Enumerating all the spanning trees in an un-oriented graph – A novel approach

Cristian E. Onete
NXP Semiconductors
IP&L Department
Eindhoven, The Netherlands
cristian.onete@ieee.org

Maria Cristina C. Onete
CASED & TU Darmstadt,
Mornewegstr. 32
64293 Darmstadt, Germany
cristina.onete@cased.de

***Abstract*—In this paper, we use a modified version of the incidence matrix of an un-oriented graph so as to enumerate all the spanning trees. In particular, we formally describe the problem and then enumerate the spanning trees, also Showing how to use this method in finding the symbolic determinant of a passive circuit.**

Keywords: spanning trees; symbolic; determinant; passive circuit

I. INTRODUCTION

Finding all spanning trees of a graph is an important task having many applications in technical areas such as electrical circuit analysis, routing etc. One application of finding all spanning trees of an undirected simple graph is in computing the symbolic determinant of passive electrical circuits [2,10,11,12]. In this approach, any spanning tree represents the product of all the admittances included in it, and the sum of all spanning trees of the undirected graph is associated with the determinant of the circuit (see also the remainder of this section). Previous solutions that find/enumerate all spanning trees in a graph require $O(N^N)$ operations, where N is the number of nodes of the circuit [3, 4, 5, 6, 7, 8, 9, 13].

One can compute symbolic determinants algebraically, by using Wang Algebra [2, 11, 12]; here, the following logical operations are true:

$$\begin{gathered} X \oplus X=0 \\ X \circ X=0 \\ X \circ Y=Y \circ X \end{gathered} \quad (1)$$

We denote logical product by $\circ$ and logical sum by $\oplus$.

A key part in this method is played by the so-called short-circuit admittance matrix Y_{sc}, which is (N-1)x(N-1) for a circuit comprising only bi-terminal devices, and whose main diagonal elements equal the sum of the admittances connected to the nodes, whereas all the other entries equal the negative of the admittance bound between the nodes. For a complete, 4-node circuit, Y_{sc} is as in (2).

$$Y_{sc}=\begin{bmatrix} (y_{12}+y_{13}+y_{14}) & -y_{12} & -y_{13} \\ -y_{12} & (y_{12}+y_{23}+y_{24}) & -y_{23} \\ -y_{13} & -y_{23} & (y_{13}+y_{23}+y_{34}) \end{bmatrix} \quad (2)$$

Previous work [2,10,11,12] shows that the determinant of Y_{sc} is the logical Wang-algebra product of the main diagonal entries of this matrix. For our example on 4 nodes, the determinant is as in (3).

$$Det(Y_{sc})=(y_{12}+y_{13}+y_{14}) \circ (y_{12}+y_{23}+y_{24}) \circ (y_{13}+y_{23}+y_{34}) \quad (3)$$

Denote n:=N-1. The total number of required multiplications is, trivially, n^n. In our case, this is 27.

[1] shows that for a simple complete graph on N nodes there are $N^{(N-2)}$, or $(n+1)^{(n-1)}$, spanning trees. The ratio

$$R=\frac{n^n}{(n+1)^{(n-1)}} \quad (4)$$

is then the inefficiency indicator for this approach, indicating the additional operations required for the determinant computation. If the determinant is symbolic, the complexity cannot be less than $O(N^{(N-2)})$, namely the total number of spanning trees for the complete graph with N nodes.

A. Our Results

Our results are two-fold:

- We find a method to generate all the spanning trees of an undirected joint graph with less nodes than there are links, such that the inefficiency factor is minimized.
- We use this method in further applications.

B. Structure of the paper

Paragraph II shows important structural preambles in setting up an algorithm that finds all spanning trees in a graph; the algorithm is then outlined in section III. Section IV shows how one can use the results of section III towards

978-1-4244-6817-1/10/$26.00 ©2010 IEEE

finding Hamiltonian circuits and solving the Travelling Salesman Problem. Finally, some conclusions are outlined in section V.

II. PASSIVE CIRCUITS AND THEIR INCIDENCE MATRICES

Consider a complete circuit on 4 nodes; its graph is as in Fig. 1. For simplicity we have indicated only the links and their orientation, ignoring any admittances. If this is a passive circuit with only bi-terminal devices, however, each link represents an admittance e.g. $L_1=y_{14}$. For general graphs, links may also be associated with distances, costs, etc. In any (and thus also in a passive) circuit, there is a so-called *reference node*. The entries of Y_{sc} are as in (2) if node 4 is the reference (the entries corresponding to this node are removed). Note that the determinant of the circuit does not depend on the choice of the reference node. For any directed graph, the nodes-links incidence matrix **Inc** is an N x l matrix (for l the number of links) whose (i,j) entries are -1, 1, or 0 depending on whether link j enters node i, exits it, or respectively it is not linked to i at all. Each column of **Inc** has precisely 2 non-zero entries (an entry and an exit node, which are different as the graph is simple). In Fig. 1 we have N = 4, l = 6, and **Inc** as in (5).

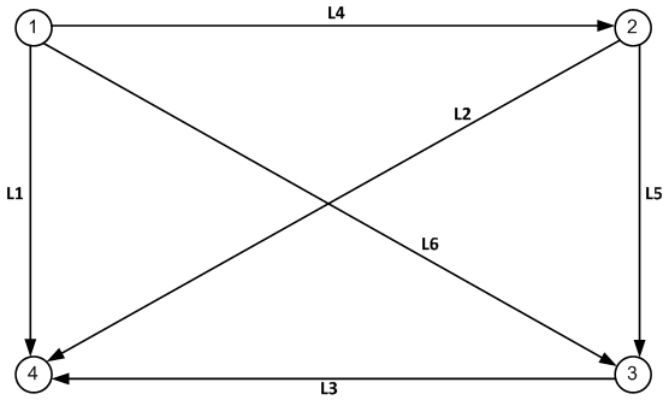

Figure 1. A complete graph on four nodes

$$\mathbf{Inc}=\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

In circuit theory and for graphs in general, one uses the so-called reduced incidence matrix **RInc**, which is obtained after removing the row of the reference node from **Inc** [14] (in our example, row 4). Now each column of **RInc** may also contain a single non-zero entry if the link corresponding to the column contained the reference node. Assume that the graph in Fig. 1 represents a passive circuit having as links only passive bi-terminal devices Y_{ij}. For other applications, Y_{jj} may represent other costs. For passive circuits, it holds that:

$$\mathbf{Y_{sc}=RInc*Diag*(RInc)^T} \quad (6)$$

Here, the lxl diagonal matrix **Diag** has non-zero entries equaling Y_{ij}. Though this example may appear restrictive, note that one can write such a matrix for any graph!

Consider a graph with l > (N-1). For this graph, we call **RInc** *wide* i.e. it has more columns than rows; similarly, $\mathbf{(RInc)^T}$ is *tall* as it has more rows than columns. Obviously the product of a square and a wide matrix of appropriate size is wide and the product of a square and a tall matrix is tall.
We formulate a first Lemma.

Lemma 1

The short-circuit admittance matrix $\mathbf{Y_{sc}}$ of a linear passive circuit comprising only bi-terminal devices (inductors, capacitors, and resistors) can be written as $\mathbf{Y_{sc}=WU}$, where **W** is wide, **U** is tall, and either (7a) or (7b) holds:

$$\mathbf{W=RInc} \quad ; \quad \mathbf{U=Diag*(RInc)^T} \quad (7a)$$

$$\mathbf{W=RInc*Diag} \quad ; \quad \mathbf{U=(RInc)^T} \quad (7b)$$

Proof.
$\mathbf{WU=RInc*Diag*(RInc)^T}$ in both cases and it is identical with equation (6), which completes the proof. □

We generally write **U** and **W** as in (7a). Lemma 2 follows.

Lemma 2

The entries in $\mathbf{U= Diag*(RInc)^T}$ have an absolute value of Y_{ij} and the same sign as corresponding entries in $\mathbf{(RInc)^T}$.

Proof.
This Lemma follows by the construction of **Diag**. □

For the circuit in Fig. 1 $\mathbf{U= Diag*(RInc)^T}$ is as in (8).

$$\mathbf{U}=\begin{bmatrix} Y_{14} & 0 & 0 \\ 0 & Y_{24} & 0 \\ 0 & 0 & Y_{34} \\ Y_{12} & -Y_{21} & 0 \\ 0 & Y_{23} & -Y_{32} \\ Y_{13} & 0 & -Y_{31} \end{bmatrix} \quad (8)$$

Recall that $Y_{ij}=Y_{ji}$ for any (i,j) and that for **any** graph, the number of non-zero entries of **any** row of $\mathbf{(RInc)^T}$ and thus of **U** is 1 or 2. Also, the rank of **RInc** and thus of **U** is n=N-1 [14]. For the graph in Fig. 1, we may write $\mathbf{Y_{sc}}$ as in (9).

$$\mathbf{Y_{sc}}=\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y_{14} & 0 & 0 \\ 0 & Y_{24} & 0 \\ 0 & 0 & Y_{34} \\ Y_{12} & -Y_{21} & 0 \\ 0 & Y_{23} & -Y_{32} \\ Y_{13} & 0 & -Y_{31} \end{bmatrix} \quad (9)$$

In terms of node-adjacencies, (9) becomes (10):

$$\mathbf{Y_{sc}}=\begin{bmatrix}1&0&0&1&0&1\\0&1&0&-1&1&0\\0&0&1&0&-1&-1\end{bmatrix}\begin{bmatrix}(1,4)&0&0\\0&(2,4)&0\\0&0&(3,4)\\(1,2)&-(2,1)&0\\0&(2,3)&-(3,2)\\(1,3)&0&-(3,1)\end{bmatrix} \quad (10)$$

III. FINDING ALL THE SPANNING TREES

Our method of finding all the spanning trees of undirected joint graphs with l > n heavily relies on **U**, thus we begin by looking at it in more detail. Recall the notation n = N-1.

We have already stated that **U** is of rank n, meaning that it contains at least one (nxn) nonsingular submatrix. We generalize the form of **U** in Fig. 1 in (11), where the function x could stand for admittance values, distances, costs, etc.

$$\mathbf{U}=\begin{bmatrix}x(1,4)&0&0\\0&x(2,4)&0\\0&0&x(3,4)\\x(1,2)&-x(2,1)&0\\0&x(2,3)&-x(3,2)\\x(1,3)&0&-x(3,1)\end{bmatrix} \quad (11)$$

The parameters of the function x are: the node the link leaves from and the node it arrives in. We hence associate each entry of **U** with triplets (i,j,k) in which k is the row number. The nodes of the graph are associated with the columns.

Furthermore, we know from e.g. [11] that all the terms of the determinant shown in equation 2 are positive and therefore we may remove the signs in (11) as we show in (12).

$$\mathbf{U}=\begin{bmatrix}x(1,4)&0&0\\0&x(2,4)&0\\0&0&x(3,4)\\x(1,2)&x(2,1)&0\\0&x(2,3)&x(3,2)\\x(1,3)&0&x(3,1)\end{bmatrix} \quad (12)$$

Recall that rows with only one non-zero entry correspond to connections to the reference node. Clearly, only at most n such rows exist, as there are no parallel links (the graph is simple). All other rows have 2 non-zero entries each.

We associate spanning trees with submatrices of **U** of rank n. More concretely the edges in each spanning tree are the links (i,j), s.t. i and j are the parameters of x for the diagonal entries of the non-singular nxn submatrices. We show that there is a one-to-one correspondence between non-singular nxn submatrices and spanning trees in this sense. Furthermore, Lemmas 3 and 4 show how to reduce the complexity of finding such submatrices.

Lemma 3

Submatrices whose rows **all** contain two non-zero entries each are not associated with spanning trees.

Proof.

All rows containing two non-zero entries show that the nodes associated with the non-zero entries are not connected to the reference node. If all the rows of the submatrix ignore the reference node, the graph cannot possibly be spanned by the diagonal entries. □

As a direct consequence of Lemma 3, it results that a spanning tree can be obtained only from matrices having at least one row having only one entry.

Let us further note that including only matrices **U** with l-n out of l rows having two entries removes $\binom{n}{l-n}$ combinations from investigation. We must also exclude singular submatrices, i.e. those submatrices whose rows or columns are not linearly independent. This second restriction is done by imposing an additional criterion in choosing our submatrices, namely that the submatrices that we choose do not contain zero columns; (2) . In practice, this is achieved very easily by inspection. We call a submatrix **U** with at least one row with 1 entry only and having this additional property *permissible*.

Lemma 4

If a permissible matrix is singular, then the graph given by this submatrix is disjoint.

Proof.

The additional property which we have requested ensures that no zero columns exist. Hence, permissible submatrices may only be singular if there exists a linear combination of more than one row. This only occurs if the edges described by the inter-dependent rows form a cycle. Consider a submatrix in whose associated graph there is a cycle k nodes. This is described by k rows of the matrix. At least one row of the graph must have a single entry (otherwise the matrix is not permissible). Therefore, since n-k nodes remain, and only at most n-k rows, the matrix either contains a zero column, or the graph is disjoint. We note that such a submatrix can always be changed, by row and column permutations, into a block-diagonal matrix similar to a Jordan form:

$$U^{*}=\begin{bmatrix}A&0&0\\0&\dots&0\\0&0&B\end{bmatrix}$$

Here A and B are blocks. In the graph, the nodes in A are disjoint from the nodes in every subsequent block.

By relating permissible matrices with n-th order determinants, we now show how to find the latter.

Lemma 5

Consider non-singular permissible matrices. The n-th order determinants are obtained such submatrices matrices $\mathbf{U}^{*}$ by means of a permutation. The determinant is then equal to the product of the elements on either diagonal of the permuted matrix.

Proof.

If the matrix is non-singular, clearly it describes a spanning tree of the graph. Furthermore, the rows can be permuted as follows: for each row i, one looks if the diagonal entry is occupied. If not, one looks for another row i^* that has an entry at position i and a possible second entry at a position greater than i. Such a row always exists, since the matrix is permissible.

The entries on the diagonal will also indicate, if they are symbolic, the spanning tree corresponding to the chosen non-singular submatrix. If the function x gives the length or the weights corresponding to each edge, we also have the length/weight of the spanning tree. □

For example, we may have a spanning tree as $Y_{14}Y_{24}Y_{34}$ which indicates a term of the determinant of the admittance matrix, but also $Y_{14} + Y_{24} + Y_{34}$ representing the length of the same tree but also {(1,4), (2,4), (3,4)} indicating the spanning tree through its nodes. The method described above can be summarized by the following algorithm.

Spanning tree algorithm

0. As a speed-up, we can order the rows of U such that first we list all the rows containing a non-zero entry on the first position (note that the possible second non-zero entry is at a different position on each row, so we can order the rows in increasing order), then list the remaining rows containing a non-zero entry on the second position, and so on.1. Start with the first one-nonzero-entry row of U. Choose, in a strictly top-down fashion, another N-2 rows such that the resulting matrix is permissible. Now try to form the Jordan-like form shown in the proof of Lemma 4: as one goes from row 1 to N-1, one shifts the columns to the left so as to form blocks.

If no such matrix can be formed, the matrix U^* was nonsingular. Repeat this step for all possible combinations with the first row and write the spanning trees obtained thus,

2. Repeat step 1 for the next row containing a single non-zero entry, and then for all remaining rows with a single non-zero entry.

Observations

1. Note that each resulting spanning tree is unique, as the algorithm works only top-down.
2. The algorithm is fully parallel: each submatrix U^* is independent of previous submatrices.
3. There is no backtracking.

The total number of determinants that are considered starting step 1 from row i $\binom{n}{l-n-i+1}$. The maximum complexity for the worst case scenario (a complete graph) is

$$\sum_{i=l}^{n}\binom{n}{l-n-i+1} \tag{13}$$

Let us apply the algorithm to the graph shown in Fig. 1.

For row 1 we obtain trees: x(1,4)x(2,4)x(3,4);,
x(1,4)x(2,4)x(3,2); x(1,4)x(2,4)x(3,1);, x(1,4)x(3,4)x(2,1);,
x(1,4)x(3,4)x(2,3);, x(1,4)x(2,1)x(3,2);, x(1,4)x(2,1)x(3,1);,
x(1,4)x(2,3)x(3,1);,
For row 2: x(2,4)x(3,4)x(1,2);, x(2,4)x(3,4)x(1,3) ;,
x(2,4)x(1,2)x(3,2);, x(2,4)x(1,2)x(3,1);, x(2,4)x(3,2)x(1,3);,
For row 3: x(3,4)x(1,2)x(2,3);, x(3,4)x(2,1)x(1,3);,
x(3,4)x(2,3)x(1,3)
.

IV. COMPLEXITY OF THE ALGORITHMS

The complexity of this method is rather high. As before, we denote N-1 = n. The complexity of the method for a complete graph is computed as follows. Let l be the number of links, hence also the number of rows of the matrix U. There are n rows with a single non-zero entry. For step 1, we have $\binom{n-1}{1-1}$ possible choices of U^*. For the second, we have $\binom{n-1}{1-2}$ combinations (the algorithm runs top-down). For a complete graph, the total number of combinations is

$$T=\binom{n-1}{1-1}+\binom{n-1}{1-2}+\ldots+\binom{n-1}{1-n-1} \tag{14}$$

Testing for singularity is a linear process in the number of nodes, as is permuting the rows in order to obtain spanning trees. This complexity is exponential, but no worse than previous work (algorithms [5-9] have a complexity O(N+L+NT), where T is the number of spanning trees, which is N^{N-2} for a complete graph).

The advantage of our method, however, is the fact that it requires no backtracking, and the process shown in step 1 can be run in parallel, thus reducing the method to the complexity of a single step (if fully parallel executions are considered). Moreover, in a circuit comprising different elements which are bridged by a single connection, it is not necessary to search for spanning trees in the entire circuit! The circuit can be divided into sub-sections that are connected by a single isthmus, and the method can be applied to each component. A final spanning tree will then contain the trees found in each component and the isthmi between the components. The complexity then is reduced to the complexity of the method for the biggest component.

V. CONCLUSIONS

In this paper we show a new algorithm for finding all the spanning trees of an undirected graph. The algorithm may be applied for finding the determinant of a symbolic or not matrix or for finding the length of the spanning trees. Even though the method has exponential complexity, the fact that it is fully parallel and requires no backtracking makes it useful in practice, particularly when graphs with many components are considered.

REFERENCES

[1] B. E. Wu, K-M Chao, "Spanning trees and Optimization Problems", Chapmann & Hall/CRC Press, 2004.

[2] R. F. Duffin, T. D. Morley, "Wang Algebra and Matroids", *IEEE Trans.on Circuits and Systems*, vol. CAS-25, pp 758 – 762, 1978.

[3] H. N. Gabow, E.W. Myers, "Finding all spanning trees of directed and undirected graphs", SIAM J. Comput. Vol 7, pp 280-287, 1978.

[4] S. Kapoor, H. Ramesh, "Algorithms for generating all spanning trees of undirected, directed and weighted graphs", in Lecture Notes in Computer Science, (F. Dehne, J – R Sack, N. Santoro eds.), pp 461-472, Smith, Springer-Verlag, 1992.

[5] R. Tarjan, "Depth-first first and linear graph algorithms", SIAM J. Comput., vol. 1, pp 146-160, 1972.

[6] A. Shioura, A. Tamura, "Efficiently Scanning All Spanning Trees of an Undirected Graph", J. Operation Research of Japan, vol. 38, pp 331-344, 1993.

[7] A. Shioura, A. Tamura, T. Uno, "An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graphs", SIAM J. Comput., vol.26, pp. 678-692, 1994.

[8] S. Kapoor, H. Ramesh, "Algorithms for Enumerating All Spanning Trees of Undirected and Weighted Graphs", SIAM J. Comput., vol.24, pp. 247-265, 1995.

[9] W. Mayeda, S. Seshu, "Generation of trees without duplication", IEEE Trans. On Circuit Theory, vol. Pp. 181- 185, 1965.

[10] W-K Chen, "Topological Analysis For Active Networks", *IEEE Trans. On Circuit Theory,* pp. 85–91, March 1965.

[11] W-K Chen"Graph Theory and Its Engineering Applications", World Scientific Publishing Co. Pte. Ltd, 1997.

[12] W-K Chen, "Active Network Analysis", World Scientific Publishing Co. Pte. Ltd, 1991.

[13] V. Ejov, J. A. Filar, S. K. Lucas and J. L. Nelson, "Solving The Hamiltonian Cycle Problem Using Symbolic Determinants", Taiwanese Journal Of Mathematics, Vol. 10, No. 2, pp. 327-338, February 2006.

[14] Balabanian N., Bickart T.A., "Electrical Network Theory", Wiley 1969.