

Edge ranking of weighted trees

Dariusz Dereniowski*

Department of Algorithms and System Modeling,
Gdańsk University of Technology, Poland

deren@eti.pg.gda.pl

Abstract: In this paper we consider the edge ranking problem of weighted trees. We prove that a special instance of this problem, namely edge ranking of multitrees is NP-hard already for multitrees with diameter at most 10. Note that the same problem but for trees is linearly solvable. We give an $O(\log n)$ -approximation polynomial time algorithm for edge ranking of weighted trees.

Keywords: approximation, edge ranking, multigraph, NP-completeness, polynomial algorithm

1 Introduction

For a given multigraph $G = (V(G), E(G))$, an *edge k -ranking* of its edges is defined as a function $c : E(G) \rightarrow \{1, \dots, k\}$ such that for every pair of edges x, y such that $c(x) = c(y)$ each path connecting x and y contains an edge with a greater color. Symbol $\chi'_r(G)$ is used to denote the smallest integer k such that G has an edge k -ranking.

The edge ranking problem of trees was intensively studied. The first published algorithm was 2-approximate [5]. However, it was an open question if an optimal edge ranking could be found efficiently. Authors in [2] gave the first polynomial time algorithm and thus placed the problem in class P. Now it is known a linear time algorithm for edge ranking of trees [7].

In the case of general graphs it has been shown that the edge ranking problem is NP-hard [6]. In addition, no polynomial time approximation algorithm using at most $m^{1/2-\epsilon}$ more labels than the optimal value can exist unless P=NP, where $\epsilon > 0$ is fixed [6]. However, there exists an $O(\log^2 n)$ -approximation algorithm for ranking the vertices of a graph [1] (definition of the vertex ranking problem is similar to the definition of edge ranking given above). Thus, we can use it for line graphs, which gives efficient algorithm for finding edge ranking with approximation ratio equal to $O(\log^2 m)$. Clearly, this method can be used to

*Partially supported by KBN grants 3 T11C 01127 and 4 T11C 04725

the edge ranking problem of multigraphs as well, but it cannot be applied in the case of weighted graphs.

The edge ranking problem has several interesting generalizations. The *c-edge ranking* of a graph G is a function mapping its edges into integers such that each connected component of the subgraph of G containing edges with colors less than or equal to i , has at most c edges labeled with i . Although the problem is defined for general graphs, it is interesting for trees because of potential practical applications. Note that if $c = 1$ then this problem is equivalent to the ordinary edge ranking defined previously. There exists an optimal algorithm for finding the c -edge ranking of a given tree, with running time $O(n^2 \log \Delta)$, where Δ is the maximum vertex degree of the graph [10].

In the paper we study another generalization of the edge ranking problem, i.e. edge ranking of weighted graphs. The next section gives necessary definitions and describes a potential application of the edge ranking problem of weighted trees. We show that edge ranking of multigraphs can be considered as a special case of edge ranking of weighted graphs. Section 3 gives a polynomial time reduction from the satisfiability problem to the edge ranking problem of multitrees with diameter bounded by 10. This implies that edge ranking of weighted graphs is hard as well. In Section 4 we analyze $O(n \log n)$ -time approximate algorithm for edge ranking of weighted trees. The approximation ratio of the algorithm is $O(\log n)$ and we prove that this is asymptotically the best bound for this algorithm.

2 Definitions and motivation

The edge ranking problem of trees can be applied in parallel query processing in large database systems [3, 8, 9], or in parallel assembly of multi-part products from their components [4]. The second application is of special interest for us. We have a set V of parts of a product which are denoted by v_1, \dots, v_n and a set of operations

$$E \subset \{\{v_i, v_j\} : i, j = 1, \dots, n, i \neq j\}.$$

If $\{v_i, v_j\} \in E$ for some i, j then parts v_i, v_j have to be connected during the assembly of the product. We assume that the operations corresponding to elements in E can be performed in any order and for any two elements e_1, e_2 of E such that $e_1 \cap e_2 = \emptyset$ operations e_1, e_2 can be done in parallel. If we create the graph $G = (V, E)$ then the assembly operation $\{v_1, v_2\}$ of parts v_1 and v_2 can be modeled as transforming G such that the above vertices are replaced by a new vertex $[v_1; v_2]$ adjacent to all neighbors of v_1 and v_2 . In each step we can perform simultaneously many such contracting operations if they form an independent set of edges in G . We want to schedule these operations in order to minimize the number of parallel steps required to reduce G into the graph consisting of one vertex (we assume that G is connected, because otherwise we can find the schedule for each connected component of G independently). It can be shown that the minimum number of steps equals $\chi'_r(G)$. If we have an

optimal edge ranking of G then we can design such assembly by performing in the i th step operations corresponding to edges labeled with i .

If we use edge rankings of simple graphs to solve the problem then we do not care of the time of each operation, i.e. we assume that each assembly requires the same interval of size t . If this is not the case (each edge e has a weight $w(e) \in \mathbb{R}_+$ denoting the time required to complete this assembly operation) then we can apply an edge ranking algorithm for the corresponding unweighted graph. If we schedule the operations as described above then the i th step requires time interval of size $\max\{w(e) : e \in c^{-1}(i)\}$. However, this is an approximate algorithm.

If edges are weighted then we generalize the definition of edge ranking. The edges of graph G are labeled by intervals, i.e. we define a function $c: E(G) \rightarrow \{[a, b) : 0 \leq a < b\}$ such that $|c(e)| = w(e)$ for each $e \in E(G)$ and if for any two edges $e_1, e_2 \in E(G)$ we have $c(e_1) \cap c(e_2) \neq \emptyset$ then each path connecting e_1 to e_2 contains an edge e such that for each $x \in c(e)$ and $y \in c(e_1) \cup c(e_2)$ it holds $x > y$. Observe that without loss of generalization we can consider only such edge rankings c that $c(E(G))$ is an interval. An edge ranking c is *optimal* if $|c(E(G))|$ is as small as possible. The *edge ranking number* of a weighted graph G is defined as $\chi'_r(G) = |c(E(G))|$, where c is an optimal edge ranking. Thus, for each edge $e \in E(G)$, $c(e)$ defines the time interval for performing assembly operation corresponding to e . If there exists an integer I such that for any edge $e \in E(G)$, $I \cdot w(e)$ is an integer bounded by a polynomial in n then we convert the edge ranking problem of weighted graphs to the edge ranking problem of multigraphs by replacing each edge e of G by $I \cdot w(e)$ parallel edges. The next theorem shows that we can do so because if we can find any edge k -ranking of a multigraph G then we can also find an edge k -ranking of G such that colors assigned to parallel edges form a consecutive set of integers. In addition, the proof of Theorem 1 gives an efficient algorithm for transforming any edge ranking of a multigraph to a ranking with desired property. If $S \subset E(G)$ then $G - S$ denotes the subgraph $(V(G), E(G) \setminus S)$.

Theorem 1 *If G is a multigraph then there exists an edge $\chi'_r(G)$ -ranking c of G such that for each pair of adjacent vertices, the edges between them are labeled with consecutive integers under c .*

Proof: We prove this theorem by induction on the number of vertices of G . Clearly, the claim holds for $n = 1$. Let us assume that the hypothesis is true for some $n \geq 1$. Let G be a multigraph on $n + 1$ vertices and let c' be any edge k -ranking of G . Let $k' < k$ be the largest integer such that k' is not a unique label under c' , and let S be the set of edges labeled by integers $k' + 1, \dots, k$. Clearly, $G - S$ is disconnected, so S contains all edges e_1, \dots, e_l between some pair of vertices u and v . Thus, we can shuffle labels assigned to S so that e_1, \dots, e_l get colors $k - l + 1, \dots, k$. We can apply the induction hypothesis for two connected components of $G - \{e_1, \dots, e_l\}$. In this way we recursively create the edge k -ranking c . \square

Using Theorem 1 we can reduce the edge ranking problem of multitrees to the problem of edge ranking of weighted trees.

Corollary 1 *Let a multitree T be given. Define a tree T' and a weight function $w: E(T') \rightarrow \mathbb{R}_+$ such that for each pair of adjacent vertices u and v in T we have $\{u, v\} \in E(T')$ and $w(\{u, v\})$ equals the number of parallel edges between u and v in T . Then we have $\chi'_r(T) = \chi'_r(T')$. \square*

Fig. 1(a) shows a graph with 9 vertices, where the numbers denote the weights on the edges, i.e. the time needed to complete the corresponding assembly operation. Figures 1(b) and 1(c) depict an edge ranking of the simple graph and an edge ranking of the corresponding multigraph, respectively, where the number of edges between two vertices in the multigraph is equal to the weight of the edge between them in the source graph. We want to use an edge

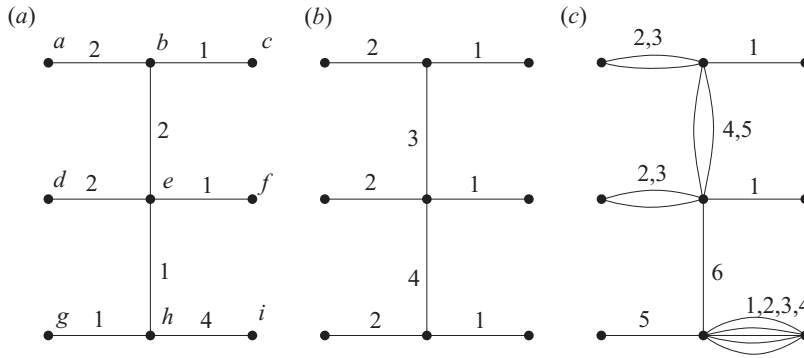


Figure 1: (a) a weighted graph; (b) edge ranking of the corresponding simple graph; (c) edge ranking of the corresponding multigraph

ranking algorithm to schedule the assembly operations of a multi-part product. If we use the straightforward approach, i.e. if we schedule the operations using ranking of a simple graph then the assembly can be depicted as shown in Fig. 2(a). However, if we use the same method of scheduling based on the edge ranking of a multigraph then we obtain the solution from Fig. 2(b) which is better than the previous one.

Note that in order to convert the weighted edge ranking problem to the problem of edge ranking of multigraphs we assumed that there exists integer I satisfying given conditions. If the integer I does not exist or I is too large then we can define I such that its value is bounded and place $\lceil I \cdot w(e) \rceil$ parallel edges in multigraph for each $e \in E(G)$. However, if we find the schedule by using edge ranking of the corresponding multigraph then, in general, the length of the schedule may not be optimal, i.e. by solving the weighted edge ranking problem we can create a schedule with shorter completion time. In Section 4 we consider the general case, i.e. our algorithm takes a weighted graph as an

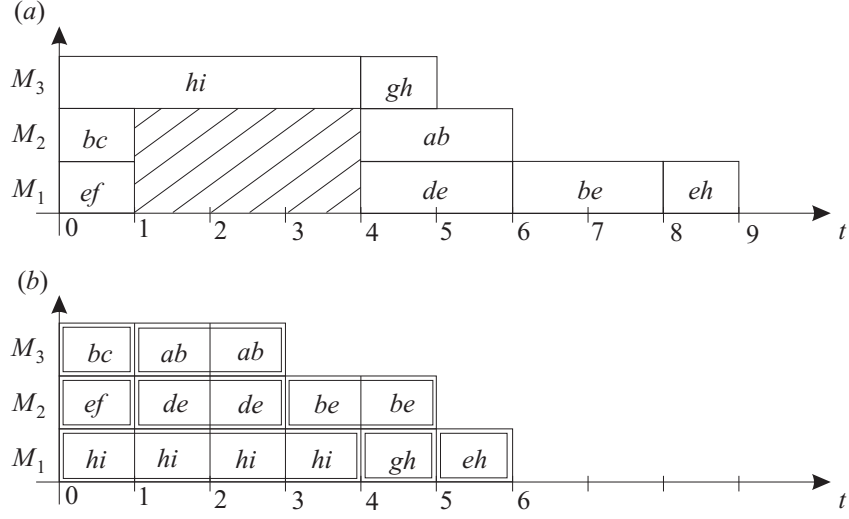


Figure 2: Assigning operations to machines according to (a) edge ranking of simple graph in Fig. 1(b); (b) edge ranking of multigraph in Fig. 1(c)

input. The NP-completeness result given in Section 3 clearly implies that the edge ranking problem of weighted graphs is also NP-hard.

Now we will give some basic definitions which will be used later on. If G is a rooted multitree then its *height* is the length (the number of edges) of a longest path from the root to a leaf. The *diameter* of a tree is defined as the length of its longest path. The graph $G + e$ stands for $(V(G), E(G) \cup \{e\})$, where e is an edge. Similarly we define graph $G + S$, where S is some set of edges. The *edge multiplicity* between two adjacent vertices of a multigraph G means the number of parallel edges between these nodes. We say that a color i is *visible* for edge (node) x if there exists an edge y labeled with i and all edges of some path between x and y have colors smaller than i . If the above path is empty then we say that the color i is *adjacent* (*incident*, respectively) to x . Symbol $G[S]$, where $S \subset V(G)$ is used to denote the *induced* subgraph of G defined as

$$(S, \{\{u, v\} \in E(G) : u, v \in S\}).$$

If c is an edge ranking of multigraph G and $L \subset E(G)$ then $c(L) = \{c(e) : e \in L\}$.

3 NP-hardness of edge ranking of multitrees

In this section we describe a polynomial reduction from the satisfiability problem (in particular 3-SAT) to the edge ranking problem of multitrees, proving that the latter is NP-complete. We use symbol $F(x_1, \dots, x_k)$ to denote the Boolean

expression of the form

$$(l_{1,1} + l_{1,2} + l_{1,3}) \cdot \dots \cdot (l_{s,1} + l_{s,2} + l_{s,3}),$$

where $l_{i,j} = x_t$ or $l_{i,j} = \overline{x_t}$ for each $i = 1, \dots, s, j = 1, 2, 3$ and $t \in \{1, \dots, k\}$, assuming that $\{x_1, \dots, x_k\}$ is the set of Boolean variables of formula F . We denote

$$F_i = (l_{i,1} + l_{i,2} + l_{i,3}) \quad \text{for each } i = 1, \dots, s.$$

In the decision problem 3-SAT we ask whether there exists an assignment of values *true* and *false* to variables x_1, \dots, x_k such that $F(x_1, \dots, x_k) = \text{true}$. Later in the section we will write F instead of $F(x_1, \dots, x_k)$.

We define $G_{k,s}$ as a rooted multtree as follows. Vertex v_0 is the root of $G_{k,s}$ and it is connected to nodes v'_0, v_1, \dots, v_k . For each $i = 1, \dots, k$ vertex v_i has two descendants: $v(x_i)$ and $v(\overline{x_i})$. These nodes correspond to the literals of the Boolean expression. In addition, if $i \geq 2$ then vertex v_i is connected to v'_i (edge multiplicity between v_i and v'_i equals to $(i-1)s$) which has a son denoted by v''_i (a leaf of $G_{k,s}$ with $2s$ incident edges). The edge multiplicity between v_0 and v'_0 (v_0 and v_i , $i = 1, \dots, k$) is $2s$ (s , respectively). Thus, the subscript s is the minimum edge multiplicity between every pair of adjacent nodes in $G_{k,s}$. In the following, symbol T_v is used to denote subgraph of $G_{k,s}$ induced by v and all its descendants. Fig. 3 shows the graph $G_{k,s}$. The numbers labeling edges of $G_{k,s}$ denote the edge multiplicity for each pair of adjacent vertices.

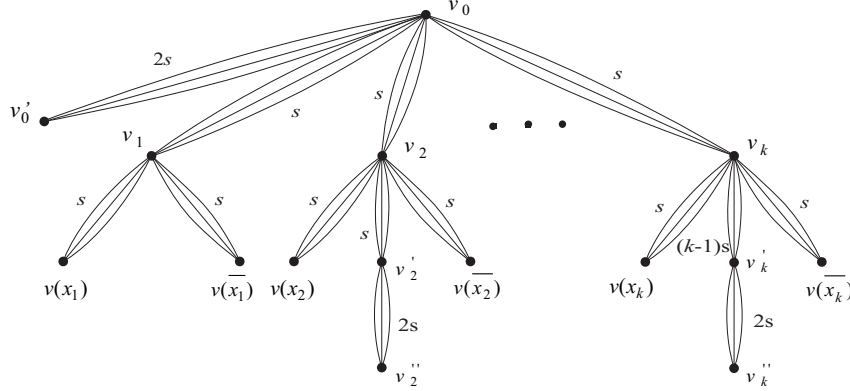


Figure 3: Graph $G_{k,s}$

Lemma 1 $\chi'_r(G_{k,s}) = (k+2)s$.

Proof: Define an edge ranking c of $G_{k,s}$ as follows:

$$\begin{aligned} c(E(G[\{v'_j, v''_j\}])) &= \{1, \dots, 2s\}, \\ c(E(G[\{v(x_i), v_i, v(\overline{x_i})\}])) &= \{1, \dots, 2s\}, \\ c(E(G[\{v_j, v'_j\}])) &= \{2s+1, \dots, (i+1)s\}, \\ c(E(G[\{v_0, v_i\}])) &= \{(i+1)s+1, \dots, (i+2)s\}, \\ c(E(G[\{v_0, v'_0\}])) &= \{1, \dots, 2s\}, \end{aligned}$$

for $i = 1, \dots, k, j = 2, \dots, k$. Clearly, c is a valid edge ranking, thus $\chi'_r(G_{k,s}) \leq (k+2)s$. The reverse inequality holds because $\Delta(G_{k,s}) = \deg(v_0) = (k+2)s$ and for any multigraph G , $\chi'_r(G) \geq \Delta(G)$. \square

Definition 1 Let G' be a multigraph and let $u_1, v_1, u_2, v_2, \dots, u_j, v_j$ be vertices of G' . Define $G'[u_1 \equiv v_1, \dots, u_j \equiv v_j]$ to be a multigraph obtained from G' by replacing vertices u_i and v_i by a new vertex $[u_i; v_i]$ for each $i = 1, \dots, j$ and $\{[u_i; v_i], x\} \in E(G)$ iff $\{u_i, x\} \in E(G')$ or $\{v_i, x\} \in E(G')$.

In the following, we write u_i or v_i instead of $[u_i; v_i]$, i.e. we assume that $u_i = v_i = [u_i; v_i]$ in graph G .

Now, for the purposes of the next two lemmas, we give the definitions of some multigraphs. Let $H_1, \dots, H_k, \overline{H}_1, \dots, \overline{H}_k$ be multigraphs with disjoint sets of vertices,

$$\chi'_r(H_i) \leq s \text{ and } \chi'_r(\overline{H}_i) \leq s$$

for each $i = 1, \dots, k$ and let $u_i \in V(H_i), \overline{u}_i \in V(\overline{H}_i), i = 1, \dots, k$. Let us denote $G' = G_{k,s} \cup H_1 \cup \dots \cup H_k \cup \overline{H}_1 \cup \dots \cup \overline{H}_k$. Finally, define

$$G = G'[v(x_1) \equiv u_1, v(\overline{x}_1) \equiv \overline{u}_1, \dots, v(x_k) \equiv u_k, v(\overline{x}_k) \equiv \overline{u}_k].$$

Less formally, the above operation can be considered as adding subgraphs H_i, \overline{H}_i to the multigraph $G_{k,s}$ by connecting them, respectively, to the vertices $v(x_i), v(\overline{x}_i)$ in $G_{k,s}$, $i = 1, \dots, k$. In the next two lemmas we identify the cases when $\chi'_r(G) = \chi'_r(G_{k,s})$ and $\chi'_r(G) > \chi'_r(G_{k,s})$, respectively.

Lemma 2 *If $\chi'_r(H_i) = 0$ or $\chi'_r(\overline{H}_i) = 0$ (i.e. H_i or \overline{H}_i , respectively, contains a single vertex) for each $i = 1, \dots, k$ then $\chi'_r(G) = \chi'_r(G_{k,s}) = (k+2)s$.*

Proof: We prove by induction on k that the edge ranking c , defined in the proof of Lemma 1 is the only optimal coloring of $G_{k,s}$.

If $k = 1$ then the claim holds. Assume that the hypothesis is true for some $k \geq 1$ and we will prove it for $k+1$. We know that

$$\chi'_r(G_{k,s}) = (k+2)s = \chi'_r(T_{v_{k+1}})$$

and the parallel edges connecting these subgraphs have minimum multiplicity in G which implies that $c(E(G[\{v_0, v_{k+1}\}])) = \{(k+2)s+1, \dots, (k+3)s\}$. It is easy to check that

$$c(E(G[\{v_{k+1}, v'_{k+1}, v''_{k+1}, v(x_{k+1}), v(\overline{x}_{k+1})\}]))$$

is the only edge $((k+2)s)$ -ranking of $T_{v_{k+1}}$. The claim follows.

We proved that for any optimal ranking c of $G_{k,s}$ we have

$$c(E(G[\{v(x_i), v_i, v(\bar{x}_i)\}])) = \{1, \dots, 2s\}$$

and all labels bigger than $2s$ are forbidden both for $E(T_{v(x_i)})$ and $E(T_{v(\bar{x}_i)})$ for each $i = 1, \dots, k$. Without loss of generality we may assume that $\chi'_r(H_i) > 0$, where $i \in \{1, \dots, k\}$. We can shuffle labels assigned to $E(G[\{v(x_i), v_i, v(\bar{x}_i)\}])$ such that

$$c(E(G[\{v_i, v(x_i)\}])) = \{s+1, \dots, 2s\}$$

and

$$c(E(G[\{v_i, v(\bar{x}_i)\}])) = \{1, \dots, s\}$$

which means that we extended edge ranking of $G_{k,s}$ to

$$(G_{k,s} \cup H_i \cup \bar{H}_i)[v(x_i) \equiv u_i, v(\bar{x}_i) \equiv \bar{u}_i]$$

without using any new labels. This holds for every $i \in \{1, \dots, k\}$ and the proof is complete. \square

Lemma 3 *If for some $i \in \{1, \dots, k\}$ $\chi'_r(H_i) > 1$ and $\chi'_r(\bar{H}_i) > 1$ then $\chi'_r(G) > \chi'_r(G_{k,s})$.*

Proof: In the proof of Lemma 2 we showed that in any optimal edge ranking of $G_{k,s}$ and for each $j = 1, \dots, k$ all colors $1, \dots, (k+2)s$ are visible for $E(T_{v(x_j)})$ or for $E(T_{v(\bar{x}_j)})$. This completes the proof. \square

For each $F_i, i = 1, \dots, s$ we create a graph corresponding to this clause and denote it by $G(F_i)$. This graph contains three identical connected components. Each component is a path on four vertices of degrees $1, 2, \chi'_r(G_{k,s}) + 3(i-1) + 1, \chi'_r(G_{k,s}) + 3(i-1)$, respectively. Fig. 4 presents graph $G(F_i)$ and shows symbols used to denote vertices and edges which will be referred later on. The connected component of $G(F_i)$ containing vertex $v(l_{i,j})$ corresponds to the clause $l_{i,j}$ of a Boolean formula F .

Now, we can complete the reduction and create graph $G(F)$ by joining graphs $G_{k,s}, G(F_1), \dots, G(F_s)$. The partial subgraph $G_i(F)$ obtained from the graphs $G_{k,s}, G(F_1), \dots, G(F_i), i \in \{1, \dots, s\}$ is created as follows. $G_0(F) = G_{k,s}$ and

$$G_i(F) = (G_{i-1}(F) \cup G(F_i))[v(l_{i,1}) \equiv u_1, v(l_{i,2}) \equiv u_2, v(l_{i,3}) \equiv u_3],$$

for each $1 \leq i \leq s$, where we choose vertices u_1, u_2, u_3 using the formula:

$$u_j = \begin{cases} v(x_t) & \text{if } l_{i,j} = x_t, \\ v(\bar{x}_t) & \text{if } l_{i,j} = \bar{x}_t, \end{cases}$$

for $j = 1, 2, 3$ and $t \in \{1, \dots, k\}$. Finally, we define $G(F) = G_s(F)$.

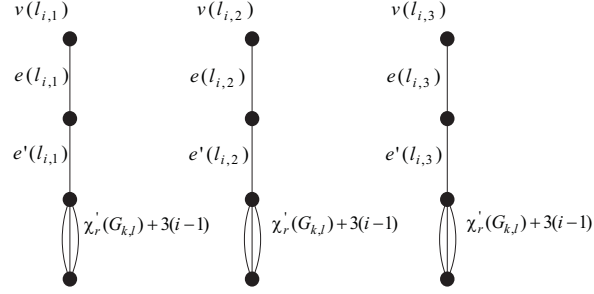


Figure 4: Graph $G(F_i)$

Lemma 4 *If c is any edge ranking of $G(F)$ using $\chi'_r(G_{k,s}) + 3s$ colors then*

$$c^{-1}(\{\chi'_r(G_{k,s}) + 1, \dots, \chi'_r(G(F))\}) \subset E(G(F_1) \cup \dots \cup G(F_s)), \quad (1)$$

and colors bigger than $\chi'_r(G_{k,s}) + 3i$ cannot be assigned to the edges of $G_{k,s} \cup G(F_1) \cup \dots \cup G(F_i)$ for each $i = 1, \dots, s$.

Proof: We will prove by induction on i that labels greater than $\chi'_r(G_{k,s}) + 3(s-i)$ cannot be assigned to the edges of subgraphs $G_{k,s}, G(F_1), \dots, G(F_{s-i-1})$ under edge ranking c .

Let $i = 0$. Consider the case when color $\chi'_r(G_{k,s}) + 3s$ is not assigned to any edge of $G(F_s)$. Some connected component of the graph obtained by removing this edge from $G(F)$ contains all connected components of $G(F_s)$. Each connected component of $G(F_s)$ has a vertex of degree $\chi'_r(G_{k,s}) + 3(s-1) + 1$, so each component contains an edge with label greater than $\chi'_r(G_{k,s}) + 3(s-1)$ under any edge $(\chi'_r(G_{k,s}) + 3s)$ -ranking of $G(F)$. Thus, there are two free labels bigger than $\chi'_r(G_{k,s}) + 3(s-1)$ and each component of $G(F_s)$ contains one of them. This is a contradiction, because each path between these components contains edges of smaller colors. This means that the edge with color $\chi'_r(G_{k,s}) + 3s$ belongs to $G(F_s)$. This color is unique, so we can similarly prove that color $\chi'_r(G_{k,s}) + 3s - 1$ belongs to $G(F_s)$. Finally, color $\chi'_r(G_{k,s}) + 3s - 2$ also is in $G(F_s)$, because each component contains a vertex of that degree. In addition, these three colors are visible for all edges in the connected component of graph

$$G(F) - c^{-1}(\{\chi'_r(G_{k,s}) + 3s - 2, \dots, \chi'_r(G_{k,s}) + 3s\})$$

containing the root of $G(F)$. If we denote this connected component by G' then we have

$$E(G_{k,s} \cup G(F_1) \cup \dots \cup G(F_{s-1})) \subset E(G').$$

So, we have proved the case $i = 0$.

Let us assume that the claim holds for some $i \geq 0$ and consider the case $i + 1$. By induction hypothesis, colors bigger than $\chi'_r(G_{k,s}) + 3(s-i)$ are not

assigned to the edges of $G(F_{s-i-1})$ which means that the proof is similar to that of case $i = 0$. The lemma follows. \square

Lemma 5 *If $\chi'_r(G(F)) \leq \chi'_r(G_{k,s}) + 3s$ and c is an optimal edge ranking of $G(F)$ then some connected component of subgraph*

$$G(F)[c^{-1}(\{1, \dots, \chi'_r(G_{k,s})\})] \quad (2)$$

contains edges of $G_{k,s} + E'$, where $E' \cap \{e(l_{i,1}), e(l_{i,2}), e(l_{i,3})\} \neq \emptyset$ for each $i = 1, \dots, s$.

Proof: Assume that $c(e(l_{i,j})) > \chi'_r(G_{k,s})$ for some $i \in \{1, \dots, s\}$ and each $j = 1, 2, 3$. By Lemma 4 $c(e(l_{i,j})) \leq \chi'_r(G_{k,s}) + 3i$. In addition, for each connected component of $G(F_i)$, $\Delta = \chi'_r(G_{k,s}) + 3(i-1) + 1$, which implies that colors $\chi'_r(G_{k,s}) + 3(i-1) + 1, \dots, \chi'_r(G_{k,s}) + 3i$ are visible for all edges of $G_{i-1}(F)$. It cannot hold $c(e(l_{i,j})) > \chi'_r(G_{k,s}) + 3(i-1)$ for each $j = 1, 2, 3$, because this implies that some edge in subgraph $G(F_i)$ gets color bigger than $\chi'_r(G_{k,s}) + 3i$ under edge ranking c and, by Lemma 4, c cannot be extended to an optimal edge ranking of $G(F)$. If

$$c(e(l_{i,j})) \in \{\chi'_r(G_{k,s}) + 1, \dots, \chi'_r(G_{k,s}) + 3(i-1)\}$$

for some $j \in \{1, 2, 3\}$ then color $c(e(l_{i,j}))$ is visible for edges of subgraphs $G(F_1), \dots, G(F_{i-1})$ and Lemma 4 implies that this coloring cannot be extended to $(\chi'_r(G_{k,s}) + 3s)$ -ranking of $G(F)$, a contradiction. Thus, we proved that

$$\{c(e(l_{i,1})), c(e(l_{i,2})), c(e(l_{i,3}))\} \not\subseteq \{\chi'_r(G_{k,s}) + 1, \dots, \chi'_r(G_{k,s}) + 3s\},$$

for each $i = 1, \dots, s$, which completes the proof. \square

Lemma 6 *If $\chi'_r(G(F)) \leq \chi'_r(G_{k,s}) + 3s$ then F is satisfiable.*

Proof: By Lemma 4 we have that if c is an edge ranking of $G(F)$ then the cutset of $G(F)$ containing edges with labels $\chi'_r(G_{k,s}) + 1, \dots, \chi'_r(G(F))$ is a subset of $E(G(F_1)) \cup \dots \cup G(F_s)$. The connected component

$$G(F)[c^{-1}(\{1, \dots, \chi'_r(G_{k,s})\})]$$

containing the root of $G(F)$ will be denoted by G . We have that $\chi'_r(G) = \chi'_r(G_{k,s})$ (by assumption $\chi'_r(G) + 3s \leq \chi'_r(G_{k,s}) + 3s$ and $\Delta(G) = \Delta(G_{k,s}) = \chi'_r(G_{k,s})$). This means that for each $i = 1, \dots, k$ it holds $\chi'_r(T_{v(x_i)}) = 0$ or $\chi'_r(T_{v(\overline{x_i})}) = 0$ in G , because otherwise by Lemma 3 we would have $\chi'_r(G) > \chi'_r(G_{k,s})$. Then, we can define the values of variables of F as follows:

$$x_t = \begin{cases} true & \text{if } E(T_{v(x_t)}) \neq \emptyset, \\ false & \text{if } E(T_{v(\overline{x_t})}) \neq \emptyset, \end{cases}$$

where $T_{v(x_t)}$ and $T_{v(\overline{x}_t)}$ are subtrees of G , $t = 1, \dots, k$. We know that the above definition is correct, i.e. both conditions cannot hold simultaneously. All variables not modified by the above formula can be assigned arbitrarily. Lemma 5 implies that for each $i = 1, \dots, k$ we can find an edge $e(l_{i,j})$, where $j \in \{1, 2, 3\}$ which also belongs to $E(G)$. Thus, for each F_i there exists literal $l_{i,j}, j \in \{1, 2, 3\}$ such that $l_{i,j} = \text{true}$, which completes the proof. \square

Lemma 7 *If F is satisfiable then $\chi'_r(G(F)) \leq \chi'_r(G_{k,s}) + 3s$.*

Proof: Without loss of generality we may assume that $l_{i,1} = \text{true}$ for each $i = 1, \dots, s$. We define an edge ranking c of $G(F)$ such that

$$\begin{aligned} c(e'(l_{i,1})) &= \chi'_r(G_{k,s}) + 3(i-1) + 1, \\ c(e(l_{i,2})) &= \chi'_r(G_{k,s}) + 3(i-1) + 2, \\ c(e(l_{i,3})) &= \chi'_r(G_{k,s}) + 3i, \end{aligned} \tag{3}$$

$i = 1, \dots, s$. Then, unlabeled elements in $E(G(F_1) \cup \dots \cup G(F_s))$ which are incident to vertices $v(x_i), v(\overline{x}_i)$, $i = 1, \dots, k$ get labels in the set $\{1, \dots, s\}$. This definition is correct, because each vertex $v(x_i)$ ($v(\overline{x}_i)$) is incident to at most s such edges, because without loss of generality we may assume that each literal $l_{i,j}$ appears at most s times in F . Edges adjacent to the leaves of multigraph $G(F_i)$ get colors $\{1, \dots, \chi'_r(G_{k,s}) + 3(i-1)\}$ and $c(e'(l_{i,2})) = c(e'(l_{i,3})) = \chi'_r(G_{k,s}) + 3(i-1) + 1$. Edges of $G_{k,s}$ are labeled as described in the proof of Lemma 1. Consider the connected component G of

$$G(F) - \{e'(l_{i,1}), e(l_{i,2}), e(l_{i,3}) : i = 1, \dots, s\}$$

containing the root of $G(F)$. We have that $E(T_{v(x_i)}) = \emptyset$ or $E(T_{v(\overline{x}_i)}) = \emptyset$ in G for each $i = 1, \dots, k$, because otherwise we would have that $e(l_{r,1})$ is adjacent to $v(x_i)$ and $e(l_{t,1})$ is adjacent to $v(\overline{x}_i)$ in G for some $r, t \in \{1, \dots, s\}$. This means that for these clauses it holds $l_{r,1} = l_{t,1} = \text{true}$ and $l_{r,1} = x_i$ and $l_{t,1} = \overline{x}_i$, a contradiction. In addition, the nonempty subgraph among $T_{v(x_i)}, T_{v(\overline{x}_i)}$ does not contain more than s edges. Thus, $\chi'_r(G) = \chi'_r(G_{k,s})$ by Lemma 2. This proves that $\chi'_r(G(F)) = \chi'_r(G_{k,s}) + 3s$. \square

As an example consider a Boolean formula

$$F' = (x_1 + x_3 + \overline{x_3})(\overline{x_1} + \overline{x_2} + \overline{x_3})(x_1 + x_2 + x_3)(x_1 + \overline{x_2} + \overline{x_3}).$$

In this case $k = 3$ and $s = 4$. Fig. 5 presents the graph $G(F')$.

If $x_1 = \text{true}, x_2 = \text{false}$ then $F = \text{true}$. There is more than one way of constructing an optimal edge 32-ranking of $G(F')$ and Fig. 5 gives an example of such a coloring. Symbol $l_{i,j}, i = 1, \dots, 4, j = 1, 2, 3$ labeling a connected component of subgraph $G(F'_i)$ shows which literal of F' corresponds to this component. For each edge (set of parallel edges) the figure shows color (the set of colors) used to label the edge (set of edges). For each $i = 1, \dots, 4$ one edge among $e(l_{i,1}), e(l_{i,2}), e(l_{i,3})$ gets color smaller than $\chi'_r(G_{3,4})$ and it is denoted with heavy line in Fig. 5.

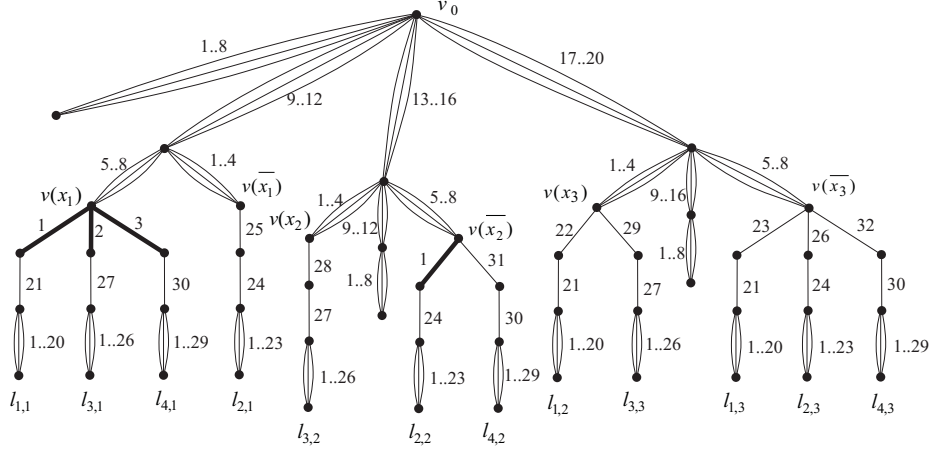


Figure 5: Graph $G(F')$ and its edge 32-ranking

Theorem 2 *The edge ranking problem of multitrees with diameter at most 10 is NP-complete.*

Proof: The problem is clearly in NP. The size of the graph $G(F)$ is polynomial in s . From Lemmas 6 and 7 we have that $\chi'_r(G(F)) \leq \chi'_r(G_{k,s}) + 3s = (k+5)s$ if and only if F is satisfiable. Clearly, graph $G(F)$ is a multitree and its height is at most 5. \square

Using the above theorem we can write the following

Corollary 2 *The edge ranking problem is NP-complete for bipartite and planar multigraphs.*

4 A polynomial time algorithm for weighted trees

The set of vertices adjacent to vertex v in a graph G is denoted by $N(v)$. In the following $n(T)$ denotes the number of vertices of T . We often write n instead of $n(T)$. In this section we assume that $0 < b \leq \frac{1}{2}$ is fixed. We are interested in edge rankings of weighted trees, because by Corollary 1 this problem is more general than the edge ranking problem of multitrees. For brevity we say that $|c(E(T))|$ is the *number of colors* used by edge ranking c of T .

We consider an algorithm denoted by A which takes a tree T and a weight function $w : E(T) \rightarrow \mathbb{R}_+$ as an input and returns the number of colors assigned to the edges of T . In addition, algorithm creates an edge ranking c of T . If the tree T is empty (i.e. T has one vertex) then A returns 0. Otherwise, we find a set $C = \{e_1, \dots, e_{l-1}\}$ of pairwise adjacent edges in T such that $T - C = T_1 \cup \dots \cup T_l$

($T - C$ is a forest containing trees T_1, \dots, T_l as connected components) and for each $i = 1, \dots, l$, $|V(T_i)| \leq (1-b)n$. The algorithm A recursively colors subtrees T_1, \dots, T_l . Let $d = \max\{A(T_i) : i = 1, \dots, l\}$ and let c_i be the edge ranking of T_i . Then we extend edge rankings $c_i, i = 1, \dots, l$ as follows

$$\begin{aligned} c(e_i) &= [d + \sum_{j=1}^{i-1} w(e_j), d + \sum_{j=1}^i w(e_j)], \quad i = 1, \dots, l-1 \\ c(e) &= c_i(e), \quad \text{where } e \in E(T_i), i \in \{1, \dots, l\}. \end{aligned} \quad (4)$$

The correctness of this algorithm follows from the following

Lemma 8 *Function c defined by (4) is an edge ranking of T .*

Proof: We have to show that there exists a set $C \subseteq \{\{v, x\} : x \in N(v)\}$ such that each connected component of $T - C$ has size at most $(1-b)n$. It is sufficient to show that there exists a vertex $v \in V(T)$ such that each connected component of $T - \{\{u, v\} : u \in N(v)\}$ has size at most $\frac{1}{2}n$, because $(1-b)n \geq \frac{1}{2}n$ for $0 < b \leq \frac{1}{2}$. Let v_0 be any vertex of T . If $T - \{\{u, v_0\} : u \in N(v_0)\}$ has a connected component T_1 of size greater than $\frac{1}{2}n$ then let v_1 be the neighbor of v_0 in T_1 . The size of the connected component T'_0 of $T - \{\{u, v_1\} : u \in N(v_1)\}$ containing v_0 does not exceed $\frac{1}{2}n$, because $|V(T'_0)| \leq |V(T) \setminus V(T_1)| = n - n(T_1) \leq \frac{1}{2}n$. So, after a finite number of above steps we find vertex v_i such that each connected component of $T - \{\{u, v_i\} : u \in N(v_i)\}$ has size at most $\frac{1}{2}n$. Thus, the definition of C is correct and one can prove, by induction on the number of the vertices of tree T , the correctness of the definition of c . \square

In the following we give an approximation ratio of A and prove that the bound is asymptotically tight.

Lemma 9 *The algorithm A uses at most $\log_{1/(1-b)}(n) \cdot \chi'_r(T)$ labels, where T is a weighted tree on n vertices.*

Proof: We prove this lemma by induction on n . If $n = 1$ then the tree has no edges and the hypothesis holds for T . Assume that the hypothesis holds for all trees with at most n vertices and consider a tree T on $n + 1$ vertices. The algorithm removes such a set of edges C from T that graph $T - C = T_1 \cup \dots \cup T_l$ is disconnected and for each $i = 1, \dots, l$ we have $n(T_i) \leq (1-b)n$. From (4) it follows that edges in C get $w(e_1) + \dots + w(e_{l-1})$ colors. Thus,

$$\begin{aligned} A(T) &= \sum_{i=1}^{l-1} w(e_i) + \max\{A(T_i) : i = 1, \dots, l\} \\ &\leq \chi'_r(T) + \max\{A(T_i) : i = 1, \dots, l+1\} & (5) \\ &\leq \chi'_r(T) + \chi'_r(T_s) \log_{1/(1-b)}((1-b)n) & (6) \\ &\leq \chi'_r(T) + \chi'_r(T) \log_{1/(1-b)} n - \chi'_r(T) & (7) \\ &= \chi'_r(G) \log_{1/(1-b)} n. \end{aligned}$$

Inequality (5) is fulfilled because

$$\chi'_r(T) \geq \max\left\{ \sum_{u \in N(v)} w(\{u, v\}) : v \in V(T) \right\} \geq \sum_{i=1}^{l-1} w(e_i).$$

Inequality (6) follows from the induction hypothesis applied for each $T_i, i = 1, \dots, l$ and $n(T_i) \leq (1-b)n, i = 1, \dots, l$. Index s in (6) is defined in such a way that

$$\chi'_r(T_s) \geq \chi'_r(T_i), \quad i \neq s.$$

Finally, $\chi'_r(T_s) \leq \chi'_r(T)$ implies (7). \square

Now, we are going to show that the bound given in Lemma 9 is asymptotically the best possible for this algorithm. For each $k \in \mathbb{N}$ we construct a family of trees

$$\mathcal{T}_b^k = \{T_i : i = 1, \dots, k\}.$$

Each $T_i \in \mathcal{T}_b^k$ has a vertex v_i . We define T_0 as follows. Vertex v_0 has $\lceil 1/b \rceil$ neighbors and each neighbor x of v_0 is adjacent to one additional vertex which is a leaf in T_0 . The weight $w(\{v_0, x\}) = p$ (p will be defined later) and the weights of all edges which are not incident to v_0 are equal to 1. Assume that tree T_i has been created. We define $T_{i+1} \in \mathcal{T}_b^k$ in such a way that we get $\lceil 1/b \rceil$ copies of T_i and add one additional vertex v_{i+1} . In each copy of T_i contained in T_{i+1} there is one leaf which is adjacent to v_{i+1} in T_{i+1} . In order to define the weight function w for T_{i+1} let w restricted to a copy of T_i be equal to the weight function defined for T_i and let $w(\{v_{i+1}, x\}) = p$ for each $x \in N(v_{i+1})$. The trees $T_0, T_{i+1} \in \mathcal{T}_{0.4}^k$ are shown in Fig. 6.

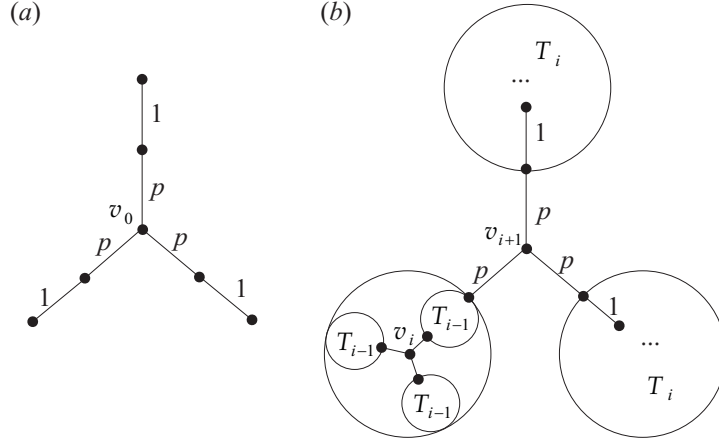


Figure 6: Trees (a) T_0 and (b) T_{i+1} .

We prove two lemmas, one gives an upper bound for the edge ranking number of the weighted tree $T_i \in \mathcal{T}_b^k$ and the other gives a lower bound for the number of colors used by the algorithm A for T_i .

Lemma 10 For each tree $T_i \in \mathcal{T}_b^k$ it holds $\chi'_r(T_i) \leq (p+i)\lceil 1/b \rceil + 1$.

Proof: We have $\chi'_r(T_0) = p \cdot \lceil 1/b \rceil + 1$. We show this lemma by constructing an edge ranking of T_i which uses the required number of colors. Consider the vertex u_j which belongs to the j th copy of T_{i-1} and $\{u_j, v_i\} \in E(T_i)$. The vertex u_j is a leaf in T_{i-1} and the edge incident to it has weight equal to 1. Denote by U_i the set of edges of weight 1 incident to vertices $u_j, j = 1, \dots, \lceil 1/b \rceil$ in all copies of T_{i-1} in T_i . Graph $T_i - U_i$ has $\lceil 1/b \rceil + 1$ connected components, one component is the star containing all edges incident to v_i and other components are subtrees $T_{i-1} - \{\{u_j, x\} : x \in V(T_{i-1})\}$. Thus, we have

$$\begin{aligned} \chi'_r(T_0) &= p \cdot \lceil 1/b \rceil + 1, \\ \chi'_r(T_i) &\leq \lceil 1/b \rceil + \chi'_r(T_{i-1}), \quad i > 0, \end{aligned} \tag{8}$$

where the inequality follows from

$$\chi'_r(T_{i-1}) \geq \max\{\chi'_r(T_{i-1} - \{\{u_{i-1}, x\} : x \in V(T_{i-1})\}), p \cdot \lceil 1/b \rceil\}.$$

So, on the basis of (8) we have

$$\chi'_r(T_i) \leq i \cdot \lceil 1/b \rceil + \chi'_r(T_0) = i \cdot \lceil 1/b \rceil + p \cdot \lceil 1/b \rceil + 1,$$

which completes the proof. \square

Lemma 11 For each tree $T_i \in \mathcal{T}_b^k$ it holds $A(T_i) \geq p \cdot (i + \lceil 1/b \rceil)$.

Proof: The algorithm A finds some vertex $v \in V(T_i)$ and disconnects T_i by removing a set C edges incident to v . Consider the case when $v \neq v_i$. Then, one connected component of $T_i - \{\{v, x\} : x \in N(v)\}$ contains v_i and $\lceil 1/b \rceil - 1$ copies of subgraph T_{i-1} . Denote this connected component by T' . We have

$$\begin{aligned} n(T') &\geq 1 + (\lceil 1/b \rceil - 1)n(T_{i-1}) \\ &= 1 + (\lceil 1/b \rceil - 1)\frac{n-1}{\lceil 1/b \rceil} \\ &= 1 + \left(1 - \frac{1}{\lceil 1/b \rceil}\right)(n-1) \\ &> \left(1 - \frac{1}{\lceil 1/b \rceil}\right)n \\ &\geq (1-b)n, \end{aligned}$$

where n stands for $n(T_i)$. This leads to a contradiction, because algorithm A disconnects the graph in such a way that each connected component has size at

most $(1-b)n$. This implies that A removes edges incident to v_i in T_i in order to perform recursive calls for connected components. This means that one of the recursively colored subgraphs is T_{i-1} . Hence $A(T_i) \geq p + A(T_{i-1})$. The above inequality and $A(T_0) = p \cdot \lceil 1/b \rceil + 1$ imply

$$A(T_i) \geq p \cdot i + p \cdot \lceil 1/b \rceil + 1.$$

□

Now, let us define the value of parameter p . For $k \in \mathbb{N}$ all trees $T_i \in \mathcal{T}_b^k$ have $p = \log_2(n)$, where $n = n(T_k)$. Let $k = \Theta(\log n)$. We construct the infinite sequence of graphs $T_i, i \in \mathbb{N}$ such that the graph T_k in this sequence belongs to \mathcal{T}_b^k . Then, by Lemmas 10 and 11 we have that

$$\begin{aligned} \frac{A(T_k)}{\chi_r'(T_k)} &\geq \frac{p \cdot (k + \lceil 1/b \rceil)}{(p + k)\lceil 1/b \rceil + 1} \\ &= \Theta\left(\frac{\log^2 n}{\log n}\right) = \Theta(\log n). \end{aligned}$$

This implies the following

Corollary 3 *The approximation ratio given in Lemma 9 is asymptotically the best possible for the algorithm A .*

Theorem 3 *The running time of A is $O(n \log n)$.*

Proof: We have to show that we can compute set C for a tree T in linear time. We pick an arbitrary vertex $v_0 \in V(T)$. If each connected component of $T - \{\{u, v_0\} : u \in N(v_0)\}$ has size at most $(1-b)n$ then $C := \{\{v_0, u\} : u \in N(v_0)\}$. Otherwise, we find the neighbor v_1 of v_0 , which belongs to connected component of $T - \{\{u, v_0\} : u \in N(v_0)\}$ with size bigger than $(1-b)n$. In the proof of Lemma 8 we showed that after at most n such steps we get the desired vertex v_i . In the j th step $j = 0, \dots, i$ we have to check the sizes of connected components in the graph obtained from T by removing v_j from T . Since vertices v_0, \dots, v_i are all different, we have to compute the sizes of appropriate subtrees $O(n)$ times. Consider T as a rooted tree at vertex r . For each vertex $v \in V(T)$ we compute in the preliminary phase of algorithm A the size of a subtree T_v . All values $T_v, v \in V(T)$ can be determined in $O(n)$ time. Then, in order to check the size of a connected component of $T - \{\{u, v_j\} : u \in N(v)\}$ containing vertex $u \in N(v_j)$ we have to read the size of T_u if u is a descendant of v_j in T or compute $|V(T_r)| - |V(T_{v_j})|$ if u is the father of v_j . Thus, the size of this connected component can be determined in constant time which means that C can be constructed in linear time. So, the running time of A is $O(n \log n)$ because the depth of the recursion is $O(\log n)$. □

References

- [1] H. Bodlaender, R. J. Gilbert, H. Hafsteinson, T. Kloks, Approximating treewidth, pathwidth, frontsize and shortest elimination tree, *J. Algorithms* **18** (1995) 238-255.
- [2] P. de la Torre, R. Greenlaw, A.A. Schäffer, Optimal edge ranking of trees in polynomial time, *Algorithmica* **13** (1995) 529-618.
- [3] D. Dereniowski, M. Kubale, Efficient parallel query processing by graph ranking, *Fundamenta Informaticae* **69** (2005/06) 273-285.
- [4] A.V. Iyer, H.D. Ratliff, G. Vijayan, Parallel assembly of modular products - an analysis, *Tech. Report 88-06*, Georgia Institute of Technology, 1988.
- [5] A.V. Iyer, H.D. Ratliff, G. Vijayan, On an edge ranking problem of trees and graphs, *Discrete Appl. Math.* **30** (1991) 43-52.
- [6] T.W. Lam, F.L. Yue, Edge ranking of graphs is hard, *Discrete Appl. Math.* **85** (1998) 71-86
- [7] 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.
- [8] K. Makino, Y. Uno, T. Ibaraki, On minimum edge ranking spanning trees, *J. Algorithms* **38** (2001) 411-437.
- [9] K. Makino, Y. Uno, T. Ibaraki, Minimum edge ranking spanning trees of threshold graphs, *LNCS* **2518** (2002) 428-440.
- [10] X. Zhou, M.A. Kashem, T. Nishizeki, Generalized edge-rankings of trees, *Proc. of the 22nd International Workshop on Graph-Theoretic Concepts in Computer Science, LNCS* **1197** (1997) 390-404.