COMMON TERMINAL MULTICOMMODITY FLOW

by

B. Rothfarb

N. P. Shein

I. T. Frisch

Memorandum No. ERL-M 209

26 April 1967

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

COMMON TERMINAL MULTICOMMODITY FLOW[*]

B. Rothfarb, N. P. Shein
and I. T. Frisch

University of California, Berkeley, California

ABSTRACT

An algorithm is presented to determine a multicommodity flow pattern which maximizes the objective function $\sum_{i=1}^{n} \alpha_i f_i$ for graphs having n sources and a common terminal where f_i is the amount of flow of the ith commodity and $\{\alpha_i\}$ is a given set of nonnegative constants. The algorithm can also be used to minimize the function $\sum_{i=1}^{n} \beta_i |r_i - f_i|$ for a given set of nonnegative constants $\{r_i\}$ and $\{\beta_i\}$.

[*] The research reported herein was supported in part by the U. S. Army Research Office--Durham under Grant DA-ARO-D-31-124-G776 and in part by the Joint Services Electronics Program (U. S. Army, U. S. Navy and U. S. Air Force) under Grant AF-AFOSR-139-66.

I. INTRODUCTION

In this paper we investigate an important special case of the general multicommodity flow problem.[1] The networks to be considered consist of a finite set of nodes, n of which are source nodes and one of which is the common terminal node. Between pairs of nodes are directed branches having nonnegative capacities. The flow of any commodity must be conserved at all nodes except its source and the terminal. The sum of all flows through an arc cannot exceed the capacity of that arc.

Let f_i denote the amount of flow of commodity i from the ith source node, i, to the terminal node t. We will present methods for solving the following problems:

(i) For a given set of nonnegative constants, $\{\alpha_i\}$, find a flow pattern which maximizes $\sum_{i=1}^{n} \alpha_i f_i$.

(ii) Test the simultaneous feasibility of a given set of flow requirements, $\{r_i\}$.

(iii) Given a set of flow requirements and a set of nonnegative constants, $\{\beta_i\}$, find a flow pattern which minimizes $\sum_{i=1}^{n} \beta_i |r_i - f_i|$.

For the general multicommodity flow problem, necessary and sufficient conditions for the feasibility of a set of simultaneous flow requirements are unknown. However, for networks with a common terminal, we can given a simple procedure for testing feasibility, and also perform the optimizations indicated in (i) and (iii). This network model corresponds to many practical situations. Some examples are:

(i) n warehouses each shipping a commodity having a specified relative value to a common destination through a given road network, and

(ii) a communication network with a headquarters requiring simultaneous communication with certainfield locations having different priorities.

II. ALGORITHM FOR OPTIMIZING THE OBJECTIVE FUNCTION

We first consider the problem of finding a flow pattern which maximizes the linear objective function $\sum_{i=1}^{n} \alpha_i f_i$. In applications this function usually represents some performance criterion such as profit. It is convenient to represent those α's with the ith largest numerical value by a single constant $\hat{\alpha}_i$. More explicitly

$$\hat{\alpha}_1 = \max_i \alpha_i, \ \hat{\alpha}_2 = \max_{i \ni \alpha_i \neq \hat{\alpha}_1} \alpha_i, \ \ldots, \ \hat{\alpha}_d = \max_{i \ni \alpha_i \neq \hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{d-1}} \alpha_i$$

where d is the number of distinct values in the set $\{\alpha_i\}$. The algorithm below presents a method for maximizing the objective function:

COMMON-TERMINAL MULTICOMMODITY FLOW ALGORITHM

(1) For each $\hat{\alpha}_i$, create a new node s_i. Direct branches of infinite capacity from s_i to all source nodes, j, for which $\alpha_j = \hat{\alpha}_i$. Also create a new source node s.

(2) Direct branch of infinite capacity from node s to node s_i, where initially $i = 1$.

(3) Maximize the s-t flow using the single-commodity labeling algorithm[1].

(4) If $i < d$, increase i by one and return to step (2). If $i = d$, go to step (5).

(5) Perform an arc-chain decomposition on the flow pattern. Assign to commodity i those flow chains using that branch created in step (1) which is incident to the ith source node.

We now prove that the flow values obtained from the algorithm maximize the objective function $\sum_{i=1}^{n} \alpha_i f_i$. Let $\hat{f}_i$ be the sum of those flow values whose coefficients in the cost function are equal to $\hat{\alpha}_i$, i.e., $\hat{f}_i = \sum_{j \ni \alpha_j = \hat{\alpha}_i} f_j$. Define $F_k = \sum_{i=1}^{k} \hat{f}_i$ and $C_k = \sum_{i=1}^{k} \hat{\alpha}_i \hat{f}_i$ and denote by $\tilde{F}_k$ and $\tilde{C}_k$ the respective maximum values of these sums. The subscript "o" will be used to denote values obtained from the algorithm. In successive repetitions of step (3) of the algorithm, F_k is maximized in the order $k = 1, 2, \ldots, d$. Thus

$$\hat{f}_{ko} = \tilde{F}_k - \tilde{F}_{k-1} \quad \text{for } k = 2, 3, \ldots, d \tag{1a}$$

and

$$\hat{f}_{1o} = \tilde{F}_1 . \tag{1b}$$

We will now show by induction on k that the algorithm maximizes C_k. For $k = 1$, $C_1 = \hat{\alpha}_1 \hat{f}_1$. Therefore to maximize C_1 we need only maximize $\hat{f}_1$. Thus from (1b) $C_{1o} = \tilde{C}_1$ and the statement is true for $k = 1$.

We now assume

$$C_{ko} = \tilde{C}_k \tag{2}$$

and show that $C_{(k+1)o} = \tilde{C}_{k+1}$. From the definition of $C_{(k+1)o}$ we have

$$C_{(k+1)o} = C_{ko} + \hat{\alpha}_{k+1} \hat{f}_{(k+1)o} . \tag{3}$$

Combining (1a), (2), and (3) gives

$$C_{(k+1)o} = \tilde{C}_k + \hat{\alpha}_{k+1} \tilde{F}_{k+1} - \hat{\alpha}_{k+1} \tilde{F}_k . \tag{4}$$

We will prove that $C_{(k+1)o} = \tilde{C}_{k+1}$ by showing that if $\hat{f}_{k+1}$ is either smaller or larger than $\hat{f}_{(k+1)o}$, the resulting C_{k+1} is less than $C_{(k+1)o}$.

<u>Case A</u>. In (3) $C_{ko} = \tilde{C}_k$ by the inductive hypothesis (2). Thus an $\hat{f}_{k+1} < \hat{f}_{(k+1)o}$ results in a $C_{k+1} < C_{(k+1)o}$.

<u>Case B</u>. Next consider a flow pattern for which $\hat{f}_{k+1} > \hat{f}_{(k+1)o}$, i.e.,

$$\hat{f}_{k+1} = \tilde{F}_{k+1} - \tilde{F}_k + \beta, \ \beta > 0 . \tag{5}$$

By definition we have $F_k = F_{k+1} - \hat{f}_{k+1}$, which together with (5) yields

$$F_k = (F_{k+1} - \tilde{F}_{k+1}) + \tilde{F}_k - \beta . \tag{6}$$

Since $F_{k+1} \le \tilde{F}_{k+1}$, (6) gives

$$F_k \le \tilde{F}_k - \beta . \tag{7}$$

As an intermediate step we now use (7) to show that the corresponding C_k must satisfy

$$C_k \le \tilde{C}_k - \hat{\alpha}_k \beta \tag{8}$$

Suppose to the contrary, i.e., $C_k > \tilde{C}_k - \hat{\alpha}_k \beta$. Since by (7) we could increase F_k by β, we could increase C_k by at least $\hat{\alpha}_k\beta$ since $\hat{\alpha}_j > \hat{\alpha}_k$ for $j < k$. But then the new value of C_k would be greater than $\tilde{C}_k$. This contradiction establishes (8).

Combining the definition $C_{k+1} = C_k + \hat{\alpha}_{k+1} \hat{f}_{k+1}$ with (8) gives

$$C_{k+1} \le \tilde{C}_k + \hat{\alpha}_{k+1} \hat{f}_{k+1} - \hat{\alpha}_k \beta . \tag{9}$$

Substituting (5) into (9) we have

$$C_{k+1} \le \left[\tilde{C}_k + \hat{\alpha}_{k+1} \tilde{F}_{k+1} - \hat{\alpha}_{k+1} \tilde{F}_k\right] - (\hat{\alpha}_k - \hat{\alpha}_{k+1})\beta \tag{10}$$

Comparing (4) and (10) and noting that $\hat{\alpha}_k > \hat{\alpha}_{k+1}$ gives $C_{k+1} < C_{(k+1)o}$.

Case A and Case B establish that the algorithm yields the optimum value of $\hat{f}_{k+1}$. Since C_{ko} was assumed maximum, we have $C_{(k+1)o} = \tilde{C}_{k+1}$ and, in particular, $C_{do} = \tilde{C}_d$. Noting that $C_d = \sum_{i=1}^{d} \hat{\alpha}_i \hat{f}_i = \sum_{i=1}^{n} \alpha_i f_i$, it follows that the algorithm maximizes the given objective function.

We now illustrate the algorithm by maximizing the objective function $14f_1 + 14f_2 + 13f_3 + 6f_4 + 15f_5$ for the graph of Fig. 1a, where f_i denotes the amount of flow from source node i. The set $\{\hat{\alpha}_i\}$ corresponding to this cost function is: $\hat{\alpha}_1 = 15$, $\hat{\alpha}_2 = 14$, $\hat{\alpha}_3 = 13$, $\hat{\alpha}_4 = 6$. The added nodes and branches specified in step (1) and an s-t flow pattern obtained by the initial execution of step (3) are shown in Fig. 1b. The dashed line indicates a saturated s-t cut set. Figure 1c shows the result of a second iteration of step (3). The third iteration increases the flow by one unit along the path s, s_3, e, f, a, b, c, d, t and the final iteration increases the flow along the path s, 4, d, t. The resulting flow pattern is shown in Fig. 1d. In Fig. 1e an arc-chain decomposition has been used to identify commodity 5. Note that the order of the identification process is arbitrary. The final flow pattern is shown in Fig. 1f and the maximum value of the objective function is seen to be 490.

It is important to note that the validity of the algorithm depends upon waiting until the final step to identify the flows of the individual commodities. Simply sending maximum amounts of each commodity in priority order will not in general yield an optimum solution. For instance, if in the above example we had initially sent a maximum amount of flow of commodity 5 according to the pattern shown in Fig. 1b, an optimal solution could not result without subsequent rearrangement of this pattern. In fact, it can be shown that every optimum solution to the above example has 3 units of commodity 5 in the branch from f to e and 4 units of commodity 5 in the branch from f to a.

IV. NETWORKS WITH FLOW REQUIREMENTS

The algorithm suggests a simple procedure for testing the simultaneous feasibility of a set $\{r_i\}$ of flow requirements, where r_i is the flow required from source node i. We modify the network by forming a node s and directing from s to each source node i a branch, called a requirement branch, having capacity equal to r_i.

The requirements $\{r_i\}$ are feasible if and only if the requirement branches form a minimum s-t cut set. If these branches do form a minimum s-t cut set, then they are saturated by every maximal s-t flow. The flows of the individual commodities can be identified by performing an arc-chain decomposition and assigning to source node i those flow chains which use the requirement branch from s to i. On the other hand, if the requirement branches do not form a minimum s-t cut-set, then an s-t flow equal to the sum of the requirements cannot be achieved and the set of requirements is not feasible.

We consider next the situation where in addition to the given set of flow requirements we have a penality, β_i, associated with the ith commodity. Specifically, β_i represents the loss incurred for each unit of the ith requirement left unfulfilled. We seek a flow pattern which minimizes the function $\sum_{i=1}^{n} \beta_i |r_i - f_i|$. The common-terminal multicommodity algorithm can be applied here, with the following modifications.

(i) The set $\{\alpha_i\}$ is replaced by $\{\beta_i\}$.

(ii) The branches from the nodes s_i to the source nodes are now given capacities equal to the source requirements instead of infinite capacities.

As an example, we use the network of Fig. 1a with the requirements $r_i = 10$, $i = 1, 2, 3, 4, 5$ and the penalities $\beta_1 = \beta_2 = 10$, $\beta_3 = 5$, $\beta_4 = 3$, $\beta_5 = 70$. For these β_i's, the commodities have the same relative priorities as in the previous example. However, the introduction of the requirements changes the solution to that shown by Figs. 2a and 2b. The minimum value of the penality function is seen to be 65.

V. DISCUSSION OF RESULTS

The class of graphs treated in this paper includes those which contain undirected branches. Such branches pose no new problems since all the technqiues used in the algorithm can be applied without modification. In addition, the algorithm can be used to solve the reverse problem: given a network ha ring a common source and n terminals, find a set of flows, $\{f_i\}$, such that $\sum_{i=1}^{n} \alpha_i f_i$ is maximized. An optimum solution can be found by first reversing the direction of all branches and interchanging the roles of source and terminal, then applying the algorithm, and finally reversing the flows thus obtained.

The algorithm presented is very efficient for both hand computation and computer implementation. In particular, only one execution of the single commodity labeling algorithm is, in effect, required, and only two commodities need be simultaneously considered when performing the arc-chain decomposition.

REFERENCES

[1] L. R. Ford Jr., and D. R. Fulkerson, Flows in Networks, Princeton University Press, 1962.

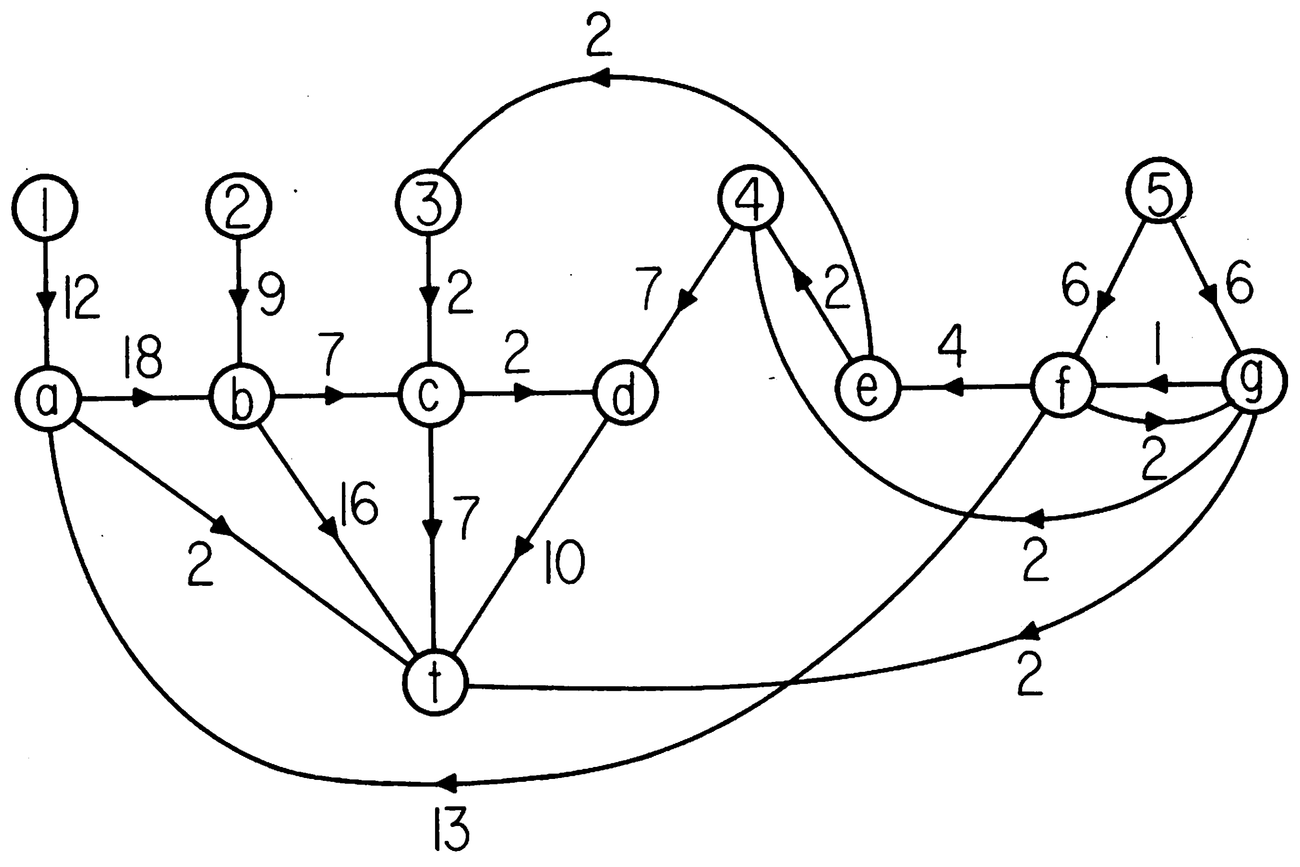

Fig. 1a. Network configuration and branch capacities.

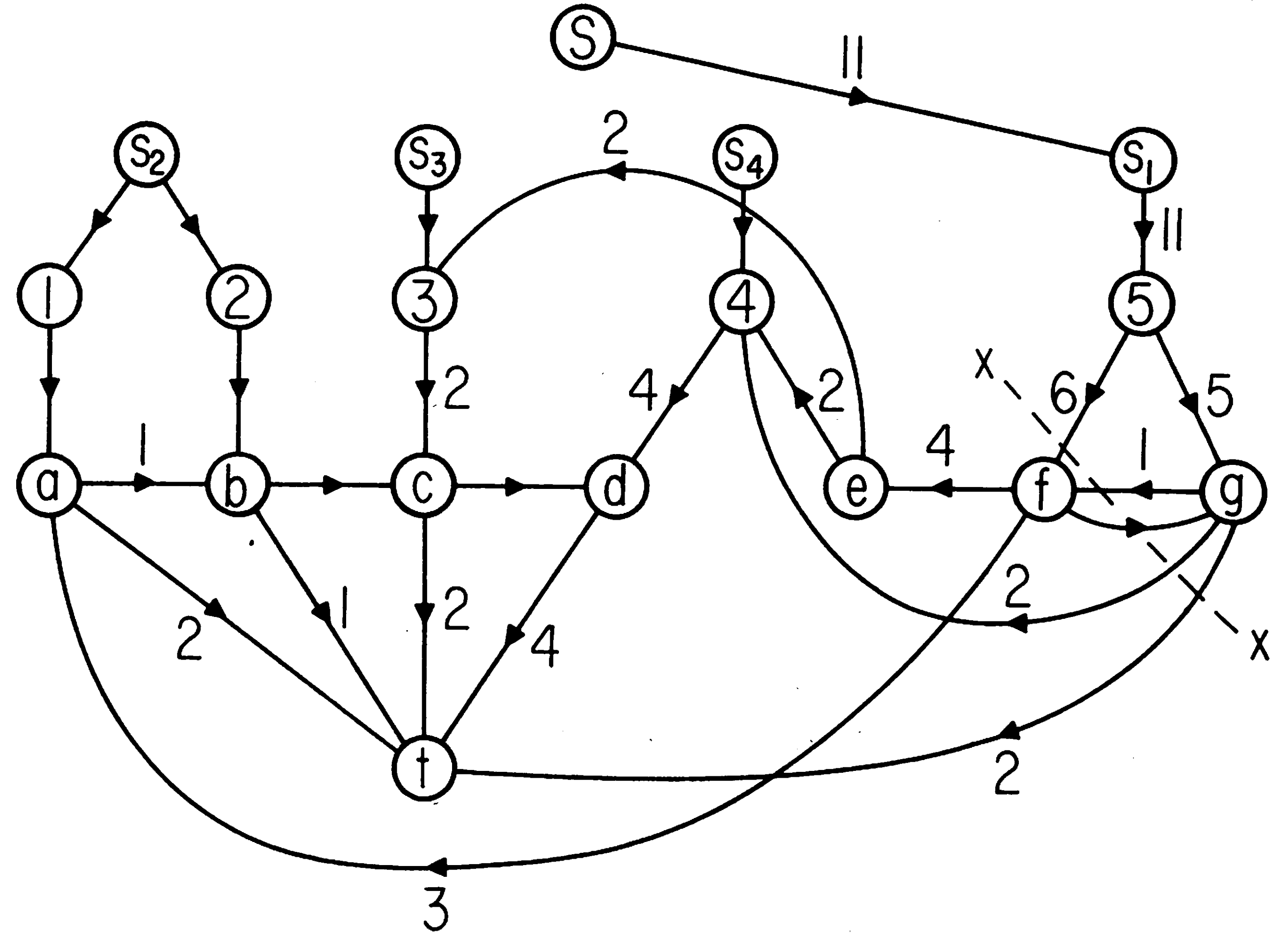

Fig. 1b. Flow pattern resulting from first iteration of step 2.

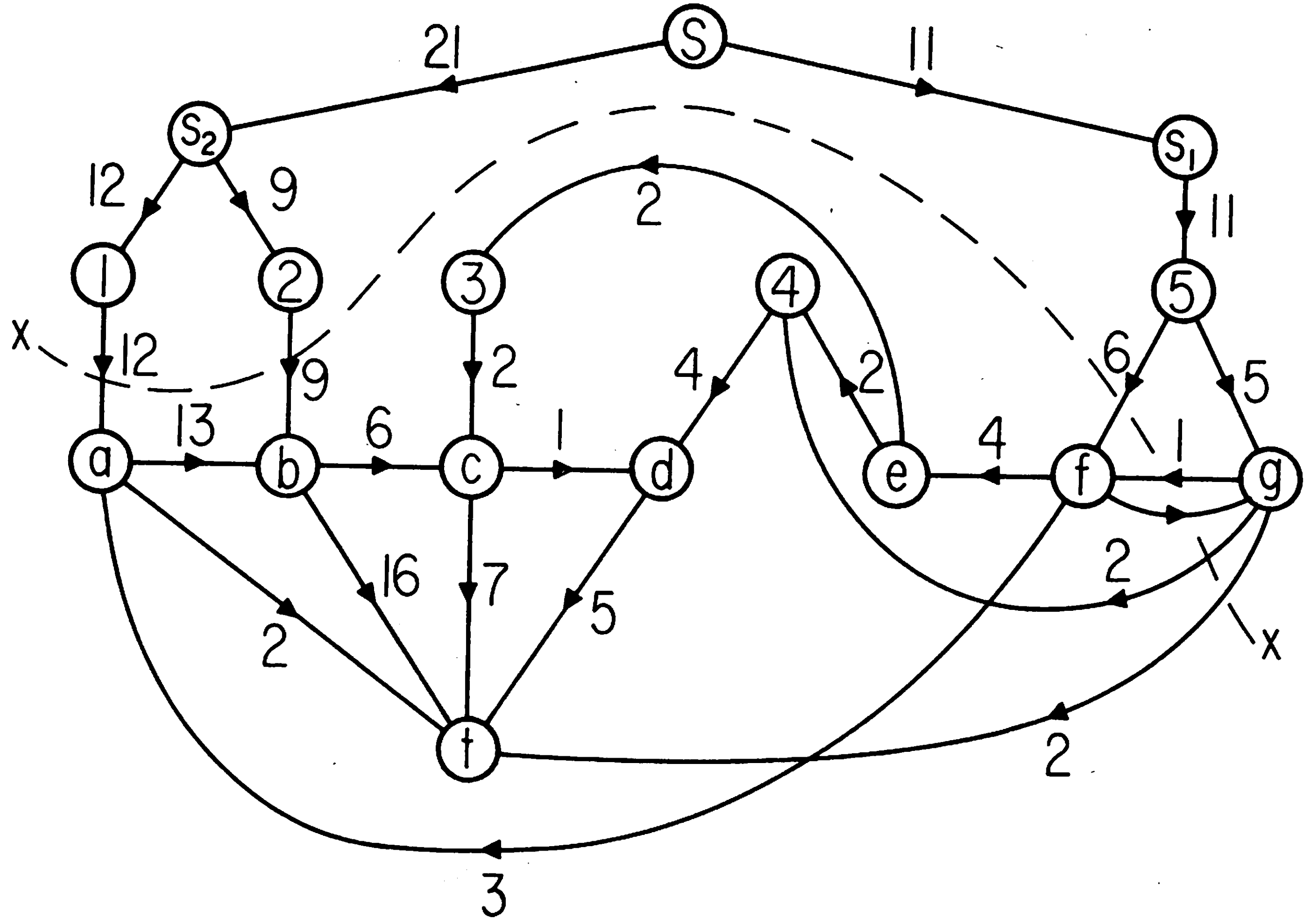

Fig. 1c. Flow pattern resulting from second iteration of step 2.

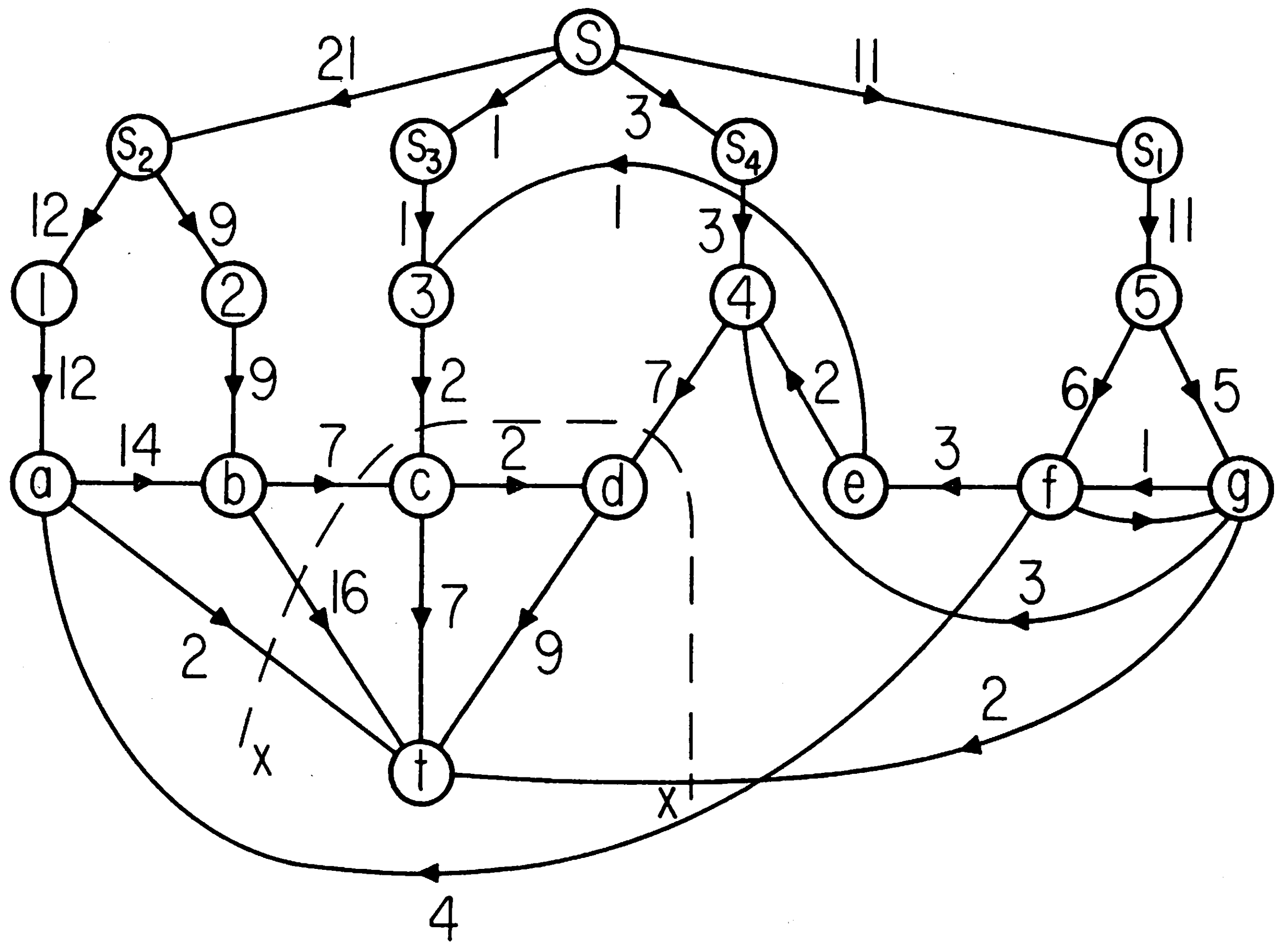

Fig. 1d. Final single commodity flow pattern.

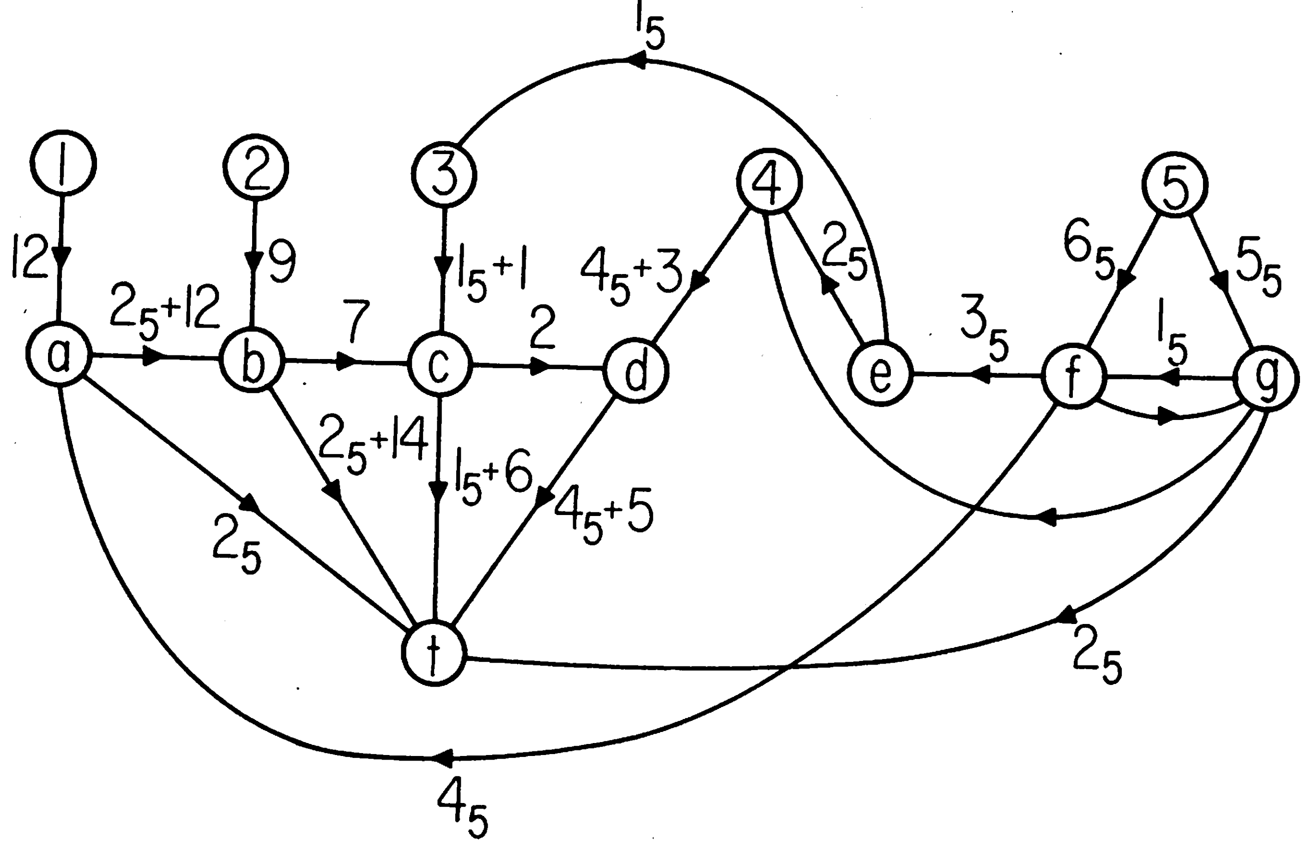

Fig. 1e. Identification of commodity 5 using arc-chain decomposition.

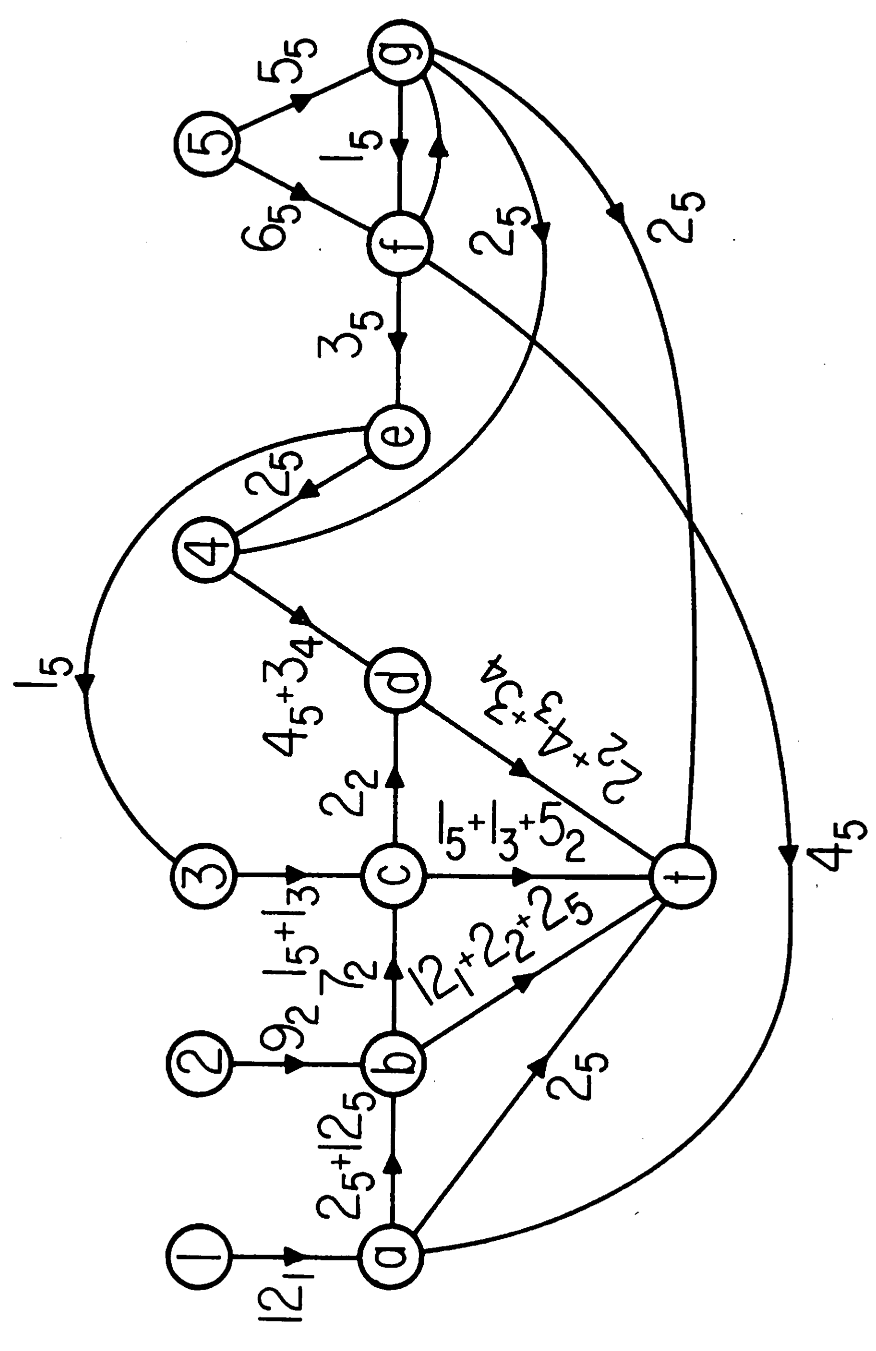

Fig. 1f. Optimum flow pattern.

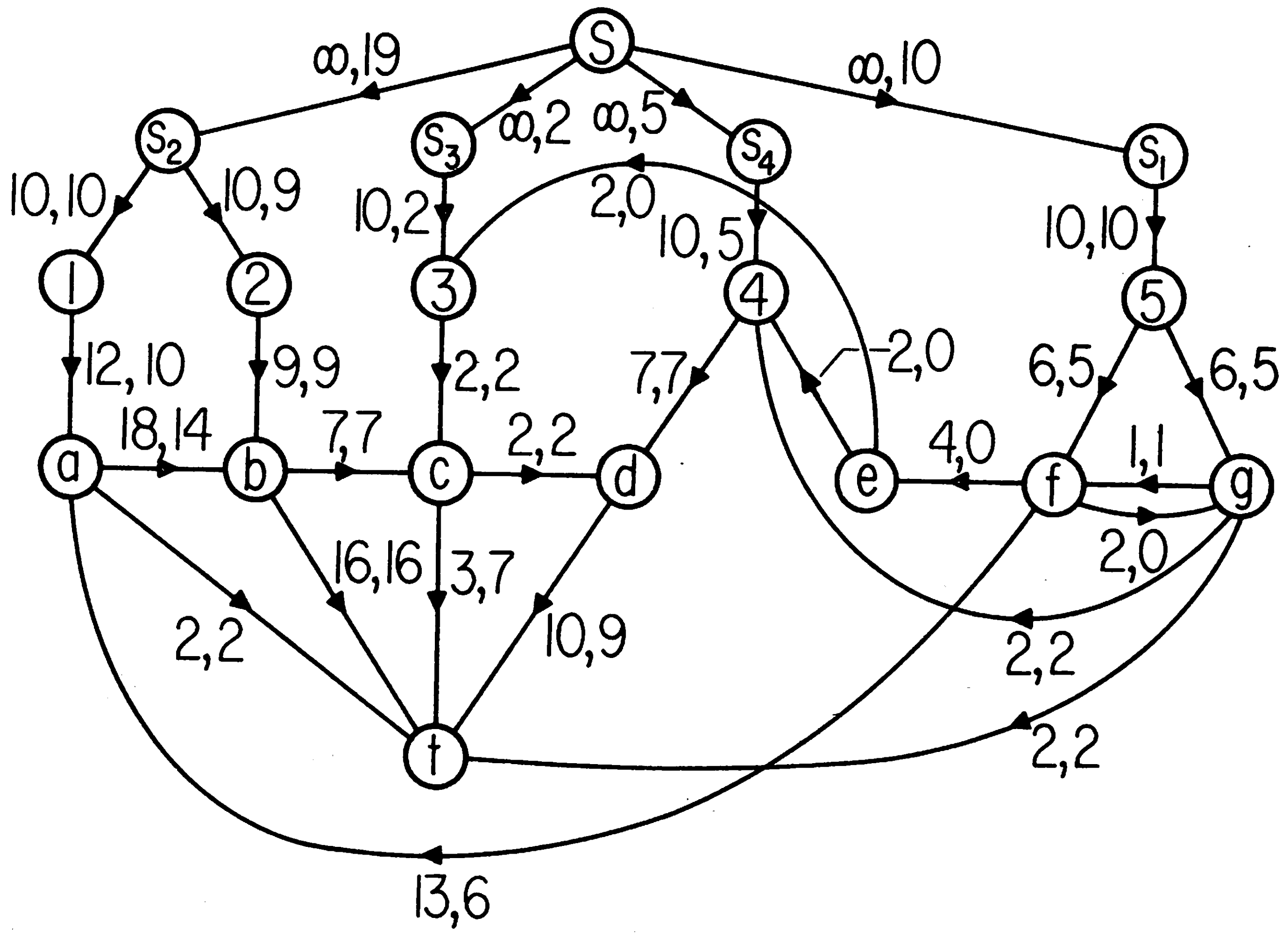

Fig. 2a. Final single commodity flow pattern. Branches are labelled: capacity, flow.

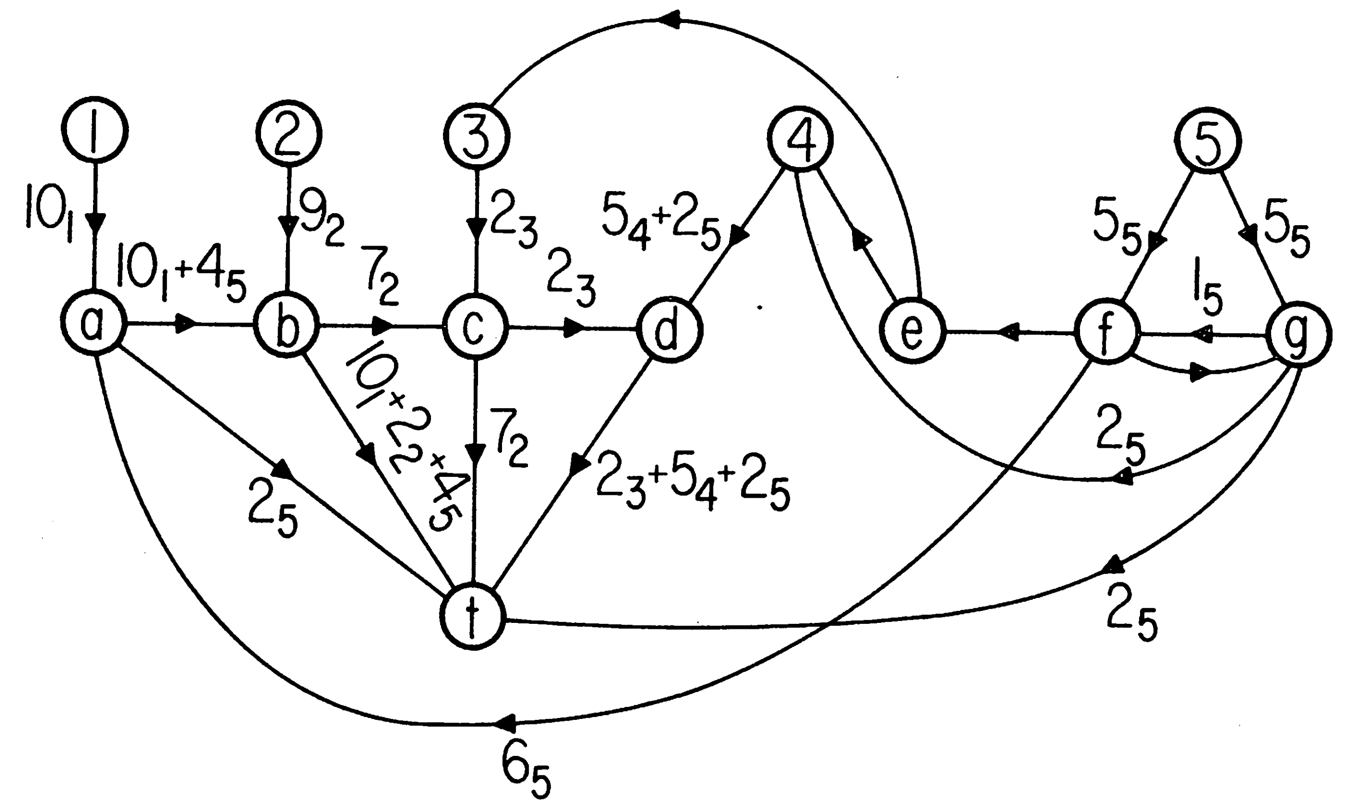

Fig. 2b. Optimum flow pattern.