Discrete Applied Mathematics 13 (1986) 157–163
North-Holland

MULTI-TERMINAL MAXIMUM FLOWS IN NODE-CAPACITATED NETWORKS*

Frieda GRANOT
Faculty of Commerce, University of British Columbia, Vancouver, B.C., Canada

Refael HASSIN
Department of Statistics, Tel-Aviv University, Ramat-Aviv, Israel

Received July 1984
Revised March 1985

The problem of finding the maximum flows between each pair of nodes in a subset of k nodes of a node- and arc-capacitated undirected network is studied. Gomory and Hu (1961) showed that in the arc-capacitated case, it suffices to solve only $k-1$ of the $k(k-1)/2$ required maximum-flow problems. The node-capcitated case apparently cannot be reduced to the arc-capacitated case, although the converse is possible. The purpose of this paper is to extend Gomory and Hu's result to the general node- and arc-capacitated case.

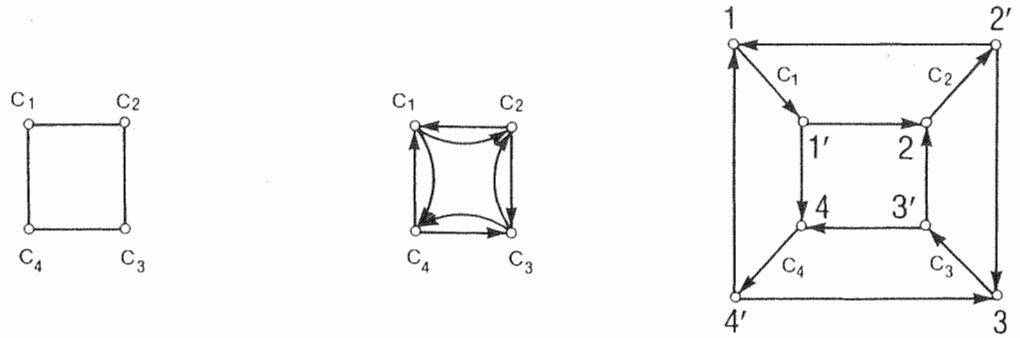
1. Introduction

The problem of finding the maximum flows between each pair of nodes in a subset of k nodes of a node- and arc-capacitated undirected network arises in the study of communication and transportation networks. Gomory and Hu [1] showed that in the arc-capacitated case, it suffices to solve only $k-1$ of the $k(k-1)/2$ required maximum-flow problems. Apparently the node-capcitated case cannot be reduced to the arc-capacitated case. Indeed, the reduction of an undirected node-capacitated network to an arc-capacitated one leads to a network that is not symmetric, i.e., the capacity of an arc joining two nodes depends upon the direction in which the arc is traversed (see Fig. 1). Since Gomory and Hu's results are valid only for the symmetric case, they do not apply to the equivalent directed arc-capacitated network.

By contrast, symmetry is not lost in reducing the arc- to the node-capacitated problem. this reduction entails replacing each capacitated arc by a pair of uncapacitated arcs incident to a node whose capacity equals the capacity of the original arc.

Therefore, we consider only networks with node capacities in the sequel. We emphasize that the above transformation of an arc- to a node-capacitated problem is

*This research was done while the first author was visiting Tel-Aviv University. The research was partially supported by Natural Sciences and Engineering Research Council Canada Grant 67-3998, SSHRC leave fellowship 451-83-0031, and NSF Grant ECS83-12356.



(a) Undirected Graph with Node Capacities

(b) Directed Equivalent with Node Capacities

(c) Directed Equivalent with Arc Capacities

Fig. 1.

used for expository purposes, but not when computing the maximum flow between any given pair of nodes. Moreover, this transformation leaves unchanged the set of pairs of nodes between which one must compute maximum flows. Thus, the computational effort required for the arc-capacitated case is the same as for Gomory and Hu's approach.

2. Minimum cut capacities

Let $G=(N, A)$ be an undirected connected graph with positive capacities on its nodes. For each $i \in N$, denote by c_i the capacity of node i, and set $c_M \equiv \sum_{i \in M} c_i$ for $M \subseteq N$. Now for $i \neq j \in N$, we say that $M \subseteq N$ is an *ij-cut* if either $i \in M$, or $j \in M$, or i and j are in distinct connected components of the subgraph of G induced by $N \setminus M$. Equivalently, an ij-cut is a subset M of the nodes such that each path from i to j contains a node from M. If M is an ij-cut and $c_M \leq c_L$ for every ij-cut L, call M a *minimum ij-cut* and $C_{ij} \equiv c_M$ the *minimum ij-cut capacity*. Of course, $C_{ij}=C_{ji}$. Let $\mathbf{C}_{ij}$ be a minimum ij-cut.

Let $\mathbb{N}^K$ be the network consisting of the complete graph with node set K and capacity C_{ij} associated with each arc (i, j) of the graph.

Theorem 1. *For each cycle in $\mathbb{N}^K$, the minimum of the arc capacities on the cycle is attained by at least two arcs therein.*

Proof. The proof is by induction on the number m of arcs in the cycle. The claim is trivial if $m=2$. If $m=3$ and the claim is false, there is a labeling i, j, k of the nodes in the cycle for which

$$C_{ik} < C_{ij} \leq C_{jk}. \tag{1}$$

Clearly $j \notin \mathbf{C}_{ik}$, since otherwise $C_{ik} \geq c_j \geq C_{jk}$, contradicting (1). Now let I be the connected component of the subgraph induced by $N \setminus \mathbf{C}_{ik}$ that contains i if $i \notin \mathbf{C}_{ik}$,

and let $I=\emptyset$ otherwise. If $j\in I$, then $\mathbf{C}_{ik}$ is a jk-cut, whence $C_{jk}\leq C_{ik}$; while if $j\notin I$, then $\mathbf{C}_{ik}$ is an ij-cut, so $C_{ij}\leq C_{ik}$. Both cases contradict (1) and so establish the result when $m=3$.

Now suppose that the result holds for all integers less than $m>3$, and consider m. Label the nodes of the cycle by $1, 2, \ldots, m$ so that the capacity of $(1, m)$ is minimum. If the theorem is false, then $C_{1m}<C_{i,i+1}$ for $i=1, \ldots, m-1$. By applying the induction hypothesis to the cycles $(1, 3, 4, \ldots, m)$ and $(1, 2, 3)$, we see that $C_{1m}=C_{13}=\text{Min}\{C_{12}, C_{23}\}$, which is a contradiction. □

A *cut-tree* of K is a spanning tree T of $\mathbb{N}^K$ such that for each arc (i, j) in $\mathbb{N}^K\setminus T$, C_{ij} is the minimum of the arc capacities on the unique path in T that connects i and j. The sum of the capacities of the arcs in a tree is called its *weight*.

Theorem 2. *The cut-trees of K are precisely the maximum-weight spanning trees of $\mathbb{N}^K$.*

Proof. It follows from Gomory and Hu [1, pp. 552–53] that a maximum-weight spanning tree T of $\mathbb{N}^K$ is a cut-tree of K. It remains to establish the converse. To that end, let T be a cut-tree of K and use a specialization of Kruskal's [3] algorithm that is described below to find a maximum-weight spanning tree T'. Recall that Kruskal's algorithm constructs a sequence of forests that begins with the empty forest and ends with T' as follows. First list the arcs of $\mathbb{N}^K$ in order of decreasing capacities. Then repeatedly remove the arcs of $\mathbb{N}^K$ in order of decreasing capacities. Then repeatedly remove the arc from the top of the list, add the arc to the current forest if that does not create a cycle, and discard the arc otherwise. The specialization of Kruskal's algorithm is that when two arcs have the same capacity, one lies in T and the other does not, then the arc in T is placed above the other arc in the list. We claim that $T'=T$. For if not, there is a first arc (i, j) that is added to the then current forest that is not in T. Then since T is a tree, there is at least one arc (k, l), say, of the unique path in T that joins i and j and that was not in the then current forest at the time (i, j) was added thereto. For in the contrary event, (i, j) would not have been added because a cycle would then have been formed. However, since (k, l) is in T and T is a cut tree, $C_{ij}\leq C_{kl}$. Thus (k, l) is above (i, j) on the list and so would have been added to the then current forest instead of (i, j), which is a contradiction. □

3. Construction of a cut tree

The goal of this section is to show how to construct a cut-tree by solving minimum-cut problems between at most $|K|-1$ pairs of nodes in K. For a fixed st-cut $\mathbf{C}_{st}$, if s (resp., t) $\notin \mathbf{C}_{st}$, let N_s (resp., N_t) be the nodes of the connected component G_s (resp., G_t) of the subgraph of G induced by $N\setminus \mathbf{C}_{st}$ that contains s

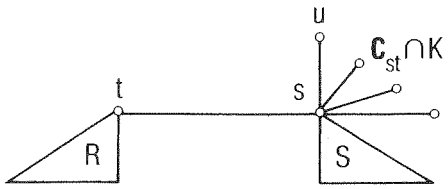


Fig. 2.

(resp., t). If s (resp., $t) \in \mathbf{C}_{st}$, then $\mathbf{C}_{st} = \{s\}$ (resp., $\{t\}$) and N_s (resp., $N_t) = \emptyset$. We suppress the dependence of N_s and N_t on $\mathbf{C}_{st}$ in the sequel for ease of exposition.

Algorithm Cut-Tree

Step 0. Set $A^K = \emptyset$. If $|K| = 1$, stop; otherwise, let s and t be any two distinct nodes in K.

Step 1. Find $\mathbf{C}_{st}$. Add arc (s, t) to A^K with capacity C_{st}.

Step 2. For each $u \in \mathbf{C}_{st} \cap K$ with $u \neq s, t$, add arc (u, s) to A^K with capacity c_u.

Step 3. Let $S = N_s \cap K$, $R = K \setminus ((\mathbf{C}_{st} \setminus \{t\}) \cup S)$ and A^S (resp., A^R) be the set of arcs generated by Algorithm Cut-Tree when applied to S (resp., R). Replace A^K by the set $A^K \cup A^S \cup A^R$.

Observe that execution of Step 3 requires one to find the sets A^R and A^S of arcs. Since, $s \notin R$ and $t \notin S$, R and S each have cardinality less than $|K|$, and thus Algorithm Cut-Tree can be applied inductively to find A^R and A^S (see Fig. 2).

Theorem 3. *Algorithm Cut-Tree generates a cut tree $T^K = (K, A^K)$ of K by solving at most $|K| - 1$ minimum-cut problems.*

The proof requires a few preliminary results.

Lemma 4. *If $u \in \mathbf{C}_{st} \setminus \{s, t\}$, then $C_{us} = C_{ut} = c_u$.*

Proof. It suffices to prove that $C_{ut} = c_u$, since the other assertion follows by interchanging the roles of s and t. The claim is true if $\mathbf{C}_{st} = \{u\}$. For then every ut-cut is an st-cut and $\{u\}$ is a ut-cut, whence $c_u = C_{st} \leq C_{ut} \leq c_u$, so equality holds throughout. Thus assume $|\mathbf{C}_{st}| > 1$. Suppose $C_{ut} < c_u$. Let G' be the subgraph of G obtained by deleting all arcs joining u and some $v \in \mathbf{C}_{st}$. Denote by $\mathbf{C}'_{ut}$ a minimum ut-cut in G'. Then

$$C'_{ut} \leq C_{ut} < c_u. \tag{2}$$

Now consider the st-cut $\mathbf{C}'' \equiv \mathbf{C}'_{ut} \cup (\mathbf{C}_{st} \setminus \{u\})$ in G. But by (2), the capacity of $\mathbf{C}''$ is less than C_{st}, contradicting the minimality of $\mathbf{C}_{st}$. □

Corollary 5. *If $u \neq v \in \mathbf{C}_{st}$, then $C_{uv} = \text{Min}\{c_u, c_v\}$.*

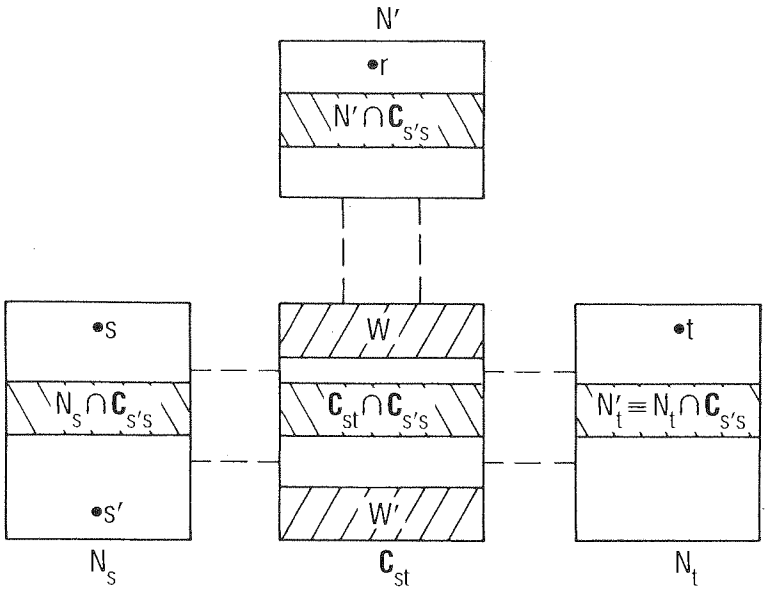


Fig. 3.

Proof. Since $\{u\}$ and $\{v\}$ are uv-cuts, $C_{uv} \leq c_u, c_v$. Thus since $c_u = C_{ut}$ and $c_v = C_{tv}$ by Lemma 4, the result follows by applying Theorem 1 to the cycle (t, u, v). □

Put $N' \equiv N \setminus (N_s \cup N_t \cup \mathbf{C}_{st})$. Fig. 3 illustrates an instance of the proof of the next lemma.

Lemma 6. *If* $s \neq t \in N \setminus \mathbf{C}_{st}$, $s' \in N_s \setminus \{s\}$ *and* $r \in N \setminus (N_s \cup \{t\} \cup \mathbf{C}_{st})$, *then we can choose* $\mathbf{C}_{s's}$ *(resp.,* $\mathbf{C}_{tr}$*) so that* $\mathbf{C}_{s's} \cap N_t$ *(resp.,* $\mathbf{C}_{tr} \cap N_s$*)* $= \emptyset$.

Proof. We begin by proving the result reading without parentheses. Let $\mathbf{C}_{s's}$ be a minimum $s's$-cut and put $N'_t \equiv \mathbf{C}_{s's} \cap N_t$. If $N'_t = \emptyset$, we are done. If not, let W (resp., W') be the subset of $\mathbf{C}_{st}$ rechable from s (resp., s') through nodes not in $\mathbf{C}_{s's}$. Thus since $\mathbf{C}_{st}$ separates N_s from N_t, it follows that $M \equiv (\mathbf{C}_{s's} \setminus N'_t) \cup W$ and $M' \equiv (\mathbf{C}_{s's} \setminus N'_t) \cup W'$ are $s's$-cuts that do not meet N_t. It remains to show that one of them is a minimum $s's$-cut. To that end, we show first that either $Q \equiv (\mathbf{C}_{st} \setminus W) \cup N'_t$ or $Q' \equiv (\mathbf{C}_{st} \setminus W') \cup N'_t$ is an st-cut. For if neither set is an st-cut, we show that there exists a path $\mathbf{P}$ from s to s' that avoids $\mathbf{C}_{s's}$, which contradicts the fact that $\mathbf{C}_{s's}$ is an $s's$-cut. With that aim, since Q (resp., Q') is not an st-cut, there exists a path P (resp., P') from s to t that avoids Q (resp., Q'). However, since every path from s to t passes through $\mathbf{C}_{st}$, P (resp., P') passes through W (resp., W'). Let w (resp., w') be the last node of P (resp., P') in W (resp., W') on the way to t. Then since P (resp., P') avoids Q (resp., Q') and $\mathbf{C}_{st}$ is an st-cut, all nodes of the subpath $\ddot{P}$ (resp., $\ddot{P}'$) of P (resp.,P') from w (resp., w') to t except w (resp., w') lie in $N_t \setminus N'_t$. Now by definition of W (resp., W'), there is a path $\hat{P}$ (resp., $\hat{P}'$) from s (resp., s') to w (resp., w') that avoids $\mathbf{C}_{s's}$. Thus $\hat{P} \cup \ddot{P} \cup \ddot{P}' \cup \hat{P}'$ is the desired path $\mathbf{P}$. Now if Q (resp., Q') is an st-cut, then $c_W \leq c_{N'_t}$ (resp., $c_{W'} \leq c_{N'_t}$), so

$c_W = c_{N_t'}$ (resp., $c_{W'} = c_{N_t'}$). Hence, M (resp., M') is a minimum $s's$-cut. On replacing $\mathbf{C}_{s's}$ by this minimum cut, the proof reading without parentheses is complete. The result reading with parentheses is proved in a similar manner. □

Proof of Theorem 3. The proof is by induction on the cardinality of $K \subseteq N$. The claim is trivial for $|K| = 1,2$. Assume the result is true for all subsets K with cardinality less than $k > 2$, and consider a subset K with $|K| = k$. The proof has two parts. The first part shows that T^K is a tree spanning K and is obtained by solving at most $k-1$ minimum-cut problems. The second part shows that T^K is a cut-tree.

For the first part, recall that $|R|, |S| < k$. Hence, by the induction hypothesis, Algorithm Cut-Tree terminates with trees T^R and T^S that span R and S respectively. Now $T' \equiv (K, A^K)$ at the end of Step 2 is a tree that spans $(\mathbf{C}_{st} \cup \{s, t\}) \cap K$. Moreover, since R (resp., S) $\neq \emptyset$, R (resp., S) shares only node t (resp., s) with T'. Thus the union of the trees T^R, T^S and T' forms the tree T^K that spans K. Now when $|S| \geq 1$, then by the induction hypothesis, at most $|S| - 1 + |R| - 1 \leq k-2$ minimum-cut problems are solved in finding T^R and T^S. If, however, $|S| = \emptyset$, then $|R| = k-1$, so again by the induction hypothesis, at most $k-2$ minimum-cut problems are solved in finding T^R and T^S. Now since $\mathbf{C}_{st}$ is the only other minimum cut generated by Algorithm Cut-Tree, at most $k-1$ minimum-cut problems are solved in finding T^K in both cases.

For the second part, we claim that for each $(i, j) \in (K \times K) \setminus A^K$ with $i \neq j$, C_{ij} is the smallest of the capacities of the arcs on the unique path in T^K that joins i and j. This claim is certainly so for each $(i, j) \in A^K$ since the capacity of (i, j) is C_{ij} by construction and Lemma 4. The claim is also true for each $(i, j) \in A^R \cup A^S$ by the induction hypothesis. It remains to establish the claim where $(i, j) \notin A^K$ is one of the following seven possibilities: (u, t), (u, v), (u, s'), (u, r), (s', t), (s, r), (s', r) where $u, v \in \mathbf{C}_{st} \setminus \{s, t\}$, $s' \in S \setminus \{s\}$, and $r \in R \setminus \{t\}$.

(u, t): Since $C_{ut} = c_u = C_{us}$ by Lemma 4, $C_{ut} = \text{Min}\{C_{us}, C_{st}\}$ by Theorem 1.

(u, v): By Corollary 5 and Lemma 4, $C_{uv} = \text{Min}\{c_u, c_v\} = \text{Min}\{C_{us}, C_{sv}\}$.

(u, s') and (u, r): Since $C_{us'}, C_{ur} \leq c_u = C_{us} = C_{ut}$ by Lemma 4, we have that $C_{us'} = \text{Min}\{C_{us}, C_{ss'}\}$ and $C_{ur} = \text{Min}\{C_{ut}, C_{tr}\}$ by Theorem 1.

(s', t) and (s, r): Since $\mathbf{C}_{st}$ is both an $s't$- and sr-cut, $C_{s't}, C_{sr} \leq C_{st}$. Thus, $C_{s't} = \text{Min}\{C_{s's}, C_{st}\}$ and $C_{sr} = \text{Min}\{C_{st}, C_{tr}\}$ by Theorem 1.

(s', r): By Lemma 6, there is a $\mathbf{C}_{rt}$ for which $N_s \cap \mathbf{C}_{rt} = \emptyset$, so N_s can be condensed into a single node that does not belong to $\mathbf{C}_{rt}$. Thus $\mathbf{C}_{rt}$ is either an rs- and an rs'-cut, or an st- and an $s't$-cut. In the former event, $C_{rs'} \leq C_{rt}$, and in the latter, $C_{st} \leq C_{rt}$. Also, since $\mathbf{C}_{st}$ is an rs'-cut, $C_{rs'} \leq C_{st}$. Thus, by Theorem 1, $C_{rs'} = \text{Min}\{C_{rt}, C_{st}, C_{s's}\}$. □

In general not all $|K| - 1$ arcs in T^K have distinct capacities. An extreme case is the star graph $G = (N, A)$ with $N = \{1, \ldots, n\}$, $A = \{(1, i)\}$ $i = 2, \ldots, n$. If $c_1 < c_i$, $i = 2, \ldots, n$ and $K = N$, then all $|K|(|K| - 1)/2$ cuts are equal to $\{1\}$.

4. Computations

Notice that Algorithm Cut-Tree requires one to solve up to $|K| - 1$ maximum-flow problems. For the arc-capacitated case, the number of computations required to find the maximum flow between a given pair of nodes is $O(|N|^3)$. (See, e.g., [2], [4].) The node- and arc-capacitated problem can be solved in one of two ways. One is by transforming the problem to an arc-capacitated problem as in Fig. 1 and using the algorithm for the arc-capacitated case with appropriate data structures to take account of the special structure of the network. The other is by using a straightforward generalization of the arc-capacitated maximum-flow algorithm [4] to allow node capacities. With both methods the running time of the maximum-flow algorithm remains $O(|N|^3)$.

It is also important to notice that when applying Algorithm Cut-Tree, the maximum-flow problems solved involve successively smaller networks as in Gomory and Hu [1]. Indeed, a straight forward extension of the proof of Lemma 6 shows that for each pair $i \neq j$ of nodes in R (resp., S), there is a minimum ij-cut that does not meet N_s (resp., $N_t \cup N'$). Thus, in computing maximum flows between pairs of nodes in R (resp., S), one may condense each component of N_s (resp., $N_t \cup N'$) into a single node with infinite capacity.

Acknowledgment

The authors gratefully acknowledge extensive helpful and constructive suggestions made by Professor A.F. Veinott Jr. which contributed greatly to the formulation, clarity and completeness of the ideas and proofs in this paper.

References

[1] R.E. Gomory and T.C. Hu, Multi-terminal network flows, SIAM J. Appl. Math. 9 (1961) 551–570.

[2] A.V. Karzanov, Determining the maximal flow in a network by the method of preflows, Soviet Math. Dokl. 15 (1974) 434–437.

[3] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc. 7 (1956) 48–50.

[4] V.M. Malhotra, M. Pramodh Kumar and S.N. Maheshwari, An $O(|V|^3)$ algorithm for finding maximum flows in networks, Inform. Process. Lett. 7 (1978) 277–278.