The Complexity of List Ranking of Trees

Dariusz Dereniowski*
Department of Algorithms and System Modeling,
Gdańsk University of Technology, Poland
deren@eti.pg.gda.pl

Abstract: A vertex k-ranking of a graph G is a function c : $V(G) \rightarrow \{1, \ldots, k\}$ such that if $c(u) = c(v), u, v \in V(G)$ then each path connecting vertices u and v contains a vertex w with $c(w) > c(u)$. If each vertex v has a list of integers $L(v)$ and for a vertex ranking c it holds $c(v) \in L(v)$ for each $v \in V(G)$ then c is called $\mathcal{L}$-list k-ranking, where $\mathcal{L} = \{L(v) : v \in V(G)\}$. In this paper we investigate both vertex and edge (vertex ranking of a line graph) list ranking problems. We prove that both problems are NP-complete for several classes of acyclic graphs, like full binary trees, trees with diameter at most 4 and comets. The problem of finding vertex (edge) $\mathcal{L}$-list ranking is polynomially solvable for paths and trees with bounded number of nonleaves, which includes trees with diameter less than 4.

1 Introduction

Let $G = (V(G), E(G))$ be a graph and let $\mathcal{L} = \{L(v) : v \in V(G)\}$, where $L(v)$ is a set of permissible colors for vertex v. In the list coloring problem G and $\mathcal{L}$ are given and the goal is to determine whether there exists a proper vertex coloring of G such that each vertex v gets a color from its list $L(v)$. This problem is a generalization of the classical coloring problem because coloring of a graph G with k colors is a list coloring with $L(v) = \{1, \ldots, k\}$ for all v. Thus, this problem is NP-complete for general graphs.

Vertex k-ranking of a graph G is a coloring of the vertices of G with k colors such that for all vertices u, v with the same color each path connecting u and v contains a vertex w colored with a greater color. The smallest integer k for which there exists a vertex k-ranking of G is called the *vertex ranking number* of G and denoted by $\chi_r(G)$. The problem of determining $\chi_r(G)$ is NP-hard for chordal graphs [4], bipartite and co-bipartite graphs [1]. On the other hand, the vertex ranking problem is polynomially solvable for interval, circular-arc

and permutation graphs [2], partial k-trees, where k is bounded [1] and, in particular, it is linear for trees [11].

A function c mapping the set of edges of G into integers $1, \ldots, k$ is an *edge k-ranking* of G if each path connecting two edges x, y colored with the same color contains an edge z such that $c(z) > c(x)$. The smallest number k such that there exists an edge k-ranking of G is the *edge ranking number* of G and it is denoted by $\chi'_r(G)$. Finding edge rankings using minimal number of colors is NP-hard for general graphs [7]. This problem can be solved in linear time for trees [8], but it is hard for bounded diameter multitrees [3].

It is natural to consider a generalization of ranking problems to list rankings. More formally, given a graph G and a list assignment $\mathcal{L} = \{L(v) : v \in V(G)\}$ for G, a function c is a *vertex $\mathcal{L}$-list k-ranking* of G if c is a vertex k-ranking of G and for each vertex $v \in V(G)$, $c(v) \in L(v)$. If the value of k is not important then we write vertex $\mathcal{L}$-list ranking or simply vertex list ranking. Vertex list ranking was recently defined by Jamison in [6]. In the *Vertex List Ranking problem* for a given G and $\mathcal{L}$ we ask whether there exists a vertex $\mathcal{L}$-list ranking of G. Finally, we define *edge $\mathcal{L}$-list ranking* and *Edge List Ranking problem* to be, respectively, the vertex $\mathcal{L}$-list ranking and Vertex List Ranking problem for the line graphs. Note that if all vertices of graph G have the same list of available colors and the vertex ranking problem can be solved in polynomial time for G then the Vertex List Ranking problem is also polynomially solvable for G.

The vertex ranking problem has applications in the parallel Cholesky factorization of matrices [9]. The edge ranking problem can be applied in the parallel query processing [10]. Consider the query graph G for a given database query. Vertices of G correspond to relations and two vertices of this graph are adjacent if the corresponding relations have to be joined during execution of the query. The goal is to find edge ranking of some spanning tree of G. The execution of a query is divided into steps and in the ith step we process operations which correspond to the edges of the spanning tree colored with label i. Another application of the edge ranking problem is the parallel assembly of modular products [3, 5], where we have a tree T such that the parts of the product correspond to vertices of T and two vertices are adjacent in T if and only if the corresponding parts have to be connected during the assembly. If we find an edge ranking of T then in the ith step we perform operations corresponding to edges labeled by i. The List Ranking Problems can be interesting not only because they are generalizations of the ranking problems, but because of their potential applications. By using list rankings in both applications we can specify some additional properties of the input problem (i.e. query processing or product assembly). For example, if some operation corresponding to edge x has to be performed in one of the prespecified time windows then we can express such a restriction by defining the list of available colors for edge x in such a way that $i \notin L(x)$ if the corresponding operation cannot be performed in the ith step.

The paper is organized as follows. In the next section we describe a polynomial-time reduction from the Set Cover problem to the Vertex List Ranking of graphs. This reduction is given in a general form, i.e. we list properties which must be

satisfied by a graph and a list assignment for vertices of the graph, created on the basis of the input for the Set Cover problem. Then, we use this general theorem to show that the Vertex List Ranking problem is hard for several classes of trees. In Section 3.4 we use previously described reduction to show that the Edge List Ranking problem is hard for those classes of trees as well. Section 4 describes two examples of polynomial-time algorithms. In particular, we use a dynamic programming technique for solving the problem for paths and a reduction to the matching problems, which allows to find a list rankings for graphs with bounded number of nonpendant vertices.

2 General reduction from the Set Cover problem

Let us recall the Set Cover problem. Given a set $S = \{a_1, \ldots, a_n\}$ and l its subsets $S_1, \ldots, S_l$, find the smallest subset of indices $I \subseteq \{1, \ldots, l\}$ such that the sets S_i *cover* S, i.e.

$$\bigcup_{i \in I} S_i = S.$$

In our reduction we use the decision version of the above problem denoted by Set Cover, where the question is: does there exist a subset $I \subseteq \{1, \ldots, l\}$ of cardinality at most t (t is part of the input to the problem) which covers S?

For a given instance of the Set Cover problem we construct a graph G and a list assignment $\mathcal{L}$ for the vertices of G such that there exists a subset I of cardinality at most t which covers S if and only if there exists a vertex $\mathcal{L}$-list ranking of G. In the following we assume that $l > 0$ and for each $j = 1, \ldots, n$ there exists a set S_i such that $a_j \in S_i$, because if $a_j \notin S_i$ for each $i = 1, \ldots, l$ and some $j \in \{1, \ldots, n\}$ then there is no set I which covers S and as an input for the Set Cover problem we pass any graph G and $\mathcal{L}$ such that vertex $\mathcal{L}$-list ranking of G does not exist.

In this section we are going to describe the reduction in a general form by determining properties which should hold for G and $\mathcal{L}$ in order to reduce the Set Cover problem to Vertex List Ranking of G. We will not restrict our theorems to trees, because we want to show hardness for list ranking of line graphs of trees as well. So, we formulate and prove theorems in this section in a more general form.

For each set $S_i, i = 1, \ldots, l$ there are vertices $\{v[S_i]\} \cup \{v_i[a] : a \in S_i\}$ in G. Vertex $v[S_i]$ corresponds to set S_i while the vertices $v_i[a]$ correspond to the elements of S_i. Denote $V(S_i) = \{v_i[a] : a \in S_i\}$. Graph G contains a set of *blocking* vertices

$$B = \{b_1, \ldots, b_{l-t}\} \cup \{b[a_1], \ldots, b[a_n]\}.$$

The vertices $b_1, \ldots, b_{l-t}$ block colors available for vertices $v[S_1], \ldots, v[S_l]$, i.e. they get colors which belong to the lists $L(v[S_1]), \ldots, L(v[S_l])$. Analogously, $b[a_1], \ldots, b[a_n]$ are blocking vertices for $v_i[a_j]$, because $b[a]$ gets a color which

can be assigned to some vertex $v_i[a]$, where $i \in \{1, \ldots, l\}$ and $a \in S_i$. Let us denote

$$W = B \cup \{v[S_1], \ldots, v[S_l]\} \cup V(S_1) \cup \cdots \cup V(S_l).$$

Graph G contains also some additional vertices which give some properties of G, like being bounded degree and we will define them in the next sections. Denote $d = \chi_r(G[V(G) \setminus W])$, where $G[X]$ is the subgraph induced by vertices of X, i.e.

$$G[X] = (X, \{\{u, v\} \in E(G) : u, v \in X\}).$$

Define $\mathcal{L} = \{L(v) : v \in V(G)\}$ for G. Let

$$L(v) = \{1, \ldots, d\}, \text{ where } v \in V(G) \setminus W. \tag{1}$$

The list of permissible colors for each vertex $v[S_i]$, $i = 1, \ldots, l$ contains two elements:

$$L(v[S_i]) = \{d + i, d + l + i + 1 + \sum_{j=1}^{i} |V(S_j)|\}. \tag{2}$$

Color $d + i$ is called *small* for vertex $v[S_i]$ and the other color is *big* for $v[S_i]$. We will use symbols $\mathrm{big}(v[S_i])$ and $\mathrm{small}(v[S_i])$, respectively. Define lists for the vertices in $V(S_i), i = 1, \ldots, l$ as follows.

$$\forall_{v \in V(S_i)} |L(v)| = 1, \tag{3}$$

$$\forall_{u,v \in V(S_i)} u \neq v \rightarrow L(u) \neq L(v) \text{ and} \tag{4}$$

$$\bigcup_{v \in V(S_i)} L(v) = \{\mathrm{big}(v[S_i]) - |V(S_i)|, \ldots, \mathrm{big}(v[S_i]) - 1\}. \tag{5}$$

In other words, for each vertex $v \in V(S_i)$, $L(v)$ contains one element, which is smaller than $\mathrm{big}(v[S_i])$ and bigger than $\mathrm{big}(v[S_{i-1}])$ (if $i > 1$). Colors permissible for different vertices in $V(S_i)$ do not overlap. We sort these vertices according to increasing values of their permissible colors. Formally, for each $i = 1, \ldots, l$ define a permutation $\pi_i : V(S_i) \rightarrow \{1, \ldots, |V(S_i)|\}$ such that $\pi_i(u) < \pi_i(v)$ if and only if $x < y$, where $L(u) = \{x\}$ and $L(v) = \{y\}$. Finally, define lists for blocking vertices:

$$L(b_k) = \{\mathrm{big}(v[S_i]) : i = 1, \ldots, l\}, \tag{6}$$

$$L(b[a]) = \bigcup \{L(v_i[a]) : i = 1, \ldots, l \text{ and } a \in V(S_i)\}, \tag{7}$$

where $k = 1, \ldots, l - t$ and $a \in S$.

The definition of $\mathcal{L}$ is correct, i.e. each vertex has nonempty list of available colors, because we assumed that for each $j = 1, \ldots, n$ element a_j belongs to some set S_i (so $L(b[a_j]) \neq \emptyset$) and we assumed that $l > 0$ (so $L(b_j) \neq \emptyset$ for $j = 1, \ldots, l - t$). In the following, if path P connects two vertices u and v then we assume (in order to simplify the notation) that u is adjacent to one end vertex of P and v is adjacent to the other end vertex of P. Thus, we assume that $u, v \notin V(P)$.

On the basis of the definition of $\mathcal{L}$ we can formulate two simple lemmas which hold regardless of the structure of graph G. Graph G' in Lemma 1 is defined so that $V(G') = W$ and $\{u,v\} \in E(G')$ iff u and v are connected in G by a path which does not contain a vertex from W.

Lemma 1 *Let $\mathcal{L}$ satisfy properties (1) – (7). If c_1 is a vertex $\mathcal{L}$-list ranking of G' and c_2 is a vertex $\mathcal{L}$-list ranking of $G[V(G) \setminus W]$ then function c such that $c|_W = c_1$ and $c|_{V(G) \setminus W} = c_2$ is a vertex $\mathcal{L}$-list ranking of G.*

Proof: If $u \in W$ and $v \notin W$ then $L(u) \cap L(v) = \emptyset$. So, we consider two cases. *Case 1:* $u, v \notin W$. Each path between u and v in G which does not appear in $G[V(G) \setminus W]$ contains a vertex $w \in W$. We have, by the definition of $\mathcal{L}$, $\min(L(w)) > \max(L(u) \cup L(v))$, which means that $c(w) > \max(c(u), c(v))$. *Case 2:* $u, v \in W$. If P is a u-v path in G then there exists a path in G' containing vertices in $V(P) \cap W$. Thus, by the definition of $\mathcal{L}$, the biggest color in path P in G equals to the biggest color in the corresponding path in G'. $\square$

The second lemma follows directly from the definition of $\mathcal{L}$.

Lemma 2 *Let $\mathcal{L}$ satisfy properties (1) – (5). If $u, v \in W \setminus B$ then $L(u) \cap L(v) = \emptyset$.* $\square$

Now, let us formulate properties (p_1), (p_2) and (p_3) of G and $\mathcal{L}$ which guarantee that the Set Cover problem reduces to the problem of Vertex List Ranking of G. The properties are as follows:

(p_1) $\mathcal{L}$ is defined by equations (1) – (7),

(p_2) for each $u \in V(S_i)$ and $v \in B$ each path connecting u and v contains vertex $v[S_i]$,

(p_3) for each $u \in V(S_i)$ and $v \in B$ there exists a path between u and v which does not contain vertices in

$$B \cup \{v[S_{i+1}], \ldots, v[S_l]\} \cup \{w \in V(S_i) : \pi_i(w) > \pi_i(u)\}.$$

In the following we assume that G and $\mathcal{L}$ satisfy properties (p_1), (p_2) and (p_3). By Lemma 1, without loss of generality we assume that subgraph $G[V(G) \setminus W]$ is properly colored and for a u-v path P in G, where $u, v \in W$ we consider only vertices of P which belong to W. Lemma 3 is a consequence of property (p_2).

Lemma 3 *For each $i = 1, \ldots, l$ graph $G - v[S_i]$ contains two (not necessary connected) components. The vertices in B belong to one component and the vertices in $V(S_i)$ belong to the other component.* $\square$
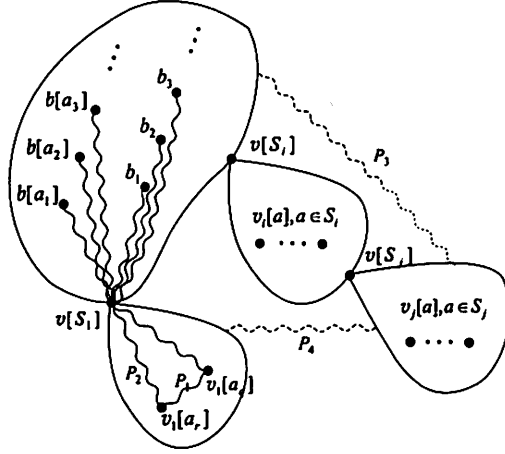
Figure 1: The structure of graph G satisfying properties (p_1) and (p_2).

Let us illustrate the structure of graph G in Fig. 1. There exists a path between the vertex $v[S_1]$ and any vertex among $b_1, \ldots, b_{l-t}, b[a_1], \ldots, b[a_n]$. Paths like P_3 and P_4 are not allowed because vertices $v[S_i]$ are separators as stated in Lemma 3. A vertex $v[S_i]$ can be directly connected to the component containing vertices in B or paths connecting $v[S_i]$ to this component may pass through vertices $v[S_{i'}], i' < i$. So, in Fig. 1 we have $j > i$. If $\pi_1(v_1[a_r]) > \pi_1(v_1[a_q])$ then there exists a path connecting $v_1[a_q]$ with $v[S_1]$, which does not contain $v_1[a_r]$. On the other hand, all paths connecting $v_1[a_r]$ to $v[S_1]$ may contain $v_1[a_q]$. Thus, the path P_2 does not have to exist, but the existence of P_2 does not exclude the path P_1.

Now we prove several technical lemmas which we use to prove the main result of this section formulated as Theorem 1.

Lemma 4 *For all vertices $u, v \in B$ there exists a path P connecting u and v such that $V(P) \cap W \subseteq \{v[S_1]\}$.*

Proof: By (p_3) there exists a path P_1' connecting u and some vertex $w \in V(S_1)$ which does not contain vertices in $B \cup \{v[S_2], \ldots, v[S_l]\}$. Note that by Lemma 3, if $v[S_i] \notin V(P_1')$, $i > 1$ then no vertex in $V(S_i)$ belongs to P_1', because otherwise P_1' would contain a cycle. Thus, $V(P_1') \cap (V(S_2) \cup \cdots \cup V(S_l)) = \emptyset$. So, $P_1 = P_1'[V(P_1') \setminus (V(S_1) \cup \{v[S_1]\})]$ connects u and $v[S_1]$ and does not contain vertices in $V(S_1) \cup \cdots \cup V(S_l) \cup \{v[S_1], \ldots, v[S_l]\}$. Similarly, there exists a path P_2 connecting v and $v[S_1]$ which does not contain vertices in $V(S_1) \cup \cdots \cup V(S_l) \cup \{v[S_2], \ldots, v[S_l]\}$. If we concatenate P_1, $v[S_1]$ and P_2 and remove cycles if necessary, then we obtain the desired path P. □

Lemma 5 *All vertices in B get pairwise different colors in each vertex $\mathcal{L}$-list ranking of G.*

Proof: By (7), for each $b[a_i], b[a_j]$, $i, j = 1, \ldots, n$ if $i \neq j$ then $L(b[a_i]) \cap L(b[a_j]) = \emptyset$. By (6) and (7), for each pair $b[a_i], b_j$, $i = 1, \ldots, n, j = 1, \ldots, l-t$ we also have $L(b[a_i]) \cap L(b_j) = \emptyset$. Consider two vertices b_i, b_j, $i \neq j$ and assume that $c(b_i) = c(b_j)$ in some vertex $\mathcal{L}$-list ranking c of G. From (6) we know that $c(b_i)$ is big with respect to some vertex $v[S_k]$, $k \in \{1, \ldots, l\}$. By (2) we have that $\mathrm{big}(v[S_i]) < \mathrm{big}(v[S_j])$ for $1 \le i < j \le l$. By Lemma 4 there exists a path P connecting b_i and b_j, such that no vertex in $W \setminus \{v[S_1]\}$ belongs to P. Thus the biggest possible color assigned to a vertex in P is $\mathrm{big}(v[S_1])$. This leads to a contradiction to the definition of vertex ranking. □

Lemma 6 *If $u \in V(S_i)$ and $v \in B$ then $v[S_i]$ is the only vertex in $V(G)$ which belongs to each path between u and v and can get bigger color than the only element in $L(u)$.*

Proof: We want to find a vertex $w \in W$ such that $\max(L(w)) > p$, where $L(u) = \{p\}$ and w belongs to each path connecting u and v. By property (p_3), $w \notin B$ and $w \neq v[S_j]$, $j = i + 1, \ldots, l$. By Lemma 3, $w \notin V(S_j)$ for $j > i$, because otherwise we would have that $v[S_j]$, $j > i$ also is in every path between u and v — a contradiction. Furthermore, $w \notin V(S_i)$, because if $\pi_i(w) > \pi_i(u)$ then there exists a path between u and v which does not contain w (by property (p_3)) and if $\pi_i(w) < \pi_i(u)$ then the color assigned to w in every $\mathcal{L}$-list ranking of G is smaller than the color of u (by the definition of π_i). Thus, all vertices not yet considered have lists containing colors smaller than p except $v[S_i]$, because $\mathrm{big}(v[S_i]) > p$. Furthermore, by property (p_2), $v[S_i]$ belongs to each path between u and v, which completes the proof. □

Lemma 7 *For each $i = 1, \ldots, l$ and each $j = 1, \ldots, l-t$ there exists a path P connecting $v[S_i]$ and b_j such that*

$$\forall_{u \in V(P)} \max(L(u)) < \mathrm{big}(v[S_i]).$$

Proof: By property (p_3) there exists a path P' between b_j and some vertex $v \in V(S_i)$ which does not contain vertices in

$$B \cup \{v[S_{i+1}], \ldots, v[S_l]\}.$$

By (p_2), P' contains $v[S_i]$. Let P be the subpath of P' connecting b_j and $v[S_i]$. Thus, if $w \in V(P) \cap W$ then $w = v[S_k]$ or $w \in V(S_k)$, where $k < i$. By the definition of $\mathcal{L}$ in both cases $\max(L(w)) < \mathrm{big}(v[S_i])$. □

Theorem 1 *Let G and $\mathcal{L}$ satisfy $(p_1), (p_2)$ and (p_3). There exists a solution of cardinality at most t to the Set Cover problem if and only if there exists a vertex $\mathcal{L}$-list ranking of G.*

Proof: ($\Rightarrow$) Assume that there exists a solution I of cardinality at most t to the Set Cover problem. We construct vertex $\mathcal{L}$-list ranking c of G. We assume that graph $G[V(G) \setminus W]$ has been colored by any vertex ranking algorithm (there exist linear time algorithms for vertex ranking of trees [11] and for edge ranking of trees [8]). By Lemma 1 we consider in the following vertices in W only. Each vertex $v \in V(S_1) \cup \cdots \cup V(S_l)$ gets the color from its list $L(v)$. Define

$$c(v[S_i]) = \begin{cases} \text{big}(v[S_i]) & \text{if } i \in I \\ \text{small}(v[S_i]) & \text{if } i \notin I \end{cases} \tag{8}$$

By Lemma 2 graph $G[W \setminus B]$ is properly colored. At most t vertices among $v[S_1], \ldots, v[S_l]$ get big color, because $|I| \leq t$. So, at least $l - t$ big colors are not assigned to vertices in $W \setminus B$ and, by (6), any big color can be assigned to a vertex $b_j, j = 1, \ldots, l - t$. It remains to color $b[a_1], \ldots, b[a_n]$. Let $j \in \{1, \ldots, n\}$. There exists $i \in I$ such that $a_j \in S_i$. By (8), $c(v[S_i]) = \text{big}(v[S_i])$. By (p_2), each path between $v_i[a_j]$ and $b[a_j]$ contains $v[S_i]$. From the definition of $\mathcal{L}$ it follows that $c(v_i[a_j]) \in L(b[a_j])$ and $\text{big}(v[S_i]) > c(v_i[a_j])$. This means that we can extend c so that $c(b[a_j]) := c(v_i[a_j])$. The above statement holds for each $j = 1, \ldots, n$, which means that all vertices can be colored properly.

($\Leftarrow$) Let c be a vertex $\mathcal{L}$-list ranking of G. We have to show that there exists a set $I \subseteq \{1, \ldots, l\}$ such that $|I| \leq t$ and if $a \in S$ then there exists $i \in I$ such that $a \in S_i$. Define

$$i \in I \quad \Longleftrightarrow \quad c(v[S_i]) = \text{big}(v[S_i]). \tag{9}$$

Consider a vertex $b[a] \in B$. By the definition of $L(b[a])$ we have that

$$L(b[a]) = \bigcup_{i=1,\ldots,l: a \in S_i} L(v_i[a]) \tag{10}$$

and

$$|L(b[a])| = |\{v_i[a] : a \in S_i, i = 1, \ldots, l\}|. \tag{11}$$

Equations (10), (11) and Lemma 2 imply that $c(b[a]) = c(v_i[a])$ for some $i \in \{1, \ldots, l\}$. By Lemma 6 we have that $c(v[S_i]) > c(v_i[a])$ which implies (by the definition of $\mathcal{L}$) $c(v[S_i]) = \text{big}(v[S_i])$. Thus, $i \in I$. It remains to show that $|I| \leq t$. By Lemma 5 all the vertices in B get pairwise different colors. By Lemma 7, if $v[S_i]$ gets big color then it is forbidden for all vertices in B. Thus, at most t vertices in $\{v[S_1], \ldots, v[S_l]\}$ get big color, so by (9) there are at most t indicies in I. $\qquad \square$

3 Instances of the reduction

In this section we show several examples of trees satisfying properties (p_2) and (p_3). All trees considered are rooted at vertex r. Since all graphs in this section are trees, we will use symbol T instead of G.

3.1 Bounded height trees

Let the subgraph induced by $\{v[S_i]\} \cup V(S_i)$ be a star with the central vertex $v[S_i]$ and leaves in $V(S_i)$. Tree T contains vertex r which is adjacent to all vertices $v[S_1], \ldots, v[S_l], b[a_1], \ldots, b[a_n], b_1, \ldots, b_{l-t}$. So, $T = (V(T), E(T))$, where

$$
\begin{aligned}
V(T) &= \{r\} \cup W \\
E(T) &= \{\{r, v\} : v \in B \cup \{v[S_1], \ldots, v[S_l]\}\} \\
&\quad \cup \{\{v[S_i], v\} : i = 1, \ldots, l, v \in V(S_i)\}.
\end{aligned}
$$

Theorem 2 *Vertex List Ranking problem is NP-complete for trees of height bounded by 2.*

Proof: The problem is in NP. If we root T at vertex r then each path connecting r to a leaf is of the form r-$v[S_i]$-$v_i[a_j]$ or r-b_i or else r-$b[a_i]$. Thus, the height of T is 2. One can check that T satisfies (p_2) and (p_3). $\mathcal{L}$ is defined as in (1) – (7). Thus, in this case $V(T) \setminus W = \{r\}$ and $d = 1$. By Theorem 1 we have that for a given instance of the Set Cover problem there exists a solution of cardinality at most t if and only if there exists a vertex $\mathcal{L}$-list ranking of T. The size of $V(T)$ is $1 + 2l - t + n + \sum_{i=1}^{l} |S_i|$, which is polynomial in the size of the input for the Set Cover problem. $\qquad \square$

3.2 Full binary trees

Let T_v, where $v \in V(T)$ be the subtree rooted at vertex v, i.e. subtree of T induced by v and all its descendants. Let T be a full binary tree such that the number of leaves in T is the smallest power of 2 greater than or equal to

$$(l + 1) \cdot 2^{\lceil \log_2(n+l-t) \rceil}.$$

Let $X_i, i = 1, \ldots, l+1$ be any subset of leaves of T such that

$$|X_i| = 2^{\lceil \log_2(n+l-t) \rceil}$$

and for their closest common ancestor x_i we have that the set of leaves in T_{x_i} equals to X_i. For each $i = 1, \ldots, l$ we place vertices $v[S_i], v_i[a_j]$ in tree T in such a way that $v[S_i] = x_i$ and $V(S_i) \subseteq X_i$. We assume that $x_i \neq x_j$ for $i \neq j$. This definition is correct because

$$|V(S_i)| \leq n \leq 2^{\lceil \log_2(n+l-t) \rceil} = |X_i|$$

105

for $i = 1, \ldots, l$. Finally, let

$$\{b[a_i] : i = 1, \ldots, n\} \cup \{b_i : i = 1, \ldots, l - t\} \subseteq X_{l+1}. \tag{12}$$

Theorem 3 *The Vertex List Ranking problem is NP-complete for full binary trees.*

Proof: It is easy to check that (p_2) and (p_3) hold for T. The size of T is polynomial in $l + n$. By Theorem 1 there exists a solution I, $|I| \leq t$, to the Set Cover problem if and only if there exists a vertex $\mathcal{L}$-list ranking of T. □

3.3 Comets

A *comet* C_{n_1,n_2} is a tree with $n_1 + n_2$ vertices, which contains a path P_{n_1} as a subgraph and one end vertex of this path is adjacent to n_2 leaves. Define C_{n_1,n_2} such that the path P_{n_1} contains vertices in $(W \setminus B) \cup \{r\}$ in the following order

$$r, v[S_1], \pi_1^{-1}(1), \ldots, \pi_1^{-1}(|V(S_1)|), \ldots$$

$$\ldots, v[S_i], \pi_i^{-1}(1), \ldots, \pi_i^{-1}(|V(S_i)|), \ldots, v[S_l], \pi_l^{-1}(1), \ldots, \pi_l^{-1}(|V(S_l)|).$$

Finally, all the vertices in B are adjacent to r. The graph created in this way is a comet C_{n_1,n_2} with $n_1 = 1 + l + (|S_1| + \cdots + |S_l|)$ and $n_2 = |B|$. As before, the above graph fulfills (p_2) and (p_3) and its size is polynomial in $n + l$. Thus, by Theorem 1 we obtain the following

Theorem 4 *Vertex List Ranking problem is NP-complete for comets.* □

3.4 Hard instances for edge list ranking

We use symbol $p[v]$, where $v \in V(T) \setminus \{r\}$ to denote the parent of vertex v in tree T. All trees T are rooted at vertex r and $r \notin W$.

Let G be the line graph of T. In order to distinguish the sets W, B for T and G we denote them by $W(T), B(T)$ and $W(G), B(G)$, respectively. The vertices of G are denoted as follows. The vertex v in $V(G)$ corresponds to edge $\{v, p[v]\} \in E(T)$. Thus, in particular, $W(G) = \{\{v, p[v]\} \in E(T) : v \in W(T)\}$.

Lists $\mathcal{L}$ for vertices of G are defined by equations $(1) - (7)$, where parameter $d = \chi'_r(T[V(T) \setminus W(T)]) = \chi_r(G[V(G) \setminus W(G)])$.

Lemma 8 *If T satisfies properties $(p_2), (p_3)$ and for all vertices $u \in B(T), v \in W(T) \setminus B(T)$ it holds that r is the only common ancestor of u and v in T, then its line graph G defined above also satisfies (p_2) and (p_3).*

Proof: First, we prove (p_2) for G. Assume, that P is a u-v path in G, where $u \in B(G)$ and $v \in V(S_i) \subseteq V(G)$ for some $i \in \{1, \ldots, l\}$. Consider the subgraph T' of T, such that $E(T') = V(P)$. Assume, for a contrary, that the vertex $v[S_i]$ of G does not belong to P. Some subgraph of T' is a u'-v' path P' in T,

106

where $\{u', p[u']\} = u$ and $\{v', p[v']\} = v$. The only vertex of P', denoted by w', such that $\{w', p[w']\} \notin V(P)$ is the closest common ancestor of u' and v'. By assumption $w' \neq v[S_i]$, which means that (p_2) does not hold for T – a contradiction.

To prove (p_3) for G we use the fact argumented above: if some path P connects $u \in B(G)$ and $v \in V(S_i) \subseteq V(G)$ then the corresponding path P' contains only one vertex w' such that $\{w', p[w']\} \notin V(P)$. Since $w' = r$, we have that if (p_3) does not hold for G then this property does not hold for T as well. □

Observe that if we root trees T used in Theorems 2, 3 and 4 at vertex r, then T fulfills assumptions of Lemma 8. We obtain the following

Theorem 5 *Edge List Ranking problem is* NP-*complete for the following trees: with diameter at most 2, full binary and comets.*

Proof: Let T be a tree created for Theorems 2, 3 and 4. By Lemma 8 the line graph of T satisfies (p_2) and (p_3). $\mathcal{L}$ is defined by (p_1) for the edges of T. By Theorem 1 there exists a vertex $\mathcal{L}$-list ranking of G if and only if there exists a solution of cardinality at most t for the Set Cover problem. The thesis of the theorem follows. □

4   Polynomial-time cases

Define $\chi_r(G, \mathcal{L})$ to be the smallest number k such that there exists a vertex $\mathcal{L}$-list k-ranking of G. If no vertex $\mathcal{L}$-list ranking of G exists then $\chi_r(G, \mathcal{L})$ is *undefined*. Symbol $\chi_r(G, \mathcal{L})$ is called the $\mathcal{L}$-*list ranking number* of G. In this section we consider the optimization version of the List Ranking problem, where for given G and $\mathcal{L}$ we want to find a vertex $\mathcal{L}$-list $\chi_r(G, \mathcal{L})$-ranking of G. Denote $L = \sum_{i=1,\ldots,n} |L(v_i)|$.

A separator S of a graph G is *minimal* if no proper subset of S is a separator in G. There exists a formula for computing the vertex ranking number for any graph $G \neq K_n$ [2]:

$$\chi_r(G) = \min_{S \in \mathcal{S}(G)} \{|S| + \max\{\chi_r(G_i) : i = 1, \ldots, j\}\}, \qquad (13)$$

where $\mathcal{S}(G)$ is the set of all minimal separators of G and $G_1, \ldots, G_j$ are the connected components of $G - S$. We want to give a formula analogous to (13) which holds for list rankings. Then, we use it to prove that the list ranking number of a path can be determined in polynomial time.

If G' is a subgraph of G and $\mathcal{L}$ is the set of lists for vertices of G then, in order to simplify the formulas, we write $\chi_r(G', \mathcal{L})$ instead of $\chi_r(G', \{L(v) \in \mathcal{L} : v \in V(G')\})$. If $G_1 \cup \cdots \cup G_j$ are some graphs and all numbers $\chi_r(G_i, \mathcal{L})$, $1 \leq i \leq j$ are defined then $\chi_r(G_1 \cup \cdots \cup G_j, \mathcal{L})$ is defined and

$$\chi_r(G_1 \cup \cdots \cup G_j, \mathcal{L}) = \max\{\chi_r(G_i, \mathcal{L}) : i = 1, \ldots, j\}.$$

If at least one number $\chi_r(G_i, \mathcal{L})$, $i \in \{1, \ldots, j\}$ is undefined, i.e. the vertex $\mathcal{L}$-list ranking of G_i does not exist, then $\chi_r(G, \mathcal{L})$ is undefined because the vertex $\mathcal{L}$-list ranking of G does not exist as well. Now, let us assume that we want to assign the biggest label to some vertex $v \in V(G)$ in some $\mathcal{L}$-list ranking c_v of a connected graph G. Since c_v is a ranking, such a vertex is unique. Let $G_1, \ldots, G_j$ be the connected components of $G - v$. Then, ranking c_v can properly color all vertices of G if and only if $\chi_r(G_1 \cup \ldots \cup G_j, \mathcal{L})$ is defined and

$$\{i \in L(v) : i > \max\{\chi_r(G_i, \mathcal{L}) : i = 1, \ldots, j\}\} \neq \emptyset.$$

If a ranking c_v exists then the biggest label used by c_v is as small as possible if and only if it is defined as follows

$$c_v(v) = \min\{i \in L(v) : i > \max\{\chi_r(G_i, \mathcal{L}) : i = 1, \ldots, j\}\}.$$

If for each vertex $v \in V(G)$ vertex list ranking c_v of G does not exists then $\chi_r(G, \mathcal{L})$ is undefined. Otherwise $\chi_r(G, \mathcal{L}) = \min\{c_v(v) : v \in V(G)\}$. Thus, we obtained the following

Theorem 6 *For a given connected graph G and a list assignment $\mathcal{L}$, if $\chi_r(G, \mathcal{L})$ exists then*

$$\chi_r(G, \mathcal{L}) = \begin{cases} \min(L(v)) & \text{if } n = 1, \\ \min_{v \in V(G)}\{\min\{i \in L(v) : i > \chi_r(G_1 \cup \cdots \cup G_j, \mathcal{L})\}\} & \text{if } n > 1, \end{cases}$$

where $G_1, \ldots, G_j$ are the connected components of $G - v$. $\square$

We show that this formula can be used to design a polynomial-time algorithm for computing an optimal vertex $\mathcal{L}$-list ranking of a path P. Note that if $v \in V(P)$ then $P - v$ contains one (if v is an end vertex of P) or two (if v is an internal vertex of P) connected components. By Theorem 6 we have to compute the vertex list ranking numbers of connected components of $P - v$ for all vertices $v \in V(P)$. If for all $v \in V(P)$ the vertex list ranking number of at least one connected component of $P - v$ is undefined or the biggest color in $L(v)$ is not bigger than the list ranking number of some connected component of $P - v$ then $\chi_r(P, \mathcal{L})$ is undefined. Otherwise, the value of $\chi_r(P, \mathcal{L})$ can be computed. Let us denote $P = (\{v_1, \ldots, v_n\}, \{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}\})$ and $P_{i,j} = P[\{v_i, \ldots, v_j\}]$ for $j \geq i$. We find list ranking numbers of subpaths $P_{i,j}$ according to increasing values of $j - i$. In order to compute $\chi_r(P_{i,j}, \mathcal{L})$ we try to assign the biggest label to a vertex v_r for each $r = i, \ldots, j$. This can be done in linear time if the label for v_r can be determined in constant time. We use two 2-dimensional tables t_l and t_r such that

$$t_l[i, j] = \min\{k \in L(v_i) : k > \chi_r(P_{i+1,j}, \mathcal{L})\}$$

and

$$t_r[i, j] = \min\{k \in L(v_j) : k > \chi_r(P_{i,j-1}, \mathcal{L})\}.$$

Observe that if $i > j$ then $\chi_r(P_{i,j}, \mathcal{L}) = 0$. Then, $\max(t_r[i, r], t_l[r, j])$ is the desired value for the vertex v_r. If the value of $\chi_r(P_{i,j}, \mathcal{L})$ has been computed then we find the values of $t_l[i-1, j], i > 1$ and $t_r[i, j+1], j < n$ as follows

$$t_l[i-1, j] = \min\{i' \in L(v_{i-1}) : i' > \max\{\chi_r(P_{i,j}, \mathcal{L}), t_l[i, j]\}\},$$

$$t_r[i, j+1] = \min\{i' \in L(v_{j+1}) : i' > \max\{\chi_r(P_{i,j}, \mathcal{L}), t_r[i, j]\}\}.$$

Thus, the cost of processing the lists in $\mathcal{L}$ in this algorithm is $O(L)$. So, Theorem 6 and the dynamic programming technique gives the following

Theorem 7 *The list ranking number of a path can be computed in $O(n^3 + L)$ time.* □

In Section 3 we have proved the NP-completeness results for vertex and edge list rankings. By an *internal* vertex (edge) we mean a vertex (edge) which is not a leaf (is not adjacent to a leaf, respectively). Observe that in the case of bounded height trees defined in Subsection 3.1 the internal vertices (edges) are $\{r, v[S_1], \ldots, v[S_l]\}$ (respectively $\{v[S_1], \ldots, v[S_l]\}$). This means that the Vertex List Ranking and the Edge List Ranking problems are hard for trees even if the lists of permissible colors for the internal vertices/edges have sizes at most 2. In the remaining of this section we prove that if internal vertices (edges) are precolored (sizes of lists are equal to 1) then the vertex (edge) list ranking problem is polynomially solvable. We also show that if the sizes of mentioned lists are not bounded, then the problem is polynomially solvable if the number of internal vertices (edges) is bounded. For a given graph G, define $S(G)$ to be the subset of vertices of G such that $v \in S(G)$ if and only if there is no minimal separator in G containing v. Note that we consider a more general problem, because a vertex can be internal in G (according to the definition above) and may not belong to $S(G)$. We will show that it can be determined in polynomial time whether a vertex list ranking $c_{S(G)}$ of $G[S(G)]$ can be extended to a vertex list ranking of G.

Assume that a valid vertex $\mathcal{L}$-list ranking $c_{S(G)}$ of $G[S(G)]$ is given. For G, $\mathcal{L}$ and $c_{S(G)}$ define a bipartite graph $G_B = (V_1 \cup V_2, E)$, where

$$V_1 = V(G) \setminus S(G)$$
$$V_2 = \{v_i : v \in V(G) \setminus S(G) \text{ and } i \in L(v)\}.$$

So, vertices in V_1 correspond to uncolored vertices of G and vertices in V_2 correspond to colors available for these vertices. Note that, in general, we have two different vertices in V_2 corresponding to the same color which appears in lists of two different vertices in $V(G) \setminus S(G)$. In the case described below we "merge" two such vertices in V_2. If $u, v \notin S(G)$, color i belongs to $L(u) \cap L(v)$ and there exists a path in G connecting vertices u and v, which contains vertices in $S(G)$ colored by $c_{S(G)}$ with labels smaller than i, then let $u_i = v_i$ in G_B. Thus, u_i and v_i are in that case the two labels denoting the same vertex in G_B. Finally, for a vertex $v \in V_1$ and $i \in L(v)$ we have $\{v, v_i\} \in E(G_B)$ if and only if there

is no path P connecting v and a vertex $u \in S(G)$ such that $V(P) \subseteq S(G)$, $c_{S(G)}(u) = i$ and for each vertex $w \in V(P)$ we have $c_{S(G)}(w) < i$.

This reduction is illustrated in Fig. 2. A graph G is given in Fig. 2(a) along with a list assignment for vertices in $V(G) \setminus S(G)$. The vertices in $S(G)$ are precolored by some function $c_{S(G)}$. Fig. 2(b) depicts the bipartite graph G_B obtained from G, $\mathcal{L}$ and $c_{S(G)}$.
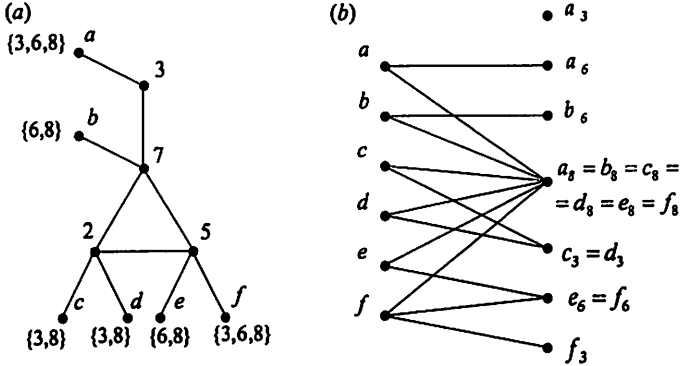

Figure 2: (a) some graph G, lists $\mathcal{L}$ for vertices in $V(G) \setminus S(G)$ and partial list ranking $c_{S(G)}$; (b) the corresponding graph G_B.

Lemma 9 *There exists a vertex $\mathcal{L}$-list ranking c of G such that $c|_{S(G)} = c_{S(G)}$ if and only if there exists a matching of size $|V_1|$ in the graph G_B.*

Proof: Assume that c is a proper vertex $\mathcal{L}$-list ranking of G such that $c|_{S(G)} = c_{S(G)}$. We have to show that each vertex in V_1 is matched in some matching M of G_B. Define

$$M = \{\{v, v_{c(v)}\} : v \in V_1\}. \tag{14}$$

Thus, for each vertex $v \in V_1$ there exists an edge $e \in M$ such that $v \in e$. Observe that if $c(u) \neq c(v)$ for $u, v \in V_1$ then edges $\{u, u_{c(u)}\}, \{v, v_{c(v)}\}$ do not share a vertex in G, which follows directly from the definition of G_B. So, we consider the case when $\{u, u_i\}, \{v, v_i\} \in M$ and $u_i = v_i$. Then, we would have that u and v get the same color i under c and there is a path P contained in $S(G)$ (and thus in G) such that each vertex in P gets a color smaller than i. This would give a contradiction, because we assumed that c is a vertex ranking of G. Thus, no two edges in M are adjacent. The size of M is $|V_1| = |V(G) \setminus S(G)|$.

Now, let us assume that M is a matching in G_B such that each vertex in V_1 is matched, i.e. for each $v \in V_1$ there exists an edge $\{v, v_i\} \in M$. Let v get color i in vertex $\mathcal{L}$-list ranking of G iff $\{v, v_i\} \in M$. We show that function c defined in this way is a vertex $\mathcal{L}$-list ranking of G. By the definition of G_B, $i \in L(v)$. Let $u \neq v$, $u \in V(G)$ be a vertex such that $c(u) = i$. Assume, for

a contradiction, that there exists a path P in G such that all vertices in P get colors smaller than i in ranking c and the end vertices of P are u and v (in this case let u, v belong to P). Consider any three consecutive vertices v_1, v_2, v_3 of P. If $v_2 \notin S(G)$ then $\{v_1, v_3\} \in E(G)$ because otherwise every separator which disconnects v_1 and v_3 in G would contain v_2, so some minimal separator would contain v_2, which means (by the definition of $S(G)$) that $v_2 \in S(G)$. Thus, if $v_2 \notin S(G)$ then we can create path P_1 by removing v_2 from P. The end vertices of P_1 are u, v and all internal vertices of P_1 have smaller colors than i in ranking c. Thus, after a finite number of such steps we obtain path P_j such that end vertices of P_j are u, v, all internal vertices of P_j belong to $S(G)$ and have colors smaller than i under ranking c. If $u \in S(G)$ then we would have a contradiction, because vertex u is precolored with i, which means that v could not be adjacent to v_i in G_B, so $\{v, v_i\} \notin M$. On the other hand, if $u \notin S(G)$ then the existence of the u-v path P_j implies that $u_i = v_i$ in graph G_B, which implies that $\{u, u_i\}, \{v, v_i\} \in M$ and these edges are adjacent in G_B — a contradiction. Thus, we have proved that if for $u, v \in V(G)$, $c(u) = c(v) = i$ then each path connecting u and v in G contains a vertex with a color greater than i. $\quad\square$

From Lemma 9 it follows that we can solve the Vertex List Ranking problem with precolored vertices in $S(G)$ by finding a matching of size $|V_1|$ in graph G_B. The complexity of finding maximum cardinality matching in a bipartite graph is $O(mn)$. The number of vertices in G_B is bounded by

$$|V(G) \setminus S(G)| + \sum_{v \in V(G) \setminus S(G)} |L(v)| = O(L)$$

and the number of edges in G_B is $O(L)$. The complexity of the above algorithm is $O(L^2)$. Furthermore, if M is an appropriate matching in G_B then each edge in M defines a color for vertex in $V(G) \setminus S(G)$. So, we obtained the following

Corollary 1 *For a given $\mathcal{L}$ and G, we can determine in $O(L^2)$ time whether vertex $\mathcal{L}$-list ranking $c_{S(G)}$ of $G[S(G)]$ can be extended to a vertex $\mathcal{L}$-list ranking of G.* $\quad\square$

We assumed that colors for vertices in $S(G)$ have been determined, or equivalently, each vertex in $S(G)$ has a list of permissible colors of size 1. The following theorem extends our results to graphs with fixed $|S(G)|$ and arbitrary sizes of list of permissible colors for vertices in $S(G)$. We also want to compute optimal list rankings.

Theorem 8 *Given a graph G and a list assignment $\mathcal{L}$ for the vertices of G. If the size of $S(G)$ is fixed then $\chi_r(G, \mathcal{L})$ and an optimal vertex $\mathcal{L}$-list ranking of G can be computed in polynomial time $O(L^3 \cdot \prod_{v \in S(G)} |L(v)|)$.*

Proof: If $|S(G)|$ is bounded then the number of all vertex labelings of vertices in $S(G)$ is polynomially bounded in the sizes of the lists in $\mathcal{L}$. For any such labeling $c_{S(G)}$ we can check in time polynomial in $|S(G)|$ whether $c_{S(G)}$ is a

vertex ranking of $G[S(G)]$. If $c_{S(G)}$ is a ranking then we create the corresponding bipartite graph G_B and compute a matching of size $|V(G) \setminus S(G)|$ in G_B. If such a matching does not exist then, by Lemma 9, we have that there is no vertex $\mathcal{L}$-list ranking c of G such that c restricted to $S(G)$ is equal to $c_{S(G)}$. On the other hand, if such a matching has been found then we know that $c_{S(G)}$ can be extended to a vertex $\mathcal{L}$-list ranking c of G. However, in order to compute $\chi_r(G, \mathcal{L})$ and an optimal vertex $\mathcal{L}$-list ranking of G we have to guarantee that the largest label used by c is as small as possible. For the graph G_B define a weight function $w \colon E(G_B) \to \mathbb{R}_+$ as follows

$$w(\{v, v_i\}) = 1 + \sum_{\{u, u_j\} \in E(G_B), j < i} w(\{u, u_j\}) \tag{15}$$

for each $\{v, v_i\} \in E(G_B)$. For a matching M define

$$w(M) = \sum_{e \in M} w(e).$$

Without loss of generality, we may assume that $|V_1| \le |V_2|$ because otherwise we know that no vertex $\mathcal{L}$-list ranking exists. We find a matching M of size $|V_1|$, such that for any other matching M' (with $|M'| = |V_1|$) it holds $w(M) \le w(M')$. We have to prove that a matching of minimum weight in G_B gives an optimal vertex $\mathcal{L}$-list ranking c in G. Let M' be a matching of size $|V_1|$, $w(M') > w(M)$ and c' be the corresponding vertex list ranking of G. Assume, for a contradiction, that the largest labels used by c' and c are j and $k > j$, respectively. By (15) we have

$$w(M') \le \sum_{\{v, v_i\} \in E(G_B), i \le j} w(\{v, v_i\}) \tag{16}$$

and from the fact that $\{v, v_k\} \in M$ it follows

$$w(M) \ge w(\{v, v_k\}). \tag{17}$$

We assumed that $k > j$, so combining equations (15), (16) and (17) we obtain

$$w(\{v, v_k\}) > \sum_{\{v, v_i\} \in E(G_B), i \le j} w(\{v, v_i\}),$$

which implies that $w(M) > w(M')$ — a contradiction. The complexity of finding minimum weight perfect matching in G_B is $O(L^3)$ and the thesis of the theorem follows. $\qquad \square$

Observe that if vertex v is a leaf in a tree T then $v \notin S(G)$. Furthermore, the vertex v' in the line graph of T (denoted by T'), corresponding to the edge adjacent to a pendant vertex in T also does not belong to set $S(T')$. Thus, Theorem 8 implies that if T is such a tree that the number of nonleaf vertices is bounded then an optimal vertex (edge) $\mathcal{L}$-list ranking of T can be computed in polynomial time. In particular, we have

Corollary 2 *If T is a tree with diameter at most 3 and $\mathcal{L}$ is a list assignment for T, then optimal vertex (edge) $\mathcal{L}$-list rankings of T can be obtained in $O(|L(u)| \cdot |L(v)| \cdot L^3)$ $(O(L(e)L^3)$, respectively) time, where u, v are internal vertices (e is an internal edge, respectively) of T.* □

5 Concluding remarks

We have proved that the Vertex and Edge List Ranking problems are NP-complete for very simple classes of graphs. The problem of determining the smallest k such that there exists vertex (edge) $\mathcal{L}$-list k-ranking is also NP-hard for trees even if $\mathcal{L}$ is created in a way that guarantees that a vertex (edge) ranking exists. To prove this observe that there exists a vertex $\mathcal{L}$-list ranking of G if and only if there exists vertex $\mathcal{L}'$-list k-ranking of G, where $\mathcal{L}' = \{L'(v) : v \in V(G)\}$,

$$k = \max(\bigcup_{v \in V(G)} L(v))$$

and

$$\forall_{v \in V(G)} L'(v) = L(v) \cup \{k+1, k+2, \ldots, k+|V(G)|\}.$$

It is easy to see that a vertex $\mathcal{L}'$-list ranking of G always exists.

The complexity results presented in this paper are summarized in Table 1, where VLR and ELR stand for Vertex List Ranking and Edge List Ranking, respectively. Symbols $\mathcal{L}_i$ ($\mathcal{L}'_i$) are used to denote that the sizes of lists of nonleaf vertices (edges not adjacent to leaves, respectively) are bounded by a constant i.

Table 1: The complexity of list ranking of graphs.

class of graphs	VLR	ELR
trees with diameter ≥ 4	NPC	NPC
trees with diameter < 4	$O(|L(u)||L(v)|L^3)$	$O(|L(e)|L^3)$
trees with $\Delta > 2$	NPC	NPC
paths ($\Delta \leq 2$)	$O(n^3 + L)$	$O(n^3 + L)$
comets	NPC	NPC
interval graphs	NPC	NPC
circular-arc graphs	NPC	NPC
permutation graphs	NPC	NPC
graphs with $\mathcal{L}_i$, $i \leq 2$	NPC	—
graphs with $\mathcal{L}_i$, $i = 1$	$O(L^3)$	—
graphs with $\mathcal{L}'_i$, $i \leq 2$	—	NPC
graphs with $\mathcal{L}'_i$, $i = 1$	—	$O(L^3)$

Since comets are interval graphs, we obtain that Vertex and Edge List Ranking problems are NP-complete for interval graphs, and thus for circular-arc and

permutation graphs. Note, that polynomial-time algorithms for vertex ranking exist for these three classes of graphs [2]. Vertices u, v in the second row of Table 1 denote the internal vertices of a graph, while e is the internal edge of a graph. In the case of polynomial-time instances Table 1 gives the complexity of finding the list ranking number of a graph.

References

[1] H. Bodlaender, J.S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, Z. Tuza, Rankings of graphs, *SIAM J. Discrete Math.* **11** (1998) 168-181.

[2] J.S. Deogun, T. Kloks, D. Kratsch, H. Müller, On the vertex ranking problem for trapezoid, circular-arc and other graphs, *Discrete Appl. Math.* **98** (1999) 39-63.

[3] D. Dereniowski, Edge ranking of weighted trees, *Discrete Appl. Math.* **154** (2006) 1198-1209.

[4] D. Dereniowski, A. Nadolski, Vertex rankings of chordal graphs and weighted trees, *Infor. Process. Lett.* **98** (2006) 96-100.

[5] A.V. Iyer, H.D. Ratliff, G. Vijayan, Parallel assembly of modular products - an analysis, *Tech. Report* **88-06**, Georgia Institute of Technology, 1988.

[6] R.E. Jamison, Coloring parameters associated with rankings od graphs, *Congressus Numerantum* **164** (2003) 111-127.

[7] T.W. Lam, F.L. Yue, Edge ranking of graphs is hard, *Discrete Appl. Math.* **85** (1998) 71-86.

[8] T.W. Lam, F.L. Yue, Optimal edge ranking of trees in linear time, *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms* (1998) 436-445.

[9] J.W.H. Liu, The role of elimination trees in sparse factorization, *SIAM J.Matrix Analysis and Appl.* **11** (1990) 134-172.

[10] K. Makino, Y, Uno, T. Ibaraki, Minimum edge ranking spanning trees of threshold graphs, *LNCS* **2518** (2002) 428-440.

[11] A.A. Schäffer, Optimal node ranking of trees in linear time, *Infor. Process. Lett.* **33** (1989/90) 91-96.