CLEARINGHOUSE FOR FEDERAL SCIENTIFIC AND TECHNICAL INFORMATION, CFSTI
DOCUMENT MANAGEMENT BRANCH 410.11

LIMITATIONS IN REPRODUCTION QUALITY

Accession # 604709

☒ 1. We regret that legibility of this document is in part unsatisfactory. Reproduction has been made from best available copy.

☐ 2. A portion of the original document contains fine detail which may make reading of photocopy difficult.

☐ 3. The original document contains color, but distribution copies are available in black-and-white reproduction only.

☐ 4. The initial distribution copies contain color which will be shown in black-and-white when it is necessary to reprint.

☐ 5. Limited supply on hand; when exhausted, document will be available in Microfiche only.

☐ 6. Limited supply on hand; when exhausted document will not be available.

☐ 7. Document is available in Microfiche only.

☐ 8. Document available on loan from CFSTI (TT documents only).

☐ 9.

Processor: GL

NBS____ 9/64

Best
Available
Copy

604709

1

COMPUTATION OF MAXIMAL FLOWS IN NETWORKS

by

D. R. Fulkerson
G. B. Dantzig

P-677

1 April 1955

Approved for OTS release

18p

COPY 1 OF 1

HARD COPY $. 1.00
MICROFICHE $. 0.50

DDC
RECEIVED
AUG 27 1964
DDC-IRA D

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

SUMMARY

A simple computational method, based on the simplex algorithm of linear programming, is proposed for the following problem:

"Consider a network (e.g., rail, road, communication network) connecting two given points by way of a number of intermediate points, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given point to the other."

COMPUTATION OF MAXIMAL FLOWS IN NETWORKS

by

D. R. Fulkerson
G. B. Dantzig

The linear programming formulation of the network-flow problem given in [1] is not only useful as a theoretical tool: When suitably interpreted, it provides in addition a simple and efficient hand-computing scheme. It is our aim in section I to describe this computation for networks with capacity constraints on arcs only. It is unnecessary to use programming terminology in this description, and so we dispense with it. A direct proof will be given in section II that the computational method assures one of finding a maximal flow in an arbitrary network.

I. Description of Method

The problem (see [1], [2]) may be stated as follows: One is given a connected network of arcs and nodes with two distinguished nodes, called source and sink respectively. All other nodes are called intermediate. Each arc in the network has associated with it a positive integer, its flow capacity. The direction of flow is assumed to be away from the source in arcs having the source as endpoint, and into

the sink in sink arcs; otherwise no flow direction is specified. Subject to the conditions that the flow in an arc does not exceed its capacity, and that the total flow into any intermediate node is equal to the flow out of it, it is desired to find a maximal flow in the network, i.e., a flow which maximizes the sum of the flows in source (or sink) arcs.

In order to illustrate the computing method, let us consider a simple example. Suppose we have the network of Fig. 1 with source A,

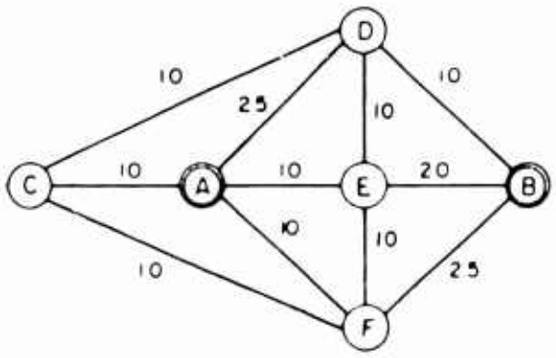


Fig 1

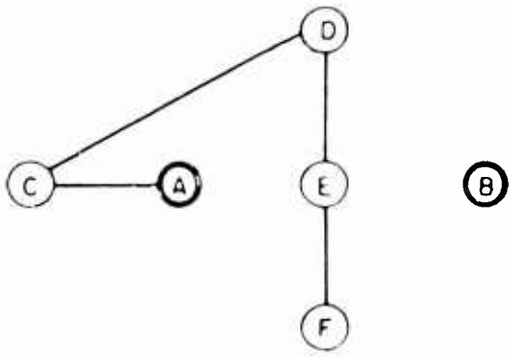


Fig 1a

sink B, and arc capacities as indicated. To start out, select two trees* of arcs, one branching out from A, the other from B,

---

*A tree is a connected linear graph without circuits; that is, there is one and only one chain of arcs joining each pair of nodes.

so that every intermediate node is reached by just one of the trees. Call these T_A and T_B. For example, T_B might consist of B alone, and T_A might be the set AC, CD, DE, EF (see Fig. 1a). Notice that since the network is connected, it is always possible to select two such trees.* Next introduce any arc which leads from T_A to T_B. There will then be just one chain from A to B; flow as much as possible along this chain. In the example, EB is such an arc, and we have then the flow diagram of Fig. 2 with all of the arcs AC,

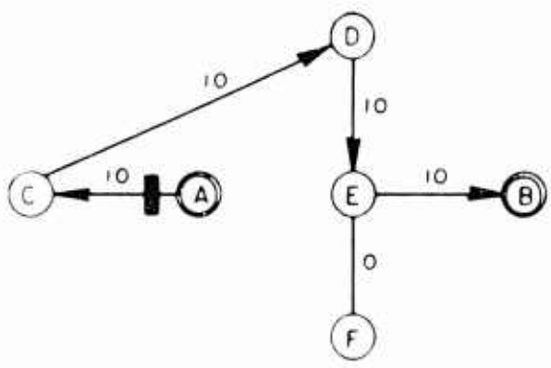


Fig 2

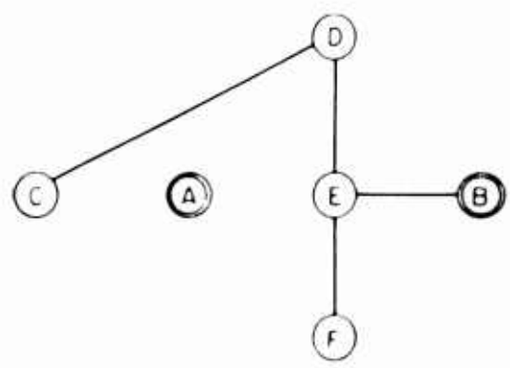


Fig 2a

CD, DE saturated. Select any one of these saturated arcs, say AC, and place some identifying mark on it for future reference. In Fig. 2 we have used a bar (▮) ; this symbol will be used

---

*It is clear that arcs may be removed until a tree is left. There is then a unique chain joining A and B. Elimination of any arc of this chain gives two trees of the kind described.

throughout to designate a subset of the saturated arcs. Now observe that if the barred arc is dropped from the picture, we again have two trees (Fig. 2a) T_A = A and T_B = EB, ED, DC, EF . Again introduce any unbarred arc leading from T_A to T_B, say AF. This creates a flow along the chain AF, FE, EB of 10 units and saturates each arc of this chain. Select one of these, say AF, and bar it. We now have the following diagram (Fig. 3),

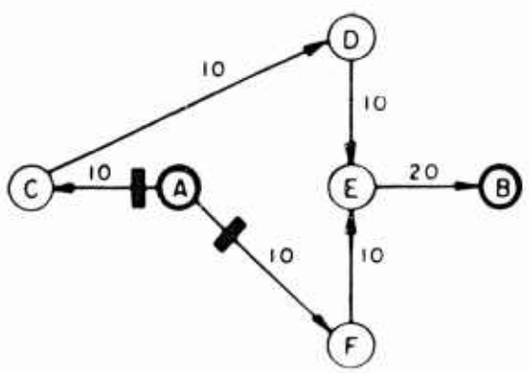


Fig 3

and we have achieved a flow of 20. Dropping barred arcs gives the same trees as in Fig. 2a. Introduce arc AE from T_A to T_B. This leads to a situation we have not met previously in that the chain thus constructed, namely AE, EB, can not take any

more flow because EB, though unbarred, is at its upper limit. Bar EB and leave AE in with a flow of zero, obtaining Fig. 4 with new trees as shown in Fig. 4a.

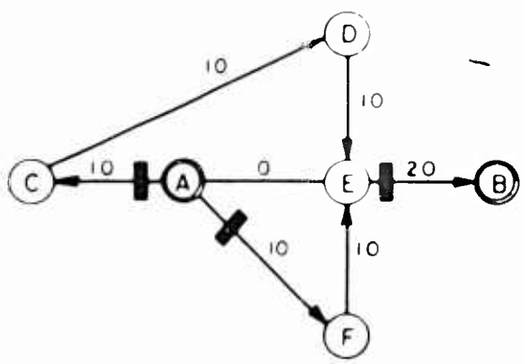


Fig 4

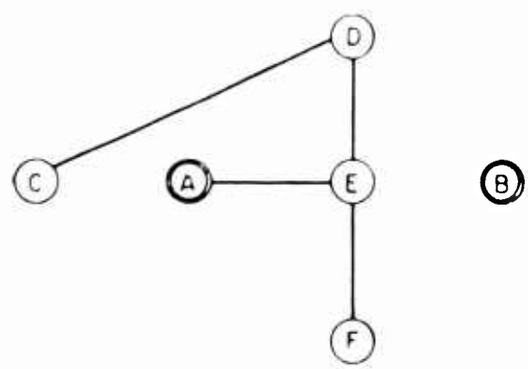


Fig 4a

Introduce DB to get the chain AE, ED, DB. This time we can get an increase even though DE is saturated, since the flow in Fig. 4 is from D to E. Thus if the flow from A to E is increased by $\epsilon \geq 0$, the flow from D to E must be decreased by ϵ, and the flow from D to B increased by ϵ (see Fig. 4b) in order to preserve the conservation equations at E and D.

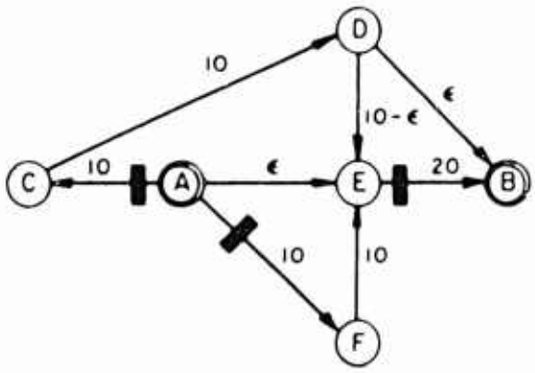


Fig 4b

The largest possible value of ε is 10, since the capacity of DB (and of AE) is 10. This cancels the flow from D to E. Bar DB and proceed.

A repetition of steps of the kind described produces the sequence of flows depicted in Figs. 5 – 8 below.

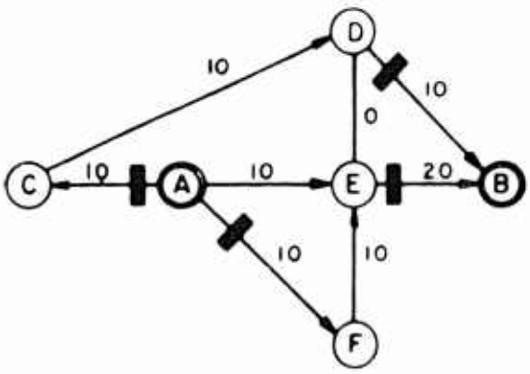


Fig 5

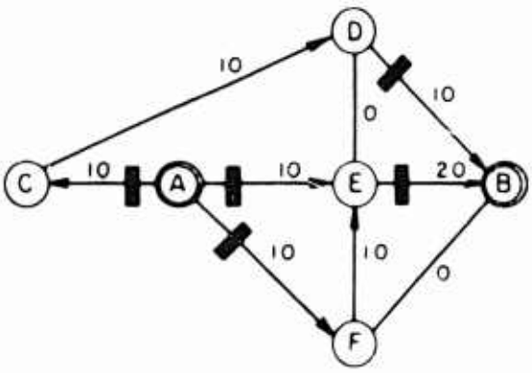


Fig. 6

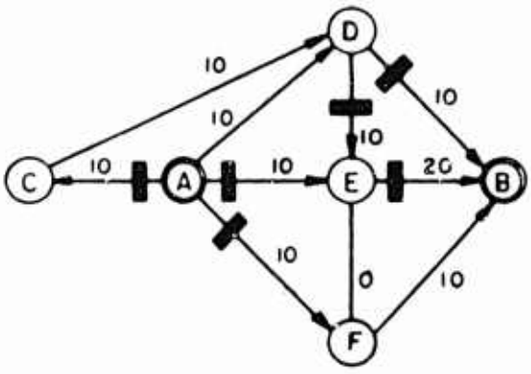


Fig 7

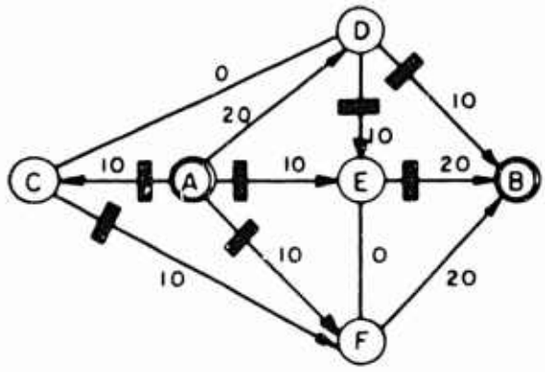


Fig 8

Now in Fig. 8 the trees are $T_A = \{AD, DC\}$, $T_B = \{FB, EF\}$, and there are no more arcs to introduce from T_A to T_B. At this stage examine the barred arcs connecting nodes of T_A to those of T_B. If the flow in each of these is in the "right" direction, that is, from T_A to T_B, an optimum has been reached, because these arcs now constitute, in the original network, a cut separating A and B (i.e., every chain joining A and B contains one of these arcs), and the total flow through the network is equal to the sum of the capacities of the arcs forming the cut. Hence, since it is clear that no flow can exceed the sum of capacities of arcs forming a cut, the flow is maximal in this case. If, on the other hand, the flow in one of the barred arcs which join T_A to T_B is in the wrong direction, an increase in total flow may possibly be obtained by decreasing the flow in this arc. To see this, notice that the arc in question, together with arcs

of T_A and T_B, will form a (unique) chain joining A and B which might look, for example, like this

the arrowheads denoting directions of flow. Now we may subtract $\epsilon \geq 0$ from the flow in all arcs having the same direction as the barred arc, add ϵ to the flow in the other arcs. Since these latter arcs will of necessity include the source (and sink) arc in the chain, the total flow will be increased provided there is any slack in the chain, that is, if none of the arcs where $+\epsilon$ appears is already saturated (though unbarred in our procedure). In any case, locate an arc in the chain which determines the maximum value of ϵ and place a bar on it. This either gives rise to two new trees or, if the position of the bar has not been changed, reverses the flow completely in the original arc.

In the example (see Fig. 8) the arcs DF, DE, AE, AF, CF joining nodes of T_A to those of T_B all have flows in the right direction. Hence the flow shown in Fig. 8 is maximal, and consequently these arcs form a minimal cut, i.e., a cut the sum of whose arc capacities is minimal.

It is clear from the preceding discussion that the proposed computation provides a nondecreasing sequence of flows. All practical experience with the method indicates that a maximal flow is reached very quickly. There is, however, the theoretical possibility that an infinite sequence of flows having the same value may be obtained. In the rest of the paper we shall indicate one way of removing this difficulty. The idea, not a new one, is to perturb the capacities of the arcs slightly so that the decision at each stage as to the placement of the bar becomes unique, thus assuring a finite process.*

II. Proof of Convergence

Given an arbitrary network N, let $a_1,\ldots,a_n$ be a listing of the arcs in some order and let $c_1,\ldots,c_n$ be their capacities.

At each stage of the computation, the arcs a_i, $i=1,\ldots,n$; are divided into four mutually exclusive classes T_A, T_B, S, Z where

---

*From the linear programming point of view the method thus far corresponds to the original simplex algorithm without modifications to account for degeneracy, with this difference. When "ties" occur in the method (i.e., when a bar may be placed in one of several positions at some step), we have not indicated a unique choice, but rather have used language like "pick any one..." If we interpret this to mean "choose at random among the possibilities," then we are at least assured that with probability one the computation terminates, as we shall see.

T_A is a tree branching out from A,

T_B is a tree branching out from B,

S consists of the barred arcs,

Z contains all other arcs,

and each node is an endpoint of some arc of T_A or T_B. Call such a division a <u>basic partition</u> of N.

<u>Theorem</u>: For a given basic partition of N, let arbitrary flows be assigned to arcs of S and Z. Then the conservation equations uniquely determine flows in the arcs of T_A and T_B. Moreover, if $x_1, \ldots, x_m$ are the amounts of the assigned flows, then the amount of flow in each arc of N is equal to $\sum_{j=1}^{m} \delta_j x_j$, where $\delta_j = 1$, 0, or -1.

The proof can be made clear by considering the following example:

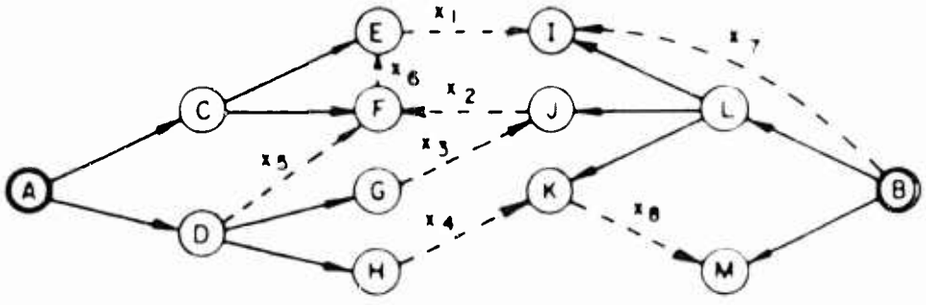


Fig 9

where the dotted flows $x_1, \ldots, x_8$ are assigned.

Orient each arc of the two trees in some fashion, say away from A in T_A and away from B in T_B (as shown in Fig. 9), the convention being that a positive number on an arc represents flow in the same direction as its orientation. Thus if x, y, z are the flows in arcs AC, CE, CF, respectively, the equation at C is $x - y - z = 0$.

That the flows in the arcs of each tree are uniquely determined can be seen by solving the equations recursively in the following manner. Start with an outermost node in one of the trees, E for example. There is then only one unknown flow into E, hence its value can be found; here it is $x_1 - x_6$. Similarly determine the flows in all arcs branching out of F, G, H, and then work backward in the tree.

For the last assertion of the theorem, it is convenient to solve the equations in another way. Select any one of the assigned flows x_1. There are two cases to consider. Either the arc with flow x_1 joins two nodes of the same tree or not. Suppose it joins two nodes of T_A, as EF does, for instance. Since there is a unique chain joining E and F in T_A, namely FC, CE, the flow x_6 in EF can be taken care of by assigning $\pm x_6$ appropriately to the arcs of this chain. Here the assignment is $+ x_6$ to CF, $- x_6$ to CE. On the other hand, if the arc

in question joins T_A to T_B, as x_1 does, consider the chain joining A and B and having this arc as one of its links. Here it is AC, CE, EI, IL, LB, and the assignments are x_1, x_1, x_1, $-x_1$, $-x_1$, respectively. If we continue in this fashion, it is evident that a solution to the system of equations will be built up having the form described in the theorem. For the example, the solution is shown in Fig. 10.

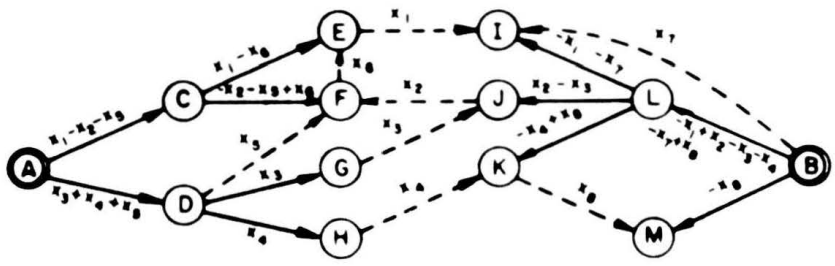


Fig. 10

In the algorithm outlined in section I, arcs of S are saturated and arcs of Z have zero flow. Hence we may state

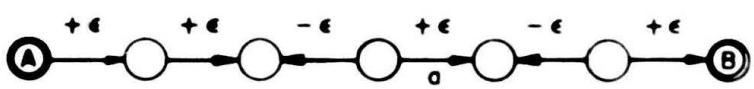
<u>Corollary</u>: Suppose S consists of arcs $a_{i_1}, \ldots, a_{i_s}$ at any given step. Then the flow in each arc of N is given by a sum of the form $\sum_{k=1}^{s} \delta_{i_k} c_{i_k}$, where $\delta_{i_k} = 1, 0$, or -1.

Let us examine in more detail how ties occur in the decision as to which arc to put into S in going from one step to the next in the computation. Each step is of one of the following two types:

(a) a flow from T_A to T_B is introduced along some arc <u>a</u> of Z, thus shifting one of the arcs of T_A or T_B, or <u>a</u> itself, into S;

(b) a flow from T_B to T_A along some arc b of S is decreased, again shifting one of the arcs of T_A or T_B into S, or leaving b in S with its direction of flow reversed.

It suffices to consider (a), since (b) is essentially no different. We have then a chain from A to B



consisting of <u>a</u> and arcs of T_A and T_B, with flows in one direction or the other in each arc of the chain, the direction of flow in the source (sink) arc being away from A (into B). Suppose the arcs in the chain are $a_{j_1}, a_{j_2}, \ldots, a_{j_m} = a, \ldots, a_{j_r}$ and let S consist of $a_{i_1}, \ldots, a_{i_s}$.

By the corollary, the magnitude of flow in a_{j_ℓ} is given by $\sum_{k=1}^{s} \delta_{i_k}^{j_\ell} c_{i_k}$. It follows that we may increase the flow in arc $\underline{a}$ by an amount

$$\epsilon = \min_{j_\ell} \left(c_{j_\ell} \pm \sum_{k=1}^{s} \delta_{i_k}^{j_\ell} c_{i_k} \right)$$

where the plus sign is taken if a_{j_ℓ} has flow directed toward A, the minus sign otherwise. Since the c_{j_ℓ} are integers, it is clear that if, instead of using the original capacities, we had altered them to begin with by setting

$$c_i' = c_i + 10^{-i} , \quad i = 1,\ldots,n ,$$

the minimum would be achieved for just one j_ℓ. Then at no stage do ties occur, and consequently arcs not in S will, at each step, be unsaturated. This means that the total flow through N is increased with each iteration, and hence, by the theorem, no basic partition having the same directions of flow in arcs of S can reoccur. Since there are only a finite number of basic partitions, the procedure terminates for the perturbed problem. Thus we must arrive at a flow in which all arcs of S leading from T_A to T_B form a cut and are in the right direction; hence the flow is maximal. Once an optimum has been attained, a solution to the original problem is given by rounding each arc flow to the nearest integer.

Notice that the theorem (see [1], [2]) which states that the maximal flow value in a network is equal to the minimal cut value is apparent from the algorithm just outlined, since the point at which the computation terminates occurs when some cut in the network is saturated by flows in the right direction.

We also point out, for what it may be worth, that the assertion in the footnote on page 9 can now be verified. To see this, let us define a state of the system to be any flow corresponding to a basic partition, where the arcs of S are saturated and those of Z have zero flow, i.e. a state is any flow that might conceivably be reached in the computation. There are only a finite number of states, say $\sigma_1, \cdots, \sigma_t$. Thus, if we start with any state σ_i and randomize among ties at each step, there is a positive probability p_i of reaching a state satisfying the maximality criterion within t steps, since the possibilities at each step include the one which would be selected in the perturbed problem. Let $p = \min_i p_i > 0$. Then the probability of not terminating is $\lim_{m \to \infty} (1-p)^m = 0$.

For anyone who may be interested in computing maximal flows in networks by this method, we do not recommend altering the capacities as described in this section. All empirical evidence, both in this particular problem and with the simplex algorithm in general (of which this is a special case), indicates that the insurance bought in this way is not worth the effort.

REFERENCES

[1] Dantzig, G. B., and D. R. Fulkerson, On the Min Cut Max Flow Theorem of Networks, The RAND Corporation, Research Memorandum RM-1418, 1 January 1955.

[2] Ford, L. R., and D. R. Fulkerson, Maximal Flow through a Network, The RAND Corporation, Research Memorandum RM-1400, 19 November 1954.