NETWORK FLOWS AND COMBINATORIAL OPTIMIZATION

R. Chandrasekaran
UT Dallas

February 4, 2002

Chapter 1

Multiterminal Flows

1.1 Single Commodity

This section deals primarily with undirected networks. The three major problems dealt with in this area are:

(i) *Realizability Problem*: Find necessary and sufficient conditions for a given set of numbers to represent maximal flow values between all pairs of nodes in some network.

(ii) *Analysis Problem*: Find the max flows between all pairs of nodes in an undirected network in a more efficient manner than doing them one by one.

(iii) *Design or Synthesis Problem:* Find the arc capacities for a network which meets specified lower bounds on all the maximum flows with the least sum of capacities (or more generally, the total cost). Such a problem arises in the context of designing a communication network.

1.2 Realizability

We begin with the analysis of another well known problem known in the literature as *Maximum Spanning Tree Problem*. This is useful in deriving necessary and sufficient conditions for realizability.

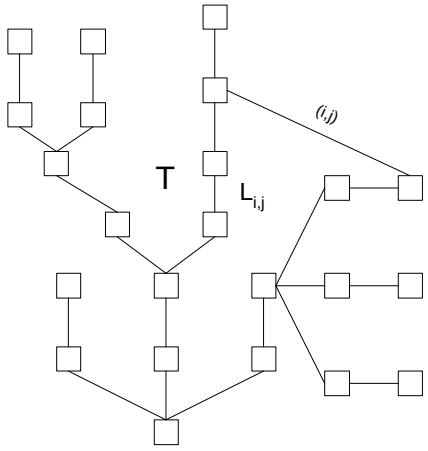
1.2.1 Spanning Tree Problem

Given an undirected network $G = [N; E]$ and arc numbers $r_{i,j}$, find a spanning tree T (if the network is connected and what is called a *principal forest* if it is not) with maximum value for $\sum_{(i,j)\in T} r_{i,j}$.

Theorem 1 *A necessary and sufficient condition for a spanning tree T^* to be optimal for the above problem is that the following condition holds:*

$$(i,j) \notin T^* \Longrightarrow r_{i,j} \leq \min_{(k,l)\in L^*_{i,j}} r_{k,l} \tag{1.1}$$

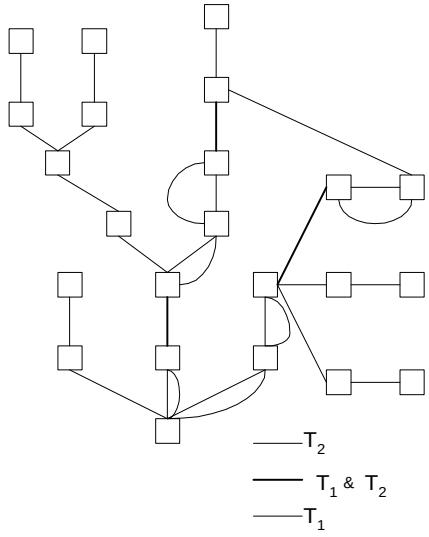
where $L^*_{i,j}$ is the loop formed by adding arc $(i,j) \notin T^*$ to the tree T^*.



Proof: That this condition is necessary is clear from the fact that otherwise an exchange of arc (i,j) with the arc violating the above condition would improve the solution. In order to show sufficiency, we will show that any two trees, say T_1 and T_2, that satisfy the above condition have equal value. Actually, much more is shown: the set of arc values in two such trees is the same.

If trees T_1 and T_2 are distinct, then we can partition the arcs in $T_1 \cup T_2$ as T_1- arcs, T_2- arcs and $T_1 \cap T_2$- arcs. Let (x_1, x_k) be a T_1-arc. This forms a loop with tree T_2, and let the remaining arcs of this loop be (x_1, x_2), (x_2, x_3) ... (x_{k-1}, x_k). We claim that of these there is an arc that is a T_2-arc with value equal to that of (x_1, x_k). Exchanging these arcs produces a tree that has one more arc in common with tree T_2 and has the same value as the original trees.

Repeating this process will prove the theorem.



To prove the claim, suppose it is not true. Since T_1 is a tree, not all of the arcs in the loop can be $T_1 \cap T_2$- arcs; also condition of the theorem holds. If none of the T_2- arcs has a value equal to that of (x_1, x_k), then each of these T_2-arcs makes a loop with tree T_1, and hence, it can be verified that T_1 has a loop which leads to contradiction. Please note that each of the T_2- arcs that are in the loop formed by (x_1, x_k) with T_2 has a value larger than that of (x_1, x_k), and hence, each of the arcs that are in the loops formed by these arcs with tree T_1 also satisfy this condition, and hence, they are distinct from (x_1, x_k). Thus, the claim and, hence, the theorem is proved.□

A direct implementation of the ideas in the theorem leads to a strongly polynomial algorithm. However, more efficient algorithms exist. Using this theorem, we can show that the well known algorithms of **Kruskal** (**Boruvka**) as well as that of **Prim** do produce optimal solutions. We will take up these algorithms in the section on *matroids*. Now we turn to realizability.

Let $G = [N, E]$ be an undirected graph. Let $v_{i,j} = v_{j,i}$ be the maximal flow in G (with $u_{i,j}$ as arc capacities) if (i, j) is the origin — destination combination. Since the network is undirected, v is symmetric. For the sake of completeness, let $v_{i,i} = \infty$ for all $i \in N$. The problem of realizability is to find the conditions necessary and sufficient for a symmetric function v to be the maximum flows in some undirected network, and this is given by the next theorem.

Theorem 2 *A necessary and sufficient condition for a nonnegative symmetric function v to be realizable is that it satisfy:*

$$v_{i,k} \geq \min[v_{i,j}, v_{j,k}] \; \forall \; i, j, k \in N. \tag{1.2}$$

Further, if v is realizable, then it is realizable by a tree network.

Proof:

Necessity: Let $[X, \bar{X}]$ be a minimal cut separating i and k (with $i \in X$ and $k \in \bar{X}$). Hence, $v_{i,k} = u(X, \bar{X})$ and j is either in X or in $\bar{X}$. If we have the first case, then $v_{j,k} \leq u(X, \bar{X})$ since this cut separates j and k, and in the second case, $v_{i,j} \leq u(X, \bar{X})$ since this cut separates i and j in this case. In either case, our result holds. Using this relation recursively, we have:

$$v_{i,p} \geq \min[v_{i,j}, v_{j,k}, v_{k,l}, ..., v_{*,p}] \tag{1.3}$$

Sufficiency: This is proved by exhibiting a network that achieves this flow. Using the given $v_{i,j}$ as arc lengths in G, find a maximal spanning tree. Now consider the tree with $v_{i,j}$ interpreted as arc capacities. This, we claim, is the required network. To show that this claim is true, the fact that this is the maximal spanning tree implies the relation:

$$v_{i,p} \leq \min[v_{i,j}, v_{j,k}, ..., v_{*,p}] \tag{1.4}$$

where (i, p) is an arc outside the tree and the rest are in the loop formed by it with the tree. But we are given that the reverse inequality holds by assumption on v. Hence equality holds and this proves sufficiency as well as the remaining part of the theorem.□

The above proof shows that the *triple inequality* above is actually true for directed networks as well; of course, symmetry does not hold for the directed case. We have proved an additional result that in an undirected network, there are no more than $n - 1$ distinct values for maximal flows. We now move on to finding these values for a given network and this is the problem of analysis.

1.3 Analysis

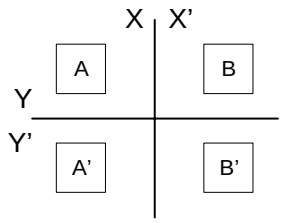
Two n — node networks are defined to be *flow equivalent* if they have the same flow function v. We have just shown that every network is flow equivalent to a tree. The analysis problem deals with finding these flows and the equivalent tree. Indeed, the tree found not only identifies the flows but the minimum cuts that are found in flow labeling as well. And the entire process involves only $n-1$ max-flow problems, often on smaller networks. This result is due to **T.C. Hu and R. Gomory** [*see also the works of* **Mayeda, Chien, Tang, Resh, Wing**, *and the book by* **Frank and Frisch** *and the recent work by* **Gusfield**]. This process does not find the actual arc flows but only the maximum flow values.

The algorithm involves an operation that produces a *condensed network*. This operation can be viewed as *contraction of certain edges* of the network which, in turn, is equivalent to making the corresponding capacities equal to

∞. When we condense a group B, of nodes, this is equivalent to contracting edges of the form (i, j) for all $i, j \in B$. It is more convenient for this operation to think of the original network as having an arc for each pair (i, j) for $i, j \in N$ (with possibly some having zero capacities). Then, all the edges between nodes in B are contracted to produce the condensed network. Clearly, condensed networks are often smaller and, hence, involve less work in computing maximal flows. The capacities of the arc (N_i, N_j) of the condensed network (remember that a node of the condensed network may actually represent many nodes of the original network) is given by $\bar{u}_{i,j} = \sum_{k \in N_i} \sum_{l \in N_j} u_{k,l}$. It should be clear that this operation can not decrease the maximum flow between any pair of nodes. The use of this operation will be clear from the following lemma.

Lemma 3 *Let $(X, \bar{X})$ be a minimal cut separating s and t in G. Let p and $q \in X$. Then, $\exists$ a minimal cut $(S, \bar{S})$ separating p and q in the original network with either $\bar{X} \subset S$ or $\bar{X} \subset \bar{S}$. Thus, condensing the nodes in $\bar{X}$ does not change the value of minimal cuts separating p and q and, hence, the maximal flow between them.*

Proof:Let $(Y, \bar{Y})$ be a minimal cut separating p and q. Since p and q are both in X, and one is in Y and other is in $\bar{Y}$, we may assume, without loss, that $p \in X \cap Y$ and $q \in X \cap \bar{Y}$. Since s is in X, without loss, we may also assume that $s \in X \cap Y$. Now there are two possibilities for t :(i)$t \in \bar{X} \cap Y$ and (ii)$t \in \bar{X} \cap \bar{Y}$. For the sake of convenience, we shall name these sets as follows: $A = X \cap Y$; $B = \bar{X} \cap Y$; $\bar{A} = X \cap \bar{Y}$; and $\bar{B} = \bar{X} \cap \bar{Y}$. We will only consider case (i) since other is similar.



(i)$t \in B$: Since $(X, \bar{X})$ is a minimal cut separating s and t and $(A \cup \bar{A} \cup \bar{B}, B)$ is another cut separating s and t, we have

$$\begin{aligned} u(X, X) &= u(A, B) + u(\bar{A}, B) + u(A, \bar{B}) + u(\bar{A}, \bar{B}) \\ &\leq u(A \cup \bar{A} \cup \bar{B}, B) \\ &= u(A, B) + u(\bar{A}, B) + u(\bar{B}, B) \end{aligned} \tag{1.5}$$

and this implies

$$u(A, \bar{B}) + u(\bar{A}, \bar{B}) - u(\bar{B}, B) \leq 0 \tag{1.6}$$

Similarly, $(Y, \bar{Y})$ is a minimal cut separating p and q and $(A \cup B \cup \bar{B}, \bar{A})$ is another cut separating p and q, we have

$$\begin{aligned} u(Y,Y) &= u(A,\bar{A}) + u(A,\bar{B}) + u(B,\bar{A}) + u(B,\bar{B}) \\ &\leq= u(A \cup B \cup \bar{B}, \bar{A}) \\ &= u(A,\bar{A}) + u(B,\bar{A}) + u(\bar{B},\bar{A}) \end{aligned} \tag{1.7}$$

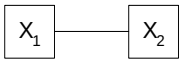
and this implies

$$u(A,\bar{B}) + u(B,\bar{B}) - u(\bar{A},\bar{B}) \leq 0 \tag{1.8}$$

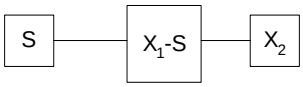
Combining (1.6) and (1.8), we get $u(A, \bar{B}) \leq 0$ which , in turn, implies that it is equal to zero. Using this once again in (I) and (II), we get $u(B, \bar{B}) = u(\bar{A}, \bar{B})$.

Using this in turn, we get $u(Y, \bar{Y}) = u(A \cup B \cup \bar{B}, \bar{A})$, and this proves the lemma. *The procedure used in this lemma is called the* uncrossing procedure *and is used in other contexts later.*□

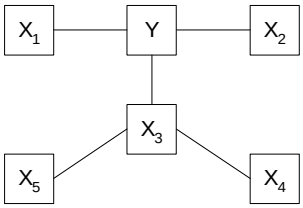
This lemma gives us the basis for the algorithm of **Hu and Gomory** to solve this problem. First, we select two nodes, say s and t, arbitrarily and solve a maximal flow problem with this pair as origin and destination. Let the minimal cut separating s and t be denoted by $(X, \bar{X})$ with $u(X, \bar{X}) = v_1$. We represent the result of this step by the diagram:

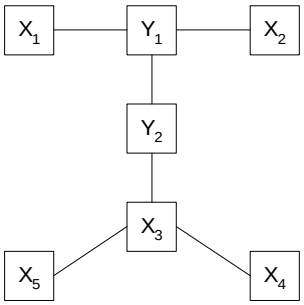


Now choose two nodes, say p and q, both of which are in one of these sets as the next origin — destination pair. If both are in X, then condense $\bar{X}$ (else condense X) and solve the maximal flow problem in the condensed network between p and q. This gives us a minimal cut $(S, \bar{S})$ separating p and q with $\bar{X}$ being entirely on one side, say $\bar{X} \subset \bar{S}$, and let the value of this cut $u(S, \bar{S}) = v_2$. This gives us the second stage of the final diagram for the cut tree as follows:



Of course, the diagram would have been different if $\bar{X} \subset S$ (in this case, the roles of S and $\bar{S}$ would be reversed). At the k^{th} step, the diagram will, in general, be a tree as shown below:





Let i and j be two nodes in Y. [If no such set exists, we are done since every node of the tree is a node of the original network.] In order to solve the maximal flow between i and j in the original network, we can condense the sets X_1, X_2, and $\{X_3 \cup X_4 \cup X_5\}$ each into a single node in the new condensed network. Suppose in this condensed system the minimal cut separating i and j is $(W, \bar{W})$ and $X_1 \cup X_2 \subset W$ and $\bar{W}$ is $N \backslash W$. Then the new tree is given by:

where $v_6 = u(W, \bar{W})$. It should be clear that this process will be repeated exactly $n-1$ times, at the end of which each node of the tree will be a singleton. We will show that the resultant tree, T, is flow equivalent to the original network.

Lemma 4 *Maximal flow $v_{i,j}$ between i and j in the original network is given by:*

$$v_{i,j} = \min[v_{i,k}, v_{k,l}, ..., v_{*,j}]$$

where arcs (i,k), (k,l), ... $(*,j)$ form the path in the tree T between i and j (or in other words are in $L_{i,j}$, the loop formed by (i,j) with T).

Proof: Since each arc in the tree represents a cut separating the same set of nodes in the tree as in the original network, it follows that LHS $\leq$ RHS in the above equation. To show the reverse inequality, we show, by induction, that at any stage of the construction, for any two sets connected by an arc in the tree, $\exists$ at least one node in each set such that the maximum flow in the original network between them is given by the value on the arc in the tree. [This assertion, together with the triangle inequality of the type mentioned before, is sufficient to prove the theorem.] Clearly, the first step of the inductive process is true by construction. Suppose at some stage there are two sets X and Y in the tree connected by an arc with value v. By induction hypothesis $\exists$ nodes $i \in X$ and $j \in Y$ such that $v_{i,j} = v$. Let Y be split at the next stage into Y_1 and Y_2, and let X be connected to Y_1 in the new tree as shown in the diagram.

Let the above split be caused by a maximum flow problem between s and t in Y with $s \in Y_1$ and $t \in Y_2$, and let the capacity of this new cut be v'. Clearly for the newly created arc, we have nothing to prove. It is for the old one we have to reestablish the result. If the arc value for the new arc is v', and the value for the old arc is v, we have to prove that $\exists$ nodes $i' \in X$ and $j' \in Y$ with $v_{i',j'} = v$. We have two cases to consider.

Case (i) $j \in Y_1$: Then $v_{i,j} = v$ and $i \in X$ and $j \in Y_1$ by assumption, and we are done.

Case (ii)$j \in Y_2$: In this case, i and s are one side of an (s,t)-mincut and j and t are on the other side, and this cut has a capacity v'. Hence from lemma, when finding maximum flow $v_{i,s}$, we can condense all the nodes of Y_2 into a single node. Denoting maximum flows in the original network by the letters without and those in the condensed network (Y_2 condensed) by letters with bars, we have:

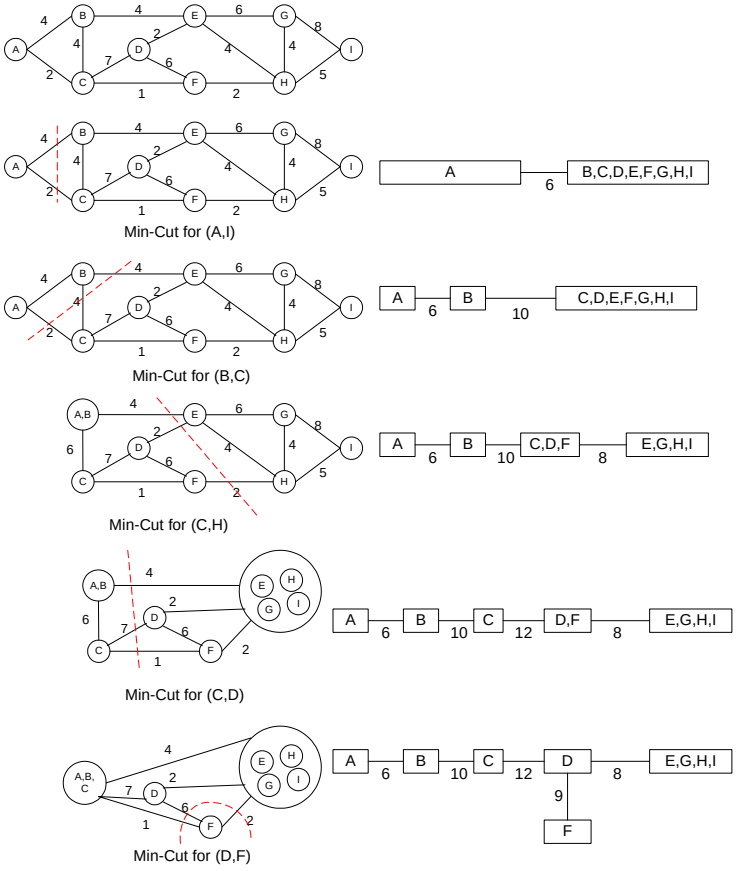
$$\bar{v}_{i,s} = v_{i,s}; \bar{v}_{i,j} \geq v_{i,j} = v; \bar{v}_{j,t} = \infty; \bar{v}_{t,s} \geq v_{t,s} = v'.$$

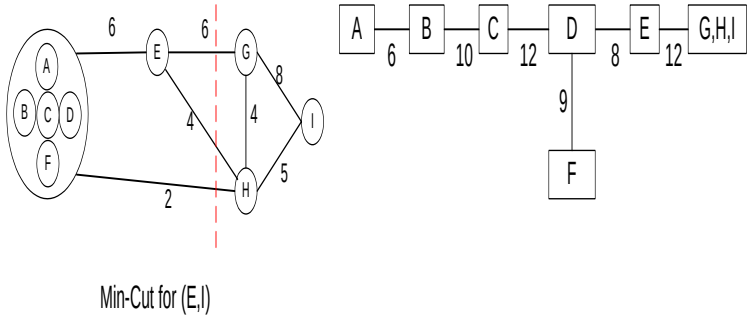
Using the inequality of the lemma, we have:

$$\bar{v}_{i,s} \geq \min[\bar{v}_{i,j},\ \bar{v}_{j,t},\ \bar{v}_{t,s}] = \min[\bar{v}_{i,j},\ \bar{v}_{t,s}]$$

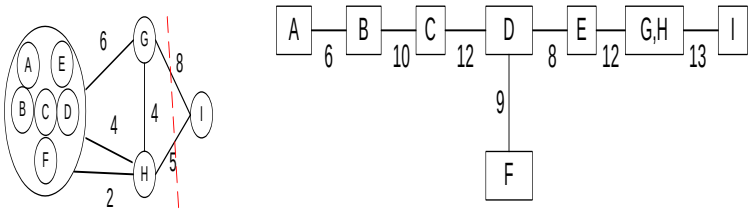
Hence, $v_{i,s} = \bar{v}_{i,s} \geq \min[v, v']$. $\exists$ an (s,t) min *cut* with a value v' that separates i and j and, hence, $v' \geq v_{i,j} = v$. Therefore, $v_{i,s} \geq v$. But there is a cut that separates i and s whose value is v and hence $v_{i,s} = v$ which is the desired result.□

1.4 Example:

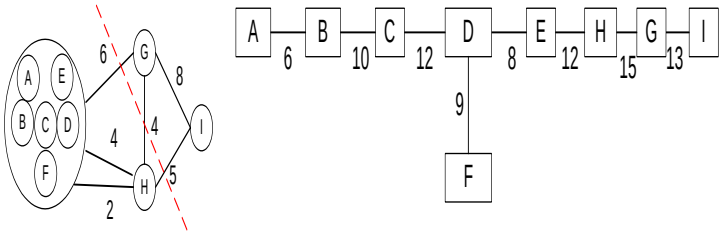

A
B
C
D
E
F
G
H
I
A
6
B,C,D,E,F,G,H,I
Min-Cut for (A,I)
A
6
B
10
C,D,E,F,G,H,I
Min-Cut for (B,C)
A,B
A
6
B
10
C,D,F
8
E,G,H,I
Min-Cut for (C,H)
A,B
E
H
G
I
A
6
B
10
C
12
D,F
8
E,G,H,I
Min-Cut for (C,D)
A,B,
C
A
6
B
10
C
12
D
8
E,G,H,I
9
F
Min-Cut for (D,F)



Min-Cut for (E,I)



Min-Cut for (G,I)



Min-Cut for (G,H)

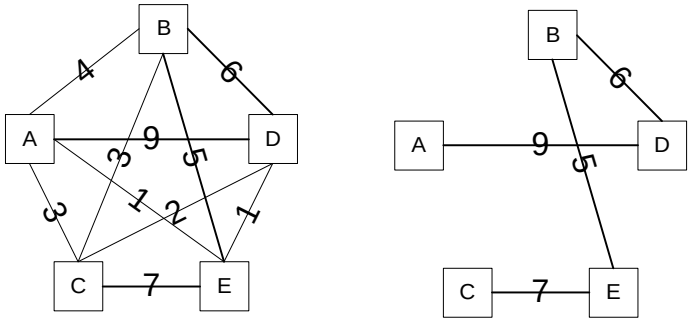
1.5 Design (Synthesis)

Given a symmetric function r (*flow requirements*) defined for all pairs of nodes of an n-node network, find capacities $u_{i,j}$ such that maximum flow on this network with u as capacities between i and j is at least $r_{i,j}$ for each pair (i,j), and among all such solutions, find the one which minimizes, $\sum_{(i,j)\in K_n} c_{i,j}u_{i,j}$. $c_{ij}(=c_{j,i})$ may be thought of as cost of installing a unit capacity on arc (i,j). This can be written as a linear program:

$$\min \sum_{(i,j)\in K_n} c_{i,j}u_{i,j}$$
$$\sum_{i\in X;j\in \bar{X}} u_{i,j} \geq \max_{\substack{i\in X\\ j\in \bar{X}}} r_{i,j} \; \forall \; X \subset N$$
$$u_{i,j} \geq 0 \; \forall \;\; i,j.$$

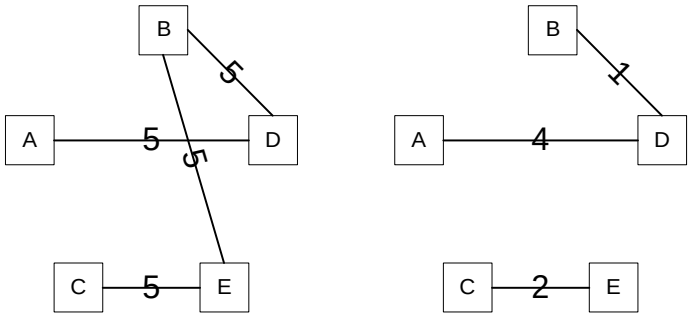
The above formulation has an exponential number of constraints. However, this problem is polynomially solvable by ellipsoid method. For the special case when all $c_{i,j}$ are equal, we have a simple solution procedure due to **T.C. Hu and R.E. Gomory** [see also **Chien, Mayeda, et al**]. *This procedure yields solutions that are multiples of half integers if the data are integers.* The integer programming version of the problem with general costs is NP-hard. As of now, we do not know how to produce the optimal integer solutions even for $c = e$. We are in the process of devising an algorithm for this problem (*one has been recently found*). We now describe this process of producing the continuous solution due to **T.C. Hu and R.E. Gomory.**

Consider a complete undirected graph on n nodes with arc numbers equal to $r_{i,j}$. A maximal spanning tree of this graph is called a *dominant requirement tree.* Thus, if these requirements are met, then all others will be automatically satisfied, and this follows from the lemma in realizability part of this section. This is the first step of the procedure. The original data and the dominant tree are illustrated in the following figure:



The second step is to decompose this tree into what are called *uniform trees;* these have the same r value for all arcs. For example, the minimum value of r on the tree will be the first one that will be *"peeled off"* which will yield some subtrees, and the process is repeated on these. An example will be the best way to illustrate this procedure.

The first step in the decomposition process is:



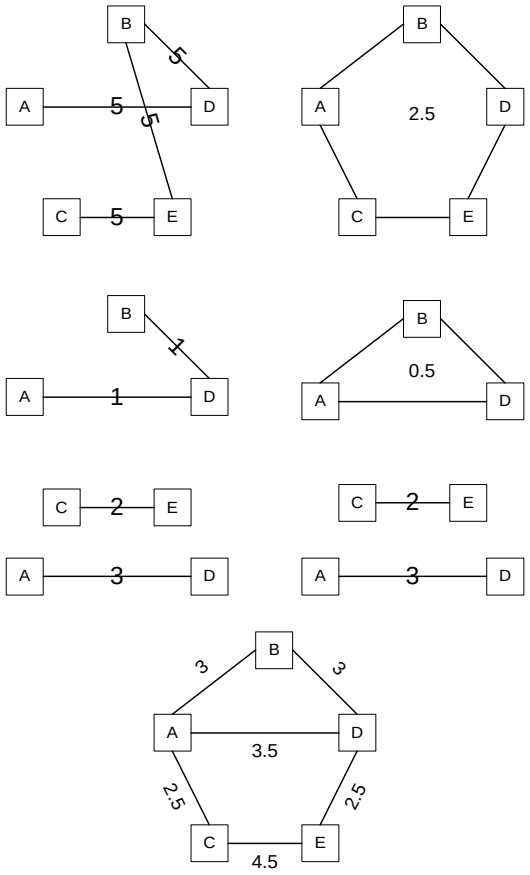
The second of these is a set of trees one of which is uniform; hence, it is further decomposed as follows:

The uniform trees are indicated by letters (a), (b), (c), and (d). Now each of the uniform trees' requirements are satisfied individually, and then all these networks are superimposed (or equivalently the arc capacities are added up). We show the individual networks and the superimposed final result.

For uniform tree (a), any cycle on the nodes with capacity of each arc equal to 2.5 will do; for example, the following network is one such:

For the uniform tree (b), it itself is the desired network since there are only two nodes; this remark also applies to (d). For the tree (c), any cycle on the nodes $1, 2$, and 3 with arc capacity equal to 1 will do; for example, the following is one such:

Superimposing all these, we get the network as shown below:



It is claimed that this is the required solution to the design problem. As can be easily seen, the solution has multiples of 1/2 if the $r_{i,j}$ are integral. That the solution produced satisfies the requirements in the dominant tree and, hence, all the requirements is easy to see. The hard part is to show that it is optimal. For this purpose, we derive a lower bound on the value of the objective function and show that this solution achieves this bound. Towards this end, define $\pi_i = \max_j r_{i,j}$ for all $i \in N$; and $\pi'_i = \max_{j:(i,j)\in T} r_{i,j}$. Clearly, $\pi'_i \leq \pi_i$ (actually they are equal, but we do not need this now). For any feasible solution $u_{i,j}$, we must have the relation:

$$u(i, N) \geq \pi_i \; \forall \; i \in N$$

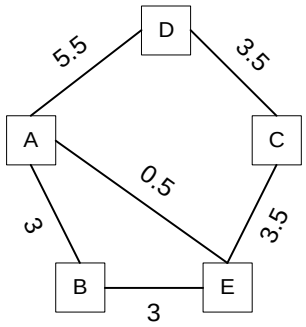
Adding these together, we get $u(N, N) \geq \Sigma\pi_i$. The first of these terms is twice the sum of the $u_{i,j}$. For the special graph, we have equality, and hence,

it is optimal. If we let $r'_{i,j} = \min[\pi_i, \pi_j] \ \forall \ (i,j)$, then it is easy to verify that $r' \geq r$. Indeed, it is the largest r for which the u values are the same as that for the original r. Since the π values have not changed, the objective value for r and r' are the same. Indeed, this is the network which produces the largest flow values for the same value of the objective function.

There is an alternative to the above algorithm. It uses the potentials right from the beginning. We first renumber the nodes so that the potentials are in decreasing (non-increasing) order: $\pi_1 = \pi_2 \geq \pi_3 \geq .... \geq \pi_n > 0$. Note that there are at least two nodes with the largest value because of the way in which these are defined.

The first step involves a circuit with capacity equal to $\frac{\pi_n}{2}$; then we reset the potentials to $\pi_i^{new} = \pi_i^{old} - \pi_n$ and drop all nodes with zero potential and repeat this operation. Finally, we superpose all the above circuits (i.e. add capacities of each to give the capacity of individual edges) to produce the final solution. We illustrate this with the same problem as before.

In this problem, $\pi_A = \pi_D = 9 > \pi_C = \pi_E = \dot{7} > \pi_B = 6$. Thus the first circuit is $(A - D - C - E - B - A)$ and has a capacity equal to 3. This reduces the potentials to: $\pi_A = \pi_D = 3 > \pi_C = \pi_E = 1$ and node B drops out. The next circuit is $(A - D - C - E - A)$ which has a capacity equal to $\frac{1}{2}$. The new potentials are: $\pi_A = \pi_D = 2$. When we have only two nodes with positive potential, then the circuit becomes just an edge with capacity equal to the potential. In this case, edge (A, D) with capacity equal to 2. Now we superpose all these to get the network shown below:



The first circuit allows us to send 6 units between any pair of nodes and hence satisfies all requirements at node B. The next circuit allows us to send 1 unit between any pair of the remaining nodes and hence all requirements at nodes C and E are now taken care of. The final edge takes care of sending two more units between A and D bringing this total to 9 as required. It is easy to verify that the sum of capacities equals $\frac{\sum_i \pi_i}{2}$ which we have already shown to be a lower bound and hence this solution is optimal.

This completes our description of multiterminal undirected networks. This work has been extended by **L.E. Trotter** in the context of matroidal flows; but

the spirit of the work is the same as that here. An excellent survey is found in the book by **Frank and Fricsh.**This work has also been extended in several other directions by Andras Frank and Robert Best.

1.6 Directed Graphs

1.6.1 Realizability

A square matrix M whose rows and columns correspond to vertices (in the same order) is said to be *semiprincipally partitionable* ($M \in SPP$) if, after possibly permuting the vertices (the same permutation is applied to both rows and columns), it satisfies the following conditions:

$$M \geq 0; m_{i,i} = \infty; M = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}$$

where all elements in $M_{1,2}$ are the same and equal the smallest element in M and $M_{i,i}$ are square and have the same property as M, and such a process continues until the submatrix is of size 1.

Theorem 5 *If M is realizable, then $M \in SPP$.*

Proof: Let $(X, \bar{X})$ be the cut corresponding to the smallest of maximum flows among all pairs of origins and destinations. Partition M into X and $\bar{X}$ at the first step. For nodes, both of which are in X, let the smallest of the max flows correspond to a cut $(S, \bar{S})$. Partition X into $X \cap S$ and $X \cap \bar{S}$. The result follows by carrying out this operation until each set in the partition is a singleton.□

To permute and partition a $M \in SPP$, we must do the above operation $n-1$ times. Every element above the diagonal in the final form is in one of the matrices of the type $M_{1,2}$, and all entries in each of such matrices are the same. Thus, there are at most $n-1$ distinct entries above the diagonal. The elements on the diagonal are all ∞. Hence, the number of distinct values of maximal flows in a network is, at most, $(n+2)(n-1)/2$. It is easy to show that this bound can be achieved. If the network is *symmetric (undirected) or pseudosymmetric* ($\sum_j c_{i,j} = \sum_j c_{j,i} \ \forall \ i$), that there are, at most, $n-1$ distinct numbers also follows from this analysis. The *triangle inequality* is also a necessary condition for M to be realizable for both undirected and directed networks.

For a set S of vertices, $M_{S,\bar{S}}$ is called the *cut matrix corresponding to* S. If $m_{k,j}$ is an element of $M_{S,\bar{S}}$, then this is said to be the cut matrix of $m_{k,j}$. If $m_{k,j}$ is the largest entry in its cut matrix, then this cut matrix is said to be a *min cut matrix* of the element $m_{k,j}$.