J. Soc. Indust. Appl. Math.
Vol. 12, No. 2, June, 1964
Printed in U.S.A.

SYNTHESIS OF A COMMUNICATION NETWORK*

R. E. GOMORY AND T. C. HU†

Abstract. A communication network is a set of nodes connected by arcs. Every arc has associated with it a nonnegative number called the branch capacity which indicates the maximum amount of flow that can pass through the arc. A communication network must have large enough branch capacities such that all message requirements (which can be regarded as flows of different commodities) can reach their destinations simultaneously. In general, these requirements vary with time. The present paper gives algorithms for min-cost synthesis of a communication network which is able to handle simultaneous flows of all time periods.

1. Introduction—Problems of analysis and synthesis in the design of communication networks. By a communication network we will mean a set of n nodes N_i and directed arcs linking nodes N_i and N_j. Associated with the arc going from N_i to N_j is a branch capacity $y_{i,j}$ and a cost coefficient $c_{i,j}$. By a p-q flow, $F_{p,q}$, we will mean the usual flow (see, for example, Ford and Fulkerson [7]) with source p and sink q; that is, a set of $n(n-1)$ nonnegative numbers $x_{i,j}^{p,q}$ such that

$$\sum_s x_{i,s}^{p,q} - \sum_r x_{r,i}^{p,q} = 0, \qquad i \neq p, q. \tag{1}$$

(1) of course has the intuitive meaning of requiring that fluid is conserved at all intermediate nodes. The flow value $f_{p,q}$ of such a flow is taken to be the amount of fluid issuing from the source p, so

$$f_{p,q} = \sum_s x_{p,s}^{p,q} = \sum_r x_{r,q}^{p,q}.$$

A set of p-q flows will be said to be feasible in a given communication net with capacity numbers $y_{i,j}$ if and only if

$$\sum_{p,q} x_{i,j}^{p,q} \leqq y_{i,j}, \qquad \text{all } i, j.$$

The two central problems associated with these networks are the problems of analysis and synthesis, both of which involve the notion of requirement, and which we state first for the case of fixed requirements $R_{p,q}$.

The problem of analysis: Given a set of $n(n-1)$ nonnegative numbers $R_{p,q}$ and a network with capacity numbers $y_{i,j}$, do there exist feasible

* Received by the editors September 19, 1963.

† Thomas J. Watson Research Center, International Business Machines Corporation, P.O. Box 218, Yorktown Heights, New York. This research was supported in part by the Office of Naval Research under Contract No. Nonr 3775(00), NR 047040.

flows such that

$$f_{p,q} \geqq R_{p,q} \,? \tag{2}$$

The problem of synthesis. Given a set of requirements $R_{p,q}$ find a network having feasible flows satisfying (2) and such that the linear cost function $\sum_{i,j} c_{i,j}\, y_{i,j}$ is minimal.

Actually, in a communication net problem there is no one set of requirements $R_{p,q}$ but rather a set $R_{p,q}(t)$ varying with a third index, time, that allows for a changing load on the network. In this article, we will assume that t takes only a finite set of values $t_1, t_2, \cdots, t_s$. The degree of difficulty of the problems of analysis and synthesis varies enormously with the assumptions that are made about the $R_{p,q}(t)$. The present status of the various problems is as follows:

Case 1. $R_{p,q}$ independent of t, or all requirements to be met simultaneously.

Analysis. The basic paper here is Ford and Fulkerson [6]. In this paper a linear programming formulation of the problem which resulted in a linear programming problem having an enormous number of columns was reduced to a reasonable problem by means of a column generating technique which is of the shortest path type. The final problem has m equations if there are m arcs in the network, and the linear programming is done with a square $m \times m$ matrix. Although the problem treated in [6] is one of maximizing total flow rather than meeting a set of requirements, the method of Ford and Fulkerson requires only minor changes to solve the problem of analysis. A special case where there are only two kinds of flows permits easy treatment instead of linear programming. See Hu [12].

Synthesis. This problem has an easy solution. Starting with a zero capacity network, it is only necessary to find the shortest (cheapest) path between the nodes p and q and then give each arc on this path an additional capacity $R_{p,q}$. This is repeated for each pair of nodes, the capacities being added, to give the minimum cost network. A more economical way of carrying out this calculation will be given later in this paper as part of another synthesis calculation.

Case 2. Completely time-shared requirements.

Here, time is broken up into distinct periods; and during any one period there is flow between one pair of nodes only. More precisely, there are times $t_{p,q}$ and $R_{p,q}(t_{p,q}) = \bar{R}_{p,q}$, and $R_{p,q}(t_{i,j}) = 0$ for $(i, j) \neq (p, q)$.

Analysis. This can always be carried out by doing $n(n-1)$ maximum flow calculations of the type of Ford and Fulkerson [5]. However, if the given network is symmetric, i.e., $y_{i,j} = y_{j,i}$, Gomory and Hu [10] showed that the analysis calculation required only $n - 1$ maximum flow calculations.

Synthesis. A subcase here is especially tractable. If we consider only symmetric networks and, in addition, impose a cost function in which all arcs are of equal cost ($c_{i,j} = 1$, all i, j), then special rapid methods of synthesis become possible; see, for example, Chien [3] and Gomory and Hu [10]. However, if the condition $c_{i,j} = 1$ is removed, we are again forced back upon linear programming. The synthesis problem, which can easily be posed as a giant linear programming problem involving an enormous number of inequalities (rows), was reduced in Gomory and Hu [11] to a more tractable size by means of a row generating technique. Again, it became possible to carry out the calculation for an m arc problem using an $m \times m$ square matrix and auxiliary calculations, that time of the maximal flow type, to produce additional rows when needed.

Case 3. Time varying requirements.

Cases 1 and 2 are extreme subcases of this more general problem.

Analysis. If we have requirements $R_{p,q}(t)$ where t is allowed s distinct values, the problem is merely s distinct repetitions of the analysis problem of Case 1. If the given network can meet the demands $R_{p,q}(t_k)$ for each time period, then it satisfies the requirements; if it fails at one or more periods, it does not meet the requirements.

Synthesis. A special case where the synthesizing network is assumed to be a tree is discussed by Tang [13]. The general case has never been treated although it is, among the problems being reviewed here, the problem of greatest practical importance. In this paper we will show that the general time varying synthesis problem for an m-arc network can also be reduced to a linear programming calculation involving only one $m \times m$ square matrix, plus auxiliary $m \times m$ linear programming calculations of the Ford and Fulkerson [6] type.

2. Methods of calculation. We will first pose the m-arc synthesis problem with time varying demands as a completely unwieldy linear programming problem. Then we will show how it can be transformed into a problem having only m columns, but an enormous number of rows. Finally, we will show how this enormous number of rows can be dealt with.

A) *Formulation as a large linear program.* Let Y be an m-vector with each component representing the capacity of an arc in an m-arc network so that Y represents an entire m-arc network. Let $\mathfrak{N}^t$ be an m-vector representing a network capable of carrying simultaneous flows with flow values $f_{p,q} \geqq R_{p,q}(t)$. It is easy to show that the networks $\mathfrak{N}^t$ form a convex (unbounded) polyhedron in m-space. Consequently, there is a finite list of networks $\mathfrak{N}_i{}^t$ such that if $\mathfrak{N}^t$ can carry the required flows, then

$$\mathfrak{N}^t = \sum_i \lambda_i{}^t \mathfrak{N}_i{}^t, \qquad 1 \leqq \sum_i \lambda_i{}^t. \tag{3}$$

If Y is to meet the requirements for each period, it is necessary and sufficient for it to contain a network of the form (3) for each t. So one way of posing the general synthesis problem is to ask for Y and λ_i^t that minimize $C \cdot Y$, subject to

$$(4) \qquad \begin{aligned} Y &\geqq \sum_i \lambda_i^t \mathfrak{N}_i^t, && t = t_1, \cdots, t_s, \\ 1 &\leqq \sum_i \lambda_i^t, && t = t_1, \cdots, t_s. \end{aligned}$$

This formulation involves $m(k+1)$ rows and an enormous number of columns including one for each $\mathfrak{N}_i^t$. To reduce the problem we first use an idea due to Benders [1].

B) *First reduction.* In (4), consider the set of $m+1$ inequalities corresponding to a particular t value t_k. Inserting the slack variables to obtain equations, we see by Farkas' theorem that there will exist λ_i^k satisfying the inequalities of (4) for a given Y if and only if

$$(5) \qquad \Pi \cdot (Y, 1) \geqq 0$$

for all the $m+1$ vectors $\Pi = (\Pi_1, \Pi_0)$ satisfying

$$(6) \qquad \Pi(\mathfrak{N}_i^k, 1) \geqq 0, \qquad \text{all } i.$$

Here Π_1 is a nonnegative m-vector and Π_0 a nonpositive scalar.

The vectors Π satisfying (6) also form a convex (unbounded) polyhedron so that there exists a finite subset $\Pi_1^k, \cdots, \Pi_q^k$ of the Π such that all Π satisfying (5) are positive combinations of these. Consequently, the solvability condition (5) which involved all Π satisfying (6) can be replaced by

$$(7) \qquad \Pi_i^k \cdot (Y, 1) \geqq 0, \qquad i = 1, \cdots, q(k).$$

Repeating this for all time periods, we find that a problem formulation equivalent to (4) is

$$(8) \qquad \text{minimize } C \cdot Y,$$

$$\text{subject to } \Pi_i^t \cdot (Y, 1) \geqq 0, \qquad i = 1, \cdots, q(t); t = t_1, \cdots, t_s.$$

This formulation involves only m variables, but generally an enormous number of rows, one for each of the Π_i^k.

However, we have not yet specified a computation that will actually produce a finite but adequate list of rows and the actual row coefficients. We turn next to this.

Calculation. We will next consider what is needed to do a calculation using the formulation (8). We will stress only those parts that are special to the present problem and will give only a short treatment of the more routine simplex steps. A full description is available in [8].

We first discuss the dual simplex calculation. This method requires as a

preliminary a starting basis that is dual feasible (with its accompanying nonfeasible Y values) and then the ability to iterate the following steps:

(9)
(i) Row selection: Find among the inequalities of (8) one that is not satisfied by the current Y values.
(ii) Column selection: Use the dual simplex rule.
(iii) Gaussian elimination on an inverse matrix only, with the pivot element the one resulting from the row and column choice in (i) and (ii).

For the primal simplex method we use a starting basis and a Y that are primal feasible, and the ability to iterate the following steps:

(10)
(i) Column selection: Choose for entry in the basis a column that will give an improvement.
(ii) Row selection: By the primal simplex rule.
(iii) Gaussian elimination on an inverse matrix as in 9(iii).

In the dual method, steps (ii) and (iii) are routine steps of the revised simplex method. (i) requires special consideration. In the primal method, (i) and (iii) are routine with (ii) requiring special treatment. We will consider the dual simplex situation first.

Dual simplex. Given a fixed dual feasible Y we will solve, for a fixed $t = t_k$, the linear programming problem:

$$\begin{aligned} &\text{Maximize } \theta = \sum \lambda_i^k, \\ &\text{subject to } Y \geqq \sum \lambda_i^k \mathfrak{A}_i^k. \end{aligned} \tag{11}$$

If we can solve (11), we will automatically obtain from the simplex calculation either (i) a $\theta \geqq 1$, or (ii) a nonnegative m-vector Π_1 such that

$$\begin{aligned} &\Pi_1 Y = \theta < 1, \\ &\Pi_1 \mathfrak{A}_i^k \geqq 1, \qquad \text{all } i. \end{aligned} \tag{12}$$

If in solving (11) we obtain (i) for all t_k ($k = 1, \cdots, s$) we have shown that Y contains a network satisfying the flow requirements for each t. Therefore Y, since it is dual feasible and primal feasible, is the optimal network. If we obtain (ii) for some t_k then using the Π_1 of (11) to form the vector $\Pi = (\Pi_1, -1)$ we see that

$$\Pi \cdot (Y, 1) < 0,$$

and

$$\Pi \cdot (\mathfrak{A}_i^k, 1) \geqq 0, \qquad \text{all } i,$$

so that we have found in Π an unsatisfied inequality of the set (8). The list of inequalities Π that can be obtained in this way from (11) is finite as there is one for each basis, yet this list contains an unsatisfied inequality

whenever Y is an infeasible network. Thus the procedure is finite and we deal with a finite but adequate list of inequalities.

Once the inequality has been obtained, the next steps are those of the ordinary revised dual simplex method. One transforms the inequality by means of an inverse, makes the dual simplex column choice by the usual ratio test (this is step 9(ii)), and carries out the Gaussian elimination only on the $m \times m$ (or with the objective function $(m + 1) \times (m + 1)$) inverse matrix (step 9 (iii)). One is then ready to iterate by looking for a new unsatisfied inequality.

What remains to finish the description of the dual method is to explain the calculation for solving (11). We do this in a manner closely related to the method of Ford and Fulkerson [6]. However, instead of dealing with a linear programming formulation involving a path for each column as in [6], we have a column representing an entire feasible network. This results in an economy both in the size of inverse required and in the column generating procedure. To start, we can obtain a feasible solution to (11) using any feasible network $\mathfrak{N}^k$ and $\lambda_i^k = 0$ and then maximize θ. We then obtain for this problem a set of linear programming prices $\bar{\Pi}$ with $\bar{\Pi}$ an m-vector whose components give a price for each arc. Since we have worked only with $\mathfrak{N}^k$ so far the next question is whether or not there are other feasible networks $\mathfrak{N}_i^k$ which will lead to an increase in θ.

Here again, we are facing a very large linear programming problem, this one having a great many columns, one for each $\mathfrak{N}_i^k$. To select a column, we want to choose the one for which $\bar{\Pi} \cdot \mathfrak{N}_i^k$ is minimal.

This column can easily be constructed since what we now want is the feasible network which would most cheaply meet the requirements of period t_k if the arc costs were given by $\bar{\Pi}$. This is merely the time independent requirement synthesis problem, and as we remarked in Case 1 of the Introduction, the synthesis of the cheapest network, which will give us the new column for the calculation, requires finding the shortest path between each pair of nodes p and q, then using an amount $R_{p,q}$ of all the arcs of this path, this procedure being repeated for all p, q.

However, the path by path construction of the feasible network involves unnecessary computation since it is overwhelmingly likely that portions of the same path will be used to connect several different pairs of nodes, and this leads to duplication in the backtracking (or path finding) part of the usual shortest path methods. We will next describe a method that avoids this.

As a preliminary, we follow many authors (see, for example, [2]) in defining as a special matrix product of two $m \times m$ matrices $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$ the matrix $C = \{c_{i,j}\}$ with

$$c_{i,j} = \min_s \{a_{i,s} + b_{s,j}\}. \tag{13}$$

For the cheapest feasible network calculation, we first form the $m \times m$ matrix $D = \{d_{i,j}\}$ where $d_{i,j}$ is the component of $\bar{\Pi}$ giving the price of arc i, j. We then form the D^{2^μ} powers of D by squaring (in the sense of (13)) the current power of D and discarding the previous. At the same time, we form and keep successive matrices B_μ when $b^\mu_{i,j}$ is an s value for which the minimum in (13) was obtained. After $L \leqq [\log_2 (n-1)]$ steps, where $[x]$ indicates the least integer greater than or equal to x, a D^{2^L} and accompanying B_L will be obtained for which $2L \geqq n-1$. Of course, the entries in D^{2^L} are the shortest path *distances* from node to node in the network using $d_{i,j}$ as distance. What we want, however, is a feasible network and for this we need the B_μ.

Define $\bar{B}_{L+1} = \{R_{p,q}(t_k)\}$ and define successive $\bar{B}_{\mu-1}$ as the matrix that results from starting with the zero matrix and running through the entries of $\bar{B}_\mu$ adding $\bar{b}^\mu_{i,j}$ to the i, k and k, j positions of the new matrix. k is determined by $b^\mu_{i,j} = k$.

Then $\bar{B}_1$ represents the desired cheapest feasible network.

To see this, consider the meaning of the various operations involved. $\bar{B}_{L+1}$ contains the numbers $R_{p,q}(t_k)$ which give the amount of flow the shortest s-step path $\mathfrak{s}$ between p and q must carry to satisfy the requirement. $\bar{B}_L$ is derived by adding the amount $R_{p,q}(t_k)$ to those two positions in $\bar{B}_L$ that in the matrix $D^{s/2}$ gave the lengths of the two $s/2$ step paths which combined to form $\mathfrak{s}$. Clearly, if $\mathfrak{s}$ is to carry $R_{p,q}(t_k)$, each half of it must too. This process is then carried back to the halves of the half-paths, etc., until finally the correct weight is assigned to the 1-step paths or arcs.

For an example of this calculation, see Tables A1–A5 which treat a 5-node example.

This is the calculation that is used to generate the improving columns $\mathfrak{A}_i^k$ for problem (11).

Primal method. Let us consider the steps outlined under (10) above. Step (i), column selection, can be done in the usual revised simplex manner if the Gaussian eliminations on the equations of (8) are recorded as right-multiplications of an $(m+1) \times (m+1)$ matrix. Step (iii) also involves only a Gaussian elimination over this matrix. Step (ii), however, involves finding out which of the enormous list of inequalities of (7) will be violated first when some currently non-basic variable is increased from its present level of zero.

To see how (ii) can be carried out, we first find the effect on the current values of Y of raising one of the current non-basic variables from its current value of zero. If we start from (8) and perform Gaussian eliminations recorded by right-multiplication of an $(m+1) \times (m+1)$ matrix R, the relation between the (z, Y) and the current non-basic variables T is given by $(z, Y) = R(z, T)$. If the ith of the non-basic variables is raised to

TABLE A1. *Values of* $[d_{i,j}]$.

	①	②	③	④	⑤
①	0	1	∞	6	∞
②	4	0	2	∞	∞
③	∞	∞	0	2	∞
④	∞	∞	∞	0	0
⑤	2	5	∞	∞	0

TABLE A2. D^2 *and* B_1 .

	$[d_{i,j}]^2 = D^2$						B_1				
	①	②	③	④	⑤		①	②	③	④	⑤
①	0	1	3	6	∞	①	0	2	2	4	5
②	4	0	2	4	∞	②	1	0	3	3	5
③	∞	∞	0	2	2	③	1	2	0	4	4
④	2	5	∞	0	0	④	5	5	3	0	5
⑤	2	3	7	8	0	⑤	1	1	2	1	0

TABLE A3. D^4 *and* B_2 .

	$[d_{i,j}]^4 = D^4$						B_2				
	①	②	③	④	⑤		①	②	③	④	⑤
①	0	1	3	5	5	①	0	2	3	2	3
②	4	0	2	4	4	②	1	0	3	4	3
③	4	5	0	2	2	③	4	5	0	4	5
④	2	3	5	0	0	④	1	1	1	0	5
⑤	2	3	5	7	0	⑤	1	2	2	2	0

a value θ, the current Y values Y_0 are increased by θY_1, where Y_1 is the $(i+1)$th column of R. Row selection involves finding the value $\theta_{\max}$ of θ and the inequality Π of (8) which do the following:

(a) $Y_0 + \theta_{\max} Y_1$ satisfies all the inequalities of (8) for $0 \leqq \theta \leqq \theta_{\max}$.

(b) $\Pi \cdot (Y_0 + \theta Y_1, 1) < 0$ for all $\theta > \theta_{\max}$.

To find θ and Π we consider for each period t the linear programming problem:

$$\text{Maximize } \theta$$

$$\text{subject to } Y_0 + \theta Y_1 \geqq \sum_i \lambda_i^k \mathfrak{N}_i^k,$$

$$1 \leqq \sum_i \lambda_i^k,$$

TABLE A4

	$\bar{B}_2$				
	①	②	③	④	⑤
①	0	5	4	0	0
②	1	0	0	2	0
③	0	0	0	0	4
④	1	0	0	0	0
⑤	0	1	0	0	0

TABLE A5

	$\bar{B}_1$				
	①	②	③	④	⑤
①	0	9	0	0	0
②	1	0	6	0	0
③	0	0	0	6	0
④	0	0	0	0	5
⑤	2	0	0	0	0

which can be rewritten as the problem:

$$\text{Maximize } \theta' = \theta$$

$$\text{subject to } Y_0 \geqq -\theta Y_1 + \sum_i \lambda_i^k \mathfrak{V}_i^k, \tag{14}$$

$$1 \leqq \sum_i \lambda_i^k.$$

On solving this problem, we will always get a finite maximum for θ because an unbounded θ would give feasible solutions with negative total cost.

On obtaining this finite maximum $\theta_{\max}^k$ by the simplex method, we automatically get a nonnegative m-vector Π^k, and a nonpositive scalar Π_0^k such that

$$(\Pi^k, \Pi_0^k) \cdot (Y_0, 1) = \theta_{\max}^k, \tag{15}$$

and since the scalar product of $(-1, \Pi^k, \Pi_0^k)$ with all the columns on the right of (14) will be nonnegative, we have from the first column

$$(\Pi^k, \Pi_0^k) \cdot (-Y_1, 0) - 1 \geqq 0. \tag{16}$$

Multiplying (16) by $-\theta$ and adding to (15) gives

$$(\Pi^k, \Pi_0^k) \cdot [(Y_0, 1) + \theta(Y_1, 0)] \leqq \theta_{\max}^k - \theta. \tag{17}$$

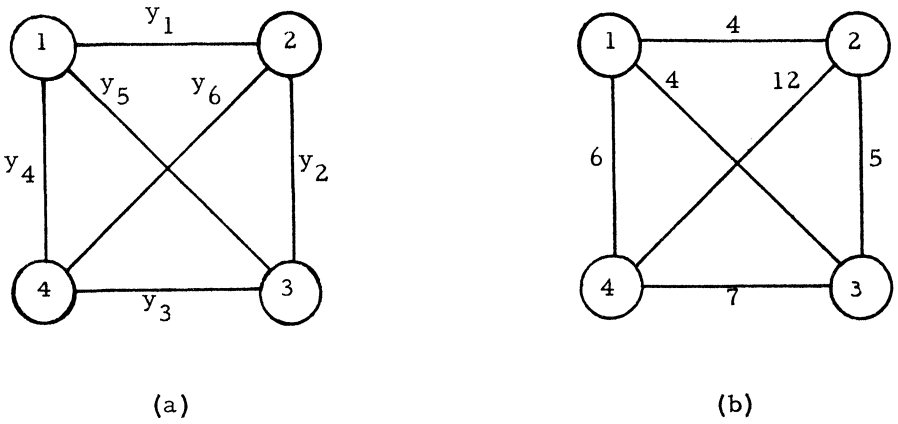

FIG. 1

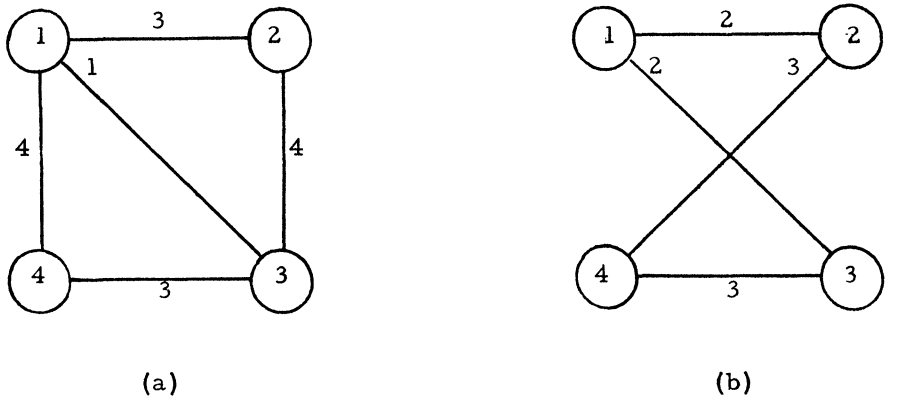

FIG. 2

Since, as we remarked above, the scalar product of $(\Pi^k, \Pi_0^k) \cdot \mathfrak{N}_i^k \geqq 0$ for all $\mathfrak{N}_i^k$, $(\Pi^k, \Pi_0^k) \cdot (Y, 1) \geqq 0$ is a valid inequality for (8).

All the inequalities of (8) that come from the condition that the network must satisfy the requirements of period t_k are satisfied by $Y_0 + \theta Y_1$ for $0 \leqq \theta \leqq \theta_{\max}^k$, because then $Y_0 + \theta Y_1$ satisfies (14) and so provides a feasible network for time period t_k. This is a step toward condition (a) above since a portion of the inequalities of (7) are satisfied. Turning now to the other condition, we see from (17) that

$$(\Pi^k, \Pi_0^k) \cdot (Y_0 + \theta Y_1, 1) < 0$$

for $\theta > \theta_{\max}^k$. So condition (b) is fulfilled with $\Pi = (\Pi^k, \Pi_0^k)$.

If we repeat the calculation (14) for each period t_k, and finally choose

$$\theta_{\max} = \min_k \theta^k_{\max} = \theta^{k_0}_{\max},$$

then by the reasoning above, condition (a) will be satisfied by $Y_0 + \theta Y_1$, $0 \leqq \theta \leqq \theta_{\max}$, and $(\Pi^{k_0}, \Pi^{k_0}_0)$ is the looked for inequality.

Taking $(\Pi^{k_0}, \Pi^{k_0}_0)$ as our selected row, we can now proceed with step (iii) of the primal procedure. This completes the description of the primal process except for the details of solving (14). However, this is so close to the procedure for solving (11) as not to require a separate description.

3. Example of the dual method. Consider the network shown in Fig. 1(a) where the costs c_i of building unit capacities are as shown in Fig. 1(b). There are two time periods with the flow requirements shown in Fig. 2(a) and Fig. 2(b).

In giving the calculation, we will also include a few simple shortcuts in the calculation. For example, we note in passing that in Fig. 1(b), the cost of y_6 is 12 where the cost of y_1 and y_4 are 4 and 6 respectively. Therefore, the arc y_6 will never be used in an optimum solution and we eliminate it from the start. This can be done for any arc whose cost equals or exceeds a sum of costs of arcs which form a path connecting the two end nodes of the arc.

To start the algorithm, we should first test the feasibility of the vector $Y = [0, 0, \cdots, 0]$ and use the inequality generated by solving (11) to start the calculation on (8). However, since any inequality, which is necessary for the network to satisfy, can be used, we can, at the very start of the calculation, get inequalities by utterly simple considerations, thus postponing the solving of (11) until more refined results are needed. The inequalities we shall use are that the sum of the branch capacities of arcs which connect one node to the others must be equal to or greater than the sum of the flow requirements between that node and the others. (This could also be done with sets of nodes and sums of requirements.)

We start then with Table B1 which represents a portion of (8) and, except for its top row, will be one inverse. The top row is the cost function to be minimized, the next five rows are identities, and the last row asserts that

$$y_1 + y_4 + y_5 - 8 = v_1 \geqq 0,$$

i.e., that node 1 must have arcs totalling at least 8, its requirement sum, connected to it. Note that the positive signs in the top row assure us of dual feasibility. The notation here is that of [8] and [11].

Using the dual simplex rule, we pivot on the starred element obtaining Table B2 (except for the bottom row). The v_1 row can now be dropped and replaced by a new inequality.

We next consider node 3, which gives the inequality

TABLE B1

	1	$-y_1$	$-y_2$	$-y_3$	$-y_4$	$-y_5$
z	0	4	5	7	6	4
y_1	0	−1	0	0	0	0
y_2	0	0	−1	0	0	0
y_3	0	0	0	−1	0	0
y_4	0	0	0	0	−1	0
y_5	0	0	0	0	0	−1
v_1	−8	−1*	0	0	−1	−1

TABLE B2

	1	$-t_1$	$-y_2$	$-y_3$	$-y_4$	$-y_5$
z	−32	4	5	7	2	0
y_1	8	−1	0	0	1	1
y_2	0	0	−1	0	0	0
y_3	0	0	0	−1	0	0
y_4	0	0	0	0	−1	0
y_5	0	0	0	0	0	−1
v_2	−8	0	−1	−1	0	−1*

TABLE B3

	1	$-t_1$	$-y_2$	$-t_4$	$-t_3$	$-t_2$
z	−81	$\frac{3}{2}$	0	$\frac{5}{2}$	$\frac{9}{2}$	$\frac{5}{2}$
y_1	7	0	1	−1	0	0
y_2	0	0	−1	0	0	0
y_3	7	$\frac{1}{2}$	1	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
y_4	0	$-\frac{1}{2}$	−1	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
y_5	1	$-\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
v_5	−8	−1	−2*	1	0	0

(18) $$-8 + y_2 + y_3 + y_5 = v_2 \geqq 0.$$

To express this inequality in terms of the current non-basic variables we turn Table B2 into the inverse by replacing the top row by (1, 0, 0, 0, 0, 0) and then multiply the row vector (−8, 0, 1, 1, 0, 1). This gives (18) in the form needed. It is placed in the position where the v_1 row was (see Table B2) and another pivot step is then made on the starred element.

We proceed in this way, considering nodes 4 and 2, with the obvious associated inequalities $y_3 + y_4 - 7 = v_3 \geqq 0$ and $y_1 + y_2 - 7 = v_4 \geqq 0$ and obtain Table B3 (except for the bottom row).

TABLE B4

	1	$-t_1$	$-t_5$	$-t_4$	$-t_3$	$-t_2$
z	-81	$\frac{3}{2}$	0	$\frac{5}{2}$	$\frac{9}{2}$	$\frac{5}{2}$
y_1	3	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	0	0
y_2	4	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	0	0
y_3	3	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{1}{2}$
y_4	4	0	$-\frac{1}{2}$	0	$-\frac{1}{2}$	$\frac{1}{2}$
y_5	1	$-\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
v_6	-3	-1^*	1	0	0	0

TABLE B5

	1	$-t_6$	$-t_5$	$-t_4$	$-t_3$	$-t_2$
z	-85.5	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{9}{2}$	$\frac{5}{2}$
y_1	$\frac{9}{2}$	$-\frac{1}{2}$	0	$-\frac{1}{2}$	0	0
y_2	$\frac{5}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	0
y_3	3	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{1}{2}$
y_4	4	0	$-\frac{1}{2}$	0	$-\frac{1}{2}$	$\frac{1}{2}$
y_5	$\frac{5}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$

At this point, these simple inequalities are satisfied and we must go through the full auxiliary calculation solving (11) to obtain a new inequality.

At the beginning the prices (which appear in the top row of Table C1) are all zero so any feasible network provides an improving column such as the one headed $\mathfrak{N}_1$. After pivoting, we have a nonzero price, and the cheapest network calculation described earlier gives a feasible network, (4, 0, 3, 4, 5), which, expressed in terms of the current variables, gives the left column in Table C2. We proceed through another pivot step and two more improving networks as shown in Tables C3 and C4 before reaching Table C5. Then the feasible network calculation shows that for a $\Pi = (0, \frac{1}{9}, 0, \frac{1}{9}, \frac{1}{9})$ there is no feasible network $\mathfrak{N}$ with $\Pi \cdot \mathfrak{N} < 1$, although $\Pi Y = \frac{1}{9} < 1$. This gives us our new inequality $\Pi Y \geqq 1$ or equivalently

$$-9 + y_2 + y_4 + y_5 = v_5 \geqq 0,$$

which is transformed to become the bottom row of Table B3. After pivoting, we obtain Table B4.

The auxiliary calculation (11) now shows that the requirements for period 1 are now met and we must now do the auxiliary calculation for the requirements of period 2. Of course, the Y values in Table B4 which give the cheapest network satisfying the first period requirements could have

been obtained much more cheaply and easily by a single feasible network calculation. However, it is necessary to obtain the inequalities represented by the rest of the matrix if one is to proceed further.

TABLE C1

$\mathfrak{N}_1$	θ	S_1	S_2	S_3	S_4	S_5	Y
−1	1	0	0	0	0	0	0
3	0	1	0	0	0	0	7
4*	0	0	1	0	0	0	0
3	0	0	0	1	0	0	7
4	0	0	0	0	1	0	0
1	0	0	0	0	0	1	1

TABLE C2

$\mathfrak{N}_2$	θ	S_1	S_2	S_3	S_4	S_5	Y
−1	1		$\frac{1}{4}$	0	0	0	0
4	0	1	$-\frac{3}{4}$	0	0	0	7
0	0	0	$\frac{1}{4}$	0	0	0	0
3	0	0	$-\frac{3}{4}$	1	0	0	7
4*	0	0	−1	0	1	0	0
5	0	0	$-\frac{1}{4}$	0	0	1	1

TABLE C3

$\mathfrak{N}_3$	θ	S_1	S_2	S_3	S_4	S_5	Y
−1	1	0	0	0	$\frac{1}{4}$	0	0
4	0	1	$\frac{1}{4}$	0	−1	0	7
1*	0	0	$\frac{1}{4}$	0	0	0	0
7	0	0	0	1	$-\frac{3}{4}$	0	7
−1	0	0	$-\frac{1}{4}$	0	$\frac{1}{4}$	0	0
9	0	0	1	0	$-\frac{5}{4}$	0	1

TABLE C4

$\mathfrak{N}_4$	θ	S_1	S_2	S_3	S_4	S_5	Y
−1	1	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	0
7	0	1	$-\frac{3}{4}$	0	−1	0	7
0	0	0	$\frac{1}{4}$	0	0	0	0
7	0	0	$-\frac{7}{4}$	1	$-\frac{3}{4}$	0	7
0	0	0	0	0	$\frac{1}{4}$	0	0
9*	0	0	$-\frac{5}{4}$	0	$-\frac{5}{4}$	1	1

TABLE C5

π_4	θ	S_1	S_2	S_3	S_4	S_5	Y
0	1	0	$\frac{1}{9}$	0	$\frac{1}{9}$	$\frac{1}{0}$	$\frac{1}{9}$
0	0	1	$\frac{2}{9}$	0	$-\frac{1}{36}$	$-\frac{7}{9}$	$\frac{56}{9}$
0	0	0	$\frac{1}{4}$	0	0	0	0
0	0	0	$-\frac{7}{9}$	1	$\frac{2}{9}$	$-\frac{7}{9}$	$\frac{56}{9}$
0	0	0	0	0	$\frac{1}{4}$	0	0
1	0	0	$-\frac{5}{36}$	0	$-\frac{5}{36}$	$\frac{1}{9}$	$\frac{1}{9}$

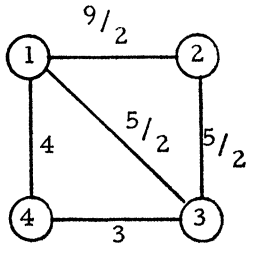

FIG. 3

An auxiliary calculation like that of Tables C1–C5 yields prices $\Pi = (\frac{1}{10}, 0, \frac{1}{10}, 0, \frac{1}{10})$ and a θ of only $\frac{7}{10}$. Hence, the inequality

$$-10 + y_1 + y_3 + y_5 = v_6 \geqq 0,$$

which is used in Table B4 to obtain Table B5. The auxiliary calculations for both periods 1 and 2 give values of $\theta \geqq 1$ so this is a feasible, and therefore optimal, network. Thus, the cheapest network for our requirements is shown in Fig. 3 and given by

$$Y = (\tfrac{9}{2}, \tfrac{5}{2}, 3, 4, \tfrac{5}{2})$$

with cost 85.5.

Primal calculation. We will redo the example using the primal method. This approach has the advantage of giving a feasible network at all times, so that calculation can be stopped if progress is too slow.

The first step is to get a starting feasible solution. This is easily done by giving the arc connecting nodes i and j a capacity equal to $\max_t R_{i,j}(t)$. Applying this to our example gives a starting solution $Y = (3, 4, 3, 4, 2, 3)$. However, just as before, there is no gain in considering an arc such as y_6 whose cost is greater than that of an alternate path connecting its end points. So we will rule out y_6 and fulfill its requirement by adding 3 units to the alternate path y_1y_4 to give the starting feasible five-arc network $Y = (5, 4, 3, 4, 2)$.

TABLE D1

	1	$-u_1$	$-u_2$	$\downarrow$ $-u_3$	$-u_4$	$-u_5$
z	−93	−4	−5	−7	−6	−4
y_1	5	1	0	0	0	0
y_2	4	0	1	0	0	0
y_3	3	0	0	1	0	0
y_4	4	0	0	0	1	0
y_5	2	0	0	0	0	1

TABLE E1

X_1	θ'	S_1	S_2	S_3	S_4	S_5	S_6	θ	Y
0	1	0	0	0	0	0	0	−1	0
3	0	1	0	0	0	0	0	0	5
4	0	0	1	0	0	0	0	0	4
3	0	0	0	1	0	0	0	1	3
4	0	0	0	0	1	0	0	0	4
1	0	0	0	0	0	1	0	0	2
1	0	0	0	0	0	0	−1	0	1

TABLE E2

X_1	θ'	S_1	S_2	S_3	S_4	S_5	S_6	θ	Y
0	1	0	0	0	0	0	0	−1	0
0	0	1	0	0	0	0	3	0	2
0	0	0	1	0	0	0	4	0	0
0	0	0	0	1	0	0	3	1*	0
0	0	0	0	0	1	0	4	0	0
0	0	0	0	0	0	1	1	0	1
1	0	0	0	0	0	0	−1	0	1

To obtain a starting basis, we introduce the variables u_i, $i = 1, \cdots, 5$, which are unrestricted in size and are defined by

$$y_1 = 5 - u_1, \qquad y_2 = 4 - u_2, \qquad y_3 = 3 - u_3,$$

$$y_4 = 4 - u_4, \qquad \text{and} \qquad y_5 = 2 - u_5.$$

This gives the starting array of Table D1. Clearly, a better solution can be obtained by increasing u_3 from its current value of 0 to some value θ. We will find the largest possible value of θ and the limiting inequality by solving the system (14) with $Y_0 = (5, 4, 3, 4, 2)$ and $Y_1 = (0, 0, 1, 0, 0)$.

TABLE E3

X_2	θ'	S_1	S_2	S_3	S_4	S_5	S_6	θ	Y
−3	1	0	0	1	0	0	3	0	0
0	0	1	0	0	0	0	3	0	2
0	0	0	1	0	0	0	4	0	0
−3	0	0	0	1	0	0	3	1	0
3*	0	0	0	0	1	0	4	0	0
3	0	0	0	0	0	1	1	0	1
1	0	0	0	0	0	0	−1	0	1

TABLE E4

X_2	θ'	S_1	S_2	S_3	S_4	S_5	S_6	θ	Y
0	1	0	0	1	1	0	7	0	0
0	0	1	0	0	0	0	3	0	2
0	0	0	1	0	0	0	4	0	0
0	0	0	0	1	1	0	7	1	0
1*	0	0	0	0	$\frac{1}{3}$	0	$\frac{4}{3}$	0	0
0	0	0	0	0	−1	1	−3	0	1
0	0	0	0	0	$\frac{1}{3}$	0	$-\frac{7}{3}$	0	1

TABLE D2

	1	$-u_1$	$-u_2$	$-u_3$	$-u_4$	$-u_5$
z	−93	−4	−5	−7	−6	−4
y_1	5	1	0	0	0	0
y_2	4	0	1	0	0	0
y_3	3	0	0	1	0	0
y_4	4	0	0	0	1	0
y_5	2	0	0	0	0	1
v_1	0	0	0	1*	1	0

TABLE D3

	1	$-u_1$	$\downarrow$ $-u_2$	$-t_1$	$-u_4$	$-u_5$
z	−93	−4	−5	7	1	−4
y_1	5	1	0	0	0	0
y_2	4	0	1	0	0	0
y_3	3	0	0	−1	−1	0
y_4	4	0	0	0	1	0
y_5	2	0	0	0	0	1
v_2	1	0	1*	−1	−1	1

TABLE D4

	1	$-u_1$	$-t_2$	$-t_1$	$\downarrow$ $-u_4$	$-u_5$
z	−88	−4	5	2	−4	1
y_1	5	1	0	0	0	0
y_2	3	0	−1	1	1	−1
y_3	3	0	0	−1	−1	0
y_4	4	0	0	0	1	0
y_5	2	0	0	0	0	1
v_3	0	0	−1	1	2*	0

TABLE D5

	1	$\downarrow$ $-u_1$	$-t_2$	$-t_1$	$-t_3$	$-u_5$
z	−88	−4	3	4	2	1
y_1	5	1	0	0	0	0
y_2	3	0	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	−1
y_3	3	0	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	0
y_4	4	0	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	0
y_5	2	0	0	0	0	1
v_4	0	1	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	1

TABLE D6

	1	$-t_4$	$-t_2$	$-t_1$	$-t_3$	$\downarrow$ $+u_5$
z	−88	4	1	2	4	−5
y_1	5	−1	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	+1
y_2	3	0	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	+1
y_3	3	0	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	0
y_4	4	0	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	0
y_5	2	0	0	0	0	−1
v_5	1	−1	0	1	−1	2

For the solution of (14), which is of course again a linear programming problem, we need a starting basis, i.e., for a starting value of θ (which will be zero) we need to express $Y_0 + \theta Y_1$ as a sum of feasible networks and slacks. This yields the starting Table E1.

Pivoting on the starred element now yields an increase in θ and Table E2.

The θ column now is an improving one so we pivot on that column giving, except for the left-most column, Table E3. Next, using the linear programming prices from the top row, we do the special matrix calculation to get

TABLE D7

	1	$-t_4$	$-t_2$	$-t_1$	$-t_3$	$-t_5$
z	85.5	$\frac{3}{2}$	1	$\frac{9}{2}$	$\frac{3}{2}$	$\frac{5}{2}$
y_1	$\frac{9}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	0	0	$-\frac{1}{2}$
y_2	$\frac{5}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	0	0	$-\frac{1}{2}$
y_3	3	0	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	0
y_4	4	0	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	0
y_5	$\frac{5}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
v_5	0	0	0	0	0	-1

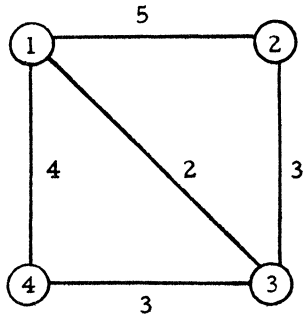

FIG. 4

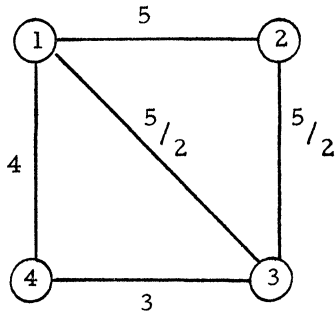

FIG. 5

a cheapest feasible network which turns out to be (3, 4, 0, 7, 4). This, after being enlarged to (0, 3, 4, 0, 7, 4, 1) by adding its coefficients in the top and bottom rows, is updated to become the left-most column in Table E3. Pivoting on the starred element gives Table E4.

θ cannot be increased any more as our column generating procedure shows. Thus, the inequality

$$-7 + y_3 + y_4 = v_1 \geqq 0 \tag{19}$$

TABLE F1. *First period requirements*

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
①	×	1	1	6	3	4	7	7	3	9
②		×	2	3	9	5	0	4	4	5
③			×	9	6	2	9	3	7	5
④				×	3	6	11	2	5	8
⑤					×	4	0	6	2	9
⑥						×	2	6	1	1
⑦							×	3	3	4
⑧								×	12	4
⑨									×	17
⑩										×

TABLE F2. *Second period requirements*

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
①	×									
②	7	×								
③	2	14	×							
④	3	9	6	×						
⑤	1	15	7	6	×					
⑥	8	4	2	12	3	×				
⑦	7	6	9	2	1	1	×			
⑧	0	5	4	3	7	11	4	×		
⑨	14	0	17	12	8	1	19	4	×	
⑩	3	17	14	4	19	2	7	3	8	×

obtained from the prices appearing in Table E4 is the binding inequality among those that insure the continued first period feasibility of our network. Ordinarily we would repeat this calculation using second period feasible networks to find the binding inequality among those that insure second period feasibility and then choose the one with the smaller θ. However, in this case with $\theta_{\max}$ already 0, this second step is unnecessary.

(19) is adjoined to Table D2 and a primal pivot step is made.

The result (except for the bottom row) appears in Table D3. Now a further improving column is (0, 1, 0, 0, 0). To find θ and the binding constraint, the calculation of (14) is repeated with $Y_0 = (5, 4, 3, 4, 2)$ and $Y_1 = (0, 1, 0, 0, 0)$. This time, both periods are considered and the binding inequality, which comes from the first period requirement, is

$$-8 + y_2 + y_3 + y_5 = v_2 \geqq 0, \tag{20}$$

with $\theta_{\max} = 1$. Updating this inequality gives the bottom row in Table D3, and a primal pivot step produces the new network of Table D4 and Fig. 4.

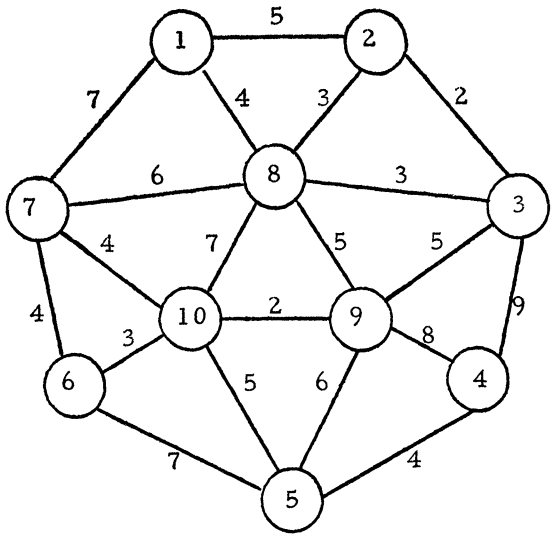

FIG. 6. *Unit costs between stations*

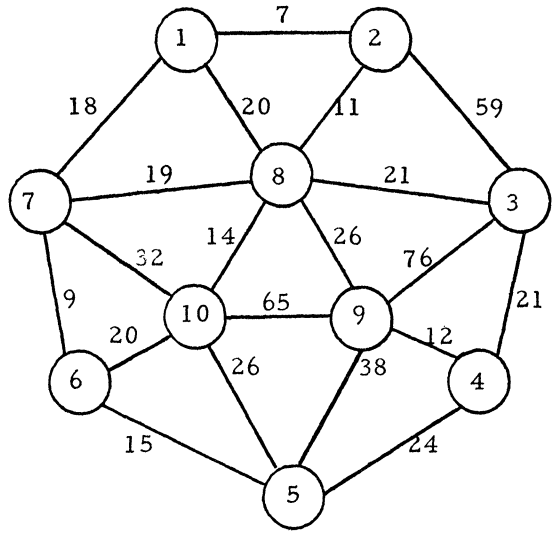

FIG. 7. *Final network, total cost* = 2375

This process is then repeated with the auxiliary calculation yielding first the inequality

$$-9 + y_2 + y_4 + y_5 = v_3 \geqq 0,$$

which becomes the bottom row of Table D4. After pivoting, Table D4

becomes Table D5. The auxiliary calculation then yields

$$-10 + y_1 + y_3 + y_5 = v_4 \geqq 0,$$

which updated appears in the bottom of Table D5. The next pivot yields Table D6. The next inequality is a recurrence of

$$-7 + y_1 + y_2 \geqq 0.$$

Pivoting yields Table D7. In Table D7, there is no longer any improving column so the optimal network has been found and is the same as the one given by the dual calculation.

4. A larger example. A ten-node twenty-arc network was considered. Unit costs are given in Fig. 6; the requirements for the two periods involved appear in Tables F1 and F2. The problem was run on the IBM 7094 using the dual method. After a run of ten minutes, the minimum synthesis shown in Fig. 7 was obtained.

REFERENCES

[1] J. F. BENDERS, *Partitioning in mathematical programming*, Thesis, Utrecht University, 1960.

[2] C. BERGE, *Theory of Graphs*, trans. Alison Doig, John Wiley, New York, 1962, pp. 138–139.

[3] R. T. CHIEN, *Synthesis of a communication net*, IBM J. Res. Develop., 4 (1960), pp. 311–320.

[4] J. FARKAS, *Uber die Theorie einfachen Ungleichungen*, J. Reine Angew. Math., 124 (1901), pp. 1–27.

[5] L. R. FORD, JR., AND D. R. FULKERSON, *A simple algorithm for finding maximal network flows and an application to the Hitchcock problem*, Canad. J. Math., 9 (1957), pp. 210–218.

[6] ———, *A suggested computation for maximal multi-commodity network flows*, Management Sci., 5 (1958), pp. 97–101.

[7] ———, *Flows in Networks*, Princeton University Press, Princeton, 1962.

[8] R. E. GOMORY, *An algorithm for integer solutions to linear programs*, Recent Advances in Mathematical Programming, McGraw-Hill (ed. Graves and Wolfe), New York, 1963.

[9] ———, *Large and non-convex problems in linear programming*, Proceedings of Symposia in Applied Mathematics, Vol. 15, Amer. Math. Soc., 1963.

[10] R. E. GOMORY AND T. C. HU, *Multi-terminal network flows*, this Journal, 9 (1961), pp. 551–570.

[11] ———, *An application of generalized linear programming to network flows*, this Journal, 10 (1962), pp. 260–283.

[12] T. C. HU, *Multi-commodity network flows*, Operations Res., 11 (1963), pp. 344–360.

[13] D. T. TANG, *Communication Networks with Simultaneous Flow Requirements*, IEEE Trans. Circuit Theory, CT-9 (1962).