	0
y_{23}	0	0	0	−1	0	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	1	−1
s''_{12}	−5	−1	1	−2	−2*	0
s''_{23}	−4	−1	1	−2	−2	0
s''_{14}	0	0	0	0	0	−1

TABLE A7

Pivoting step 4

	1	$-s_{12}$	$-s'_{23}$	$-y_{23}$	$-s''_{12}$	$-s_{14}$
z	−46	3	2	1	1	3
y_{12}	3.5	−0.5	0.5	0	−0.5	0
y_{13}	2.5	−0.5	−0.5	0	0.5	0
y_{23}	0	0	0	−1	0	0
y_{24}	2.5	0.5	−0.5	1	−0.5	0
y_{34}	1.5	−0.5	0.5	−1	0.5	−1
s''_{12}	0	0	0	0	−1	0
s''_{23}	+1	0	0	0	−1	0
s''_{14}	0	0	0	0	0	−1



culation, an optimum solution is characterized by being both primal and dual feasible.

There may be cases where one would rather settle for primal feasible solutions which cost less than a certain amount, rather than go to the end of the computation. This is discussed in the next section where a method closer to the primal simplex method is shown.

4. Algorithm II. This algorithm is essentially a primal simplex method which avoids treating all constraints simultaneously. This method has the advantage of having primal feasible solutions during calculation. Hence calculation can be stopped if one desires and a feasible, though not optimal, solution is obtained. The method is explained in three steps below and the example used in §3 will be solved again by this method.

0. *Setting up the Matrix.* We want a primal feasible matrix which ex-

TABLE A8

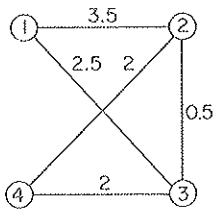
Three flow problem

	1	$-s_{12}$	$-s'_{23}$	$-s_{23}$	$-s''_{12}$	$-s_{14}$
z	−46	3	2	1	1	3
y_{12}	3.5	−0.5	0.5	0	−0.5	0
y_{13}	2.5	−0.5	−0.5	0	0.5	0
y_{23}	0	0	0	−1	0	0
y_{24}	2.5	0.5	−0.5	1	−0.5	0
y_{34}	1.5	−0.5	0.5	−1	0.5	−1
s'''_{12}	0	−1	0	0	0	0
s'''_{23}	−1	−1	0	−2*	1	−1
s'''_{14}	0	0	0	0	0	−1

TABLE A9

Pivoting step 5, and three flow problem

	1	$-s_{12}$	$-s'_{23}$	$-s'''_{23}$	$-s''_{12}$	$-s_{14}$
z	−46.5	2.5	2	0.5	1.5	2.5
y_{12}	3.5	−0.5	0.5	0	−0.5	0
y_{13}	2.5	−0.5	−0.5	0	0.5	0
y_{23}	0.5	0.5	0	−0.5	−0.5	0.5
y_{24}	2	0	−0.5	0.5	0	−0.5
y_{34}	2	0	0.5	−0.5	0	−0.5
s'''_{12}	0	−1	0	0	0	0
s'''_{23}	0	0	0	−1	0	0
s'''_{14}	0	0	0	0	0	−1



presses all y_{ij} in terms of slack variables and the y_{ij} themselves. If one sets branch capacities y_{ij} equal to the flow requirements in the dominating requirement tree, and $y_{ij} = 0$ when its corresponding flow requirement is not in the tree, then one has a primal feasible solution. These values of y_{ij} are used as current values at the beginning. Now the initial independent variables are introduced by the following rule. Let $y_{ij} = r_{ij} - u_{ij}$ if f_{ij} is in the dominating requirement tree, and $y_{ij} = -(-y_{ij})$ if otherwise. Then the u_{ij} and y_{ij} on the right-hand sides are used as independent variables.

To solve the example in §3, we would let $y_{12} = 6 - u_{12}$, $y_{23} = 5 - u_{23}$, $y_{14} = 4 - u_{14}$, and use u_{12}, u_{23}, u_{14}, y_{13}, y_{24}, and y_{34} as independent variables. Since we know y_{14} can be eliminated from the start, we shall change the dominating requirement tree. From the discussion in §2, any dominating requirement tree based on the flow requirements in Fig. 1(b) can be

used. We shall let $r_{34} = 4$ replace $r_{14} = 4$, i.e., let $y_{34} = 4 - u_{34}$. Now u_{12}, u_{23}, u_{34}, y_{13}, y_{24} are independent variables. The objective function then becomes $z = -4(6 - u_{12}) - 5y_{13} - 4(5 - u_{23}) - 6y_{24} - 3(4 - u_{34}) = -56 + 4u_{12} - 5y_{13} + 4u_{23} - 6y_{24} + 3u_{34}$. This matrix is listed as in Table P1 above the double line. Now, we want to replace u_{ij} by slack variables. This can be done by introducing slack variables which represent cuts containing y_{12}, y_{23}, and y_{34}. These slack variables are listed in Table P1 under the double line. Then the usual pivot steps are applied. The pivot element is indicated by a "*". Successive steps are shown in Tables P2, P3, and B1. In Table B1, all independent variables are y_{ij} or s_{ij}. We are ready to apply Step 1.

1. *Selecting the Inequalities.* Since we start with a primal feasible solution, we would like to change this solution into another primal feasible solution

TABLE P1

Preliminary

	1	$-u_{12}$	$-y_{13}$	$-u_{23}$	$-y_{24}$	$-u_{34}$
z	−56	−4	5	−4	6	−3
y_{12}	6	1	0	0	0	0
y_{13}	0	0	−1	0	0	0
y_{23}	5	0	0	1	0	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	0	1
s_{12}	0	1*	−1	0	0	0
s_{23}	0	0	−1	1	−1	0
s_{34}	0	0	0	0	−1	1

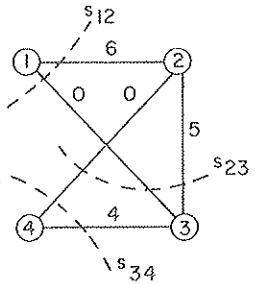


TABLE P2

Preliminary

	1	$-s_{12}$	$-y_{13}$	$-u_{23}$	$-y_{24}$	$-u_{34}$
z	−56	4	1	−4	6	−3
y_{12}	6	−1	1	0	0	0
y_{13}	0	0	−1	0	0	0
y_{23}	5	0	0	−1	0	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	0	1
s_{12}	0	−1	0	0	0	0
s_{23}	0	0	−1	1*	−1	0
s_{34}	0	0	0	0	−1	1

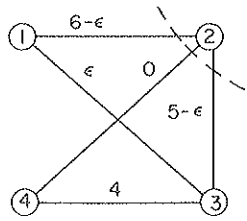
TABLE P3

Preliminary

	1	$-s_{12}$	$-y_{13}$	$-s_{23}$	$-y_{24}$	$-y_{34}$
z	−56	4	−3	4	2	−3
y_{12}	6	−1	1	0	0	0
y_{13}	0	0	−1	0	0	0
y_{23}	5	0	1	−1	1	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	0	1
s_{12}	0	−1	0	0	0	0
s_{23}	0	0	0	−1	0	0
s_{34}	0	0	0	0	−1	1*

TABLE B1

	1	$-s_{12}$	$\downarrow$ $-y_{13}$	$-s_{23}$	$-y_{24}$	$-s_{34}$
z	−56	4	−3	4	−1	3
y_{12}	6	−1	1	0	0	0
y_{13}	0	0	−1	0	0	0
y_{23}	5	0	1	−1	1	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	1	−1
s_{12}	0	−1	0	0	0	0
s_{23}	0	0	0	−1	0	0
s_{34}	0	0	0	0	0	−1



and reduce the total cost. If a reduction in cost is possible, then the matrix will have some negative constants in the top row.

Raising the value of the corresponding variable from its current value of zero will give a decrease in cost. If we give the variable the value ϵ we obtain new y_{ij} values $y_{ij}(\epsilon)$. We want to find the largest value of ϵ, call it $\epsilon_{\max}$, for which the following are true

$$y_{ij}(\epsilon) \geqq 0 \qquad \text{(all } i, j\text{)}$$
$$f_{ij}(\epsilon) = f_{ij}(y_{ij}(\epsilon)) \geqq r_{ij}. \qquad \text{(for the } n-1 \text{ dominant } r_{ij}\text{)}. \tag{4.1}$$

(In Table B1, both columns $(-y_{13})$ and $(-y_{24})$ have negative constants in the top row of the matrix. The "↓" above $(-y_{13})$ indicates that we have chosen y_{13}. Let column $(-y_{13})$ be multiplied by $-\epsilon$ and added to the first column. This gives the $y_{ij}(\epsilon)$. For example, in Table B1, if we multiply

the column under "↓" by $-\epsilon$ and add it to the first column, we have $z = -56 + 3\epsilon$, $y_{12} = 6 - \epsilon$, $y_{13} = \epsilon$, $y_{23} = 5 - \epsilon$, $y_{24} = 0$, $y_{34} = 4$, as shown in the figure beside Table B1.

As $y_{ij}(\epsilon)$ are all linear functions of ϵ, the capacity of a cut, which is a sum of $y_{ij}(\epsilon)$, is also a linear function of ϵ. Since $f_{ij}(\epsilon)$ between any pair of nodes is equal to the capacity of a minimal cut separating nodes i and j, it follows easily that $f_{ij}(\epsilon)$ is a piecewise linear and convex function of ϵ.

The procedure for finding $\epsilon_{\max}$ is as follows.

(a). To satisfy the condition $y_{ij}(\epsilon) \geqq 0$, we find, by the usual simple ratio test, the first value of ϵ, call it ϵ_0 for which some $y_{ij}(\epsilon) = 0$ and $y_{ij}(\epsilon) < 0$ for $\epsilon > \epsilon_0$. This value is then an upper bound for $\epsilon_{\max}$.

(b). Take one of the $n - 1$ dominant flow requirements and compute the corresponding flow $f_{ij}(\epsilon)$ as a function of ϵ by the process described below until:

(b_1). A value of ϵ' is reached such that $\epsilon' < \epsilon_{\max}$ and

$$f_{ij}(\epsilon') = r_{ij}$$

$$f_{ij}(\epsilon) < r_{ij} \qquad (\epsilon > \epsilon').$$

In this case take a new $\epsilon_{\max} = \epsilon'$, and repeat (b) with the next flow requirement. Or,

(b_2). No such value is reached for $\epsilon < \epsilon_{\max}$. Then keep the current $\epsilon_{\max}$ and repeat (b) with the next flow requirement.

After repeating (b) $n - 1$ times we emerge with the final value $\epsilon_{\max}$ satisfying (4.1).

Having explained the computation of $\epsilon_{\max}$ we turn to its use.

If $\epsilon_{\max}$ is obtained from a condition $y_{ij} \geqq 0$, then the row of y_{ij} represents that restriction among the equations (4.1) which first prevents increase in ϵ, so an ordinary simplex step can be made using this row (i.e., the intersection of this row and the column under the arrow is the pivot element).

If $\epsilon_{\max}$ is obtained from a flow requirement $f_{ij}(\epsilon) = r_{ij}$ and $f_{ij}(\epsilon) < r_{ij}$ for $\epsilon > \epsilon_{\max}$, then, in the course of the $f_{ij}(\epsilon)$ computation described below, we will have obtained a cut whose capacity equals the maximal flow for some range of ϵ including $\epsilon_{\max}$. Using this cut, updating it as in the first algorithm we have the restriction which first prevents the increase in ϵ, and we pivot on this row.

These pivot steps are iterated just as in the first method until the optimum solution is reached.

We will now describe and illustrate the computation of $f_{st}(\epsilon)$. This could be done by a repetition of ordinary network flow computations, but there is a considerable economy to be obtained by using a variant of the "out of kilter" method of D. R. Fulkerson [11] as we do here.

We start with a network which consists of links whose capacities $a_{ij} + b_{ij}\epsilon$ are linear functions of the parameter ϵ (Fig. 3a).

Here and in what follows, we will often have expressions of the form $a + b\bar{\epsilon}$ with $\bar{\epsilon}$ a certain parameter. In comparing two such expressions, we will say $a + b\bar{\epsilon} = a' + b'\bar{\epsilon}$ if and only if $a = a'$ and $b = b'$, we will say $a + b\bar{\epsilon}$ is greater than $a' + b'\bar{\epsilon}$ if and only if either (1) $a > a'$ or (2) $a = a'$ and $b > b'$.

The "out of kilter" algorithm actually obtains a minimal cost feasible circulation in a directed network having upper and lower bounds to the flow through the ares. These words have the following meanings. A circulation is simply a set of arc flows x_{ij} (flow from node i to node j in the arc ij) such that $\sum_{i,k} (x_{ij} - x_{jk}) = 0$ for all j, i.e., fluid is conserved at the nodes. A feasible circulation is a circulation when the x_{ij} satisfy $l_{ij} \leqq x_{ij} \leqq y_{ij}$ with prescribed upper and lower bounds y_{ij} and l_{ij}. The cost of a circulation is, for given $\bar{c}_{ij}$, $\sum_{i,j} \bar{c}_{ij}x_{ij}$.

To recast our problem as a minimal cost circulation problem, we assign arbitrary directions to each arc and use the capacity and its negative for upper and lower bounds, i.e. $-(a_{ij} + b_{ij}\epsilon) \leqq x_{ij} \leqq (a_{ij} + b_{ij}\epsilon)$. We adjoin an arc from t to s and give it infinite capacity and set all cost $\bar{c}_{ij} = 0$ except $\bar{c}_{ts}$ which is taken as -1.

The minimum cost circulation in this network (Fig. 3b) is the one that puts the most flow through the arc ts from t to s. This amount is clearly also the maximum flow from s to t.

To start the "out of kilter" algorithm, we need only some (not necessarily feasible) circulation ($x_{ij} = 0$ for all i, j will do), and a starting set of "node prices" π_i, $i = p, \cdots, n$. These π_i are arbitrary, so $\pi_i = 0$ for all i, is satisfactory.

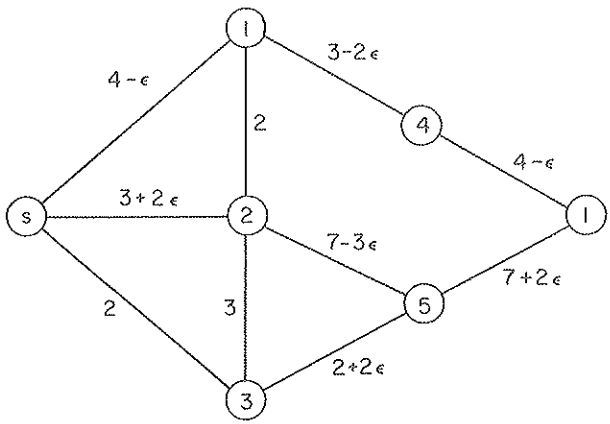


FIG. 3a

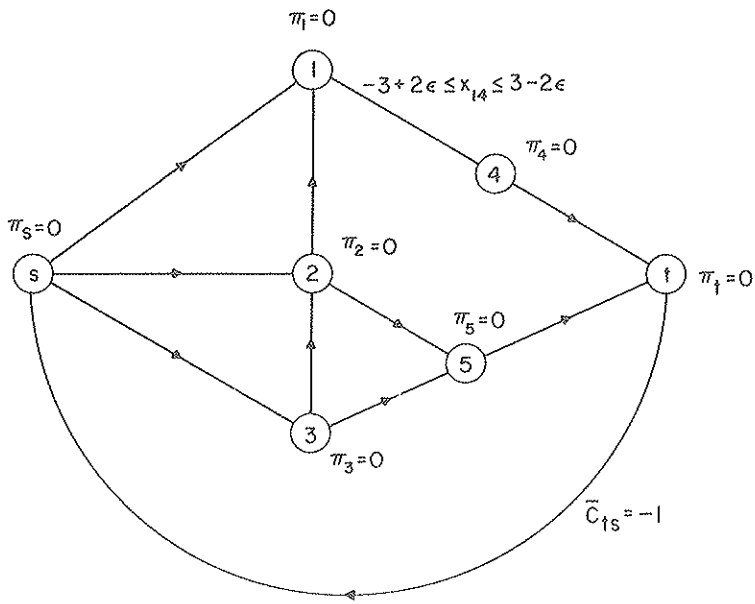


FIG. 3b

We apply the "out of kilter" method as described in [11]. Since we are dealing with capacities and later flows that are of the form $a + b\epsilon$, in following the instructions of [11], we must apply the definitions of $>$ and $=$ as given earlier. The finiteness proof of [11] can be modified to cover this case.

The modified out-of-kilter method then gives us a maximal flow with flow value $f_{st}(\epsilon) = x_{ts}(\epsilon) = a_{ts} + b_{ts}\epsilon$. There is also a corresponding cut, discovered during the course of the computation, whose capacity is $a_{ts} + b_{ts}\epsilon$. The existence of this cut assures us that $f_{st}(\epsilon) \leqq a_{ts} + b_{ts}$ for all ϵ, and hence our current flow is maximal for any range of ϵ for which no restrictions are violated. Fig. 4a shows the maximal flow in our example. Each arc is accompanied by three expressions; the first is the capacity, the second the flow, the third the first ϵ value, call it ϵ_1, after which capacity becomes negative or is exceeded by the arc flow. Then our flow is valid and maximal for $-0 \leqq \epsilon \leqq \epsilon_1$. In our example $\epsilon = \frac{1}{2}$ as the flow in the arc 2-5 exceeds the capacity for $\epsilon > \frac{1}{2}$. For $\epsilon \leqq \frac{1}{2}$, $f_{st}(\epsilon) = 9 + \epsilon$.

We have now obtained $f_{st}(\epsilon)$ for a range $0 \leqq \epsilon \leqq \epsilon_1$. To get f_{st} for the next interval, we simply rewrite all expressions $a + b\epsilon$ in the form $\bar{a} + \bar{b}\bar{\epsilon}$ with $\bar{\epsilon} = \epsilon - \epsilon_1$ (see Fig. 4b).

We now apply the out-of-kilter method of this new network to obtain the minimal cost circulation. The difference is that our comparisons are now based on the coefficients $\bar{a}$, $\bar{b}$ of the expressions $\bar{a} + \bar{b}\bar{\epsilon}$. We can use the

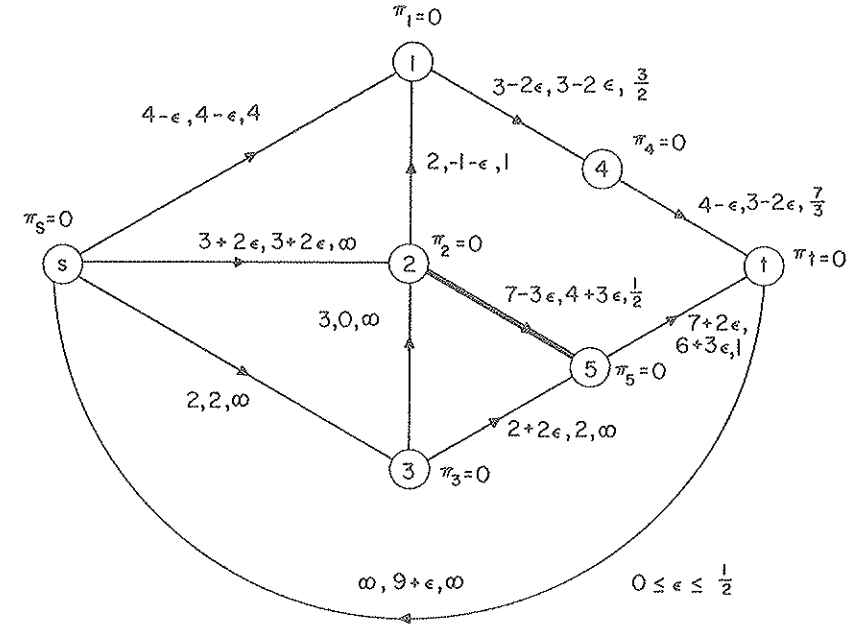

$\pi_1=0$
1
$3-2\epsilon, 3-2\epsilon, \frac{3}{2}$
$4-\epsilon, 4-\epsilon, 4$
$2, -1-\epsilon, 1$
$\pi_4=0$
4
$4-\epsilon, 3-2\epsilon, \frac{7}{3}$
$\pi_s=0$
s
$3+2\epsilon, 3+2\epsilon, \infty$
$\pi_2=0$
2
t
$\pi_t=0$
$7-3\epsilon, 4+3\epsilon, \frac{1}{2}$
$3, 0, \infty$
$7+2\epsilon,$
$6+3\epsilon, 1$
5
$\pi_5=0$
$2, 2, \infty$
$2+2\epsilon, 2, \infty$
3
$\pi_3=0$
$\infty, 9+\epsilon, \infty$
$0 \le \epsilon \le \frac{1}{2}$

FIG. 4a

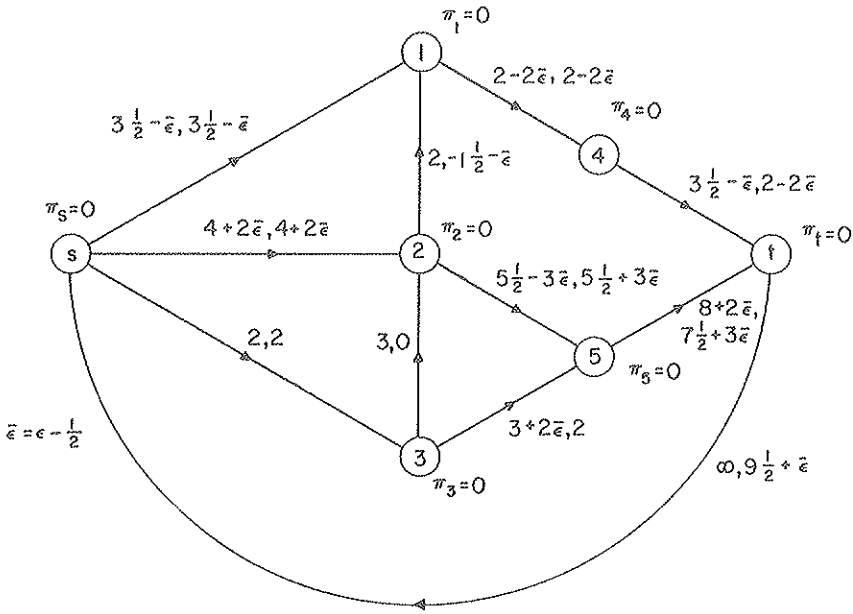

$\pi_1=0$
1
$2-2\bar{\epsilon}, 2-2\bar{\epsilon}$
$3\frac{1}{2}-\bar{\epsilon}, 3\frac{1}{2}-\bar{\epsilon}$
$\pi_4=0$
$2, -1\frac{1}{2}-\bar{\epsilon}$
4
$3\frac{1}{2}-\bar{\epsilon}, 2-2\bar{\epsilon}$
$\pi_s=0$
s
$4+2\bar{\epsilon}, 4+2\bar{\epsilon}$
$\pi_2=0$
2
t
$\pi_t=0$
$5\frac{1}{2}-3\bar{\epsilon}, 5\frac{1}{2}+3\bar{\epsilon}$
$8+2\bar{\epsilon},$
$7\frac{1}{2}+3\bar{\epsilon}$
$2, 2$
$3, 0$
5
$\pi_5=0$
$\bar{\epsilon}=\epsilon-\frac{1}{2}$
$3+2\bar{\epsilon}, 2$
3
$\pi_3=0$
$\infty, 9\frac{1}{2}+\bar{\epsilon}$

FIG. 4b

present flow as a starting circulation even though it is not (in terms of $\bar{a}$, $\bar{b}$) a feasible circulation. (In our example in Fig. 4b, the arc flow $5\frac{1}{2} + 3\bar{\epsilon}$ is greater than the upper bound $5\frac{1}{2} - 3\bar{\epsilon}$.) The node prices from the previous calculation can also be used. Thus repeating the out-of-kilter algorithm on this network, we obtain a new maximal flow f_{st} valid for a new range of $\epsilon > \epsilon_1$. This procedure is iterated until the graph of $f_{st}(\epsilon)$ dips below the required amount r_{st} or until ϵ exceeds the current ϵ_{max} as defined in (a), (b_1) and (b_2) above. That this will occur after a finite number of repetitions is shown in the Appendix.

Various stages of the calculation for our example are shown in Fig. 5a and 5b. Fig. 5a shows the result of applying the out-of-kilter algorithm to the network of Fig. 4b. A new maximal flow is obtained by a slight adjustment of the previous one. (The advantage of the out-of-kilter method is that it does allow us to start with the previous flow.) Now, $f_{st}(\epsilon) = 9\frac{1}{2} + \bar{\epsilon} = 9 + \epsilon$, for a new range $\frac{1}{2} \leqq \epsilon \leqq \frac{3}{4}$. At $\epsilon = \frac{3}{4}$ the arc 3-5 is becoming oversaturated so $\bar{\bar{\epsilon}} = \bar{\epsilon} - \frac{1}{4}$ is introduced and all quantities put into the form $\bar{\bar{a}} + \bar{\bar{a}} \cdot \bar{\bar{\epsilon}}$. The result of calculating on that network is shown in Fig. 5b. Where a maximal flow is shown valid for $0 \leqq \bar{\bar{\epsilon}} \leqq \frac{1}{4}$, i. e., for $\frac{3}{4} \leqq \epsilon \leqq 1$, and giving $f_{st}(\epsilon) = 9\frac{3}{4} - 3\bar{\epsilon} = 12 - 3\epsilon$ for that range. For $\epsilon > 1$, $f_{st} < r_{st}$ so it is not necessary to pursue the example further. The function $f_{st}(\epsilon)$, $0 \leqq \epsilon \leqq 1$, is given in Fig. 6.

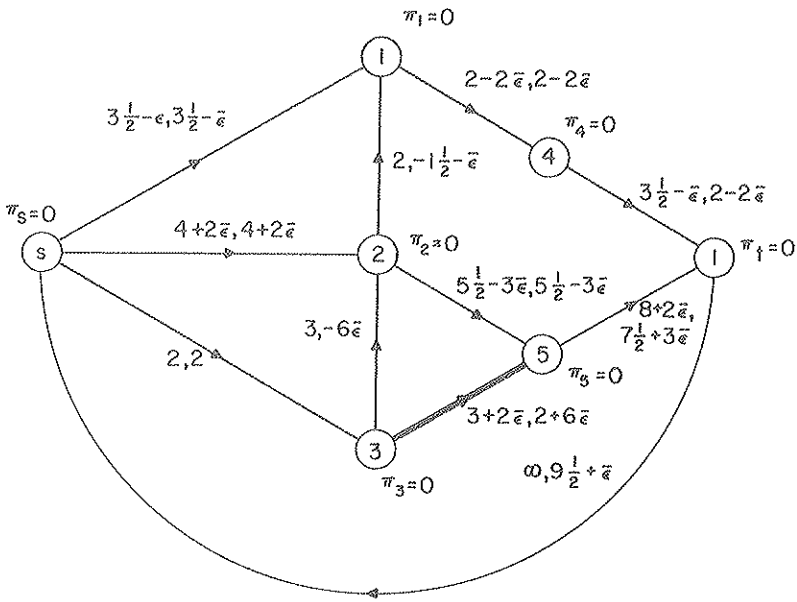


FIG. 5a

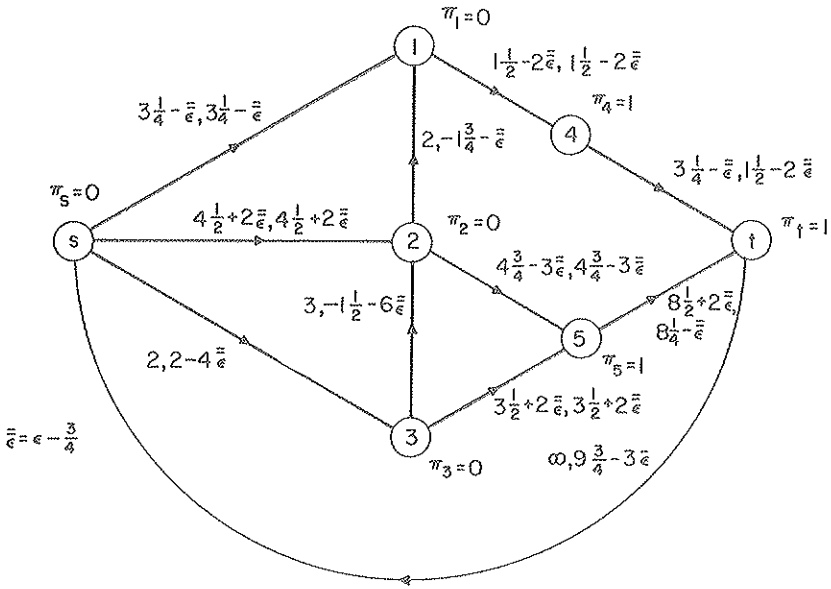


FIG. 5b

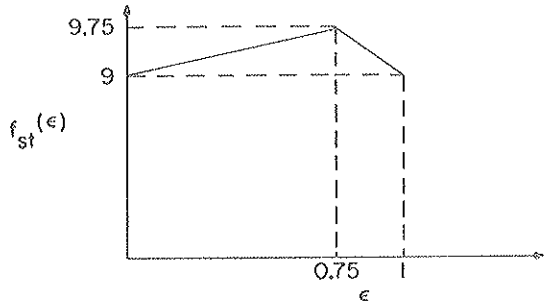


FIG. 6

Turning now to the original network example already done by Algorithm I, we have (from Table B1),

$$y_{12} = 6 - \epsilon$$

$$y_{13} = \epsilon$$

$$y_{23} = 5 - \epsilon;$$

hence the first value for $\epsilon_{\max} = 5$. Turning to the flow requirement $r_{12} = 6$,

we have

$$f_{12} = 6 \qquad (\epsilon \leqq 2.5)$$

$$f_{12} = 11 - 2\epsilon \quad \text{for a range of } \epsilon > \epsilon_0 = 2.5;$$

hence $\epsilon_{\max} \leqq 2.5$. Consider now $r_{23} = 5; f_{23} = 5$ for $\epsilon \leqq 3$. So $\epsilon_{\max} = 2.5$. As $f_{34} \geqq 4$, we have $\epsilon_{\max} = 2.5$. The cut $(2|\ 1, 3, 4)$ obtained from computing f_{12} is used. Here the minimum cut consists of y_{12}, y_{23}, and y_{24}, and is updated and added to the bottom of the matrix in table B2. Now we can apply Step 2.

2. *Primal Simplex Method.* The usual pivot step can be done for Table B2. Since this is a primal simplex method, we want a positive constant in the first row below the double line, and choose a positive element in its column. The pivot element in Table B2 is indicated by a "*" and the result of the pivot step, is shown in Table B3. Now the inequality below the double line in Table B3 can be neglected and Step 1 can again be applied.

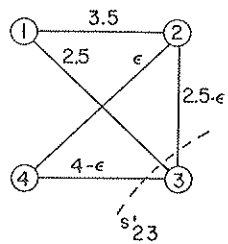
1. Now we want to replace $-y_{24}$ as it is the only column with negative constant in the top row. The $y_{ij}(\epsilon)$ can be seen from the first column and the column in question, i.e., $y_{12} = 3.5$, $y_{13} = 2.5$, $y_{23} = 2.5 - \epsilon$, $y_{24} = \epsilon$,

TABLE B2

	1	$-s_{12}$	$-y_{13}$	$-s_{23}$	$-y_{24}$	$-s_{34}$
z	-56	4	-3	4	-1	3
y_{12}	6	-1	1	0	0	0
y_{13}	0	0	-1	0	0	0
y_{23}	5	0	1	-1	1	0
y_{24}	0	0	0	0	-1	0
y_{34}	4	0	0	0	1	-1
s'_{12}	5	-1	2*	-1	0	0

TABLE B3

	1	$-s_{12}$	$-s'_{12}$	$-s_{23}$	$\downarrow$ $-y_{24}$	$-s_{34}$
z	-48.5	2.5	1.5	2.5	-1	3
y_{12}	3.5	-0.5	-0.5	0.5	0	0
y_{13}	2.5	-0.5	0.5	-0.5	0	0
y_{23}	2.5	0.5	-0.5	-0.5	1	0
y_{24}	0	0	0	0	-1	0
y_{34}	4	0	0	0	1	-1
s'_{12}	0	0	-1	0	0	0



$y_{34} = 4 - \epsilon$. Based on these values of $y_{ij}(\epsilon)$ we have

$$y_{23} = 2.5 - \epsilon$$
$$y_{24} = \epsilon \qquad (\epsilon \leqq 2.5)$$
$$y_{34} = 4 - \epsilon;$$

hence $\epsilon_{\max} \leqq 2.5$. For the flow requirements, we have

$$r_{12} = 6 \qquad f_{12} = 6$$
$$r_{23} = 5 \qquad f_{23} = 5 \qquad (\text{for} \quad \epsilon \leqq 2)$$
$$f_{23} = 9 - 2\epsilon \quad (\text{for a range} \quad \epsilon > 2)$$
$$\epsilon_{\max} = 2.$$

But

$$r_{34} = 4 \qquad f_{34} = 4;$$

$\epsilon_{\max}$ still equals 2.

Hence $\epsilon = 2$, and a new inequality is added below the double line in Table B4.

TABLE B4

	1	$-s_{12}$	$-s'_{12}$	$-s_{23}$	$-y_{23}$	$-s_{34}$
z	−48.5	2.5	1.5	2.5	−1	3
y_{12}	3.5	−0.5	−0.5	0.5	0	0
y_{13}	2.5	−0.5	0.5	−0.5	0	0
y_{23}	2.5	0.5	−0.5	−0.5	1	0
y_{24}	0	0	0	0	−1	0
y_{34}	4	0	0	0	1	−1
s'_{23}	4	0	0	−1	2*	−1

TABLE B5

	1	$-s_{12}$	$-s'_{12}$	$-s_{23}$	$-s'_{23}$	$-s_{34}$
z	−46.5	2.5	1.5	2	0	0
y_{12}	3.5	−0.5	−0.5	0.5	0	0
y_{13}	2.5	−0.5	0.5	−0.5	0	0
y_{23}	0.5	0.5	−0.5	0	−0.5	0.5
y_{24}	2	0	0	−0.5	0.5	−0.5
y_{34}	2	0	0	0.5	−0.5	−0.5
s'_{23}	0	0	0	0	−1	0

2. Another pivoting step is shown in Table B5. In that table all constants in the top row of the matrix are positive. This indicates that no reduction in cost is possible, i.e., the solution is also dual feasible. Therefore it is the optimum solution.

The finiteness of the procedure can be obtained along the same lines as in §3.

APPENDIX

In the process of the ϵ-computation all arcs are of the form $a_{ij} + b_{ij}\epsilon$ and all arc flows are of the form $a'_{ij} + b'_{ij}\epsilon$. An arc is called identically saturated in an interval if $a_{ij} = a'_{ij}$ and $b_{ij} = b'_{ij}$ in that interval. If a set of arcs is identically saturated for $\epsilon_0 < \epsilon \leqq \epsilon_1$, not identically saturated for $\epsilon_1 < \epsilon \leqq \epsilon_2$, and identically saturated again for $\epsilon_2 < \epsilon \leqq \epsilon_3$, then from the convexity of $f_{st}(\epsilon)$, we can simply by combining flows have a maximal flow $f_{st}(\epsilon)$ for which the set of arcs is identically saturated for $\epsilon_0 < \epsilon \leqq \epsilon_3$.

To make sure that sets of saturated arcs cannot recur, we modify the algorithm as follows. When using the labeling process in the out-of-kilter algorithm, we first label using only not identically saturated arcs. Only if non-breakthrough then results do we label using the identically saturated arcs also. If this procedure is always used to modify the labeling part of the process, the new maximal flow that results for the new interval will unsaturate an identically saturated arc only if there is *no* maximal flow that does leave all the identically saturated arcs saturated. This implies that identically saturated arcs can not all be identically saturated again for some interval.

Let S_i be the set of identically saturated arcs in the ith interval. Since at least one unsaturated arc becomes identically saturated in going from the ith to the $(i + 1)$th interval, it follows that $S_i \neq S_{i+1}$. Assume that the ϵ computation is an infinite process. This means that there is an infinity of intervals and hence, an infinity of sets S_i. Then, some of the sets in disjoint intervals must be the same. However, this is impossible if the modified out-of-kilter method is used. This proves the finiteness of the procedure.

REFERENCES

1. R. E. Gomory and T. C. Hu, *Multi-terminal network flows*, this Journal, 9 (1961), pp. 551–570.
2. L. R. Ford, Jr. and D. R. Fulkerson, *Maximal flow through a network*, Canad. J. Math., 8 (1956), pp. 399–404.
3. ———, *A simple algorithm for finding maximum network flow and an application to the Hitchcock problem*, Ibid., 9 (1957), pp. 210–218.
4. ———, *A suggested computation for maximal multi-commodity network flows*, Management Sci., 5 (1958), pp. 97–101.
5. G. B. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Operations Res., 8 (1960), pp. 100–110.

6. G. B. DANTZIG, *Linear Programming and Extensions*, Chapter 22 (to be published by Princeton University Press).
7. O. WING, *Synthesis of optimal communication nets—a linear programming approach*, IBM Research Report RC-293, July 1960.
8. J. B. KRUSKAL, JR., *On the shortest subtree of a graph and the traveling salesman problem*, Proc. Amer. Math. Soc., 7 (1956), pp. 48–50.
9. R. C. PRIM, *Shortest connection networks and some generalizations*, Bell System Tech. J., 36 (1957), pp. 1389–1401.
10. R. E. GOMORY, *An algorithm for integer solutions to linear programs*, Princeton-IBM Mathematics Research Project, Technical Report No. 1, November 17, 1958.
11. D. R. FULKERSON, *An out-of-kilter method for minimal cost flow problems*, this Journal, 9 (1961), pp. 18–27.