Counting Spanning Trees in Cographs

S.D. Nikolopoulos and C. Papadopoulos

24 - 2002

Preprint, no 24 - 02 / 2002

Department of Computer Science
University of Ioannina
45110 Ioannina, Greece

Counting Spanning Trees in Cographs

Stavros D. Nikolopoulos and Charis Papadopoulos

Department of Computer Science, University of Ioannina
GR-45110 Ioannina, Greece
e-mail: {stavros, charis}@cs.uoi.gr

Abstract: In this paper we propose a linear-time algorithm for determining the number of spanning trees in cographs; we derive formula for the number of spanning trees of a cograph G on n vertices and m edges, and prove that the problem of counting the number of spanning trees of G can be solved in $O(n+m)$ time. Our proofs are based on the Kirchhoff matrix tree theorem which expresses the number of spanning trees of a graph as a function of the determinant of a matrix that can be easily construct from the adjacency relation of the graph. Our results generalize previous results regarding the number of spanning trees.

Keywords: Spanning trees, Kirhhoff matrix tree theorem, cographs, tree graphs, combinatorial problems.

1 Introduction

We consider finite undirected graphs with no loops nor multiple edges. Let G be such a graph on n vertices. A *spanning tree* of G is an acyclic $(n-1)$-edge subgraph. The problem of calculating the number of spanning trees on the graph G is an important, well-studied problem in graph theory. Deriving formulas for different types of graphs can prove to be helpful in identifying those graphs that contain the maximum number of spanning trees. Such an investigation has practical consequences related to network reliability [11, 15].

Thus, for both theoretical and practical purposes, we are interested in deriving formulas for the number of spanning trees of a graph based on its time complexity in order to calculate the formula. Many cases have been examined depending on the choice of G. It has been studied when G is a labelled molecular graph [2], when G is a circulant graph [20], when G is a complete multipartite graph [18], when G is a cubic cycle and quadruple cycle graph [19], when G is a threshold graph [7] and so on (see Berge [1] for an exposition of the main results; also see [4, 9, 12, 13, 14, 16, 17, 18]).

The purpose of this paper is to study the problem of finding the number of spanning trees and propose a fast algorithm regarding the number of spanning tree of a cograph G. A graph G on n vertices is called a tree graph if it is a connected $(n-1)$-edge graph; G is called a cograph, or complement reducible graph, if it contains no induced subgraph isomorphic to P_4 [6]. A cograph G has a unique tree representation $T(G)$ called cotree. Our proofs are based on a classic result known as the *Kirchhoff Matrix Tree* theorem [8], which expresses the number of spanning trees of a graph G as a function of the determinant of a matrix (Kirchhoff Matrix) that can be easily construct from the adjacency relation (adjacency matrix, adjacency lists, ect) of the graph G. Calculating the determinant of the Kirchhoff Matrix seems to be a promising approach for computing the number of spanning trees of families of graphs (see [1, 4, 5, 12, 18]). In our case, we compute the number of spanning trees of a cograph G, using standard techniques from linear algebra and matrix theory. Our ideas and techniques will be formalized and further clarified in the sequel.

2 Definitions and Background Results

For an $n \times n$ matrix A, the ijth *minor* is the determinant of the $(n-1) \times (n-1)$ matrix M_{ij} obtained from A deleting row i and column j. The ith *cofactor* denoted A_i equals $det(M_{ii})$.

Let $G = (V, E)$ be a graph on n vertices. Then the *Kirhhoff matrix* K for the graph G has

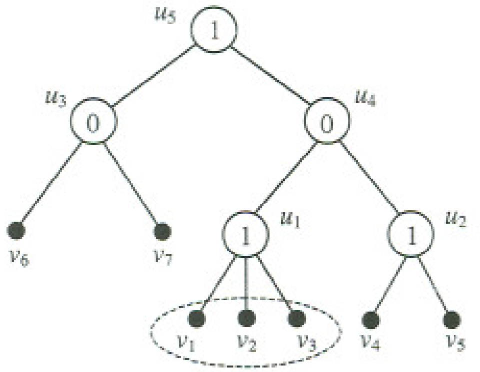
$$k_{i,j} = \begin{cases} d_i & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

elements, where d_i is the number of edges incident to vertex v_i in the graph G. The Kirhhoff Matrix Tree Theorem is one of the most famous results in graph theory. It provides a formula for the number of spanning trees of a graph G, in terms of the cofactors of its Kirhhoff Matrix.

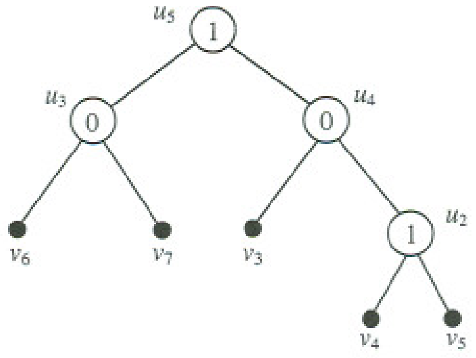
Theorem 2.1. (Kirchhoff Matrix Tree Theorem [8]) *For any graph G with K defined as above, the cofactors of K have the same value, and this value equals the number of spanning trees of G.*

The *complement reducible graphs*, or so-called *cographs*, are defined as the class of graphs formed from a single vertex under the closure of the operations of union and complement. More precisely, the class of cographs is defined recursively as follows: (i) a single-vertex graph is a cograph; (ii) the disjoint union of cographs is a cograph; (iii) the complement of a cograph is a cograph.

Cographs themselves were introduced in the early 1970s by Lerchs [10] who studied their structural and algorithmic properties. Lerchs has shown, among other properties, the following two very nice algorithmic properties: (i) cographs are exactly the P_4 restricted graphs, and (ii) cographs admit a unique tree representation, up to isomorphism, called cotree [3].



(a)



(b)

Figure 1: (a) A cograph on 6 vertices and (b) the corresponding cotree.

Let G be a cograph, and let $T(G)$ be its corresponding cotree. We define the following nodes/vertices sets on the cotree $T(G)$:

- L_i, which contains the 0(1)-nodes of the ith level of $T(G)$, $1 \leq i \leq h$, and
- $ch(u_i)$, which contains the children of the 0(1)-node $u_i \in T(G)$, $1 \leq i \leq k$.

The parent of a node/vertex x in $T(G)$ is denoted by $p(x)$. The root of the cotree is a 1-node and denoted by r. Figure 1 features a cotree $T(G)$ with the corresponding level sets.

3 The Number of Spanning Trees

Let G be a cograph on n vertices and m edges and let $L_1, L_2, \ldots, L_h$ be the nodes sets of its cotree $T(G)$. In order to compute the number of spanning trees of the graph G we make use of Theorem 2.1; that is, we delete an arbitrary vertex v of the set $V(G)$ and all its edges incident to vertex v. Now the vertex set $V(G)$ of the resulting cograph G and the leaves of the cotree $T(G)$ is of size $n-1$.

We set $s(v) := d_v$ for every vertex $v \in V(G)$, where d_v denotes the degree of the vertex v. This labeling of the vertices is called s-labeling.

We define a function which we call *Replace-Update* and contracts the cotree $T(G)$ into one vertex tree. Let u_i be a 0(1)-node of $T(G)$ such that the set $ch(u_i) = \{v_1, v_2, \ldots, v_p\}$ contains leaf vertices of $T(G)$, $1 \leq i \leq k$. The function Replace-Update is applied to the node u_i and works as follows:

- It changes the s-labels $s(v_1), s(v_2), \ldots, s(v_p)$ of the vertices $v_1, v_2, \ldots, v_p$, respectively, in the case where u_i is a 1-node,
- computes specific values for the s-labels of the vertices u_{p-1} and u_p,
- delete the vertices $v_1, v_2, \ldots, v_p$ from the cotree $T(G)$, and replace the node u_i with vertex v_p.

Figure 1 shows the application of the function Replace-Update on node u_4 on a cotree $T(G)$. The formal description of the function Replace-Update is as follows:

Replace_Update$(u, T(G))$

---

1. Compute the vertex set $ch(u) = \{v_1, v_2, \ldots, v_p\}$;

2. **for** every vertex $v_i \in ch(v)$ **do**
  if u is 1-node **then** $s(v_i) := s(v_i) + 1$;

3. Compute $e := \sum_{i=1}^{p-1} \frac{1}{s(v_i)}$;

4. Update the s-label $s(v_{p-1})$ as follows:
  $s(v_{p-1}) := s(v_{p-1}) \cdot (1 + s(v_p) \cdot e)$;

5. Update the s-label $s(v_p)$ as follows:
  if u is 0-node **then** $s(v_p) := \frac{s(v_p)}{1 + s(v_p) \cdot e}$
  else $s(v_p) := \frac{s(v_p) \cdot (1-e) - 1}{1 + s(v_p) \cdot e}$;

6. Delete vertices $v_1, v_2, \ldots, v_p$ from $T(G)$ and replace node $u \in T$ with vertex v_p;

7. Return the s-labels $s(v_1), s(v_2), \ldots, s(v_{p-1})$ and the resulting cotree $T(G)$;
  If u is the root of the cotree $T(G)$ return the s-label $s(v_p)$;

Algorithm 1: *Replace_Update*$(u, T(G))$

We next describe an algorithm which computes the number of spanning trees $\tau(G)$ of a cograph G; it works as follows: First it computes the graph $G := G - v$, where $v \in V(G)$, and constructs it cotree $T(G)$; the resulting graph G has $n-1$ vertices. Then, it computes the degree d_i of each vertex $v_i \in T(G)$ (i.e., v_i is a leaf of $T(G)$) and assigns $s(v_i) := d_i$, $1 \leq i \leq n-1$. Next, it repeatedly applies the function Replace-Update$(u, T(G))$ to each node u with the property that all its children are leaves in $T(G)$, and computes the s-labels $s(v_1), s(v_2), \ldots, s(v_{n-1})$ of the vertices of $T(G)$. Finally, it computes the number of spanning trees $\tau(G) := \prod_{i=1}^{n-1} s(v_i)$. The formal description of the above described algorithm is as follows:

Number_Spanning_Trees

Input: A cograph G on n vertices and m edges;

Output: The number of spanning trees $\tau(G)$ of the cograph G;

---

1. Set $G := G - v$, where $v \in V(G)$ and let $v_1, v_2, \ldots, v_{n-1}$ be the vertices of G;

2. Construct the cotree $T(G)$ of the cograph G;

3. Compute the sets $L_1, L_2, \ldots, L_h$ such that:
L_i contains the 0-nodes and 1-nodes of the ith level of $T(G)$, $1 \leq i \leq h$;

4. **for** each leaf v_i of $T(G)$ **do**
$s(v_i) := d_i$;

5. **for** $i = h - 1$ down to 1 **do**
for every 0-node or 1-node $u \in L_i$ **do** *Replace_Update*$(u, T, ch(u))$;

6. Compute $\tau(G) := \prod_{i=1}^{n-1} s(v_i)$;

Algorithm 2: Number_Spanning_Trees

Given a cograph $G = (V, E)$ we construct its corresponding cotree $T(G)$. We partition the vertices of the cotree $T(G)$ according to L-function. Afterwards, we apply function *Replace_Update*$(u, T, ch(u))$ on each node of level L_{h-1} and calculate values $s(v)$, for vertices $v \in L_h$, until we get level L_1. At the end we compute the product of the final $n - 1$ values $s(v)$ which expresses the number of spanning trees $\tau(G)$ of cograph G.

4 Correctness and Time Complexity

The correctness is established thought Theorem 2.1, and, thus, we construct the Kirhhoff matrix and calculate one of its cofactors.

Let G be a cograph, and let $T(G)$ be the rooted cotree of G. Let $L_1, L_2, \ldots, L_h$ be the level sets of the cotree $T(G)$ and let $v_1, v_2, \ldots, v_n$ and $u_1, u_2, \ldots, u_k$ be the vertices (leaves) and the 0(1)-nodes of $T(G)$, respectively. Then, we form the Kirhhoff $n \times n$ matrix K of the cograph G; it has the following form:

$$K = \begin{bmatrix} d_1 & & (*)_1 & & & & & & \\ & \ddots & & & & & & & \\ (*)_1 & & d_p & & & & (-1)_{ji} & & \\ & & & d_{p+1} & & (*)_2 & & & \\ & & & & \ddots & & & & \\ & & & (*)_2 & & d_q & & & \\ & & & & & & \ddots & & \\ & & (-1)_{ij} & & & & & d_\ell & (*)_k \\ & & & & & & & \ddots & \\ & & & & & & (*)_k & & d_n \end{bmatrix},$$

where, according to the definition of the Kirhhoff matrix, d_i is the degree of the vertex v_i of the cograph G. The off-diagonal positions in every block $(*)_i$ is -1 if u_i is a 1-node and 0 otherwise, $1 \leq i \leq k$. The entries $(-1)_{ij}$ and $(-1)_{ji}$ of the off-diagonal position (i,j) and (j,i) are both -1 if the vertices v_i and v_j are adjacent in G and 0 otherwise, $1 \leq i,\ j \leq n$.

Starting from the upper left part of the matrix, the first p rows of the matrix correspond to the p vertices $v_1, v_2, \ldots, v_p$ with the same parent $u_1 \in L_{h-1}$ in cotree $T(G)$; the next $q-p$ rows correspond to the $q-p$ vertices with the parent node u_2 and so forth. Moreover, leaf vertices v_i and v_j of cotree $T(G)$ with the same parent $p(v_i) = p(v_j)$ have the same degree on cograph G, $d_i = d_j$.

From the Kirchhoff Matrix Theorem the value of any *cofactor* of matrix K equals the number of spanning trees of cograph G. Arbitrary, we focus on the *cofactor* A_{nn} of matrix K. Recall that the nn-th *minor* is the determinant of the $(n-1) \times (n-1)$ matrix M_{nn} obtained from K deleting the last row and the last column. Substituting the values $s'(v_i) = d_i$, $1 \leq i \leq n-1$ the form of matrix M_{nn} becomes:

$$M_{nn} = \begin{bmatrix} s'(v_1) & & (*)_1 & & & & & & & \\ & \ddots & & & & & & & & \\ (*)_1 & & s'(v_p) & & & & & (-1)_{ji} & & \\ & & & s'(v_{p+1}) & & (*)_2 & & & & \\ & & & & \ddots & & & & & \\ & & & (*)_2 & & s'(v_q) & & & & \\ & & & & & & \ddots & & & \\ & & (-1)_{ij} & & & & & s'(v_\ell) & & (*)_k \\ & & & & & & & & \ddots & \\ & & & & & & & (*)_k & & s'(v_{n-1}) \end{bmatrix}.$$

Thus, we focus on the computation of the determinant of matrix M_{nn}, as it is clear from Theorem 2.1 that

$$\tau(G) = det(M_{nn}). \tag{1}$$

In order to compute the determinant $\det(M_{nn})$, we start by focus on the first p rows and p columns. We multiply by -1 the p row and add it to rows $1, 2, \ldots, p-1$. Now all the non-zero off-diagonal elements in the first $p-1$ rows lie on the column p. Thus, our gain is to eliminate these elements. In the case where u_1 is a 1-node we multiply column j, with value $\frac{s'(v_p)+1}{s'(v_j)+1}$ and add it to column p, for $1 \leq j \leq p-1$. Similarly, in the case where u_1 is a 0-node we multiply column j, with value $\frac{s'(v_p)}{s'(v_j)}$ and add it to column p, for $1 \leq j \leq p-1$. Now, in rows $1, 2, \ldots, p-1$ only the diagonal positions are non-zero elements. So expanding in terms of the $p-1$ rows the determinant of matrix M_{nn} becomes:

$$det(M_{nn}) = \prod_{i=1}^{p-1} s(v_i) \cdot \begin{vmatrix} s(v_p) & & & & & (-1)_{j'i'} & & \\ & s'(v_{p+1}) & & (*)_2 & & & & \\ & & \ddots & & & & & \\ & (*)_2 & & s'(v_q) & & & & \\ & & & & \ddots & & & \\ (-1)_{i'j'} & & & & & s'(v_\ell) & & (*)_k \\ & & & & & & \ddots & \\ & & & & & (*)_k & & s'(v_{n-1}) \end{vmatrix}, \tag{2}$$

where

$$\begin{aligned} s(v_i) &= s'(v_i) + 1, \quad 1 \le i \le p \\ e &= \sum_{i=1}^{p-1} \frac{1}{s(v_i)} \\ s(v_{p-1}) &= s(v_{p-1}) \cdot (1 + s(v_p) \cdot e) \\ s(v_p) &= \frac{s(v_p)(1-e) - 1}{1 + s(v_p) \cdot e} \end{aligned}$$

if u_1 is a 1-node, and

$$\begin{aligned} s(v_i) &= s'(v_i), \quad 1 \le i \le p \\ e &= \sum_{i=1}^{p-1} \frac{1}{s(v_i)} \\ s(v_{p-1}) &= s(v_{p-1}) \cdot (1 + s(v_p) \cdot e) \\ s(v_p) &= \frac{s(v_p)}{1 + s(v_p) \cdot e} \end{aligned}$$

if u_1 is a 0-node.

Applying the above operations to other blocks of nodes $u_2, \ldots, u_k$, we can compute the determinant of matrix M_{nn}. Thus, the results of this section are the following:

Lemma 3.1. *The algorithm Number_Spanning_Trees correctly computes the number of spanning trees of a cograph G.*

Theorem 3.1. *The number of spanning trees of a cograph G on n vertices and m edges can be computed in $O(n + m)$ time.*

References

[1] C. Berge, *Graphs and Hypergraphs*, North Holland, Amsterdam, 1973.

[2] T.J. Brown, R.B. Mallion, P. Pollak and A. Roth, Some methods for counting the spanning trees in labeled molecular graphs, examined in relation to certain fullerness, *Discrete Appl. Math.* **67**, 51–66, 1996.

[3] D. G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Comput.* **14**, 926–984, 1985.

[4] K.L. Chung and W.M. Yan, On the number of spanning trees of a multi-complete/star related graph, *Inform. Process. Lett.* **76**, 113–119, 2000.

[5] B. Gilbert and W. Myrvold, Maximizing spanning trees in almost complete graphs, *Networks* **30**, 23–30, 1997.

[6] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Inc., 1980.

[7] P.L. Hammer and A.K. Kelmans, Laplacian spectra and spanning trees of threshold graphs, *Discrete Appl. Math.* **65**, 255–273, 1996.

[8] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

[9] A. K. Kelmans, On Graphs with the Maximum Number of Spanning Trees, Random Structures and Algorithms 1-2 (1996), 177-192.

[10] H. Lerchs, On cliqus and kernels, Department of Computer Science, University of Toronto, March 1971.

[11] W. Myrvold, K.H. Cheung, L.B. Page, and J.E. Perry, Uniformly-most reliable networks do not always exist, *Networks* **21**, 417–419, 1991.

[12] S.D. Nikolopoulos and P. Rondogiannis, On the number of spanning trees of multi-star related graphs, *Inform. Process. Lett.* **65**, 183–188, 1998.

[13] S.D. Nikolopoulos and C. Papadopoulos, The number of spanning trees in quasi-threshold graphs, *Technical Report TR-19-02*, Dept. of Computer Science, University of Ioannina, 2002.

[14] P.V. O'Neil, The number of trees in certain network, *NoticesAmer. Math. Soc.* **10**, 569, 1963.

[15] L. Petingi, F. Boesch and C. Suffel, On the characterization of graphs with maximum number of spanning trees, *Discrete Appl. Math.* **179**, 155–166, 1998.

[16] L. Petingi and J. Rodriguez, A new technique for the characterization of graphs with a maximum number of spanning trees, *Discrete Math.* **244**, 351–373, 2002.

[17] L. Weinberg, Number of trees in a graph, *Proc. IRE.* **46**, 1954–1955, 1958.

[18] W.M. Yan, W. Myrnold and K.L. Chung, A formula for the number of spanning trees of a multi-star related graph, *Inform. Process. Lett.* **68**, 295–298, 1998.

[19] Xue-rong Yong, Talip, Acenjian, The numbers of spanning trees of the cubic cycle C_n^3 and the quadruple cycle C_n^4, *Discrete Math.* **169**, 293–298, 1997.

[20] Y. Zang, X. Yong and M.J. Golin, The number of spanning trees in circulant graphs, *Discrete Appl. Math.* **223**, 337–350, 2000.