Efficient Algorithms For Computing The Matching And Chromatic Polynomials For Series-Parallel Graphs

N. Chandrasekharan[†‡] *and* Sridhar Hannenhalli
Department of Computer Science
University of Central Florida
Orlando, FL 32816
email: chandra@eola.cs.ucf.edu

Abstract. We present efficient algorithms for computing the matching polynomial and chromatic polynomial of a series-parallel graph in $O(n^3)$ and $O(n^2)$ time respectively. Our algorithm for computing the matching polynomial generalizes and improves the result in [13] from $O(n^3 \log n)$ time for trees and the chromatic polynomial algorithm improves the result in [18] from $O(n^4)$ time. The salient features of our results are the following: Our techniques for computing the graph polynomials can be applied to certain other graph polynomials and other classes of graphs as well. Furthermore, our algorithms can also be parallelized into NC algorithms.

1. Introduction and Previous Work

A graph $G = (V, E)$, where V is the vertex set and E is the edge set, is considered here as being finite, undirected, connected, and without multiple edges or self-loops. For most of our presentation, we follow the standard terminology of Golumbic [8], unless otherwise indicated. For obtaining complexity of algorithms, we use the standard RAM with integer operations (multiplication, addition, etc.) taking $O(1)$ time. The class of *series-parallel* (two-terminal) graphs is a subclass of planar graphs, which finds important applications in the fields such as electric networks and the scheduling problems [6, 15, 12]. Many standard graph problems which are NP-complete for general graphs such as Domination, Vertex-cover, Independent set are solvable in linear-time on series-parallel graphs [19]. All of these problems can also be solved fast in parallel on series-parallel graphs [9]. We define the class of series-parallel graphs as follows:

Definition 1.1: A graph $G = (V, E, u, v)$ where $u, v \in V$ are *distinguished* vertices called the *terminals*, is a series-parallel (sp) graph, if it can be obtained by finitely many applications of the following rules:

1. $G = (\{u, v\}, \{(u, v)\}, u, v)$ is an sp graph.
2. If $G_1 = (V_1, E_1, u_1, v_1)$ and $G_2 = (V_2, E_2, u_2, v_2)$ are sp graphs then:

   (a) the *series composition* of the graphs $G_1 \circ_s G_2 = (V_1 \cup V_2 - \{u_2\}, E_1 \cup E_2 \cup \{(x, v_1) | (x, u2) \in E_2\} - \{(x, u_2) | x \in V_2\}, u_1, v_2)$ is also an sp graph.

[†]Research supported in part by a National Science Foundation Grant No. NSF-CCR-9110159
[‡]Current address: Department of Mathematical Sciences, Loyola Universiry of Chicago, Lake Shore Campus Chicago, IL 60626. email: Chandra@math.luc.edu

(b) the *parallel composition* of the graphs $G_1 \circ_p G_2 = (V_1 \cup V_2 - (u_2, v_2), E_1 \cup E2 \cup \{(x, u_1) | (x, u_2) \in E_2\} \cup \{(x, v_1) | (x, v_2) \in E2\} - \{(x, y) | x \in V_2, y \in (u_2, v_2)\}, u_1, v_1)$ is an sp graph.

∎

Several graph polynomials have been proposed and studied extensively in the literature. These include the Tutte polynomial [21], the characteristic polynomial [7], the σ-polynomial [11], the rook polynomial [8], the matching polynomial [13] and the dichromatic polynomial [22]. We would like to study the computational aspects of the matching and chromatic polynomials here because of their significance. A *matching* in a graph $G = (V, E)$ is a set of edges $F \subseteq E$ such that no two edges in F have a vertex in common. The problem of finding a maximum cardinality matching is one of the most extensively studied graph problems [13]. Let $\phi_k(G)$ denote the number of k-element matchings in G. We define $\phi_0(G) = 1$ by convention. Let ν_G denote the matching number of G (which is the cardinality of a maximum matching set in G). Then the *matching polynomial* of G is

$$g(G; x) = \sum_{k=0}^{\nu_G} \phi_k(G) x^k$$

The complexity of computing the matching polynomial even for a bipartite graph is NP-hard, because it has been shown in [23] that the problem of counting the number of perfect matchings in it, is #P-complete. To the best of our knowledge, the only nontrivial algorithmic result for computing the matching polynomial is given in [13].

Theorem 1.2. *If G is a forest then*

$$m(G; x) = \sum_{k=0}^{\lfloor n/2 \rfloor} (-1)^k \phi_k(G) x^{n-2k}$$

(called the matching defect polynomial) is equal to the characteristic polynomial of the adjacency matrix of G. ∎

It is easy to see that from $m(G; x)$, the matching polynomial can be computed. Since the coefficients of the characteristic polynomial can be computed in $O(n^3 \log n)$ time [10], so can the matching polynomial be. Furthermore, in [13], it is noted that the matching polynomial for an outerplanar graph can be determined in polynomial time, without explicit mention of the time bound. In this paper, we generalize this result by developing an algorithm for computing the matching polynomial in $O(n^3)$ time for sp graphs.

A *proper coloring* of a graph $G = (V, E)$ is simply a function f from V into a set of colors such that $f(v) \neq f(w)$, whenever $(v, w) \in E$. For a given integer $t \geq 0$, the *chromatic polynomial* $P(G; t)$ is the number of distinct proper colorings of G using at most t colors. Chromatic polynomials are of interest in many ways. They arise in the context of graph coloring and are related to the famous graph reconstruction conjecture [2]. Furthermore, they capture a good deal of combinatorial information about a graph, describing its acyclic orientations, its all terminal reliability, and its spanning trees [5]. While computing the chromatic polynomial of an arbitrary graph is also NP-hard, polynomial time algorithms are known for chordal graphs [5] and sp graphs [18]. These algorithms take $O(m+n)$ and $O(n^4)$ time respectively. In [18], computing the chromatic polynomial of an sp graph G is reduced to counting the homomorphisms from G to K_i, for i ranging from 1 to $n+1$. We follow, a completely different approach that not only results in an improved $O(n^2)$ time algorithm, but also has the same flavor as our algorithm for computing the matching polynomial. The data structure we use for our algorithms is called the *parse-tree* of an sp graph (also called the *decomposition tree* [9]). A related result to ours is in [17] where it has been shown that for any accessible class of matroids of bounded width, the Tutte polynomial is computable in polynomial time.

As opposed to constructing an sp graph as in Definition 1, given an sp graph $G = (V, E, u, v)$, we can break G down into two sp subgraphs G_1 and G_2 such that either $G = G_1 \circ_s G_2$ or $G = G_1 \circ_p G_2$. This procedure can be performed recursively until we get to isomorphic copies of the basis graph which is a single edge. Such a procedure uncovers the history of constructing G. The binary tree representing this procedure is called the *parse-tree*. Given an sp graph G, a parse-tree for G can be found in $O(n)$ time [20]. We define the parse-tree of an sp graph G as follows:

Definition 1.3: A parse-tree of an sp graph G is a rooted tree S such that:
(a) each leaf of S corresponds to a single edge,
(b) the internal nodes of the tree S are labeled by either s or p representing the series and the parallel operation respectively. An internal vertex u of S corresponds to the sp graph obtained by composing (series or parallel) the two series-parallel subgraphs corresponding to the children of u.
(c) the sp graph corresponding to the root of S is isomorphic to G. ∎

Merits of our Results: There is a vast amount of literature on combinatorial aspects of graph polynomials, but only a few on sequential algorithmic aspects, and with the exception of parallel algorithms for obtaining the chromatic polynomial of a chordal graph in [4,16], hardly any on parallel algorithmic aspects. Our algorithmic results on computing graph polynomials either generalize or improve existing results, and in the context of parallelism provide new results. The parse-tree data structure used in this paper to compute the polynomials can also be used for certain other classes of graphs and graph polynomials.

The paper is organized along the following lines. The next section describes the design of the algorithm for computing the matching polynomial and Section 3 that of the chromatic polynomial. In Section 3 we also outline how to obtain parallel algorithms for the above problems.

2. Matching Polynomial

The main idea behind the algorithm for computing the matching polynomial is as follows: Given an sp graph G, we find ϕ_k for k ranging from 0 to v_G, by using a variation of the methodology described in [3, 24]. The methodology involves finding a homomorphism from (graph, subgraph) pairs to an appropriate set preserving certain properties. More details can be found in [3], but our presentation here is self-contained. Traditionally, the methodology has been used to solve a variety of graph optimization problems and their counting versions. Here, we vary the technique to obtain the number of subgraphs (satisfying a certain property) having a certain fixed size. We first start by considering the following four *recurrence classes* for an sp graph $G = (V, E, u, v)$:

$[U](G)$:$\{(G,F)$:F is a matching in G, u is matched and v is not matched in $F.\}$
$[V](G)$:$\{(G,F)$:F is a matching in G, v is matched and u is not matched in $F.\}$
$[W](G)$: $\{(G,F)$: F is a matching in G, both u and v are matched in $F.\}$
$[N](G)$: $\{(G,F)$: F is a matching in G, neither u nor v is matched in $F.\}$

Sometimes when the graph G is implied by the context, we will simply use [U], [V], [W] or [N] to denote the classes. We now extend the definition of the composition of two sp graphs $G_1 = (V_1 E_1, u_1, v_1) \circ G_2 = (V_2, E_2, u_2, v_2)$, where $\circ$ is either $\circ_s$ or $\circ_p$, to elements of the above classes in the following natural manner:

$$(G_1, F_1) \circ (G_2, F_2) = (G, F) = (G_1 \circ G_2, F_1 \cup F_2)$$

The next step is to consider all possible ways of composing elements taken from the classes [U], [V], [W], and [N] above. As a consequence, we can get the following *recurrence equations*. We get two sets of recurrence equations, one for the series and the other for the parallel composition operations.

Theorem 2.1. *If $G = G_1 \circ_s G_2$ is an sp graph, then the following equations hold:*

$$[U](G) = [U](G_1) \circ_s [U](G_2) \cup [U](G_1) \circ_s [N](G_2) \cup [W](G_1) \circ_s [N](G_2)$$
$$[V](G) = [V](G_1) \circ_s [V](G_2) \cup [N](G_1) \circ_s [V](G_2) \cup [N](G_1) \circ_s [W](G_2)$$
$$[W](G) = [U](G_1) \circ_s [V](G_2) \cup [U](G_1) \circ_s [W](G_2) \cup [W](G_1) \circ_s [V](G_2)$$
$$[N](G) = [V](G_1) \circ_s [N](G_2) \cup [N](G_1) \circ_s [U](G_2) \cup [N](G_1) \circ_s [N](G_2)$$

Proof: Following the definition of *series composition* operation, in order for G to be a member of $[U](G)$, u_1 has to be matched in F (hence in F_1) and v_2 cannot be matched in F (hence in F_2). Moreover, v_1 and u_2 can't be matched together.

From this the first equation readily follows. The other equations can be proved similarly. ∎

Theorem 2.2 *If $G = G_1 \circ_p G_2$ is an sp graph, then the following equations hold:*

$$
\begin{aligned}
[U](G) &= [U](G_1) \circ_p [N](G_2) \cup [N](G_1) \circ_p [U](G_2) \\
[V](G) &= [V](G_1) \circ_p [N](G_2) \cup [N](G_1) \circ_p [V](G_2) \\
[W](G) &= [U](G_1) \circ_p [V](G_2) \cup [V](G_1) \circ_p [W](G_2) \\
&\quad \cup [W](G_1) \circ_p [N](G_2) \cup [N](G_1) \circ_p [W](G_2) \\
[N](G) &= [N](G_1) \circ_p [N](G_2)
\end{aligned}
$$

The proof is similar to that for theorem 2.1. ∎

The above theorems can be used for finding a maximum cardinality matching in an sp graph. However, our interest is in counting the number of solutions of different sizes.

Theorem 2.3. *Let $G = (V, E, u, v)$, $G_1 = (V_1, E_1, u_1, v_1)$ and $G_2 = (V_2, E_2, u_2, v_2)$ be sp graphs such that $G = G_1 \circ_s G_2$. Then the following hold:*

(a) The total number of matchings in G is equal to $|[U](G)| + |[V](G)| + |[W](G)| + |[N](G)|$.

(b) $|[U](G)| = |[U](G_1)||[U](G_2)| + |[U](G_1)||[N](G_2)| + |[W](G_1)||[N](G_2)|$

(c) $|[V](G)| = |[V](G_1)||[V](G_2)| + |[N](G_1)||[V](G_2)| + |[N](G_1)||[W](G_2)|$

(d) $|[W](G)| = |[U](G_1)||[V](G_2)| + |[U](G_1)||[W](G_2)| + |[W](G_1)||[V](G_2)|$

(e) $|[N](G)| = |[V](G_1)||[N](G_2)| + |[N](G_1)||[U](G_2)| + |[N](G_1)||[N](G_2)|$

(f) If G is the base graph *i.e.*, K_2, then let $(|[U](G)|, |[V](G)|, |[W](G)|, |[N](G)|) = (0, 0, 1, 1)$

Proof: First observe that The recurrence classes $[x]$ for $x \in \{U, V, W, N\}$ are pairwise disjoint. Therefore, for each composite class $[x](G_1) \circ [y](G_2)$ shown in Theorem 2.1, we can write $|[x](G_1) \circ_s [y](G_2)| = |[x](G_1)| \times |[y](G_2)|$. For counting all the matchings in G we need to find $|[U](G)| + |[V](G)| + |[W](G)| + |[N](G)|$, establishing (a) of the Theorem. Furthermore, it can be seen that the right-hand sides of each equation in Theorem 2.1 are made of disjoint unions of *composite classes* of the form $[x] \circ [y]$. Hence, to find $|[U](G)|$, for example, it is enough to find the sum of products of the sizes of the composite classes making up the right-hand side. This fact establishes the validity of the equations (b) thru (e). The validity of statement (f) is obvious. ∎

The following theorem applies to the parallel operation and it is similar to Theorem 2.3.

Theorem 2.4. *Let* $G = (V, E, u, v), G_1 = (V_1, E_1, u_1, v_1)$ *and* $G_2 = (V_2, E_2, u_2, v_2)$ *be series-parallel graphs such that* $G = G_1 \circ_p G_2$. *Then the following hold:*

(a) The total Number of matchings in G is equal to $|[U](G)| + |[V](G)| + |[W](G)| + |[N](G)|$.
(b) $|[U](G)| = |[U](G_1)||[N](G_2)| + |[N](G_1)||[U](G_2)|$
(c) $|[V](G)| = |[V](G_1)||[N](G_2)| + |[N](G_1)||[V](G_2)|$
(d) $|[W](G)| = |[U](G_1)||[V](G_2)| + |[V](G_1)||[U](G_2)| + |[W](G_1)||[N](G_2)| + |[N](G_1)||[W](G_2)|$
(e) $|[N](G)| = |[N](G_1)||[N](G_2)|$
(f) If G is base graph i.e. K_2, then let $(|[U](G)|, |[V](G)|, |[W](G)|, |[N](G)|) = (0, 0, 1, 1)$.

Note that the empty matching is included in the set of all matchings. ∎

The above theorems are crucial to computing the number of "k-matchings" (ϕ_k) for sp graphs. Let $|[x](G)|^k$ denote the number of k-matchings belonging to class $[x]$. We then have the following theorem:

Theorem 2.5. *Let* $G = (V, E, u, v), G_1 = (V_1, E_1, u_1, v_1)$ *and* $G_2 = (V_2, E_2, u_2, v_2)$ *be sp graphs such that* $G = G_1 \circ_s G_2$. *Then the following hold:*

(a) The total number of k-matchings in G is equal to $|[U](G)|^k + |[V](G)|^k + |[W](G)|^k + |[N](G)|^k$.
(b) $|[U](G)|^k = \sum_{l+m=k}(|[U](G_1)|^l|[U](G_2)|^m + |[U](G_1)|^l|[N](G_2)|^m + |[W](G_1)|^l|[N](G_2)|^m)$
(c) $|[V](G)|^k = \sum_{l+m=k}(|[V](G_1)|^l|[V](G_2)|^m + |[N](G_1)|^l|[V](G_2)|^m + |[N](G_1)|^l|[W](G_2)|^m)$
(d) $|[W](G)|^k = \sum_{l+m=k}(|[U](G_1)|^l|[V](G_2)|^m + |[U](G_1)|^l|[W](G_2)|^m + |[W](G_1)|^l|[V](G_2)|^m)$
(e) $|[N](G)|^k = \sum_{l+m=k}(|[V](G_1)|^l|[N](G_2)|^m + |[N](G_1)|^l|[U](G_2)|^m + |[N](G_1)|^l|[N](G_2)|^m)$

Proof: Statement (a) follows from Theorem 2.3. Each class $|[x](G)|$ contains matchings of various sizes which can range from 0 to ν_G. Further, if $(G, F) = (G_1, F_1) \circ (G, F_2)$ then $|F| = |F_1| + |F_2|$. Therefore, to find the number of k-matchings in G, we have to compute all possible ways of adding up matchings in G_1 and G_2 to give rise to matchings of sizes k. Hence, we have the theorem. ∎

The proof of the following theorem is precisely along the lines as Theorem 2.5.

Theorem 2.6. *Let* $G = (V, E, u, v), G_1 = (V_1, E_1, u_1, v_1)$ *and* $G_2 = (V_2, E_2, u_2, v_2)$ *be sp graphs such that* $G = G_1 \circ_p G_2$. *Then the following hold:*

(a) The total number of k-matchings in G is equal to $|[U](G)|^k + |[V](G)|^k + |[W](G)|^k + |[N](G)|^k$.

(b) $|[U](G)|^k = \sum_{l+m=k}(|[U](G_1)|^l|[N](G_2)|^m + |[N](G_1)|^l|[U](G_2)|^m)$
(c) $|[V](G)|^k = \sum_{l+m=k}(|[V](G_1)|^l|[N](G_2)|^m + |[N](G_1)|^l|[V](G_2)|^m)$
(d) $|[W](G)|^k = \sum_{l+m=k}(|[U](G_1)|^l|[V](G_2)|^m + |[V](G_1)|^l|[U](G_2)|^m$
$+|[W](G_1)|^l|[N](G_2)|^m + |[N](G_1)|^l|[W](G_2)|^m)$
(e) $|[N](G)|^k = \sum_{l+m=k}(|[N](G_1)|^l|[N](G_2)|^m)$ ∎

In the following section we present the algorithm to compute all the k-matchings in an sp graph G, given its the parse-tree.

2.1. Algorithm

Let $G = (V, E, u, v)$ be an sp graph. The root of its parse-tree T corresponds to G. Define the structure *matching* associated with each node of T as:

$$\begin{pmatrix} & U_0 & U_1 & \dots & U_{\lfloor n/2 \rfloor} \\ SP & V_0 & V_1 & \dots & V_{\lfloor n/2 \rfloor} \\ & W_0 & W_1 & \dots & W_{\lfloor n/2 \rfloor} \\ & N_0 & N_1 & \dots & N_{\lfloor n/2 \rfloor} \end{pmatrix}$$

where U_k denotes the number of matchings of cardinality k when u is matched and v is not, V_k denotes the number of matchings of cardinality k when v is matched and u is not, W_k denotes the number of matchings of cardinality k when u is matched and v is matched, N_k denotes the number of matchings of cardinality k when neither u nor v is matched. The field "SP" takes either of the two values *viz.* "s" or "p" to denote the type of operation (series or parallel) at each node of T. The leaf nodes of the decomposition tree are single edges. The field "SP" is undefined for the leaf nodes. In order to get the total number of matchings of size k in a graph G for each k, we need to compute the value of *matching* associated with the root of the decomposition tree for G. The algorithm to compute this, takes as input, the *matchings* associated with the leaves of the decomposition tree. We refer to this as the *basis structure*. The *basis structure* for an edge (corresponding to each leaf of T) is:

$$\begin{pmatrix} & 0 & 0 & 0 & 0 & \dots & 0 \\ NULL & 0 & 0 & 0 & 0 & \dots & 0 \\ & 0 & 1 & 0 & 0 & \dots & 0 \\ & 1 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

The following algorithm computes the structure *matching* for an sp graph G, working on its parse-tree. It is based on an implementation of results in Theorems 2.5 and 2.6.

Function MATCH_SP(root of parse tree): *matching*
Begin
If root = leaf
then MATCH_SP = *basis structure*
else MATCH_SP = COMP(MATCH_SP(Left Child),
MATCH_SP(Right Child))
End

Function COMP(X1,X2: *matching*): *matching* X
Begin
If X $\cdot SP$ = "s"
For k = 0 to ν_G
$X \cdot U_k = X \cdot V_k = X \cdot W_k = X \cdot N_k = 0$
For l = 0 to k
$X \cdot U_k = X \cdot U_k + X_1 \cdot U_l * X2 \cdot U_{k-l} + X1 \cdot U_l * X2 \cdot N_{k-l}$
$+X1 \cdot W_l * X2 \cdot N_{k-l}$
$X \cdot V_k = X \cdot V_k + X1 \cdot V_l * X2 \cdot V_{k-l} + X1 \cdot N_l * X2 \cdot V_{k-l}$
$+X1 \cdot N_l * X2 \cdot W_{k-l}$
$X \cdot W_k = X \cdot W_k + X1 \cdot U_l * X2 \cdot V_{k-l} + X1 \cdot U_l * X2 \cdot W_{k-l}$
$+X1 \cdot W_l * X2 \cdot V_{k-l}$
$X \cdot N_k = X \cdot N_k + X1 \cdot V_l * X2 \cdot N_{k-l} + X1 \cdot N_l * X2 \cdot U_{k-l}$
$+X1 \cdot N_l * X2 \cdot N_{k-l}$
Endfor
Endif
Else
For k = 0 to ν_G
$X \cdot U_k = X \cdot V_k = X \cdot W_k = X \cdot N_k = 0$
For l = 0 to k
$X \cdot U_k = X \cdot U_k + X1 \cdot U_l * X2 \cdot N_{k-l} + X1 \cdot N_l * X2 \cdot N_{k-l}$
$X \cdot V_k = X \cdot V_k + X1 \cdot V_l * X2 \cdot N_{k-l} + X1 \cdot N_l * X2 \cdot V_{k-l}$
$X \cdot W_k = X \cdot W_k + X1 \cdot U_l * X2 \cdot V_{k-l} + X1 \cdot V_l * X2 \cdot U_{k-l}$
$+X1 \cdot W_l * X2 \cdot N_{k-l} + X1 \cdot N_l * X2 \cdot U_{k-l}$
$X \cdot N_k = X \cdot N_k + X1 \cdot N_l * X2 \cdot N_{k-l}$
Endfor
Endelse
End

The theorem given below addresses the time-complexity of the above algorithm.

Theorem 2.7. *The function MATCH_SP can be implemented in* $O(n^3)$ *time.*

Proof: Given an sp graph G, a parse-tree can be constructed in linear-time (the number of edges in G is $O(n)$). The computation of U_k and V_k take only $O(k)$

time. At each internal node, U_k and V_k are computed for all values of k in the range 0 and v_G. This makes the time-complexity for computing the structure *matching* at each internal node $O(n^2)$. The entire computation is proceeds bottom-up walking thru all of the nodes in T. Since there are $O(n)$ internal nodes in T, the total time-complexity is $O(n^3)$. ∎

3. Chromatic Polynomial

The chromatic polynomial $P(G;t)$ of a graph G, evaluated at $t = k$, gives the number of ways of properly coloring G using at most k colors. We call such a coloring a k-coloring of G. It does not seem possible to adopt the same strategy for computing the number of k-colorings for various values of k, as we did for the k-matchings. We will therefore, follow a different approach as follows: Using the parse-tree we compute $P(G;t)$ at $n+1$ distinct values of k, where n is the number of vertices of G. Using these values we interpolate to obtain the polynomial of degree n, which is really the $P(G;t)$.

Definition 3.1: A *k-coloring* of a graph $G = (V, F)$ is defined as a function f from V to the set C of k colors such that:

$$(x, y) \in F \Rightarrow f(x) \neq f(y).$$

Consider the following *classes* over the instances of a *k-colored* sp graph $G = (V, F, u, v)$.
$[E](G) = \{(G, f) : f(u) = f(v)\}$.
$[N](G) = \{(G, f) : f(u) \neq f(v)\}$.
The proof of the following lemma is straightforward.

Lemma 3.2. *Let $\alpha, \beta, \gamma \in C$. Let $|[E]_\alpha(G)|$ denote the number of colorings in $[E]$ such that $f(u) = f(v) = \alpha$. Let $|[N]_{\alpha,\beta}(G)|$ denote the number of colorings in $[N]$ such that $f(u) = \alpha$ and $f(v) = \beta$. Then we have,*

$$|[E]_\alpha(G)| = \frac{|[E](G)|}{k}$$

$$|[N]_{\alpha,\beta}(G)| = \frac{|[N](G)|}{k \cdot (k-1)}$$

∎

Let $G = (V, E, u, v)$, $G_1 = (V_1, E_1, u_1, v_1)$ and $G_2 = (V_2, E_2, u_2, v_2)$ be sp graphs such that $G = G_1 \circ_s G_2$. Let f, f_1 and f_2 be the respective colorings. The composition $\circ_s$ is feasible only if $f_1(v_1) = f_2(u_2)$ since v_1 and u_2 are *glued*. Now consider the result of a parallel composition operation *i.e.*, $G = G_1 \circ_p G_2$. The composition $\circ_p$ is feasible only if $f_1(u_1) = f_2(u_2)$ and $f_1(v_1) = f_2(v_2)$.

We will now describe the approach for computing the cardinalities of [E] and [N] via following theorems.

Theorem 3.3. *Given $G = G_1 \circ_s G_2$ where G, G_1, G_2 are sp graphs, we have:*

$$|[\mathrm{E}](G)| = \left(\frac{|[\mathrm{E}](G_1)|}{k} \cdot |[\mathrm{E}](G_2)|\right) + \left(\frac{|[N](G_1)|}{k \cdot (k-1)} \cdot [\mathrm{N}](G_2)|\right) \quad \text{(a)}$$

and

$$|[\mathrm{N}](G)| = \left(\frac{|[\mathrm{E}](G_1)|}{k} \cdot |[\mathrm{N}](G_2)|\right) + \left(|[N](G_1)| \cdot \frac{|[\mathrm{E}](G_2)|}{k}\right) + \left(\frac{|[N](G_1)|}{k \cdot (k-1)} \cdot |[\mathrm{N}](G_2)|\right) \quad \text{(b)}$$

Proof:

(a): Consider $G = G_1 \circ_s G_2$ such that $G \in [\mathrm{E}]$, then G can be obtained in one of the following ways:

1. $f_1(u_1) = \alpha, f_1(v_1) = f_2(u_2) = \alpha, f_2(v_2) = \alpha.$
   $|[\mathrm{E}]_\alpha(G_1)| = \frac{|[\mathrm{E}](G_1)|}{k}$
   $|[\mathrm{E}]_\alpha(G_2)| = \frac{|[\mathrm{E}](G_2)|}{k}$
   $\Rightarrow |[\mathrm{E}]_\alpha(G)| = \frac{|[\mathrm{E}](G_1)|}{k} \cdot \frac{|[\mathrm{E}](G_2)|}{k}$
   $\Rightarrow |[\mathrm{E}](G)| = \frac{|[\mathrm{E}](G_1)|}{k} \cdot \frac{|[\mathrm{E}](G_2)|}{k} \cdot k$

2. $f_1(u_1) = \alpha, f_1(v_1) = f_2(u_2) = \beta, f_2(v_2) = \alpha.$
   $|[\mathrm{N}]_{\alpha,\beta}(G_1)| = \frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)}$
   $|[\mathrm{N}]_{\beta,\alpha}(G_2)| = \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)}$
   $\Rightarrow |[\mathrm{E}]_\alpha(G)| = \frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)} \cdot \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)}$
   $\Rightarrow |[\mathrm{E}](G)| = \frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)} \cdot \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)} \cdot (k \cdot (k-1))$

G can fall in either of these categories, so

$$|[\mathrm{E}](G)| = \left(\frac{|[\mathrm{E}](G_1)|}{k} \cdot \frac{|[\mathrm{E}](G_2)|}{k} \cdot k\right) + \left(\frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)} \cdot \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)} \cdot (k \cdot (k-1))\right)$$

(b): Now consider $G = G_1 \circ_s G_2$, where $G \in [\mathrm{N}]$. Then G can be obtained in one of the following ways:

1. $f_1(u_1) = \alpha, f_1(v_1) = f_2(u_2) = \alpha, f_2(v_2) = \beta$, in which case
   $|[\mathrm{N}](G)| = \frac{|[\mathrm{E}](G_1)|}{k} \cdot \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)} \cdot (k \cdot (k-1))$
2. $f_1(u_1) = \alpha, f_1(v_1) = f_2(u_2) = \beta, f_2(v_2) = \beta$, in which case
   $|[\mathrm{N}](G)| = \frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)} \cdot \frac{|[\mathrm{E}](G_2)|}{k} \cdot (k \cdot (k-1))$
3. $f_1(u_1) = \alpha, f_1(v_1) = f_2(u_2) = \gamma, f_2(v_2) = \beta$, in which case
   $|[\mathrm{N}](G)| = \frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)} \cdot \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)} \cdot (k \cdot (k-1) \cdot (k-2))$, since $\alpha \neq \gamma \neq \beta$.

Sum of these three terms gives the expression for $|[N](G)|$ in the case of series composition. ∎

Theorem 3.4. *Given $G = G_1 \circ_p G_2$ where G, G_1, G_2 are sp graphs, we have:*

$$|[E](G)| = \frac{|[E](G_1)|}{k} \cdot |[E](G_2)| \qquad \text{(a)}$$

and

$$|[N](G)| = \frac{|[N](G_1)|}{k \cdot (k-1)} \cdot |[N](G_2)| \qquad \text{(b)}$$

Proof:
(a): Consider $G = G_1 \circ_p G_2$ such that $G \in [\mathrm{E}]$. Then $f_1(u_1) = f_2(u_2) = \alpha, f_1(v_1) = f_2(v_2) = \alpha$ in which case

$$|[\mathrm{E}](G)| = \frac{|[\mathrm{E}](G_1)|}{k} \frac{|[\mathrm{E}](G_2)|}{k} \cdot k$$

(b): Consider $G = G_1 \circ_p G_2$ such that $G \in [\mathrm{N}]$. Then $f_1(u_1) = f_2(u_2) = \alpha, f_1(v_1) = f_2(v_2) = \beta$ in which case

$$|[\mathrm{N}](G)| = \frac{|[\mathrm{N}](G_1)|}{k \cdot (k-1)} \frac{|[\mathrm{N}](G_2)|}{k \cdot (k-1)} \cdot (k \cdot (k-1))$$

∎

The above recursive equations can be employed to compute the number of *k-colorings*. We are now equipped to present the algorithm for computing the number of k-colorings for $3 \leq k \leq n+2$, where n is the number of vertices of G. It is well known that any sp graph can be properly colored using 3 colors.

3.1. Algorithm

Let $G = (V, E, u, v)$ be an sp graph. Define the structure *Colors* associated with each node of the parse-tree of G as:

$$\begin{pmatrix} SPG & E_3 & E_4 & \dots & E_{n+2} \\ & N_3 & N_4 & \dots & N_{n+2} \end{pmatrix}$$

where E_k denotes the *number of colorings* of G using at most k distinct colors, when u and v are colored by the same color, and N_k denotes the *number of colorings* of G using at most k distinct colors, when u and v are colored by different colors. The field "SP" can take three possible values *viz.* "s", or "p" depending on whether the operation at the particular node is *series* or *parallel* respectively. In order to get the total number of colorings of graph G using atmost k colors for arbitrary k, we need to compute the value of *Colors* associated with the root of the

decomposition tree for G. The total number of colorings using atmost k colors is given by $E_k + N_k$. The algorithm takes as input, the structure *Colors* associated with the leaves of the decomposition tree which are single edges. We refer to this as the *basis structure* as before. It is easy to see that $E_k = 0$ and $N_k = k*(k-1)$, $3 \leq k \leq n+2$ when the graph consists of just a single edge. After we get the values of $E_k + N_k$ for $3 \leq k \leq n+2$, which is the same as $P(G;t)$ evaluated at $n+1$ distinct points, we can use these values to interpolate the polynomial of degree n. The time-complexity for computing the $n+1$ values is $O(n^2)$. We can use a simple method of polynomial interpolation such as the Newton's taking an additional $O(n^2)$ time, for a total of $O(n^2)$ time. We give below the algorithm for computing the k-colorings.

Function COLOR_SP(root of parse tree): *Colors*
Begin
  If root = leaf
  then COLOR_SP = *basis vector*
  else COLOR_SP = COMP(COLOR_SP(Left Child),
                           COLOR_SP(Right Child))
End

Function COMP(X1,X2: COLORING): *Colors* X
Begin
  If $X \cdot SP$ = "s"
    For k = 3 to $n+2$

$$X \cdot E_k = \frac{X1 \cdot E_k * X2 \cdot E_k}{k} + \frac{X1 \cdot N_k * X2 \cdot N_k * (k-1)}{k}$$

$$X \cdot N_k = \frac{X1 \cdot E_k * X2 \cdot N_k}{k} + \frac{X1 \cdot N_k * X2 \cdot E_k}{k} + \frac{X1 \cdot N_k * X2 \cdot N_k * (k-2)}{k*(k-1)}$$

    Endfor
  Endif
  Elseif $X \cdot SP$ = "p"
    For k = 3 to $n+2$

$$X \cdot E_k = \frac{X1 \cdot E_k * X2 \cdot E_k}{k}$$

$$X \cdot N_k = \frac{X1 \cdot N_k * X2 \cdot N_k}{k*(k-1)}$$

    Endfor
  Endelse
End.

3.2. Parallel Algorithms

Results in [1, 4] show in general, how to obtain fast parallel algorithms for bottom-up binary tree (parse-tree) computations using the *tree contraction* technique. Briefly, the tree contraction technique reduces a binary tree into a 3 node (or a single node) tree by means of repeated shunt operations performed carefully on the leaves of the tree. The number of steps for this reduction is $O(\log n)$ using

$O(n/\log n)$ processors on an Exclusive Read Exclusive Write (EREW) PRAM. This technique can be used even when the computations at each internal node tale more than $O(1)$ time, as in the case of computing the matching and chromatic polynomials, for example. The parse-tree of an sp graph can be obtained using $O(n)$ processors in $O(\log^2 n)$ time on an EREW PRAM [9]. Using the tree contraction technique on the parse-tree, it is a standard exercise to obtain a fast parallel algorithm using $O(n^3)$ processors in $O(\log^2 n)$ time for computing the coefficients of the matching polynomial. In the same manner, using $O(n^2)$ processors and in $O(\log^2 n)$ time, we can compute the number of ways of coloring an sp graph using k colors, where $3 \leq k \leq n+2$. Furthermore, polynomial interpolation can be performed in $O(\log^2 n)$ time using $O(n/\log n)$ processors [10].

Acknowledgements:

We thank the referee for many excellent suggestions which helped improve the paper significatly and also the editor for much timely help.

References

1. K.Abramhson,N.Dadoun and T.Przytycka, *A Simple Parallel Tree Contraction Algorithm*, Journal of Algorithms 10 2 (1989), 287–302.
2. Bari, R., *Chromatically equivalent graphs*, in "Graphs and Combinatorics", Lecture Notes in Mathematics 406, 1974, pp. 186–200.
3. Chandrasekharan, N., S. Hedetniemi and T. Wimer, *Enumeration Techniques for Certain k-terminal Graphs*, submitted.
4. Chandrasekharan, N. and Shier D., *Algorithms for Computing the Chromatic Polynomial*, Journal of Combinatorial Mathematics and Combinatorial Computing 4 (1988), 213–222.
5. Chandrasekharan, N., *Fast parallel Algorithms and Enumeration Techniques for partial k-Trees*, Ph.D. Dissertation (1989), Clemson University.
6. Duffin, J., *Topology of Series Parallel Networks*, Journal of Mathematical Analysis and Applications 10 (1965), 303–318.
7. Harary, F., "Graph Theory", Addisson-Wesley, 1969.
8. Golumbic, M.C., "Algorithmic Graph Theory and Perfect Graphs", Academic Press, 1986.
9. He, X. and Y. Yesha, *Parallel Recognition and Decomposition of two-terminal series-parallel graphs*, Information and Computation 74 (1988).
10. JaJa, Joseph, "An Introduction to Parallel Algorithms", Addison-Wesley, 1992.
11. Korfhage, R., *σ-polynomials and graph coloring*, J. Combin. Theory B, 24 (1978), 137–153.

12. Lawler, L., *Sequencing jobs to Minimize Total Weighted Completion time Subject to Precedence Constraints*, Annals of Discrete Mathematics 2 (1978), 75–90.
13. Lovasz, L. and M. Plummer, "Matching Theory", North-Holland, Amsterdam, 1986.
14. G.Miller and J.Reif, *Parallel Tree Contraction Part I. Fundamentals*, Advances in Computing Research 5 (1989), 47–72.
15. Monma, L., Sidney, *B. A General Algorithm for Optimal Job Sequence with Series-Parallel Constraints*, Tech. Rept. No. 347 (1977), School of OR and IE, Cornell University.
16. Naor, J., M. Naor and A.A. Schaffer, *Fast parallel Algorithms for Chordal Graphs*, SIAM J. Computing 32 (1989).
17. Oxley, J.G., and D.J.A. Welsh, *Tutte Polynomials Computable in Polynomial Time*, Manuscript, Dept. of Mathematics, Louisiana State University, 1992.
18. Ravi, S.S., Hunt III, and Stearns, R. Separators, *Graph Homomorphisms and Chromatic Polynomials*, in "Proc. 26th Annual Allerton Conference on CommuniCation, Control and Computing", Urbana-Champaign, Illinois, 1988.
19. Takamizawa, T., Nishizeki, N. Saito, *Linear-Time Computability of Combinatorial Problems on Series-Parallel Graphs*, JACM 29 3 (1982), 623–641.
20. Tarjan, R., Valdes, *The Recognition of Series-parallel graphs*, Proc. 11th ACM STOC, 1979, 1–12.
21. Temperley, H., "Graph Theory and Applications", Ellis Horwood, 1981.
22. Tutte, W., *A contribution to the theory of Chromatic polynomials*, Canadian J. Mathematics 6 (1954), 80–91.
23. Valiant, L., *The Complexity of Enumeration and Reliability Problems*, SIA M J. Computing 8 3 (1984), 410–421.
24. Wimer, T., *Linear Algorithms on k-Terminal Graphs*, Ph.D. Thesis, Clemson University, Aug. 1987.