Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Note on Parametric Network Flows

Edward Minieka,

To cite this article:
Edward Minieka, (1973) A Note on Parametric Network Flows. Management Science 19(5):585-587. https://doi.org/10.1287/mnsc.19.5.585

MANAGEMENT SCIENCE
Vol. 19, No. 5, January, 1973
Printed in U.S A.

A NOTE ON PARAMETRIC NETWORK FLOWS*

EDWARD MINIEKA

University of Illinois, Chicago

In their paper [1], Doulliez and Rao present algorithms that solve two flow problems for a single source, multi-terminal network. The first problem that they solve is the construction of a flow that maximizes the value of t, where the demand at each sink is a nondecreasing, linear function of t. Given such a flow, the second problem that they solve is the construction of a flow that maximizes the value of t when the capacity of an arc is reduced. This paper supplies a finiteness proof for the first algorithm and sketches a finiteness proof for the second algorithm. The proofs are based on the well-known fact that a network possesses only a finite number of different spanning trees.

Let N be a finite, directed network with node set X and arc set E. Let $s \in X$ be the unique source node in N, and let $D \subseteq X$ be the set of sink nodes in N. Let h_e denote the capacity of arc e, and let $\beta_i + \alpha_i t$ denote the demand at sink i at time t, where all α_i and β_i are nonnegative and finite.

Time t is said to be feasible if there exists a flow that exactly satisfies all demands at time t.

In [1], an algorithm is presented that determines the maximum feasible time and constructs a flow for this maximum time. This algorithm's finiteness is asserted in a lengthy argument that compares the algorithm to a finite, parametric dual-simplex algorithm which can be suitably modified to avoid cycling. The algorithm does not describe this modification.

The purpose of this paper is to present a slightly modified version of this algorithm and a short, simple proof based on the finiteness property of spanning trees.

A spanning tree is a connected set of $|X| - 1$ arcs that is adjacent to all nodes in X. It is well known that between any two nodes there is a unique path in a spanning tree, i.e. a spanning tree contains no circuits.

LEMMA 1. *Network N contains only a finite number of different spanning trees.*

PROOF. Network N has a finite number of arcs.

Let f_e denote the flow in arc e. Arc e is called *intermediate* if $0 < f_e < h_e$.

LEMMA 2. *For any feasible time t, there exists a feasible flow whose intermediate arcs are contained in a spanning tree.*

PROOF. Given a feasible flow for time t, if the set of intermediate arcs contains no circuits, then it can be contained in a spanning tree. Otherwise, the flow within a circuit composed of intermediate arcs can be adjusted until one intermediate arc is eliminated without disturbing the conservation of flow at any node. This process can be repeated until all circuits of intermediate nodes have been eliminated. Q.E.D.

Given a spanning tree T of network N, there exists a unique path from source s to each sink $i \in D$.

Suppose only the arcs in T can be used to send additional flow from s to D. If arc $a \in T$ is directed away from s, then at most $\delta_a \equiv h_a - f_a \geqq 0$ additional flow units can traverse arc a. If arc $a \in T$ is directed towards s, then at most f_a additional flow units can traverse arc a. For arc $a \in T$, let β_a denote the number of flow units that

* Received February 1971; revised September 1971, May 1972.

must traverse arc a if t is to be increased by one unit, i.e. let β_a equal the sum of the α_i for each $i \in D$ whose path from s to i traverses arc a. Clearly, arc a prohibits a time increase of more than δ_a/β_a.

If arc $a \in T$ is removed from T, then X is partitioned into two sets X_a^1 and X_a^2. Without loss of generality, suppose that $s \in X_a^1$. Let X_a be defined as the set of arcs joining members of X_a^1 and X_a^2. Set X_a, defined by tree T, is called a cocycle.[1] If all arcs in X_a that are directed from X_a^1 to X_a^2 carry a capacity flow, and if all other arcs in X_a carry no flow, then X_a is called a critical cocycle. If X_a is a critical cocycle and $\beta_a > 0$, then from the Max-Flow Min Cut Theorem, we know that no additional flow into the sinks is possible. We can now present:

The Algorithm

Step 1. For any feasible time t, construct a flow whose intermediate arcs are contained in some spanning tree. Call this spanning tree T. Standard network flow techniques can be used. Go to Step 2.

Step 2. For each arc a in tree T, calculate β_a and δ_a. The maximum time increase that arc a can tolerate is δ_a/β_a, if $\beta_a \neq 0$. Let M denote the set of arcs a for which δ_a/β_a is minimum. Denote this minimum possible time increase by Δt. If $\Delta t = 0$, go to Step 3. Otherwise, adjust the flow along each arc $a \in T$ by the amount $\beta_a \Delta t$, and let $t = t + \Delta t$. Clearly, no arc in M remains intermediate. Go to Step 3.

Step 3. Let arc a be any arc in M such that no other arc in M lies in tree T between arc a and source s. Arc a defines the cocycle X_a. If cocycle X_a is critical, stop. (A critical cocycle has been found and the current value of t is maximum.) Otherwise, replace arc a in the tree by any arc in cocycle X_a that allows flow. Return to Step 2.

PROOF. Since the algorithm terminates with a critical cocycle, it terminates optimally. Hence, only the finiteness of the algorithm remains to be proved.

If the algorithm does not terminate finitely, then some spanning tree, say T_1 must be generated infinitely many times, since, by Lemma 1, there are only a finite number of distinct spanning trees.

Suppose spanning tree T_1 generates an infinite number of positive increments of time. After each of these positive increments of time (Step 2), no further increment is possible without generating a different tree. At each of these times, the flow on the arcs not in T_1 must assume a different set of values. There are only a finite number of distinct sets of values for the flow on the arcs not in T_1, since these arcs have either zero or capacity flow. Hence, T_1 can generate only a finite number of positive increments of time.

Hence, if the algorithm does not terminate finitely, T_1 must generate an infinite sequence of time increments Δt equal to zero.

For the initial case in this sequence, let arc $a \in T_1$ be the arc that is removed from tree T_1. Then for tree T_1 to be generated again, arc a must return to the tree.

If arc a is directed *away from* the source s in tree T_1, then $f_a = h_a$. In order for arc a to enter the tree again, arc a must be directed toward the source s with respect to the cocycle at that step. This situation occurs only if at least one arc on the path from source s to node x is removed from the tree. But this removal is impossible, since only members of M can be removed from the tree, and no arc in the path in tree T_1 from source s to node x is a member of M or can become a member of M in subsequent iterations of Step 2. Thus tree T_1 cannot be generated a second time without increasing t, which is a contradiction.

[1] A cocycle is any minimal set of arcs that disconnects the network.

A similar contradiction results when we assume that arc a is directed toward source s. Q.E.D.

The algorithm in [1] does not specify which arcs to choose from set M. This is the only difference between the original algorithm and the algorithm presented here.

Given a flow for the maximum value of t, [1] presents an algorithm that constructs a flow that maximizes t when the capacity of a single arc is reduced. Unfortunately, the finite termination of this algorithm, as presented in [1], depends upon an unspecified method that will avoid cycling in the corresponding parametric dual-simplex algorithm. This algorithm also replaces arcs in the spanning tree until a maximum, feasible flow is attained. If there is a choice of which arc to replace, the algorithm makes the choice arbitrarily.

If M denotes the set of arcs that may be replaced, then by choosing any arc in M such that no other arc in M lies between it and source s in the spanning tree, the algorithm will terminate finitely.

The proof of the finite termination of this modified algorithm is similar to the finite termination proofs of the first algorithm, i.e., it depends upon the finiteness of the number of spanning trees of a network and upon the selection of only certain members of M to enter the spanning tree.

Reference

1. Doulliez, P. J. and Rao, M. R., "Maximal Flow in a Multi-Terminal Network with Any One Arc Subject to Failure," *Management Science*, Vol. 18, No. 1 (September 1971), pp. 48–58.

Published in Management Science on January 01, 1973 as DOI: 10.1287/mnsc.19.5.585.
This article has not been copyedited or formatted. The final version may differ from this version.