Theoretical Computer Science 526 (2014) 41–57

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Counting spanning trees using modular decomposition ☆

Stavros D. Nikolopoulos [a,*], Leonidas Palios [a], Charis Papadopoulos [b]

[a] Department of Computer Science and Engineering, University of Ioannina, Greece
[b] Department of Mathematics, University of Ioannina, Greece

ARTICLE INFO

Article history:
Received 11 October 2011
Received in revised form 4 July 2013
Accepted 14 January 2014
Communicated by J. Kratochvil

Keywords:
Number of spanning trees
Modular decomposition
Kirchhoff matrix tree theorem
Linear-time algorithms
$(q, q-4)$-graph
P_4-tidy graph
Tree-cograph

ABSTRACT

In this paper we present an algorithm for determining the number of spanning trees of a graph G which takes advantage of the structure of the modular decomposition tree of G. Specifically, our algorithm works by contracting the modular decomposition tree of the input graph G in a bottom-up fashion until it becomes a single node; then, the number of spanning trees of G is computed as the product of a collection of values which are associated with the vertices of G and are updated during the contraction process. In particular, when applied on a $(q, q-4)$-graph for fixed q, a P_4-tidy graph, or a tree-cograph, our algorithm computes the number of its spanning trees in time linear in the size of the graph, where the complexity of arithmetic operations is measured under the uniform-cost criterion. Therefore we give the first linear-time algorithm for the counting problem in the considered graph classes.

1. Introduction

A *spanning tree* of a connected undirected graph G on n vertices is a connected $(n-1)$-edge subgraph of G. The number of spanning trees of a graph G, also called the *complexity* of G, is an important, well-studied quantity in graph theory, and appears in a number of applications. Most notable application fields are network reliability [27], computing the total resistance along an edge in an electrical network [7], enumerating certain chemical isomers [10], and counting the number of Eulerian circuits in a graph [21]. In particular, counting spanning trees is an essential step in many methods for computing, bounding, and approximating network reliability [11]; in a network modeled by a graph, intercommunication between all nodes of the network implies that the graph must contain a spanning tree and, thus, maximizing the number of spanning trees is a way of maximizing reliability.

Thus, both for theoretical and for practical purposes, we are interested in deriving a formula for computing the number of spanning trees of a graph G, and also of the K_n-complement of G (for any subgraph H of the complete graph K_n, the *K_n-complement* of H, denoted by $K_n - H$, is defined as the graph obtained from K_n by removing the edges of H; note that, if H has n vertices, then $K_n - H$ coincides with the complement $\overline{H}$ of H). Many cases have been examined depending on the choice of G, such as when G is a labeled molecular graph [10], a circulant graph [1,37], a complete multipartite graph [36], a multi-star related graph [30], a quasi-threshold graph [2,28]; see Berge [5] for an exposition of the main results.

The purpose of this paper is to study the general problem of finding the number of spanning trees of an input graph. Traditionally, the number of spanning trees of a graph is computed by means of the classic *Kirchhoff matrix tree theorem* [21], which expresses the number of spanning trees of a graph G in terms of the determinant of a cofactor of the so-called

☆ A preliminary version of this work was presented at WALCOM 2011.

* Corresponding author.
E-mail addresses: stavros@cs.uoi.gr (S.D. Nikolopoulos), palios@cs.uoi.gr (L. Palios), charis@cs.uoi.gr (C. Papadopoulos).

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2014.01.012

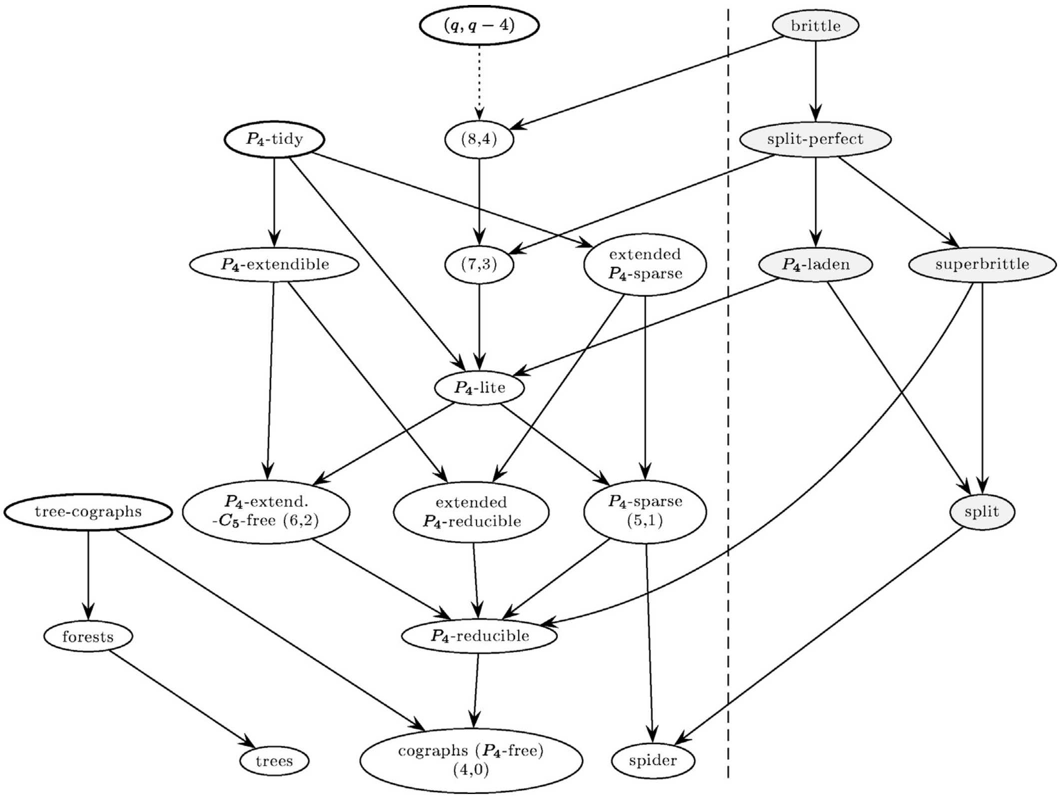

Fig. 1. A Hasse diagram of class inclusions. For the classes to the left of the dashed line, the number of spanning trees can be computed in linear time.

Kirchhoff matrix that can be easily constructed from the adjacency relation (adjacency matrix, adjacency lists, etc.) of G. Thus, counting spanning trees reduces to evaluating the determinant of an $((n-1)\times(n-1))$-size matrix, where n is the number of vertices of the input graph. This approach has been used for computing the number of spanning trees of families of graphs (see [2,5,19,30,36]), but it requires $\Theta(n^{2.376})$ arithmetic operations on matrix entries and $\Theta(n^2)$ space [13]; in fact, the algorithm that achieves this time and space complexity appears to have so large a constant factor that for practical combinatorial computations the naive $O(n^3)$-time algorithm turns out to be the sensible choice. We also mention that in some special classes of graphs, the determinant can be computed in $O(n^{1.5})$ time, using the planar separator theorem [23]. In a few cases, the number of spanning trees of a graph has been computed without the evaluation of a determinant. Colbourn et al. in [12] have proposed an algorithm which runs in $O(n^2)$ time for an n-vertex planar graph. Their algorithm is based on some particular transformations (known as the *delta-wye* technique) that can be applied on planar graphs; unfortunately, it is hard to study such or other kinds of transformations on general graphs (besides planar graphs).

In order to obtain an efficient solution for this problem, we take advantage of the modular decomposition (a form of graph decomposition which associates the graph with its maximal homogeneous sets) of the input graph G and especially the properties of its modular decomposition tree. The usage of modular decomposition has been proposed for solving several optimization problems (see the surveys [20,26] on modular decomposition). To name a few of them we refer to the problem of computing the treewidth and the minimum fill-in [6]. Also, it has been proposed to obtain efficient algorithms by expressing optimization problems in monadic second-order logic [14]. Other application areas of modular decomposition arise in graph drawing [32] and in biological strategies of proteins [17].

Our algorithm uses the modular decomposition and relies on tree contraction operations which are applied in a systematic fashion from bottom to top in order to shrink the modular decomposition tree of the graph G into a single node, while at the same time certain parameters are appropriately updated; the updating essentially implements transformations on a cofactor of the Kirchhoff matrix towards evaluating its determinant, yet the fact that we are dealing with modules (that is, any vertex outside the module is adjacent to either all or none of the vertices in the module) allows us to beat the $O(n^{2.376})$ time complexity for many classes of graphs. In the end, the number of spanning trees of G is obtained as the product of n numbers, where n is the number of vertices of G; this multiplication takes $O(n)$ time under the uniform-cost criterion [31]. Our algorithm is easy to implement; its correctness is established by means of the Kirchhoff matrix tree theorem along with standard techniques from linear algebra and matrix theory.

In particular, for certain classes of graphs, the structure of their modular decomposition trees (and in fact their prime graphs) ensures that each tree node can be processed in time linear in the size of the contracted part of the tree; thus, since the modular decomposition tree of a graph can be constructed in time and space linear in the size of the graph [15,25,34],