The Number of Spanning Trees in P4-reducible Graphs

Stavros D. Nikolopoulos and Charis Papadopoulos

ABSTRACT: *We present an efficient algorithm for determining the number of spanning trees in the class of P_4-reducible graphs, which are perfect graphs and generalize both the well-known class of cographs and the class of quasi-threshold graphs. In particular, for a P_4-reducible graph G on n vertices and m edges, our algorithm computes the number of spanning trees of G in $O(n+m)$ time and space, where the complexity of arithmetic operations is measured under the uniform cost criterion. The algorithm takes advantage of the modular decomposition tree of the input graph which it gradually shrinks in a systematic fashion until it becomes a single vertex while at the same time appropriately updating certain parameters whose product gives the desired number of spanning trees. The correctness of the algorithm is established through the Kirchhoff matrix tree theorem, and is also based on structural and algorithmic properties of the graphs with few P_4s. Our results generalize previous results and extend the family of graphs admitting linear-time algorithms for the number of their spanning trees.*

1 Introduction

The purpose of this paper is to study the problem of finding the number of spanning trees in the class of P_4-*reducible* graphs, which are defined as the graphs for which no vertex belongs to more than one induced P_4 [3]. We take advantage of the modular decomposition tree of the class of P_4-reducible graphs in order to obtain an efficient solution of the above mentioned problem.

The *modular decomposition* of a graph G is a tree denoted as $T(G)$; the leaves of $T(G)$ are the vertices of G and the set of leaves associated with the subtree rooted at an internal node, induce a strong module of G. Thus, the modular decomposition tree $T(G)$ represents all the strong modules of G. An internal node is labeled by either 0 (or p for *parallel* module), 1 (or s for *series* module), or 2 (or n for *neighborhood* module). The module corresponding to a 0-node induces a disconnected subgraph of G, that of a 1-node induces a connected subgraph of G whose complement is disconnected, and that of a 2-node induces a connected subgraph of G whose complement is also connected. A linear-time algorithm for the construction of the tree $T(G)$ can be found in [4].

Let u be an internal node of $T(G)$. We denote by $M(u)$ the module corresponding to u which consists of the set of vertices of G in the subtree of $T(G)$ rooted at node u. Let $\mathrm{ch}(u) = \{u_1, u_2, \ldots, u_p\}$ be the set of the children of u in $T(G)$. Let $G(u)$ be the *representative graph* of the module $M(u)$, which is defined as follows: $V(G(u)) = \{u_1, u_2, \ldots, u_p\}$, and $E(G(u)) = \{(u_i, u_j) \mid (v_i, v_j) \in E(G), v_i \in M(u_i) \text{ and } v_j \in M(u_j)\}$.

Let G be a graph, $T(G)$ its modular decomposition tree and u an internal 2-node of $T(G)$. The graph G is a P_4-reducible graph iff for every u of $T(G)$, $G(u)$ is either a P_4, or a bull graph. Moreover, the vertices of the P_4, or the P_4 of the bull graph are leaf vertices in $T(G)$ (Giakoumakis and Vanherpe [2]).

2 The Algorithm

Let G be a P_4-reducible graph on n vertices and m edges. In order to compute the number of spanning trees of the graph G, we make use of the Kirchhoff matrix tree theorem [1]; that is, we delete an arbitrary vertex $w \in V(G)$ and all the edges incident on vertex w.

Next, we set $s(v) \leftarrow d(v)$ for every vertex v of the graph $G-w$, where $d(v)$ is the degree of the vertex v in G; we call these labels of the vertices their *s-labels*. The algorithm works by contracting in a systematic fashion the contractible nodes of the modular decomposition tree of the graph $G-w$ and by assigning to the leaf that is produced the highest-index vertex of G which is a child of the contracted node. The contractions are done by means of two functions, namely, *Replace_Parallel-Series*() and *Replace_Neighborhood*(), which also update the s-labels of the children/vertices of the contractible node. We note that the s-labels are assumed to be global variables in our algorithm. The algorithm *Spanning_Trees_Number* and the two functions are given in detail below.

Let u be a 0(1)-node such that $\mathrm{ch}(u) = \{v_1, v_2, \ldots, v_p\}$ forms a contracted block of $T(G)$; that is, each $v_i \in \mathrm{ch}(u)$ is a leaf. The function Replace_Parallel-Series()works as follows:

- if u is a 1-node then increase the s-labels $s(v_1), s(v_2), \ldots, s(v_p)$ by 1;
- compute the parameter $e(u) := \displaystyle\sum_{i=1}^{p-1} \frac{1}{s(v_i)}$;
- update the s-labels $s(v_{p-1})$ and $s(v_p)$ as follows:
  $s(v_{p-1}) := s(v_{p-1}) \cdot (1 + s(v_p) \cdot e(u))$, and $s(v_p) := s(v_p) \ / \ (1 + s(v_p) \cdot e(u))$;
  if u is a 1-node then decrease $s(v_p)$ by 1;
- make vertex v_p a child of $p(u)$ and delete the vertices $v_1, v_2, \ldots, v_{p-1}$ and node u from $T(G)$.

Let u be a 2-node such that $\mathrm{ch}(u) = \{v_1, v_2, \ldots, v_p\}$ forms a contracted block of $T(G)$. If $G(u)$ is a P_4 then $|\mathrm{ch}(u)| = 4$, or else if $G(u)$ is a bull graph, then $|\mathrm{ch}(u)| = 5$ (see [2]). The function Replace_Neighborhood() works as follows:

- compute the endpoints v_1, v_4 and the midpoints v_2, v_3 of the P_4 $v_1v_2v_3v_4$ of the graph induced by $\mathrm{ch}(u)$, and also the values of d_s and d_k, where $d_s = d(v_1) = d(v_4)$ and $d_k = d(v_2) = d(v_3)$;
- if $|\mathrm{ch}(u)| = 4$, then update the s-labels of the vertices of $\mathrm{ch}(u)$, as follows:

$$s(v_1) = d_s \tag{1}$$

$$s(v_2) = d_k + \frac{d_s - 1}{d_s} \tag{2}$$

$$s(v_3) = \frac{d_s}{\gamma} \cdot (d_k - \frac{d_s + 1}{d_s}) \tag{3}$$

$$s(v_4) = \gamma \tag{4}$$

  where

$$\gamma = 2 \cdot (d_s + d_k + 1); \tag{5}$$

∘ if $|\mathrm{ch}(u)| = 5$, then update the s-labels of the vertices of $\mathrm{ch}(u)$, as follows:

$$\begin{aligned}
s(v_1) &= d_s && (6)\\
s(v_2) &= d_k + \frac{d_s - 1}{d_s} && (7)\\
s(v_3) &= \frac{d_s}{\gamma} \cdot \frac{s(v_5) \cdot (d_k - \frac{d_s+1}{d_s}) - 2}{s(v_5) + \frac{d_s}{d_s+1}} && (8)\\
s(v_4) &= \gamma && (9)\\
s(v_5) &= s(v_5) + \frac{d_s}{d_s + 1} && (10)
\end{aligned}$$

where

$$\gamma \;=\; 2 \cdot (d_s + d_k + 1) - (d_s - 1) \cdot \frac{2 + \frac{d_s}{d_s+1} \cdot (d_k - \frac{d_s+1}{d_s})}{s(v_5) + \frac{d_s}{d_s+1}}; \qquad (11)$$

∘ make vertex v_3 child of node $p(u)$ and delete the vertices of $\mathrm{ch}(u)$ and node u from the tree $T(G)$.

We next describe an algorithm which computes the number of spanning trees $\tau(G)$ of a P_4-reducible graph G; it works as follows: first it computes the degree $d(v_i)$ of each vertex $v_i \in V(G)$ and assigns $s(v_i) := d(v_i)$, $1 \leq i \leq n$. Then, it computes the graph $G := G - v_n$, where $v_n \in V(G)$, and constructs its modular decomposition tree $T := T(G)$; the resulting graph G has $n - 1$ vertices. Recall that, v_i is a leaf of $T(G)$, $1 \leq i \leq n-1$. Next, it repeatedly applies the functions Replace_Parallel-Series(u, T) to each contracted 0(1)-node u, and Replace_Neighborhood(u, T) to each contracted 2-node u, by a bottom-up traversal of T. With this process, it finds the s-labels $s(v_1), s(v_2), \ldots, s(v_{n-1})$ of the vertices of T. Finally, it computes the number of spanning trees $\tau(G) := \prod_{i=1}^{n-1} s(v_i)$.

Theorem 2.1. *The number of spanning trees of a P_4-reducible graph G on n vertices and m edges can be computed in $O(n + m)$ time and space.*

References

[1] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, London, 1974.

[2] V. Giakoumakis and J-M. Vanherpe, On extended P_4-reducible and P_4-sparse graphs, *Theor. Comput. Sci.* **180** (1997) 269–286.

[3] B. Jamison and S. Olariu, P_4-reducible graphs a class of tree representable graphs, *Studies Appl. Math.* **81** (1989) 79–87.

[4] R.M. McConnell and J. Spinrad, Modular decomposition and transitive orientation, *Discrete Math.* **201** (1999) 189–241.

Stavros D. Nikolopoulos and Charis Papadopoulos
Dept. of Computer Science, University of Ioannina, GR-45110 Ioannina, Greece
stavros@cs.uoi.gr