COUNTING SPANNING TREES IN GRAPHS USING MODULAR DECOMPOSITION

S.D. Nikolopoulos, L. Palios, Ch. Papadopoulos

9 – 2004

Preprint, no 9 – 7 / 2004

Department of Computer Science
University of Ioannina
45110 Ioannina, Greece

Counting Spanning Trees in Graphs using Modular Decomposition

Stavros D. Nikolopoulos Leonidas Palios Charis Papadopoulos

Department of Computer Science, University of Ioannina
P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, palios, charis}@cs.uoi.gr

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, or a P_4-tidy graph, 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. This implies that the problem has linear-time solution for many well-known classes of graphs, such as, cographs, P_4-sparse graphs, P_4-reducible graphs, P_4-lite graphs, and P_4-extendible graphs. The correctness of the algorithm is established through the Kirchhoff matrix tree theorem, and also relies on structural and algorithmic properties of the graphs in consideration. Our results generalize previous results and extend the family of graphs admitting linear-time algorithms for the number of their spanning trees.

Keywords: number of spanning trees, combinatorial method, modular decomposition, $(q, q-4)$-graph, P_4-tidy graph, algorithm.

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 (network) G, also called the *complexity* of G [5], is an important, well-studied quantity in graph theory, and appears in a number of applications. Most notable application fields are network reliability (in a network modeled by a graph, intercommunication between all nodes of the network implies that the graph must contain a spanning tree; thus, maximizing the number of spanning trees is a way of maximizing reliability) [20, 24], computing the total resistance along an edge in an electrical network [6], enumerating certain chemical isomers [10], and counting the number of Eulerian circuits in a graph [16].

Thus, both for theoretical and for practical purposes, we are interested in deriving a formula for or 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 [28, 29], a complete multipartite graph [26], a cubic cycle and quadruple cycle graph [27], a quasi-threshold graph [21] (see Berge [4] for an exposition of the main results; also see [12, 21, 22, 26]).

The purpose of this paper is to study the general problem of finding the number of spanning trees of an input graph and additionally to concentrate in two large classes of graphs and their subclasses: (i) the class of $(q, q-4)$-*graphs*, defined as the graphs for which every set of at most q vertices induces no more than $q-4$ different chordless paths on four vertices [3], and (ii) the class of P_4-*tidy* graphs, defined as the graphs for which every chordless path on four vertices has at most one partner [14]. By resolving the problem for these classes of graphs, we also resolve it for their subclasses, such as, the cographs, the P_4-reducible, the extended P_4-reducible, the P_4-sparse, the P_4-lite, the P_4-extendible, and the extended P_4-sparse graphs. The number of spanning trees of a graph can be computed by means of the classic *Kirchhoff matrix tree theorem* [16], which expresses the number of spanning trees of a graph G in terms of the determinant of a submatrix of the so-called Kirchhoff Matrix that can be easily constructed from the adjacency relation (adjacency matrix, adjacency lists, etc) of G. This approach has been used for computing the number of spanning trees of families of graphs (see [4, 12, 15, 22, 26]), but it necessitates $\Theta(n^3)$ time and $\Theta(n^2)$ space.

In order to obtain an efficient solution for this problem, we take advantage of the modular decomposition of the input graph and especially the properties of its modular decomposition tree. Our algorithm relies on tree contraction operations which are applied in a systematic fashion from bottom to top so as to shrink the modular decomposition tree of the input graph G into a single node, while at the same time certain parameters are appropriately updated. 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. In particular, for the classes of P_4-tidy and $(q, q-4)$-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 [11, 19], the processing of the entire modular decomposition tree takes time and space linear in the size of the input graph. Finally, the multiplication of the n numbers takes $O(n)$ time under the uniform-cost criterion [1, 23]. Thus, the number of spanning trees of a P_4-tidy or $(q, q-4)$-graph, for fixed q, can be computed in time and space linear in the size of the graph. We also note that other types of prime graphs can also be handled in the same way, so that the number of spanning trees of other classes of graphs (e.g., the semi-P_4-sparse and the (P_5,diamond)-free graphs) can also be computed in linear time.

2 Definitions and Background Results

We consider finite undirected graphs with no loops or multiple edges. For a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively. Let S be a subset of the vertex set of a graph G. Then, the subgraph of G induced by S is denoted by $G[S]$. Moreover, we denote by $G-S$ the subgraph $G[V(G)-S]$ and by $G-v$ the graph $G[V(G)-\{v\}]$. A *clique* is a set of pairwise adjacent vertices; a *stable set* is a set of pairwise non-adjacent vertices.

The *neighborhood* $N(x)$ of a vertex x of the graph G, also denoted as $N_G(x)$, is the set of all the vertices of G which are adjacent to x. The *closed neighborhood* of x is defined as $N[x] = N(x) \cup \{x\}$. The *degree* of a vertex x in the graph G, denoted $d(x)$, is the number of edges incident on x; thus, $d(x) = |N(x)|$. If two vertices x and y are adjacent in G, we say that x *sees* y; otherwise we say that x *misses* y. We extend this notion to vertex sets: $V_i \subseteq V(G)$ sees (misses) $V_j \subseteq V(G)$ if and only if every vertex $x \in V_i$ sees (misses) every vertex $y \in V_j$.

A *path* in a graph G is a sequence of vertices $v_0 v_1 \cdots v_k$ such that $v_{i-1}v_i \in E(G)$ for $i = 1, 2, \ldots, k$. A path is called *simple* if none of its vertices occurs more than once. A path (simple path) $v_0 v_1 \cdots v_k$ is a

cycle (*simple cycle*) if $v_0v_k \in E(G)$. A simple path (cycle) $v_0v_1 \cdots v_k$ is *chordless* if $v_iv_j \notin E(G)$ for any two non-consecutive vertices v_i, v_j in the path (cycle). Throughout the paper, the chordless path (cycle) on k vertices is denoted by P_k (respectively C_k). In particular, a chordless path on 4 vertices is denoted by P_4; in a P_4 *abcd*, the vertices b, c are the *midpoints* and the vertices a, d are the *endpoints* of the P_4.

2.1 Spider graphs

A graph is called a *spider* if its vertex set admits a partition into sets S, K, and R such that:

(S1) $|S| = |K| \geq 2$, S is a stable set, and K is a clique;

(S2) the vertex set R sees K and misses S;

(S3) there exists a bijection $f : S \longrightarrow K$ such that exactly one of the following statements holds:
- (i) for each vertex $v \in S$, $N(v) \cap K = \{f(v)\}$;
- (ii) for each vertex $v \in S$, $N(v) \cap K = K - \{f(v)\}$.

The triple (S, K, R) is called the *spider partition*. A graph G is a *prime spider* if G is a spider with vertex partition (S, K, R) and $|R| \leq 1$. Moreover, in order that cases (i) and (ii) in condition S3 are distinguished, they are referred to as the *thin spider* and the *thick spider*, respectively.

2.2 Modular Decomposition and Contractible Subtrees

A subset M of vertices of a graph G is said to be a *module* of G, if every vertex outside M is either adjacent to all the vertices in M or to none of them. The emptyset, the singletons, and the vertex set V are *trivial* modules and whenever G has only trivial modules it is called a *prime graph* (or *indecomposable*). A non-trivial module is also called a *homogeneous* set. A prime spider is a prime graph, since it does not contain any non-trivial module. Furthermore, a module M of G is called *strong*, if for any module $M' \neq M$ of G, either $M' \cap M = \emptyset$ or $M' \subset M$.

The modular decomposition of a graph G is represented by a tree $T(G)$ which we call the *modular decomposition tree* of G; the leaves of $T(G)$ are the vertices of G, whereas each internal node t corresponds to a strong module, denoted M_t, which is induced by the set of vertices/leaves of the subtree rooted at t. Thus, $T(G)$ represents all the strong modules of G. Each internal node is labeled by either P for *parallel* module, S for *series* module, or N for *neighborhood* module. The module corresponding to a P-node induces a disconnected subgraph of G, that of an S-node induces a connected subgraph of G whose complement is a disconnected subgraph and that of an N-node induces a connected subgraph of G whose complement is also a connected subgraph. Figure 1 shows a graph and its modular decomposition tree.

In particular, let t be an internal node of the modular decomposition tree $T(G)$. If t has children $u_1, u_2, \ldots, u_p$, then we define the *representative graph* G_t of the module M_t as follows:

- $V(G_t) = \{u_1, u_2, \ldots, u_p\}$, and
- $E(G_t) = \{(u_i, u_j) \mid (v_i, v_j) \in E(G), v_i \in M_{u_i} \text{ and } v_j \in M_{u_j}\}$.

Note that by the definition of a module, if a vertex of M_{u_i} is adjacent to a vertex of M_{u_j} then every vertex of M_{u_i} is adjacent to every vertex of M_{u_j}. Thus G_t is isomorphic to the graph induced by a subset of M_t consisting of a single vertex from each maximal submodule of M_t in $T(G)$. Then: (i) if t is a P-node, G_t is an edgeless graph, (ii) if t is an S-node, G_t is a complete graph, and (iii) if t is an N-node, G_t is a prime graph.

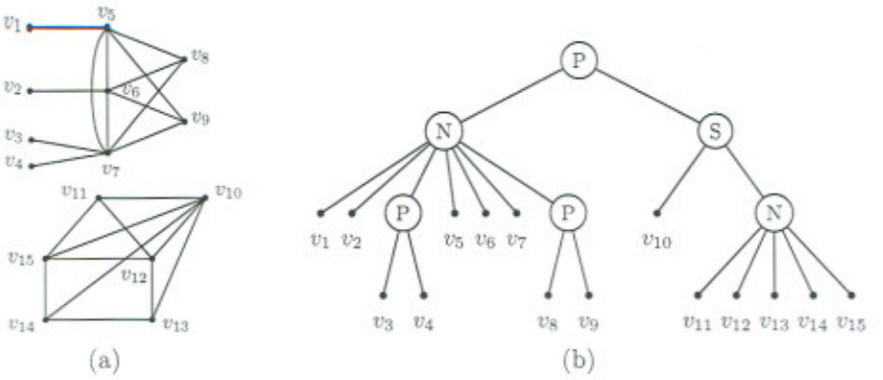

Figure 1: (a) a graph and (b) its modular decomposition tree.

The modular decomposition tree $T(G)$ of a graph G is constructed recursively as follows: parallel modules are decomposed into their connected components, series modules into their co-connected components, and neighborhood modules into their strong submodules. It is well known that for any graph G the tree $T(G)$ is unique up to isomorphism and it can be constructed in linear time [11, 19]. Note that if the tree $T(G)$ does not contain any internal N-node then G is a cograph (P_4-free) and $T(G)$ is its cotree (the P-nodes and S-nodes of the $T(G)$ are precisely the 0-nodes and 1-nodes, respectively, of the cotree).

Next, we introduce the definitions of the non-spider cost of a graph and of a contractible subtree of the modular decomposition tree which we will need in our algorithm.

Definition 2.1. Let G be a graph, $T(G)$ be its modular decomposition tree, and let $\alpha(G) = \{t_1, t_2, \ldots, t_{s'}\}$ be the set of the N-nodes of $T(G)$ such that the representative graphs $G_{t_1}, G_{t_2}, \ldots, G_{t_{s'}}$ are not spiders. We define the *non-spider cost* of G as the value $\phi(G) = \sum_{t\in\alpha(G)} |V(G_t)|^3 = \sum_{t\in\alpha(G)} |\mathrm{ch}(t)|^3$, where $\mathrm{ch}(t)$ denotes the set of children of node t in $T(G)$.

It is not difficult to see that for any n-vertex graph G, we have $\phi(G) = O(n^3)$; for an n-vertex cograph G we have $\phi(G) = 0$.

Definition 2.2. Let $T(G)$ be the modular decomposition tree of a graph G. A subtree rooted at an internal node t of $T(G)$ is a *contractible subtree* iff all the children of t are leaves of $T(G)$.

Based on the structural properties of the modular decomposition tree $T(G)$ of a graph G, it is easy to see that G has at least one contractible subtree.

2.3 Kirchhoff Matrix

For an $n \times n$ matrix M, the $(n-1)$-st order *minor* μ_j^i is the determinant of the $(n-1)\times(n-1)$ matrix obtained from M after having deleted row i and column j. The i-th *cofactor* equals μ_i^i. For an undirected graph G on n vertices, let A be its adjacency matrix and D be its degree matrix, i.e., the diagonal matrix with the degrees of the vertices of G in its main diagonal. The *Kirchhoff matrix* K for the graph G is the matrix $D - A$. The Kirchhoff 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 G's Kirchhoff matrix.

Theorem 2.1. (Kirchhoff Matrix Tree Theorem [16]): *For any graph G with matrix K defined as above, the cofactors of K have the same value, and this value equals the number of spanning trees of G.*

3 The Algorithm

In order to compute the number of spanning trees of a graph G on vertices $v_1, v_2, \ldots, v_n$, we make use of Theorem 2.1: we delete an arbitrary vertex $v_n \in V(G)$ and all the edges incident on v_n, we associate each of the remaining vertices with an *s-value* which is initialized to the vertex's degree in G, and we construct the modular decomposition tree of the graph $G - v_n$; next, in a bottom-up fashion, we process each of the contractible subtrees of the tree and we contract it into the leaf-node corresponding to its highest-index vertex/leaf, while at the same time updating the s-values of its vertices/leaves; eventually, the entire tree becomes a single vertex/leaf, and the number of spanning trees of G is then equal to the product of the final values of the s-values of all the vertices in $V(G) - \{v_n\}$. The algorithm is given in detail below; the input graph is assumed to be connected (otherwise it has no spanning trees). Furthermore, we note that the s-values are global variables.

Spanning_Trees-Number

Input: A connected graph G on n vertices $v_1, v_2, \ldots, v_n$ and m edges.

Output: The number of spanning trees $\tau(G)$ of the graph G.

1. **for** every vertex v_i, $1 \leq i \leq n-1$ **do**
   compute its degree $d(v_i)$ in G;
   $s(v_i) \leftarrow d(v_i)$;
2. Construct the modular decomposition tree T of the graph $G - v_n$;
3. Compute the node sets $L_0, L_1, \ldots, L_h$ of the levels $0, 1, \ldots, h$ of T;
4. **for** $i = h - 1$ down to 0 **do**
   for every internal node $t \in L_i$ **do**
   4.1 **if** t is a P-node **or** an S-node {*the subtree rooted at t is contractible*}
   4.2 **then** $T \leftarrow$ *Contract-Parallel_Series*(t, T);
   4.3 **else** $T \leftarrow$ *Contract-N_node*(t, T);
5. $\tau(G) \leftarrow \prod_{i=1}^{n-1} s(v_i)$;

Algorithm 1: Algorithm *Spanning_Trees-Number*.

Contract-Parallel_Series(t, T)

1. $r \leftarrow 0$;
   if t is an S-node
   then {*let* $v_1, v_2, \ldots, v_p$ *be the vertices associated with the children of node* t}
   for every vertex v_i, $1 \leq i \leq p$ **do**
   $s(v_i) \leftarrow s(v_i) + 1$;
   $r \leftarrow 1$;
2. $\alpha \leftarrow \sum_{i=1}^{p} \frac{1}{s(v_i)}$;
3. Update the s-values of vertices v_{p-1} and v_p as follows:
   $s(v_{p-1}) \leftarrow s(v_{p-1}) \cdot s(v_p) \cdot \alpha$; $\quad s(v_p) \leftarrow \frac{1}{\alpha} - r$;
4. Replace the subtree rooted at node t by the leaf-node associated with vertex v_p;
   return the resulting tree;

Algorithm 2: Function *Contract-Parallel_Series*(t, T).

3.1 Processing N-nodes (Step 4.3)

Let t be an N-node which is the root of a contractible subtree. We can show that if the (prime) graph G_t belongs to a family $\mathcal{F}$ of graphs then its processing takes time linear in the size of G_t; this implies that if all prime graphs of a graph G belong to $\mathcal{F}$ then the number of spanning trees of G can be computed in time linear in the size of G. It can be shown that such a family $\mathcal{F}$ contains spiders, trees, chordless cycles, and their complements; we call these graphs *basic prime graphs*. Due to lack of space, we will restrict our attention to spiders; yet, this suffices to show that the number of spanning trees of P_4-tidy graphs, of $(q, q-4)$-graphs (for fixed q), and of their subclasses can be computed in linear time. Then, the function Contract-N_node is as follows:

Contract-N_node(t, T)

if the representative graph G_t is a (prime) spider
then $T \leftarrow$ *Contract-Spider-N_node*(t, T);
else $T \leftarrow$ *Contract-NonBasic-N_node*(t, T);

Algorithm 3: Function *Contract-N_node*(t, T).

Contract-NonBasic-N_node(t, T)

1. Construct the $p \times p$ Kirchhoff matrix B of the graph G_t, where $v_1, v_2, \ldots, v_p$ are the vertices/children of node t;
   for $i = 1, 2, \ldots, p$ **do**
   $B[i, i] \leftarrow s(v_i)$;
   $c_i \leftarrow -1$;
2. **for** $i = 1, 2, \ldots, p-1$ **do** {*Gauss-Jordan elimination*}
   2.1 **for** $j = i+1, i+2, \ldots, p$ **do**
   $r \leftarrow B[j, i]/B[i, i]$;
   $c_j \leftarrow c_j - r \cdot c_i$;
   for $k = i, i+1, \ldots, p$ **do**
   $B[j, k] \leftarrow B[j, k] - r \cdot B[i, k]$;
   2.2 repeat step 2.1 with the row- and column-indices of array $B[\,,\,]$ exchanged, in order to complete the Gauss-Jordan elimination;
3. $\beta \leftarrow \sum_{i=1}^{p} \frac{c_i^2}{B[i, i]}$;
4. Update the s-values of vertices $v_1, v_2, \ldots, v_p$ as follows:
   for $i = 1, 2, \ldots, p-2$ **do**
   $s(v_i) \leftarrow B[i, i]$;
   $s(v_{p-1}) \leftarrow B[p-1, p-1] \cdot B[p, p] \cdot \beta$; $s(v_p) \leftarrow 1/\beta$;
5. Replace the subtree rooted at node t by the leaf-node associated with vertex v_p; return the resulting tree;

Algorithm 4: Function *Contract-NonBasic-N_node*(t, T).

Contract-Spider-N_node(t, T)

1. Compute the sets S, K, and R of the graph G_t and let $S = \{v_1, v_2, \ldots, v_k\}$ and $K = \{v_{k+1}, \ldots, v_{2k}\}$; (note that if $R \neq \emptyset$ then R contains only one vertex, i.e., $R = \{v_{2k+1}\}$)

2. Update the s-value of the vertex $v_k \in S$ as follows:
   $s(v_k) \leftarrow s(v_k) - r_1 \cdot (\lambda_1 + 1 - \delta) - \lambda_2 + 1$,
   after having computed the values of parameters $r_1, \lambda_1, \lambda_2$, and δ according to Eq. (1)-(3);

3. Update the s-values of the vertices in K as follows:
   for every vertex $v_i \in K - \{v_{2k}\}$ **do**
   $s(v_i) \leftarrow s(v_i) + \frac{s(v_{i-k})-1}{s(v_{i-k})}$;
   $s(v_{2k}) \leftarrow s(v_{2k}) - k + 1 + \mu_2 + \sigma_1 \cdot \frac{\lambda_2 - 1 - r_1 \cdot (1-\delta)}{s(v_k)}$,
   after having computed the values of parameters μ_2 and σ_1 according to Eq. (2) and (4);

4. $\gamma \leftarrow k - \mu_3 - \sigma_1 \cdot \frac{\lambda_3 - \delta - 1}{s(v_k)}$,
   after having computed the values of parameters μ_3 and λ_3 according to Eq. (2);

5. **if** $R \neq \emptyset$
   then $\{R$ *contains only one vertex, i.e.,* $R = \{v_{2k+1}\}\}$
   update the s-values of $v_{2k+1} \in R$ and $v_{2k} \in K$ as follows:
   $s(v_{2k+1}) \leftarrow s(v_{2k+1}) + \frac{s(v_k) + \lambda_2 - \lambda_1 + r_1 \cdot (1-\delta)}{s(v_k)}$;
   $s(v_{2k}) \leftarrow s(v_{2k}) - \sigma_2 \cdot \frac{s(v_k) + \lambda_2 + r_1 \cdot (1-\delta)}{s(v_k) \cdot s(v_{2k+1})}$;
   $\gamma \leftarrow \gamma + \sigma_2 \cdot \frac{s(v_k) + \lambda_3 - \delta - 1}{s(v_k) \cdot s(v_{2k+1})}$,
   after having computed the value of the parameter σ_2 according to Eq. (5);

6. Update the s-values of the vertices $v_k \in S$ and $v_{2k} \in K$ as follows:
   $s(v_k) \leftarrow s(v_k) \cdot \gamma$; $\quad s(v_{2k}) \leftarrow s(v_{2k})/\gamma$;

7. Replace the subtree rooted at node t by the leaf-node associated with vertex v_{2k};
   return the resulting tree;

Algorithm 5: Function *Contract-Spider-N_node*(t, T).

The values of the parameters $r_1, r_2, \lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2, \mu_3, \delta, \sigma_1, \sigma_2$ are given by the following formulae:

$$r_1 = \begin{cases} 0 & \text{if } G_t \text{ is a thin spider} \\ 1 & \text{otherwise} \end{cases} \qquad r_2 = 1 - 2 \cdot r_1 \tag{1}$$

$$\lambda_1 = r_2 \sum_{i=1}^{k-1} \frac{s(v_i) - s(v_k)}{s(v_i) \cdot (s(v_{k+i}) + \frac{s(v_i)-1}{s(v_i)})} \qquad \mu_1 = \sum_{i=k+1}^{2k-1} \frac{s(v_i) - s(v_{2k})}{s(v_i) + \frac{s(v_{i-k})-1}{s(v_{i-k})}}$$

$$\lambda_2 = r_2 \sum_{i=1}^{k-1} \frac{(s(v_i) - s(v_k)) \cdot (s(v_i) + r_1 r_2)}{s(v_i)^2 \cdot (s(v_{k+i}) + \frac{s(v_i)-1}{s(v_i)})} \qquad \mu_2 = \sum_{i=k+1}^{2k-1} \frac{(s(v_i) - s(v_{2k})) \cdot (s(v_{i-k}) + r_1 r_2)}{s(v_{i-k}) \cdot (s(v_i) + \frac{s(v_{i-k})-1}{s(v_{i-k})})} \tag{2}$$

$$\lambda_3 = r_2 \sum_{i=1}^{k-1} \frac{(s(v_i) - s(v_k)) \cdot (s(v_i) + r_2)}{s(v_i)^2 \cdot (s(v_{k+i}) + \frac{s(v_i)-1}{s(v_i)})} \qquad \mu_3 = \sum_{i=k+1}^{2k-1} \frac{(s(v_i) - s(v_{2k})) \cdot (s(v_{i-k}) + r_2)}{s(v_{i-k}) \cdot (s(v_i) + \frac{s(v_{i-k})-1}{s(v_{i-k})})}$$

$$\delta = s(v_k) \cdot \sum_{i=1}^{k-1} \frac{1}{s(v_i)} \tag{3}$$

$$\sigma_1 = s(v_{2k}) + \mu_2 - k + 1 - r_1 \cdot (\mu_1 - k) + r_2 \tag{4}$$

$$\sigma_2 = s(v_{2k}) + \mu_2 - \mu_1 + 1 + \sigma_1 \cdot \frac{\lambda_2 - \lambda_1 + r_1 \cdot (1 - \delta) + 1}{s(v_k)} \tag{5}$$

3.2 Correctness of the Algorithm

The correctness of the algorithm Spanning_Trees-Number follows from the Kirchhoff matrix tree theorem; we show that the value returned by the algorithm Spanning_Trees-Number is equal to the cofactor μ_n^n of the Kirchhoff matrix of the input graph. The advantage of our approach over computing the determinant of the cofactor is that we save time (and space) by processing modules, since any vertex outside the module either sees all the vertices of the module or none of them. We have the following lemma.

Lemma 3.1. *The algorithm Spanning_Trees-Number correctly computes the number of spanning trees of the input graph.*

Proof. The Kirchhoff matrix tree theorem (Theorem 2.1) implies that the number of spanning trees of the graph G is equal to any of the cofactors of the Kirchhoff matrix K defined in Section 2.3, or equivalently to the determinant of the $(n-1) \times (n-1)$ submatrix M of K formed by the first $n-1$ rows and the first $n-1$ columns. We need to consider the processing of contractible subtrees rooted at a P-node, an S-node, and an N-node (a spider or a non-basic N-node).

Processing a P-node. Let t be a P-node which is the root of a contractible subtree and assume without loss of generality that the vertices/children of t are $v_1, v_2, \ldots, v_p$. In this case, the matrix M is:

$$M = \left[\begin{array}{ccc:ccc}
s(v_1) & & 0 & (-1)_{1,p+1} & & (-1)_{1,n-1} \\
& \ddots & & \vdots & (-1)_{i,j} & \vdots \\
0 & & s(v_p) & (-1)_{p,p+1} & & (-1)_{p,n-1} \\
\hdashline
(-1)_{p+1,1} & \cdots & (-1)_{p+1,p} & s(v_{p+1}) & & \\
& & & & \ddots & (-1)_{i',j'} \\
& (-1)_{j,i} & & & s(v_j) & \\
& & & (-1)_{j',i'} & & \ddots \\
(-1)_{n-1,1} & \cdots & (-1)_{n-1,p} & & & s(v_{n-1})
\end{array}\right],$$

where $s(v_i)$ is the degree of vertex v_i in G, the off-diagonal elements in the first p rows and p columns are equal to 0 since vertices $v_1, v_2, \ldots, v_p$ are not adjacent in G, and the entries $(-1)_{i,j}$ of the off-diagonal elements are both -1 if the vertices v_i, v_j are adjacent in G, and are equal to 0 otherwise. If $p = n-1$, then, by the Kirchhoff matrix tree theorem, the number of spanning trees of G is equal to the determinant $det(M)$ of matrix M; this is equal to the product $s(v_1) \cdot s(v_2) \cdot \ldots \cdot s(v_p)$, which is precisely what Algorithm Spanning_Trees-Number in conjunction with the function Contract-Parallel_Series computes; since we process a P-node, then $r = 0$. Let us now consider the case where $p < n-1$. Then, because the vertices $v_1, v_2, \ldots, v_p$ induce a module of G, we observe that the rows of M's submatrix formed by rows $1, 2, \ldots, p$ and columns $p+1, p+2, \ldots, n-1$ are identical; similarly, the columns of the submatrix formed by rows $p+1, p+2, \ldots, n-1$ and columns $1, 2, \ldots, p$ are identical.

In order to compute the determinant $\det(M)$, we zero the off-diagonal entries in the first $p-1$ rows, which we do as follows: We first multiply row p of M by -1 and add it to rows $1, 2, \ldots, p-1$. Then, in the first $p-1$ rows of M, non-zero entries are found only in positions (i,i) and (i,p), $1 \leq i \leq p-1$, and have values $s(v_i)$ and $-s(v_p)$, respectively. Next, for all $1 \leq i \leq p-1$, we multiply column i by $\frac{s(v_p)}{s(v_i)}$ and add it to column p, which implies that in the first $p-1$ rows of M only the diagonal elements have non-zero values. Note that row p remains unchanged. Additionally, those of the elements in positions (i,p) of column p, for $i = p+1, p+2, \ldots, n-1$, which were equal to 0 in the original matrix M are still 0, whereas the remaining ones, which were equal to -1, now have the same value $-\left(1 + \sum_{i=1}^{p-1} \frac{s(v_p)}{s(v_i)}\right) = -s(v_p) \cdot \sum_{i=1}^{p} \frac{1}{s(v_i)}$. Finally, we divide the entries of column p by $s(v_p) \cdot \alpha$, where $\alpha = \sum_{i=1}^{p} \frac{1}{s(v_i)}$, so that they become equal to -1 (as in the original matrix M), and multiply column $(p-1)$ by $s(v_p) \cdot \alpha$ so that the value of the determinant of M does not change.

Then, in the matrix that results after these operations, we have: (i) all the off-diagonal elements in rows $1, 2, \ldots, p-1$ are equal to 0, (ii) the submatrix formed by rows $p, p+1, \ldots, n-1$ and columns $p, p+1, \ldots, n-1$ is identical to the corresponding submatrix in the original matrix M except for the (p,p)-element, and (iii) the diagonal elements in positions (i,i), $1 \leq i \leq p$, have values $s_0(v_i)$ which are equal to:

$$\begin{aligned} s_0(v_i) &= s(v_i), \qquad 1 \leq i \leq p-2 \\ s_0(v_{p-1}) &= s(v_{p-1}) \cdot s(v_p) \cdot \alpha \\ s_0(v_p) &= \frac{1}{\alpha} \\ \text{where} \qquad \alpha &= \sum_{i=1}^{p} \frac{1}{s(v_i)}. \end{aligned}$$

Thus, if we expand in terms of the first $p-1$ rows, we have that the determinant of M is

$$\begin{aligned} \det(M) &= \left(\prod_{i=1}^{p-1} s_0(v_i) \right) \cdot \begin{vmatrix} s_0(v_p) & (-1)_{p,p+1} & & & & & \\ (-1)_{p+1,p} & s(v_{p+1}) & & & & & \\ & & \ddots & & (-1)_{i',j'} & & \\ & & & s(v_j) & & & \\ & & (-1)_{j',i'} & & \ddots & & \\ & & & & & & s(v_{n-1}) \end{vmatrix} \\ &= \left(\prod_{i=1}^{p-1} s_0(v_i) \right) \cdot \det(M'), \end{aligned}$$

where M' is an $(n-p) \times (n-p)$ matrix similar to the original matrix M; in fact, it is identical to the submatrix of the original matrix M formed by rows $p, p+1, \ldots, n-1$ and columns $p, p+1, \ldots, n-1$, with the only exception that the value $s_0(v_p)$ is different from $s(v_p)$. Thus, if we assume (in an inductive fashion) that the determinant of the matrix M' can be expressed as the product of appropriate values $s'(v_p), s'(v_{p+1}), \ldots, s'(v_{n-1})$, then the determinant of the original matrix M is equal to the product of these values multiplied by the product of $s_0(v_1), s_0(v_2), \ldots, s_0(v_{p-1})$, just as the algorithm Spanning_Trees-Number does by using function Contract-Parallel_Series for $r = 0$.

Processing an S-node. Let t be an S-node which is the root of a contractible subtree and assume without loss of generality that the vertices/children of t are $v_1, v_2, \ldots, v_p$. In this case, the matrix M formed by rows $1, 2, \ldots, n-1$ and columns $1, 2, \ldots, n-1$ of the Kirchhoff matrix is:

$$M = \left[\begin{array}{ccc:ccc} s(v_1) & & -1 & (-1)_{1,p+1} & & (-1)_{1,n-1} \\ & \ddots & & \vdots & (-1)i,j & \vdots \\ -1 & & s(v_p) & (-1)_{p,p+1} & & (-1)_{p,n-1} \\ \hdashline (-1)_{p+1,1} & \cdots & (-1)_{p+1,p} & s(v_{p+1}) & & \\ & & & & \ddots & (-1)_{i',j'} \\ & (-1)_{j,i} & & & s(v_j) & \\ & & & (-1)_{j',i'} & & \ddots \\ (-1)_{n-1,1} & \cdots & (-1)_{n-1,p} & & & s(v_{n-1}) \end{array}\right]$$

where the off-diagonal elements in the first p rows and p columns of matrix M are all equal to -1 (because the vertices $v_1, v_2, \ldots, v_p$ are adjacent in G). Additionally, since the vertices $v_1, v_2, \ldots, v_p$ induce a module of G, the rows of M's submatrix formed by rows $1, 2, \ldots, p$ and columns $p+1, p+2, \ldots, n-1$ are identical and similarly the columns of M's submatrix formed by rows $p+1, p+2, \ldots, n-1$ and columns $1, 2, \ldots, p$ are identical.

In order to compute the determinant $\det(M)$, we zero the off-diagonal entries in the first $p-1$ rows. We work as follows: We first multiply row p of M by -1 and add it to rows $1, 2, \ldots, p-1$. Then, in the first $p-1$ rows of M, non-zero entries are found only in positions (i,i) and (i,p), $1 \leq i \leq p-1$, and have values $s(v_i)+1$ and $-(s(v_p)+1)$, respectively. Next, for all $1 \leq i \leq p-1$, we multiply column i by $\frac{s(v_p)+1}{s(v_i)+1}$ and add it to column p; this implies that in the first $p-1$ rows of M only the diagonal elements have non-zero values, and row p remains unchanged except for the (p,p)-element which becomes $s(v_p) - \sum_{i=1}^{p-1} \frac{s(v_p)+1}{s(v_i)+1} = (s(v_p)+1) \cdot \left(1 - \sum_{i=1}^{p} \frac{1}{s(v_i)+1}\right)$. Additionally, those of the elements in positions (i,p) of column p, for $i = p+1, p+2, \ldots, n-1$, which were equal to 0 in the original matrix M are still 0, whereas the remaining ones, which were equal to -1, now have the same value $-\left(1 + \sum_{i=1}^{p-1} \frac{s(v_p)+1}{s(v_i)+1}\right) = -(s(v_p)+1) \cdot \sum_{i=1}^{p} \frac{1}{s(v_i)+1}$. Finally, we divide the entries of column p by $(s(v_p)+1) \cdot \alpha$, where $\alpha = \sum_{i=1}^{p} \frac{1}{s(v_i)+1}$, so that they become equal to -1 (as in the original matrix M), and we multiply column $(p-1)$ by $(s(v_p)+1) \cdot \alpha$ so that the determinant of M does not change.

Then, in the matrix that results after these operations, we have: (i) all the off-diagonal elements in rows $1, 2, \ldots, p-1$ are equal to 0, (ii) the submatrix formed by rows $p, p+1, \ldots, n-1$ and columns $p, p+1, \ldots, n-1$ is identical to the corresponding submatrix in the original matrix M except for the (p,p)-element, and (iii) the diagonal elements in positions (i,i), $1 \leq i \leq p$, have values $s_1(v_i)$ which are equal to:

$$\begin{aligned} s_1(v_i) &= s(v_i)+1, \qquad 1 \leq i \leq p-2 \\ s_1(v_{p-1}) &= (s(v_{p-1})+1) \cdot (s(v_p)+1) \cdot \alpha \\ s_1(v_p) &= \frac{1}{\alpha} - 1 \end{aligned}$$

where

$$\alpha = \sum_{i=1}^{p} \frac{1}{s(v_i)+1}.$$

Thus, expanding in terms of the first $p-1$ rows, we find that the determinant of M is

$$\det(M) = \left(\prod_{i=1}^{p-1} s_1(v_i)\right) \cdot \begin{vmatrix} s_1(v_p) & (-1)_{p,p+1} & & & & \\ (-1)_{p+1,p} & s(v_{p+1}) & & & & \\ & & \ddots & & (-1)_{i',j'} & \\ & & & s(v_i) & & \\ & & (-1)_{j',i'} & & \ddots & \\ & & & & & s(v_{n-1}) \end{vmatrix}$$

$$= \left(\prod_{i=1}^{p-1} s_1(v_i)\right) \cdot \det(M'),$$

where M' is an $(n-p) \times (n-p)$ matrix which is identical to the submatrix of the original matrix M formed by rows $p, p+1, \ldots, n-1$ and columns $p, p+1, \ldots, n-1$, with the only exception that the value $s_1(v_p)$ is different from $s(v_p)$. Thus, if we assume (in an inductive fashion) that the determinant of the matrix M' can be expressed as the product of appropriate values $s'(v_p), s'(v_{p+1}), \ldots, s'(v_{n-1})$, then the determinant of the original matrix M is equal to the product of these values multiplied by the product of $s_1(v_1), s_1(v_2), \ldots, s_1(v_{p-1})$, just as the algorithm Spanning_Trees_Number does by using function Contract-Parallel_Series for $r = 1$.

Processing a spider N-node. Let t be an N-node such that the subtree rooted at t is contractible and the representative graph G_t is a (prime) spider. Then, if we assume without loss of generality that the vertices/children of t are $v_1, v_2, \ldots, v_p$, they can be partitioned into sets $S = \{v_1, v_2, \ldots, v_k\}$, $K = \{v_{k+1}, v_{k+2}, \ldots, v_{2k}\}$, and R which is either empty (in this case, $p = 2k$) or $R = \{v_{2k+1}\}$ (then, $p = 2k+1$). Let us suppose for the time being that $R \neq \emptyset$; then, the matrix M is:

$$M = \left[\begin{array}{cccc:cccc:c} s(v_1) & 0 & & 0 & r_1-1 & -r_1 & \cdots & -r_1 & 0 & \\ 0 & s(v_2) & & & -r_1 & r_1-1 & \cdots & -r_1 & 0 & \\ & & \ddots & & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & & & s(v_k) & -r_1 & -r_1 & \cdots & r_1-1 & 0 & \\ \hdashline r_1-1 & -r_1 & \cdots & -r_1 & s(v_{k+1}) & -1 & \cdots & -1 & -1 & Z_1 \\ -r_1 & r_1-1 & \cdots & -r_1 & -1 & s(v_{k+2}) & \cdots & -1 & -1 & \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ -r_1 & -r_1 & \cdots & r_1-1 & -1 & -1 & \cdots & s(v_{2k}) & -1 & \\ \hdashline 0 & 0 & \cdots & 0 & -1 & -1 & \cdots & -1 & s(v_{2k+1}) & \\ \hdashline \multicolumn{8}{c:}{Z_2} & Z \end{array}\right],$$

where the parameter r_1 is as defined in Eq. (1) so that the values $r_1 - 1$ and $-r_1$ are equal to -1 or 0 reflecting the adjacencies in a thin or thick spider, the elements of the matrices Z_1 and Z_2 are equal to -1 and 0 depending on whether the corresponding vertices are adjacent in the graph or not, and the matrix Z is an $(n-p-1) \times (n-p-1)$ submatrix of the form

$$Z = \begin{bmatrix} s(v_{p+1}) & & (-1)_{j',i} & & \\ & \ddots & & & \\ (-1)_{i,j'} & & s(v_i) & & \\ & & & \ddots & \\ & & & & s(v_{n-1}) \end{bmatrix}. \tag{6}$$

It is important to note that, because the spider is a module, the columns of the submatrix Z_1 are identical, and so are the rows of the submatrix Z_2.

In order to compute the determinant $\det(M)$ of matrix M, which will give us the number of spanning trees of the input graph G, we proceed with the following steps:

1. *We zero the off-diagonal entries in the first $k-1$ rows of matrix M:* To do that,

   - ▷ we multiply row k of the matrix M by -1 and add it to rows $1, 2, \ldots, k-1$;
   - ▷ we multiply column i, for $1 \leq i \leq k-1$, by $s(v_i)/s(v_k)$, and add it to column k;
   - ▷ we add columns j, for $k+1 \leq j \leq 2k-1$, to column $2k$.

   We note that after the above operations the $(2k,k)$-entry and the (i,k)-entries, for $p+1 \leq i \leq n-1$, of the matrix M have values $-1+(1-\delta)\cdot r_1$ and $-1-\delta$, respectively, where δ is as defined in Eq. (3). Finally,

   - ▷ we multiply column j, for $1 \leq j \leq k-1$, by $r_2/s(v_j)$, where r_2 is as defined in Eq. (1), and add it to column $k+j$, respectively.

   Then, all the off-diagonal elements of the first $k-1$ rows have zero values, whereas the values of the diagonal elements are equal to their initial s-values. More specifically, the resulting matrix M is:

$$M = \left[\begin{array}{ccccc:ccccc:c}
s(v_1) & & & 0 & & 0 & & & 0 & 0 & 0 \\
 & \ddots & & \vdots & & \vdots & & & \vdots & \vdots & \vdots \\
 & & s(v_{k-1}) & 0 & & 0 & & & 0 & 0 & 0 \\
0 & \cdots & 0 & s(v_k) & & -r_1 & \cdots & -r_1 & -kr_1 - r_2 & 0 & (-1)_{k,j} \\
\hdashline
r_1 - 1 & \cdots & -r_1 & -\frac{r_2 s(v_k)}{s(v_1)} - r_1(1+\delta) & & s(v_{k+1}) + \frac{(r_1-1)r_2}{s(v_1)} & \cdots & -1 - \frac{r_1 r_2}{s(v_{k-1})} & s(v_{k+1}) - k + 1 & -1 & \\
\vdots & \ddots & \vdots & \vdots & & \vdots & \ddots & \vdots & \vdots & \vdots & (-1)_{i',j} \\
-r_1 & \cdots & r_1 - 1 & -\frac{r_2 s(v_k)}{s(v_{k-1})} - r_1(1+\delta) & & -1 - \frac{r_1 r_2}{s(v_1)} & \cdots & s(v_{2k-1}) + \frac{(r_1-1)r_2}{s(v_{k-1})} & s(v_{2k-1}) - k + 1 & -1 & \\
-r_1 & \cdots & -r_1 & r_1(1-\delta) - 1 & & -1 - \frac{r_1 r_2}{s(v_1)} & \cdots & -1 - \frac{r_1 r_2}{s(v_{k-1})} & s(v_{2k}) - k + 1 & -1 & \\
\hdashline
0 & \cdots & 0 & 0 & & -1 & \cdots & -1 & -k & s(v_{2k+1}) & \\
\hdashline
 & (-1)_{j,i} & & (-1-\delta)_{j,k} & & (-1 - \frac{r_2}{s(v_1)})_{j,k+1} & \cdots & (-1 - \frac{r_2}{s(v_{k-1})})_{j,2k-1} & (-k)_{j,2k} & (-1)_{j,2k+1} & Z
\end{array}\right]$$

2. *We zero the off-diagonal entries in rows $k+1, k+2, \ldots, 2k-1$ of matrix M:* To do that,

   - ▷ we multiply row $2k$ by -1, and add it to rows j, for $k+1 \leq j \leq 2k-1$;
   - ▷ we multiply column j, for $k+1 \leq j \leq 2k-1$, by $-(r_2\frac{s(v_j)-s(v_k)}{s(v_j)})/(s(v_j) + \frac{s(v_{j-k})-1}{s(v_{j-k})})$, and add it to column k (then, in column k, the entries (j,k) for $k+1 \leq j \leq 2k-1$ become equal to 0, whereas the entries $(k,k), (2k,k), (2k+1,k)$ and (j,k) for $p+1 \leq j \leq n-1$, of matrix M have values $s(v_k) + \lambda_1 \cdot r_1$, $r_1 \cdot (1-\delta) + \lambda_2 - 1$, λ_1, and $\lambda_3 - \delta - 1$, respectively, where $\lambda_1, \lambda_2, \lambda_3$ are as defined in Eq. (2));
   - ▷ we multiply column j, for $k+1 \leq j \leq 2k-1$, by $-(s(v_j) - s(v_{2k}))/(s(v_j) + \frac{s(v_{j-k})-1}{s(v_{j-k})})$, and add it to column $2k$.

Then, all the off-diagonal elements of rows $k+1, k+2, \ldots, 2k-1$ have zero values; the diagonal elements are equal to $s(v_i) + \frac{s(v_{i-k})-r_2^2}{s(v_{i-k})} = s(v_i) + \frac{s(v_{i-k})-1}{s(v_{i-k})}$ (because $r_2^2 = 1$), which are the updated values of $s(v_{k+1}), s(v_{k+2}), \ldots, s(v_{2k-1})$ assigned in Step 3 of the function Contract-Spider-N_node. In particular, if μ_1, μ_2 and μ_3 are as defined in Eq. (2), the resulting matrix M is

$$M = \begin{bmatrix}
s(v_1) & & & 0 & 0 & & & 0 & 0 & 0 \\
 & \ddots & & \vdots & \vdots & & & \vdots & \vdots & \vdots \\
 & & s(v_{k-1}) & 0 & 0 & & & 0 & 0 & 0 \\
0 & \cdots & 0 & s(v_k)+\lambda_1 r_1 & -r_1 & \cdots & -r_1 & -kr_1 - r_2 + r_1\mu_1 & 0 & (-1)_{k,j} \\
\hline
-r_2 & \cdots & 0 & 0 & s(v_{k+1}) + \frac{s(v_1)-1}{s(v_1)} & \cdots & 0 & 0 & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & -r_2 & 0 & 0 & \cdots & s(v_{2k-1}) + \frac{s(v_{k-1})-1}{s(v_{k-1})} & 0 & 0 & 0 \\
-r_1 & \cdots & -r_1 & r_1(1-\delta)+\lambda_2-1 & -1-\frac{r_1 r_2}{s(v_1)} & \cdots & -1-\frac{r_1 r_2}{s(v_{k-1})} & s(v_{2k})+\mu_2-k+1 & -1 & (-1)_{2k,j} \\
\hline
0 & \cdots & 0 & \lambda_1 & -1 & \cdots & -1 & \mu_1 - k & s(v_{2k+1}) & (-1)_{2k+1,j} \\
\hline
 & (-1)_{j,i} & & (\lambda_3-\delta-1)_{j,k} & (-1-\frac{r_2}{s(v_1)})_{j,k+1} & \cdots & (-1-\frac{r_2}{s(v_{k-1})})_{j,2k-1} & (\mu_3-k)_{j,2k} & (-1)_{j,2k+1} & Z
\end{bmatrix},$$

and thus we have

$$\det(M) = \left(\prod_{i=1}^{k-1} s(v_i)\right) \cdot \left(\prod_{i=k+1}^{2k-1} (s(v_i) + \frac{s(v_{i-k})-1}{s(v_{i-k})})\right) \cdot \det(C), \tag{7}$$

where C is an $(n-2k+1) \times (n-2k+1)$ matrix of the following form:

$$C = \left[\begin{array}{ccc:ccccc}
s(v_k)+\lambda_1 r_1 & r_1(\mu_1-k)-r_2 & 0 & (-1)_{2k,p+1} & \cdots & (-1)_{2k,i} & \cdots & (-1)_{2k,n-1} \\
r_1(1-\delta)+\lambda_2-1 & s(v_{2k})+\mu_2-k+1 & -1 & (-1)_{2k,p+1} & & (-1)_{2k,i} & & (-1)_{2k,n-1} \\
\lambda_1 & \mu_1-k & s(v_{2k+1}) & (-1)_{2k,p+1} & & (-1)_{2k,i} & & (-1)_{2k,n-1} \\
\hline
(\lambda_3-\delta-1)_{p+1,2k} & (\mu_3-k)_{p+1,2k} & (-1)_{p+1,2k} & s(v_{p+1}) & & & & \\
 & & & & \ddots & & (-1)_{i',j'} & \\
(\lambda_3-\delta-1)_{i,2k} & (\mu_3-k)_{i,2k} & (-1)_{i,2k} & & & s(v_i) & & \\
 & & & & (-1)_{j',i'} & & \ddots & \\
(\lambda_3-\delta-1)_{n-1,2k} & (\mu_3-k)_{n-1,2k} & (-1)_{n-1,2k} & & & & & s(v_{n-1})
\end{array}\right]. \tag{8}$$

We note that because the spider is a module in the input graph G then, a vertex v_i, where $i > p$, sees v_{2k} if and only if it sees v_k and v_{2k+1} as well; this is why in the above description of the matrix C, the entries $(\lambda_3-\delta-1)_{i,j}$, $(\mu_3-k)_{i,j}$, and $(-1)_{i,j}$ have one subscript always equal to $2k$.

It is worth mentioning that Eq. (7) holds in the case where the set R is an empty set as well. In this case, matrix C has the same form as in Eq. (8) with the 3rd row and 3rd column removed.

In order to calculate the determinant of matrix C we work as follows: We first multiply the 2nd row of matrix C by -1 and add it to the 1st row. This has the effect that the elements in positions $(1,1)$, $(1,2)$, $(1,3)$, and the remaining positions in the 1st row become equal to $s'(v_k) = s(v_k) - r_1(\lambda_1+1-\delta) - \lambda_2 + 1$, $-(s(v_{2k})+\mu_2-k+1) + r_1 \cdot (\mu_1-k) - r_2 = -\sigma_1$, 1, and 0 respectively, where $s'(v_k)$ is the updated value of $s(v_k)$ (see Step 2 of function Contract-Spider-N_node), and σ_1 is as defined in Eq. (4). Then, we can zero the $(1,2)$-element by multiplying the 1st column by $\sigma_1/s'(v_k)$, and by adding it to the 2nd column. Next, we zero the $(1,3)$-element by multiplying the 1st column by $-1/s'(v_k)$ and by adding it to the 3rd column. After that, only the diagonal position of the 1st row of matrix C has non-zero value which is equal to $s'(v_k)$, whereas the elements in positions $(i,2)$ of the 2nd column, for $i = p+1, p+2, \ldots, n-1$, either have value 0 (if the corresponding entries in the original matrix M were equal to 0) or else they have the same value $\mu_3 - k + \frac{\sigma_1}{s'(v_k)}(\lambda_3-\delta-1)$ which is equal to $-\gamma$, where γ is the value assigned in Step 4 of the function Contract-Spider-N_node.

Now, if $R = \emptyset$, then $p = 2k$; if we divide the 2nd column of matrix C by γ and multiply the 1st column by γ, so that the value of the determinant of C does not change, we have

$$\det(M) = \left(\prod_{i=1}^{2k-1} s_2(v_i) \right) \cdot \begin{vmatrix} s_2(v_{2k}) & (-1)_{2k,p+1} & & & & & \\ (-1)_{p+1,2k} & s(v_{p+1}) & & & & & \\ & & \ddots & & (-1)_{i',j'} & & \\ & & & s(v_i) & & & \\ & & (-1)_{j',i'} & & \ddots & & \\ & & & & & & s(v_{n-1}) \end{vmatrix}$$

$$= \left(\prod_{i=1}^{2k-1} s_2(v_i) \right) \cdot \det(M'),$$

where M' is an $(n-p) \times (n-p)$ matrix similar to the original matrix M, except that in this case, the value $s_2(v_{2k})$ is equal to the value assigned in Step 6 of the function Contract-Spider-N_node. Moreover, the values $s_2(v_1), s_2(v_2), \ldots, s_2(v_{2k-1})$ are as assigned in Step 6 for $R = \emptyset$.

If $R \neq \emptyset$, then we zero the off-diagonal elements of the 3rd row of matrix C. We multiply the 2nd row by -1 and add it to the 3rd row. Then, the entry $(3,2)$ of the matrix C becomes equal to $-\sigma_2$, where σ_2 is as defined in Eq. (5). In order to make this entry equal to 0, we multiply the 3rd column by $\frac{\sigma_2}{s'(v_{2k+1})}$, where $s'(v_{2k+1}) = s(v_{2k+1}) + \frac{s'(v_k)+\lambda_2-1-r_1\delta+\lambda_1}{s'(v_k)}$, and add it to the 2nd column. As a result, the elements in the 2nd column of C corresponding to the vertices v_i, for $p+1 \leq i \leq n-1$, are either equal to 0 (if v_{2k} and v_i are non-adjacent in G) or else they have the same value $-\gamma$, where γ is the updated value assigned during Step 5 of function Contract-Spider-N_node. As these latter entries had corresponding values equal to -1 in the original matrix M, we divide the 3rd column by γ, and then in order to maintain the value of the determinant unchanged, we multiply the 1st column by γ. All of the above implies that Eq. (7) becomes

$$\det(M) = \left(\prod_{\substack{i=1 \\ i \neq 2k}}^{2k+1} s_2(v_i) \right) \cdot \begin{vmatrix} s_2(v_{2k}) & (-1)_{2k,p+1} & & & & & \\ (-1)_{p+1,2k} & s(v_{p+1}) & & & & & \\ & & \ddots & & (-1)_{i',j'} & & \\ & & & s(v_i) & & & \\ & & (-1)_{j',i'} & & \ddots & & \\ & & & & & & s(v_{n-1}) \end{vmatrix}$$

$$= \left(\prod_{\substack{i=1 \\ i \neq 2k}}^{2k+1} s_2(v_i) \right) \cdot \det(M''),$$

where M'' is an $(n-p) \times (n-p)$ matrix similar to the original matrix M, and the values $s_2(v_i)$, $1 \leq i \leq 2k+1$, are equal to the values given in function Contract-Spider-N_node.

Since both matrices M' and M'' are identical to the submatrix of the original matrix M formed by rows $p, p+1, \ldots, n-1$ and columns $p, p+1, \ldots, n-1$, except for the entry with value $s_2(v_{2k})$, then if we assume (again in an inductive fashion) that the determinants of M' and M'' can be expressed as the product of appropriate values $s(v_{2k}), s(v_{p+1}), \ldots, s(v_{n-1})$, the determinant of the original matrix M is equal to the product of these values multiplied by $\prod_{\substack{i=1 \\ i \neq 2k}}^{2k+1} s_2(v_i)$ or $\prod_{i=1}^{2k-1} s_2(v_i)$ respectively. In either case, this is precisely what the algorithm Spanning_Trees_Number computes by using the function Contract-Spider-N_node.

Processing a non-basic N-node. Let t be a non-basic N-node which is the root of a contractible subtree and assume without loss of generality that the vertices/children of t are $v_1, v_2, \ldots, v_p$. Step 2 of function Contract-NonBasic-N_node applies the Gauss-Jordan elimination in the submatrix formed by the first p rows and p columns of matrix M. Thus, with the values $B[i,i]$ and c_i, $1 \leq i \leq p$, computed in this step, we have

$$\det(M) = \left|\begin{array}{ccc:ccc} B[1,1] & & 0 & (c_1)_{1,p+1} & & (c_1)_{1,n-1} \\ & \ddots & & & (c_i)_{i,j} & \vdots \\ 0 & & B[p,p] & (c_p)_{p,p+1} & & (c_p)_{p,n-1} \\ \hdashline (c_1)_{p+1,1} & \cdots & (c_p)_{p+1,p} & s(v_{p+1}) & & \\ & & & \ddots & (-1)_{i',j'} & \\ & (c_i)_{j,i} & & & s(v_j) & \\ & & & (-1)_{j',i'} & & \ddots \\ (c_1)_{n-1,1} & \cdots & (c_p)_{n-1,p} & & & s(v_{n-1}) \end{array}\right|$$

where, according to the definition of the Kirchhoff matrix, the values $(-1)_{i,j}$ of the off-diagonal elements (i,j) are -1 if the vertices v_i and v_j are adjacent in G and are 0 otherwise, $p+1 \leq i,\ j \leq n-1$. Similarly, the values $(c_i)_{i,j}$ and $(c_i)_{j,i}$, $1 \leq i \leq p \leq j \leq n-1$ have value c_i if the vertices v_i and v_j are adjacent in G and are 0 otherwise.

In order to compute the determinant of matrix M, we zero the off-diagonal elements of its first $p-1$ rows: for each $i = 1, 2, \ldots, p-1$, we multiply row p by $-c_i/c_p$ and we add it to row i, and subsequently, for each $i = 1, 2, \ldots, p-1$, we multiply column i by $\frac{c_i \cdot B[p,p]}{c_p \cdot B[i,i]}$ and add it to column p. As a result, indeed only the diagonal elements of the first $p-1$ rows have non-zero values, which are equal to $B[i,i]$. Moreover, if β is the value defined in Step 3 of function Contract-NonBasic-N_node, all the non-zero elements in positions (i,p) of column p have now the same value β/c_p and all the non-zero elements in positions (p,i) of row p have now the same value c_p, $p+1 \leq i \leq n-1$; these elements were equal to -1 in the original matrix M. In order to make them equal to -1, we divide the entries of column p by β, and then in order to maintain the value of the determinant unchanged, we multiply column $p-1$ by β. Thus, expanding in terms of the first $p-1$ rows, we find that the determinant of M is

$$\begin{aligned} \det(M) &= \left(\prod_{i=1}^{p-1} s_3(v_i)\right) \cdot \left|\begin{array}{cc:ccc} s_3(v_p) & (-1)_{p,p+1} & & & \\ (-1)_{p+1,p} & s(v_{p+1}) & & & \\ \hdashline & & \ddots & & (-1)_{i',j'} \\ & & & s(v_j) & \\ & & (-1)_{j',i'} & & \ddots \\ & & & & s(v_{n-1}) \end{array}\right| \\ &= \left(\prod_{i=1}^{p-1} s_3(v_i)\right) \cdot \det(M'), \end{aligned}$$

where

$$\begin{aligned} s_3(v_i) &= B[i,i], \qquad 1 \leq i \leq p-2 \\ s_3(v_{p-1}) &= B[p-1,p-1] \cdot B[p,p] \cdot \beta \\ s_3(v_p) &= B[p,p] \cdot \frac{1}{c_p} \cdot \frac{c_p}{B[p,p] \cdot \beta} = \frac{1}{\beta} \end{aligned}$$

and M' is an $(n-p) \times (n-p)$ matrix which is identical to the submatrix of the original matrix M formed by rows $p, p+1, \ldots, n-1$ and columns $p, p+1, \ldots, n-1$, with the only exception that the value $s_3(v_p)$ is different from $s(v_p)$. Similarly with the other cases, the determinant of the matrix M' can be expressed (in an inductive fashion) as the product of appropriate values $s'(v_p), s'(v_{p+1}), \ldots, s'(v_{n-1})$, and then the determinant of the original matrix M is equal to the product of these values multiplied by the product of $s_3(v_1), s_3(v_2), \ldots, s_3(v_{p-1})$; this is precisely what the algorithm Spanning_Trees-Number computes in this case as well. ∎

3.3 Time Complexity

Lemma 3.2. *The algorithm Spanning_Trees-Number runs in $O(n + m + \phi(G))$ time, where n is the number of vertices, m is the number of edges, and $\phi(G)$ is the non-spider cost of the input graph G.*

Proof. Step 1 of the algorithm Spanning_Trees-Number clearly takes $O(n+m)$ time and so does the construction of the modular decomposition tree $T(G)$ of the graph G [11, 19]. The computation of the level sets $L_0, L_1, \ldots, L_{h-1}$ of the tree $T(G)$ in Step 3 can be performed in $O(n)$ time, since the tree $T(G)$ contains $O(n)$ nodes. Additionally, note that exactly one of the functions Contract-Parallel_Series, Contract-Spider-N_node and Contract-NonBasic-N_node is applied on each of the nodes of $T(G)$. When the function Contract-Parallel_Series is applied on a node t, it can be executed in $O(|\mathrm{ch}(t)|)$ time, where $|\mathrm{ch}(t)|$ is the number of children of node t in $T(G)$. When the function Contract-Spider-N_node is applied on t with representative graph a prime spider G_t, we must compute the sets S, K, and R of G_t (this can be easily done in $O(|V(G_t)| + |E(G_t)|)$ time after having computed the degrees of the vertices of the spider; note that, if a graph is a spider with partition (S, K, R), then for every choice of v, u, and r in S, K, and R respectively, $\mathrm{degree}(v) < \mathrm{degree}(r) < \mathrm{degree}(u)$ [18]), update the s-values (this takes $O(|\mathrm{ch}(t)|)$ time), and update the modular decomposition tree (this also takes $O(|\mathrm{ch}(t)|)$ time). Lastly, when the function Contract-NonBasic-N_node is applied on node t, it takes $O(|V(G_t)|^3)$ time, so that the execution of this function on all the non-spider N-nodes of the tree $T(G)$ requires a total of $O(\phi(G))$ time, where $\phi(G)$ is the non-spider cost of G.

Given that the number of nodes of the tree $T(G)$ is $O(n)$, the fact that the number of edges of G is no less than the sum of the numbers of edges of all the representative graphs of $T(G)$, and the fact that checking whether a graph H is a spider takes $O(|V(H)| + |E(H)|)$ time, then Step 4 of the algorithm Spanning_Trees-Number requires $O(n + m + \phi(G))$ time. Finally, Step 5 takes $O(n)$ time under the uniform-cost criterion, according to which each instruction requires one unit of time and each register requires one unit of space, implying that, no matter how large the numbers are, an arithmetic operation involving ℓ numbers takes $O(\ell)$ time. Therefore, the algorithm Spanning_Trees-Number takes $O(n + m + \phi(G))$ time. ∎

Remark 3.1. If a single computer word can store an integer as large as n^{n-2}, where n is the number of vertices of the input graph, then the uniform-cost criterion is certainly realistic (recall that the number of spanning trees of a graph on n vertices is at most n^{n-2}, which is achieved by the complete graph K_n). If however this is not the case, then the uniform-cost criterion is not realistic; but, in such a case, even the logarithmic-cost criterion (which takes into account the limited size of a real memory word which is logarithmic in the number stored) is somewhat unrealistic as well, since it assumes that two integers i and j can be multiplied in time $O(\log(i) + \log(j))$, which is not known to be possible (see [1, 23]). □

4 Counting Spanning Trees in Linear Time

Let G be a graph on n vertices and m edges and let $\phi(G)$ be its non-spider cost. From Lemma 3.2, it is clear that if $\phi(G)$ is linear in the size of G, then the algorithm Spanning_Trees-Number runs in linear

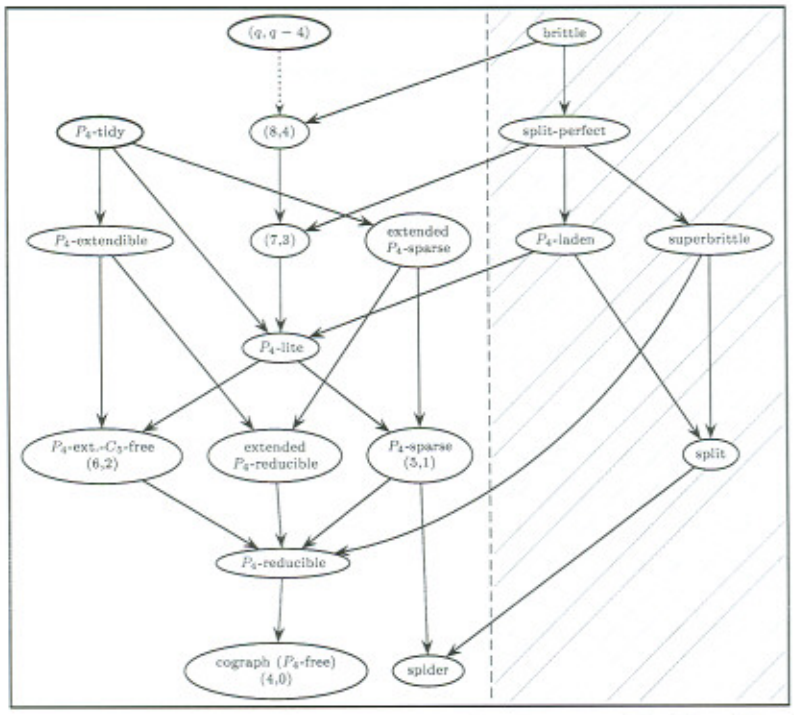

Figure 2: A Hasse diagram of class inclusions. For the classes to the left of the dashed line, the number of spanning trees can be computed in linear time.

time. We next investigate classes of graphs which have linear non-spider cost.

4.1 $(q, q-4)$-graphs

Babel and Olariu in [3] proposed the generalizing concept of (q,t)-graphs. In such a graph, no set of at most q vertices contains more than t distinct P_4s. In particular, the $(q, q-4)$-graphs possess important structural properties (they admit a unique tree representation; see Theorem 4.1) [2], are brittle[1] for $q \leq 8$ [3] but are not brittle (and not perfect) for $q \geq 9$. It turns out that the cographs are precisely the $(4,0)$-graphs, the P_4-sparse graphs are the $(5,1)$-graphs, and the C_5-free P_4-extendible graphs are the $(6,2)$-graphs. In our terminology, the structure of $(q, q-4)$-graphs can be described as follows.

Theorem 4.1. (Babel and Olariu [3]): *Let G be a $(q, q-4)$-graph. Then, every prime graph in the modular decomposition of G is either a prime spider or a graph with fewer than q vertices.*

Based on the above result, many optimization and domination problems (such as the vertex ranking, the path cover, the list coloring, the domination clique problem, etc.) can be solved in linear time for the class of $(q, q-4)$-graphs for fixed q [2]. Additionally, since computing the number of spanning trees of a graph on a fixed number of vertices takes constant time, Theorem 4.1 implies:

Lemma 4.1. *The non-spider cost of a $(q, q-4)$-graph G on n vertices is $\phi(G) = O(n)$, for every fixed $q \geq 4$.*

4.2 P_4-tidy graphs

As mentioned in [14], the class of P_4-tidy graphs was introduced by I. Rusu in order to illustrate the notion of P_4-domination in perfect graphs. A graph G is *P_4-tidy* if for any induced P_4, say *abcd*, there

[1] Chvátal defined a graph G to be *brittle* if each induced subgraph H of G contains a vertex that is not a midpoint or not an endpoint of any P_4. See also [9].

exists at most one vertex $v \in V(G) - \{a, b, c, d\}$ such that the subgraph $G[\{a, b, c, d, v\}]$ has at least two P_4s (i.e. the P_4 has at most one *partner*). The P_4-tidy graphs strictly contain the cographs, P_4-reducible, P_4-sparse, P_4-extendible, and P_4-lite graphs. The P_4-lite graphs were defined by Jamison and Olariu in [17]: a graph G is *P_4-lite* if every induced subgraph H of G with at most six vertices either contains at most two P_4s, or is a 3-sun, or is the complement of a 3-sun (a *3-sun* is a thick spider on six vertices with $R = \emptyset$). They remark that every P_4-sparse graph is P_4-lite and prove that every P_4-lite graph is brittle and is thus perfect. We mention here that the P_4-lite graphs coincide with the C_5-free P_4-tidy graphs.

Theorem 3.2 in[14] implies the following result for the modular decomposition of P_4-tidy graphs:

Theorem 4.2. (Giakoumakis *et al.* [14]): *Let G be a P_4-tidy graph. Then, every prime graph in the modular decomposition of G is a P_5, a $\overline{P_5}$, a C_5, an urchin, or a starfish*[2].

In [14], Theorem 4.2 played a crucial role in the linear-time recognition of P_4-tidy graphs and in the linear-time solution of the problems of calculating the clique and stability number, the chromatic number, and the hamiltonian path. In connection to our work, it implies that the non-spider cost $\phi(G)$ of a P_4-tidy graph G is linear in the number of vertices of G.

Lemma 4.2. *The non-spider cost of a P_4-tidy graph G on n vertices is $\phi(G) = O(n)$.*

Concluding, in light of Lemmata 4.1 and 4.2, the number of spanning trees of the classes of $(q, q-4)$-graphs for any fixed $q \geq 4$ and of P_4-tidy graphs can be efficiently computed. Moreover, it is not difficult to see that for these graphs the space needed by the algorithm Spanning_Trees-Number is $O(n + m)$; recall that the modular decomposition tree of a graph and its construction require space linear in the size of the graph [11, 19]. Thus, the results of this section are summarized in the following theorem.

Theorem 4.3. *The number of spanning trees of a $(q, q-4)$-graph for any fixed $q \geq 4$ or of a P_4-tidy graph can be computed in $O(n+m)$ time and space, where n and m are the number of vertices and edges of the input graph.*

Figure 2 depicts the classes of graphs that we mentioned in this paper and their inclusions; our approach allows for the linear-time computation of the number of spanning trees of the classes on the left-side of the figure.

5 Concluding Remarks

In this paper we have presented a general approach for computing the number of spanning trees of a graph using modular decomposition, which yields a linear-time algorithm for the problem on P_4-tidy graphs and $(q, q-4)$-graphs for any fixed $q \geq 4$. We have taken advantage of the structural properties of the modular decomposition tree of these graphs and used the Kirchhoff matrix tree theorem as a tool for proving the correctness of the proposed algorithm. Besides spider graphs, other prime graphs, such as, trees, cycles, and their complements, can be handled in the same way, which allows the computation of the number of spanning trees for additional classes as well, e.g., the *semi-P_4-sparse* graphs [13] and the *(P_5, diamond)-free* graphs [7]. Additionally, due to the central role that split graphs and their several extensions (which generalize the spider graphs) play in the modular decomposition tree of the *split-perfect* graphs [8], we pose as an open problem the efficient handling of the former class of graphs towards the linear-time computation of the number of spanning trees of the latter class.

[2] An *urchin* is a thin prime spider; a *starfish* is a thick prime spider.

Finally, as mentioned in the introduction, a uniformly-most reliable network (defined in [20]) must maximize the number of spanning trees. Thus, it is interesting to determine the types of graphs which have the maximum number of spanning trees for fixed numbers of vertices and edges (see [15, 24, 25]). The problem may be approached as an optimization question on the s-values of the vertices, which are calculated by the algorithm Spanning_Trees-Number. Work along these lines is currently in progress and some preliminary results suggest that almost regularity seems to be the key to the solution.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] L. Babel, T. Kloks, I. Kratochvil, D. Kratsch, H. Mueller, and S. Olariu, Efficient algorithms for graphs with few P_4s, *Discrete Math.* **235** (2001) 29–51.

[3] L. Babel and S. Olariu, On the structure of graphs with few P_4s, *Discrete Appl. Math.* **84** (1998) 1–13.

[4] C. Berge, *Graphs and Hypergraphs*, North-Holland, 1973.

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

[6] B. Bollobás, *Graph Theory, an Introductory Course*, Springer-Verlag, New York, 1979.

[7] A. Brandstädt, (P_5,Diamond)-free graphs revisited: structure and linear time optimization, *Discrete Appl. Math.* **138** (2004) 13–27.

[8] A. Brandstädt and V.B. Le, Split-perfect graphs: characterizations and algorithmic use, *SIAM J. Discrete Math.* **17** (2004) 341–360.

[9] A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[10] T.J.N. 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 fullerenes, *Discrete Appl. Math.* **67** (1996) 51–66.

[11] A. Cournier and M. Habib, A new linear algorithm for modular decomposition, *Proc. 19th Int'l Colloquium on Trees in Algebra and Programming (CAAP'94)*, LNCS **787** (1994) 68–84.

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

[13] J.-L. Fouquet and V. Giakoumakis, On semi-P_4-sparse graphs, *Discrete Math.* **165-166** (1997) 277–300.

[14] V. Giakoumakis, F. Roussel, and H. Thuillier, On P_4-tidy graphs, *Discrete Math. and Theoret. Comput. Science* **1** (1997) 17–41.

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

[16] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

[17] B. Jamison and S. Olariu, A new class of brittle graphs, *Studies Appl. Math.* **81** (1989) 89–92.

[18] B. Jamison and S. Olariu, A tree representation for P_4-sparse graphs, *Discrete Appl. Math.* **35** (1992) 115–129.

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

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

[21] S.D. Nikolopoulos and C. Papadopoulos, The number of spanning trees in K_n-complements of quasi-threshold graphs, *Graphs and Combinatorics* (to appear).

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

[23] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

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

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

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

[27] X. 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** (1997) 293–298.

[28] Y. Zhang and M.J. Golin, Chebyshev polynomials and spanning tree formulas for circulant and related graphs, HKUST Theoretical Science Center Research Report TCSC-02-09, 2002.

[29] Y. Zhang, X. Yong, and M.J. Golin, The number of spanning trees in circulant graphs, *Discrete Math.* **223** (2000) 337–350.