A NEW CHARACTERIZATION OF TREES AND CO-TREES

by S. D. AGASHE, *Department of Electrical Engineering, Indian Institute of Technology, Powai, Bombay-400076*

(Communicated by Prof. P. K. Menon, F.N.A.)

(*Received 15 May 1974*)

The following characterizations of a tree and co-tree are proved : "*a tree is a minimal set of edges containing at least one element of every cutset*" and "*a co-tree is a minimal set of edges containing at least one element of every circuit*". A procedure for generating all trees of a connected graph, based on the new characterization of a tree is presented.

INTRODUCTION

According to Seshu and Reed (1961), Whitney (1935) has proved a characterization of a circuit as "*a minimal set containing at least one chord of every tree*". Seshu and Reed themselves have stated and proved the dual characterization of a cutset as "*a minimal set of edges which contains at least one branch of every tree*" (Seshu and Reed 1961, Th: 2–15). These two characterizations of circuits and cutsets in terms of co-trees and trees suggest that the reciprocal relations may hold and lead us to ask whether the following is a characterization of a tree : "*a minimal set of edges containing at least one element of every cutset*" and whether the following is a characterization of a co-tree : "*a minimal set of edges containing at least one element of every circuit*". We prove below that these two properties are indeed characteristic of trees and co-trees respectively.

CHARACTERIZATION THEOREMS

Theorem 1: A set of edges T of a connected graph G is a tree if and only if it is a set minimal with respect to the property that it contains at least one element of every cutset of G.

Proof : The 'only if' part of Theorem 1, viz., that every tree contains at least one element of every cutset, or to put it more symmetrically, given any tree and any cutset the two must have at least one element in common, is already well-known (Seshu and Reed 1961, Th: 2–14). However, we give below a proof in which the 'if' and 'only if' parts are tackled together.

We show that the following two properties of a set T of edges of a connected graph G are equivalent : (a) for every pair of vertices of G, there is a path between them consisting of elements of T only; (b) T contains at least one element from every cutset of G. (This equivalence seems to be well-known but does not appear to have been explicitly stated and proved. Therefore, although its proof is simple, we give it).

(a) *implies* (b): Suppose a set of edges T has property (a). We want to show that if C is any cutset of G, T and C have at least one element in common. C being a cutset, is a minimal set of edges which when removed separates G into two parts. If T

and C were to be disjoint, the graph obtained by deleting C from G would contain T and, therefore, would have a path (in fact, a T-path) between any pair of vertices of G and would thus be connected. This is a contradiction.

(*b*) *implies* (*a*): Suppose T is a set of edges with property (*b*). On the vertices of G, the relation of being connected by a T-path is an equivalence relation. To show that T has property (*a*), it suffices to show that this equivalence relation has only one equivalence class, namely, the set of all vertices of G. If not, by grouping the equivalence classes into two parts, we will get a partitioning of the vertices of G into two parts, say P and Q, such that there is no T-path between any vertex of P and any vertex of Q. Now, G is connected, so that there is at least one G-path between any pair of vertices of G. Therefore, the set of edges of G that "go between" P and Q is non-empty and in fact, it must be a disjoint union of cutsets. Also, it does not contain any element of T. This contradicts property (*b*) of T.

Since properties (*a*) and (*b*) are equivalent, a set T of edges is minimal with respect to property (*a*) if it is minimal with respect to property (*b*). But, by the usual definition, a tree is a set of edges that is minimal with respect to property (*a*). Hence the statement of Theorem 1 above.

Corollary 1: A set of $(n-1)$ edges of a connected graph of n vertices is a tree of the graph if the set contains at least one element from each cutset of the graph.

Theorem 2: A set of edges T^* of a connected graph G is a co-tree if and only if it is a set minimal with respect to the property that it contains at least one element of every circuit of G.

Proof: Once again, the 'only if' part, namely, that every co-tree contains at least one element of every circuit, or equivalently, that every circuit contains at least one chord or link of every tree, is well-known. However, we give below a proof in which the 'if' and 'only if' parts are tackled together.

We first observe that the following two properties of a set T^* of edges of a connected graph G are equivalent : (*a*) when T^* is deleted from G, the remaining graph is circuit-free; (*b*) T^* contains an element from every circuit of G. (This equivalence too seems to be well-known, and perhaps intuitively too simple to warrant a proof, and so, we omit a proof).

Since properties (*a*) and (*b*) are equivalent, a set T^* of edges is minimal with respect to property (*a*) if it is minimal with respect to property (*b*). However, with the usual definition of a tree, and that of a co-tree as the complement of a tree, it is easy to prove that a set T^* of edges is a co-tree if it is minimal with respect to property (*a*). Hence the statement of Theorem 2.

Corollary 2: A set of $(e-n+1)$ edges of a connected graph with n vertices and e edges is a co-tree if the set contains at least one element from each circuit of the graph.

Application : A New Procedure for Generating All Trees

Corollary 1 of Theorem 1 above immediately suggests a new procedure for generating all trees of a connected graph.

Computationally, it is very easy to generate all the cutsets of an n-vertex e-edge connected graph G, starting with, say, its incidence matrix **A**. Also, it is easy to generate, one after another or enumeratively, all the $(n-1)$-edge subgraphs of

G. To find out whether a particular $(n-1)$-edge subgraph is a tree or not, by Theorem 1, Corollary 1, it suffices to find out whether it intersects every outset or not. Further, it is enough to consider only cutsets with $(e-n+1)$ or fewer edges; a cutset (or any set of edges for that matter) with more than $(e-n+1)$ edges is bound to intersect a set with $(n-1)$ edges, the total number of edges being e. Note that it is not necessary to generate and store all the cutsets, as they can be generated one by one (with some repetition, perhaps) when needed for checking whether a given set of $(n-1)$ edges is a tree or not.

Like Theorem 1, Theorem 2 could be used to generate all co-trees of a connected graph, by first generating all the circuits or loops. However, computationally it seems easier to generate all cutsets rather than all circuits.

As an example of an algorithm for generating all trees based on Theorem 1, Corollary 1, consider the following.

Let **A** denote the $n \times e$ incidence matrix of the connected graph G, modulo 2, i.e., with entries 0 and 1 only. Let t denote an $e \times 1$ column vector of 0's and 1's, with exactly $(n-1)$ 1's; let v denote a $1 \times n$ row vector of 0's and 1's. t is to correspond to the set of $(n-1)$ edges which is to be tested for being a tree; v corresponds to a partitioning of vertices of the graph. Then, t corresponds to a tree if for every v, the scalar $(vA)t$ is non-zero. Note that the product (vA), *computed modulo* 2, is a $1 \times e$ row vector of 0's and 1's and gives the cutset corresponding to the vertex partitioning given by v. The product of (vA) with t is to be calculated in ordinary arithmetic and gives the number of edges that the set being tested has in common with the cutset. It is not necessary to compute the actual value of the scalar $(vA)t$; it is enough to check whether it is greater than zero or not. It is not necessary to consider all the 2^n vectors v; it is enough to consider $\frac{1}{2}(2^n-2)$, i.e., only $2^{n-1}-1$ of them.

Example: Consider a bridge network; $n = 4$, $e = 6$, $n-1 = 3$. $C_3^6 = 20$, so that there are at most 20 candidates for being tested as trees. It is enough to consider only 7 v's. If a (vA) has more than $c-n+1$, i.e., 3, 1's. it can be ignored. The table below shows the computations for a particular t.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}, \quad t = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

v				vA						$(vA)t$
0	0	0	1	0	1	0	1	1	0	2
0	0	1	0	0	0	1	0	1	1	1
0	0	1	1	0	1	1	1	0	1	√
0	1	0	0	1	0	0	1	0	1	2
0	1	0	1	1	1	0	0	1	1	√
0	1	1	0	1	0	1	1	1	0	√
0	1	1	1	1	1	1	0	0	0	2
			t =	1	1	0	0	1	0	

Conclusion

New characterizations of trees and co-trees have been given. An algorithm for generating all trees of a connected graph, based on the new characterization of a tree, has been described. It should be worthwhile developing more efficient algorithms based on the same characterization and to compare them with existing ones.

References

Seshu, S., and Reed, M. B. (1961). *Linear Graphs and Electrical Networks.* Addison-Wesley Publishing Co. Inc., Reading, Mass., U.S.A.

Whitney, H. (1935). On the Abstract Properties of Linear Dependence. *Am. J. Math.*, **57**, 509–533.