アルゴリズム 70－3
（1999. 11. 8）

最小辺ランキング全域木問題について

牧野和久[†], 宇野裕之[¶], 茨木俊秀[§]

[†] 大阪大学大学院 基礎工学研究科 システム人間系, 560-8531 豊中市待兼山 1-3
[¶] 大阪府立大学 総合科学部 数理・情報科学科, 599-8531 堺市学園町 1-1
[§] 京都大学大学院 情報学研究科 数理工学専攻, 606-8501 京都市左京区吉田本町

あらまし 本論文では, 最小辺ランキング全域木問題 (MERST), すなわち, 与えられたグラフ G の全域木で, その辺ランキングが最小になるものを計算する問題を紹介する. 与えられた木に対しては, その辺ランキングは多項式時間で計算できることが知られているが, MERST は NP 困難であることを示す. そこでわれわれは, MERST に対して, 近似比が $\frac{\min\{(\Delta^*-1)\log n/\Delta^*, \Delta^*-1\}}{\log(\Delta^*+1)-1}$ である近似アルゴリズムを提案する. ただし, n は G の節点数で, Δ^* は最大次数が最小となる全域木の最大次数である. この近似アルゴリズムは, 最小次数全域木問題と木の辺ランキング問題に対する既存の 2 つのアルゴリズムの組合わせであるが, その近似比の解析は, 木の辺ランキングのいろいろな新しい性質にもとづいている.

和文キーワード: 辺ランキング, 最小辺ランキング全域木, 近似アルゴリズム

On minimum edge ranking spanning trees

Kazuhisa MAKINO[†], Yushi UNO[¶] and Toshihide IBARAKI[§]

[†] Department of Systems and Human Science, Graduate School of Engineering Science,
Osaka University, Toyonaka, Osaka, 560-8531 Japan
email:makino@sys.es.osaka-u.ac.jp

[¶] Department of Mathematics and Information Sciences, College of Integrated Arts and Sciences,
Osaka Prefecture University, Sakai 599-8531, Japan
email:uno@mi.cias.osakafu-u.ac.jp

[§] Department of Applied Mathematics and Physics, Graduate School of Informatics,
Kyoto University, Kyoto 606-8501, Japan
email:ibaraki@i.kyoto-u.ac.jp

Abstract In this paper, we introduce the problem of computing a minimum edge ranking spanning tree (MERST); i.e., find a spanning tree of a given graph G whose edge ranking is minimum. Although the minimum edge ranking of a given tree can be computed in polynomial time, we show that problem MERST is NP-hard. Furthermore, we present an approximation algorithm for MERST, which realizes its worst case performance ratio $\frac{\min\{(\Delta^*-1)\log n/\Delta^*, \Delta^*-1\}}{\log(\Delta^*+1)-1}$, where n is the number of vertices in G and Δ^* is the maximum degree of a spanning tree whose maximum degree is minimum. Although the approximation algorithm is a combination of two existing algorithms for the restricted spanning tree problem and for the minimum edge ranking problem of trees, the analysis is based on novel properties of the edge ranking of trees.

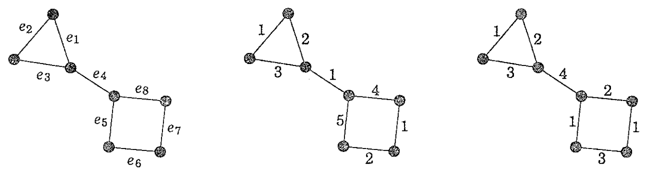
英文 key words: edge ranking, minimum edge ranking spanning tree, approximation algorithm

1 Introduction and Preliminaries

Let $G = (V,E)$ be an undirected graph, which is simple and connected. An *edge ranking* of a graph $G = (V,E)$ is a labeling $r\colon E \to Z^+$, with the property that every path between two edges with the same label i contains an intermediate edge with label $j > i$. By definition, every edge ranking r has exactly one edge with the largest rank. An edge ranking by integers $1,2,\ldots,k$ is called a *k-edge ranking*. A graph G is said to be *k-edge rankable* if it has a k-edge ranking. An edge ranking is *minimum* if the largest rank k in it is the smallest among all edge rankings of G; such k is called the *minimum edge rank* of G and is denoted by $\mathrm{rank}(G)$. The *minimum edge ranking problem* (MER) asks to compute a minimum edge ranking of a given graph G and formally defined as follows:

MER (minimum edge ranking problem)
Input: A simple undirected graph $G = (V,E)$ which is connected, and a nonnegative integer k.
Question: Is G k-edge rankable (i.e., does G satisfy $\mathrm{rank}(G) \le k$)?

In the affirmative case, such a $\mathrm{rank}(G)$-edge ranking is also required. Fig. 1 is an input graph $G = (V,E)$ with $E = \{e_1,e_2,\ldots,e_8\}$, while (b) and (c) give 5- and 4-edge rankings of G, respectively. Note that (c) in fact gives a minimum edge ranking of G, i.e., $\mathrm{rank}(G) = 4$.



(a) A graph G (b) An edge ranking of G (c) A minimum edge ranking of G

Figure 1: Edge rankings of a graph.

MER has applications in the context of assembling a multi-part product from its components in the smallest number of parallel processing (integration) stages [1, 7, 12]. It is known that MER is in general NP-hard [9], but it can be solved in polynomial time when the graph is a tree [2, 8, 12].

In this paper, we newly consider the following problem, which resembles MER but is essentially different. Given a simple undirected graph $G = (V,E)$ which is connected, we repeat contraction steps until all the vertices are contracted into a single vertex. Here one contraction step consists of many simultaneous contractions of edges which do not share any of their end vertices. In this process, all self-loops created are simply ignored. Under this setting, we like to minimize the number of steps required before contracting all vertices into one.

It is easy to see that this problem is equivalent to finding a spanning tree of G whose edge ranking is minimum. To see this, first assume that a spanning tree of G as well as its edge ranking is given. Then no two edges with the same rank share their end vertices, and in the i-th step all edges with rank i can be contracted simultaneously. Thus the required number of steps is equal to the largest rank in this edge ranking. Conversely, given a series of steps that contracts G into a single vertex, we assign rank i to all the edges contracted in the i-th step. Then it can be seen that G contains a spanning tree whose edge ranking is defined by the above ranks.

Now we call this problem *the minimum edge ranking spanning tree problem* (MERST).

MERST (minimum edge ranking spanning tree problem)
Input: A simple undirected graph $G = (V,E)$ which is connected, and a nonnegative integer k.
Question: Does G have a k-edge rankable spanning tree (i.e., does there exist a spanning tree $T = (V,E_T)$ of G with $\mathrm{rank}(T) \le k$)?

Similarly to MER, in the affirmative case, such a spanning tree T as well as its k-edge ranking is also required. We say that T is a *minimum edge ranking spanning tree* of a graph G if T is a spanning tree of G having the minimum $\mathrm{rank}(T)$ among all spanning trees of G. Fig. 2 (a) gives an example of a minimum edge ranking spanning tree of the graph G in Fig. 1 (a), together with its edge ranking. Fig. 2 (b) demonstrates the contraction steps defined by the T.

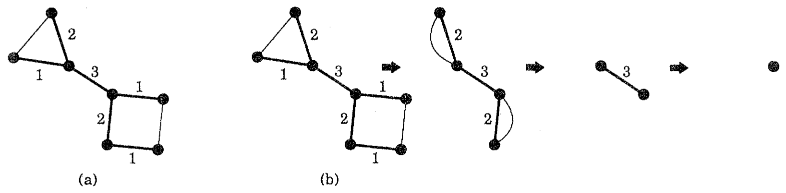


Figure 2: (a) A minimum edge ranking spanning tree T of the graph G in Fig. 1 (a), and (b) its edge contraction according to the minimum edge ranking spanning tree.

Problem MERST can be found in many practical applications. Among them, we pick up here one example found in relational database theory.

Let us consider the "query graph (join graph)" [6, 11], where its vertex set corresponds to the set of relations and its edge set represents the pairs of relations which are joined. In this context, join operations which are represented by non-adjacent edges can be joined in parallel, but no two join operations corresponding to adjacent edges can be performed simultaneously, since a relation can participate only in one join operation at a time. The join operations are then performed until all the relations are merged into a single relational table. The whole process can be formulated as MERST.

In this paper, we show that MERST is NP-hard, and present an approximation algorithm for MERST, with its worst case performance ratio $\min\{(\Delta^*-1)\log n/\Delta^*, \Delta^*-1\}/(\log(\Delta^*+1)-1)$, where n is the number of vertices in G and Δ^* is the maximum degree of a spanning tree whose maximum degree is minimum. This algorithm is a combination of two existing algorithms for the minimum degree spanning tree problem (MDST) and for the minimum edge ranking problem for trees (MERT), respectively.

2 NP-hardness of MERST

In this section, we show that MERST is intractable after showing several lemmas on the edge ranking. The idea of our proof is based on the NP-hardness proof of the connected size-k-partition problem for planar bipartite graphs [3]. We assume that a graph $G=(V,E)$ is simple, undirected and connected. For a vertex set $W \subseteq V$, $G[W]$ denotes the subgraph of G induced by W.

Lemma 1 *Any connected graph G with* $\text{rank}(G)=k$ *has at most* 2^k *vertices.*

Let us introduce some additional notions related to minimum edge ranking spanning trees. For a graph $G=(V,E)$ and a positive integer k, a *size-k-partition* (this notion is referred to as k-partition in [3]) of V is a $(|V|/k)$-tuple $(V_1,V_2,\ldots,V_{|V|/k})$ and $V=V_1\cup V_2\cup\cdots\cup V_{|V|/k}$, $V_i\cap V_j=\emptyset$ for all $i\neq j$ such that $|V_i|=k$ for $i=1,2,\ldots,|V|/k$. Each V_i is called an *element* of the partition. A size-k-partition of V is *connected* if the graphs $G[V_i]$ are connected for all i. Let $G=(V,E)$ be a graph with $|V|=2^k$, where $k\geq 0$. We say that G has a *nested partition* if it recursively satisfies one of the following conditions:

(i) $k=0$, or (ii) G has a connected size-2^{k-1}-partition (V_1,V_2) such that both $G[V_1]$ and $G[V_2]$ have nested partitions.

Lemma 2 *Let $G=(V,E)$ be a graph with $|V|=2^k$ $(k\geq 0)$. Then G has a k-edge rankable spanning tree if and only if it has a nested partition.*

This lemma provides the essential idea of NP-completeness proof of MERST, i.e., to find a k-edge rankable spanning tree of G is equivalent to find a nested partition of G.

Theorem 1 *MERST is NP-complete.*

Given a spanning tree $T = (V, E_T)$, we can check whether T is k-edge rankable in polynomial time [8]. Hence, MERST belongs to NP. To prove the completeness, we reduced 3-dimensional mathching, which is known to be NP-complete [5] to MERST. However, the detail of the proof is omitted here due to space reasons.

3 An Approximation Algorithm for MERST

Since MERST is NP-hard, we propose in this section an approximation algorithm, which is a combination of two existing algorithms for the minimum degree spanning tree problem (MDST, defined below) and for the minimum edge ranking problem of trees (MERT, which is MER whose input graphs are restricted to be trees). We also state its approximation ratio here, but detailed analysis of the algorithm will be given in Sect. 4.

MDST
Input: A graph $G = (V, E)$.
Output: A minimum degree spanning tree $T = (V, E_T)$ of G; i.e., a spanning tree T of G whose maximum degree is minimum.

We denote the maximum degree of vertices in a graph G by Δ_G, and the maximum degree of the minimum degree spanning tree T of G by Δ^* $(= \Delta_T)$. Although MDST is known to be NP-hard [5], Fürer and Raghavachari [4] developed a polynomial time approximation algorithm which computes a spanning tree T satisfying

$$\Delta^* \leq \Delta_T \leq \Delta^* + 1 \ (\leq \Delta_G). \tag{1}$$

Our approximation algorithm for MERST first computes a spanning tree T_{Approx} of G satisfying (1) (by using the algorithm in [4]), and then computes its minimum edge ranking. Recall that MERT is polynomially solvable (e.g., [8]). Thus, our algorithm described below can be executed in polynomial time.

Algorithm APPROX_MERST
Input: A graph $G = (V, E)$.
Output: A spanning tree T of G and its edge ranking r.

```
    Step 1: Compute a spanning tree $T_{\text{Approx}}$ of $G$ satisfying (1).
    Step 2: Compute a minimum edge ranking $r$ of $T_{\text{Approx}}$.
    Step 3: Output $T = T_{\text{Approx}}$ and its edge ranking $r$.
```
□

Theorem 2 *For a graph $G = (V, E)$ with $|V| = n$, let T_{Min} denote a minimum edge ranking spanning tree of G, and let T_{Approx} denote a spanning tree of G computed by algorithm* APPROX_MERST *for the input G. Then, the approximation ratio of algorithm* APPROX_MERST *can be bounded from above by*

$$\frac{\text{rank}(T_{Approx})}{\text{rank}(T_{Min})} \leq \frac{\min\{(\Delta^* - 1)\log n/\Delta^*, \Delta^* - 1\}}{\log(\Delta^* + 1) - 1},$$

where Δ^* is the maximum degree of the minimum degree spanning tree of G. □

4 Analysis of Edge Ranking of Trees

In this section, we discuss some properties of the minimum edge ranking of trees in order to prove the approximation ratio of algorithm APPROX_MERST. In particular, we derive upper and lower bounds on rank(T) of a tree $T = (V, E_T)$ in terms of the number of vertices $n = |V|$ and its maximum degree Δ_T.

4.1 Lower and Upper Bounds on the Edge Rank of Trees

Lemma 3 *For any tree* $T = (V, E_T)$, $\mathrm{rank}(T) \geq \max\{\Delta_T, \lceil \log n \rceil\}$ *holds, where* Δ_T *is the maximum degree of vertices in* T *and* $n = |V|$.

Both lower bounds Δ_T and $\lceil \log n \rceil$ are tight, that is, there exist trees T_1 and T_2 such that $\mathrm{rank}(T_1) = \Delta_{T_1}$ and $\mathrm{rank}(T_2) = \lceil \log n \rceil$, respectively.

Lemma 4 *Let* $T = (V, E_T)$ *be a tree with* $|V| = n$. *Then it holds that*

$$\mathrm{rank}(T) = \lceil \log n \rceil \qquad \text{if } \Delta_T = 0, 1, 2 \tag{2}$$

$$\mathrm{rank}(T) \leq \frac{(\Delta_T - 2)\log n}{\log \Delta_T - 1} \qquad \text{if } \Delta_T \geq 3. \tag{3}$$

This lemma, together with Lemma 3, proves Theorem 2, since the algorithm of Fürer and Raghavachari [4] can find a spanning tree T of G such that $\Delta^* \leq \Delta_T \leq \Delta^* + 1$ in the first step of APPROX_MERST.

4.1.1 Edge Ranking of a Tree by Top-down Approach

For the purpose of proving (3), we present an algorithm which gives a consistent (but not always minimum) edge ranking of trees.

All the existing exact algorithms for MERT [2, 8, 12] are based on the bottom-up approach: Choose an arbitrary vertex as a root (which is placed at the top) and assign ranks from leaf edges to top edges in the resulting rooted tree. However, we describe here a top-down edge ranking algorithm, which may not give exact minimum solution but is easy to analyze.

It starts from the given tree T, and in each step, removes one edge from a generated subtree of T to split it into two, until all the generated subtrees become singletons. This process can be visualized by the partition tree as illustrated in Fig. 3.

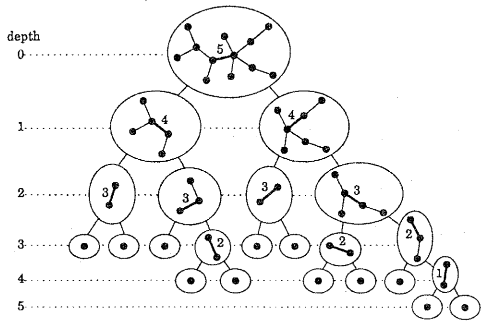


Figure 3: Partition tree constructed by top-down approach.

Algorithm EDGE_RANKING_OF_TREES (ERT)
Input: A tree $T = (V, E_T)$.
Output: An edge ranking of T.

Step 1: Start with the partition tree consisting of exactly one component T.

Step 2: If there is a subtree T', which has more than one vertex and is located in a leaf position of the current partition tree, choose an edge e in T' according to the criterion to be described later and remove it after giving it temporal ranking $r'(e) = i$, where i is the depth of the chosen subtree in the partition tree. The two subtrees resulting from the removal of e become the children of T' in the partition tree. Return to Step 2. On the other hand, if there is no subtree satisfying the above condition, then go to Step 3.

Step 3: Let h be the height of the resulting partition tree, and rank all edges e by $r(e) = h - r'(e)$. □

It is easy to see that this algorithm in fact gives an edge ranking. The main point of this algorithm is how to determine an edge e to be removed from T' in Step 2. Our criterion selects an edge, which makes the resulting two subtrees "most balanced" in the sense that the difference of their sizes is smallest. Formally, such an edge is defined as follows.

Definition For a tree $T = (V, E_T)$, a partition (V_1, V_2) of V is called *connected* if the induced subgraphs $T[V_1]$ and $T[V_2]$ are both connected (i.e., subtrees of T). A connected partition (V_1, V_2) is called *optimal* if the difference of their sizes $||V_1| - |V_2||$ is minimum. An edge is called *optimal* if its removal produces an optimal connected partition. Furthermore, the *weight* $w(v)$ of a vertex $v \in V$ is the size of the subtree, which has the maximum number of vertices among all subtrees in $T[V \setminus \{v\}]$, where $T[V \setminus \{v\}]$ denotes the subgraph of T induced by $V \setminus \{v\}$. The *centroid* of a tree T is the set of all the vertices having the minimum weight. A vertex in the centroid is called a *centroid vertex.*

Now, take a vertex $v_0 \in V$ of a tree $T = (V, E_T)$, and consider that T is rooted at v_0. Assume that v_0 has $b \equiv \deg(v_0)$ children $v_1, v_2, \ldots, v_b$. If we remove v_0 from T, then there remain b subtrees $T_1, T_2, \ldots, T_b$, where each T_j is rooted at v_j. We index these T_j in the nonincreasing order of their sizes (i.e., the number of vertices). By definition, $|T_1| \geq |T_2| \geq \cdots \geq |T_b|$, and hence $w(v_0) = |T_1|$ holds.

Lemma 5 *Let $T = (V, E)$ be a tree with $n = |V|$. Then v_0 is a centroid vertex of T if and only if $w(v_0) \leq n/2$ holds. Furthermore, any tree T has either one centroid vertex v_0 or two centroid vertices v_0 and v_1 which are adjacent to each other. In the latter case, $w(v_0) = w(v_1) = n/2$ holds.*

In the subsequent discussion, we choose v_0 and v_1 so that v_0 is the centroid vertex if there is only one centroid vertex (in this case, v_1 is the root of subtree T_1), and v_0 and v_1 are both centroid vertices if there are two centroid vertices.

Now we present the following criterion.

Criterion in Step 2 of ERT: Choose the optimal edge (v_0, v_1) in the subtree under consideration.

4.1.2 Analysis of Algorithm ERT

Let $\gamma = \dfrac{(\Delta_T - 2)\log n}{\log \Delta_T - 1}$. In this subsection, we prove the inequality (3) of Lemma 4; i.e., Algorithm ERT always gives a γ-edge ranking to a tree $T = (V, E_T)$, where $n = |V|$ and we assume $\Delta_T \geq 3$. Note that γ is non-decreasing in Δ_T as can be proved by direct calculation.

The proof is done by induction on n.

In case of $n = 1$, Lemma 4 clearly holds. Assuming that Lemma 4 holds for all trees T' with n' $(< n)$ vertices and its maximum degree $\Delta_{T'}$ $(\leq \Delta_T)$, we consider the case of n vertices.

For an optimal partition (V_1, V_2) of a tree $T = (V, E_T)$, the subtree which includes the centroid vertex v_0 of T is called the *main* subtree. By the definition of v_0, the main subtree is always greater than or equal to the other subtree.

Now, given a tree $T = T^{(0,0)} = (V_{(0,0)}, E_{(0,0)})$, we recursively define its vertices $v_0^{(i)}$, $v_1^{(i)}$ and subtrees $T^{(i,0)} = (V_{(i,0)}, E_{(i,0)})$ and $T^{(i,1)} = (V_{(i,1)}, E_{(i,1)})$ for $i = 1, 2, \ldots$, as follows: (i) $(v_0^{(i)}, v_1^{(i)})$ is an optimal edge in $T^{(i-1,0)}$ (i.e., $v_0^{(i)}$ is a centroid vertex of $T^{(i-1,0)}$), (ii) $T^{(i,0)} = T[V_{(i,0)}]$ and $T^{(i,1)} = T[V_{(i,1)}]$, where $(V_{(i,0)}, V_{(i,1)})$ is the optimal partition of $V_{(i-1,0)}$ obtained by the deletion of edge $(v_0^{(i)}, v_1^{(i)})$, such that $V_{(i,0)}$ is its main subtree, i.e., $v_0^{(i)} \in V_{(i,0)}$.

Note that $V_{(i-1,0)} = V_{(i,0)} \cup V_{(i,1)}$ and $E_{(i-1,0)} = E_{(i,0)} \cup E_{(i,1)} \cup \{(v_0^{(i)}, v_1^{(i)})\}$ hold for all i (≥ 1). The above condition (ii) implies that

$$|T^{(i,0)}| \geq |T^{(i,1)}| \tag{4}$$

for all i. Of course, this partition process of T $(= T^{(0,0)})$ constitutes a part of the partition tree of ERT, as shown in Fig. 3.

In the above partition process, let α_1 be the positive integer such that $v_0^{(1)} = v_0^{(2)} = \cdots = v_0^{(\alpha_1)}$ and $v_0^{(\alpha_1)} \neq v_0^{(\alpha_1+1)}$ if such $v_0^{(\alpha_1+1)}$ exists, and α_2 be the positive integer such that $2|T^{(i,1)}| < |T^{(i,0)}|$ for all $i = 1,2,\ldots,\alpha_2 - 1$ and $2|T^{(\alpha_2,1)}| \geq |T^{(\alpha_2,0)}|$. Notice that $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ hold by definition. Now define

$$\alpha = \min\{\alpha_1, \alpha_2\}.$$

Let $v_0 = v_0^{(1)}$. Since $v_0 = v_0^{(i)}$ holds for $i = 1,2,\ldots,\alpha$, we have $\alpha \leq \deg(v_0)$. Let us consider the case of $\alpha = \deg(v_0)$; i.e., $T^{(\alpha,0)}$ consists of a single vertex v_0. In this case, since $v_0 = v_0^{(\alpha)}$ is a centroid vertex of $T^{(\alpha-1,0)}$, $|T^{(\alpha,1)}| \leq |T^{(\alpha-1,0)}|/2$ holds by Lemma 5. By $|T^{(\alpha-1,0)}| = |T^{(\alpha,0)}| + |T^{(\alpha,1)}|$ and $|T^{(\alpha,0)}| = 1$, we have $|T^{(\alpha-1,0)}| = 2$. However, since $T^{(\alpha-1,1)} \geq 1$, it holds $2|T^{(\alpha-1,1)}| \geq |T^{(\alpha-1,0)}|\,(=2)$, contradicting the definition of α. Hence, we have

$$\alpha \leq \deg(v_0) - 1 \quad (\leq \Delta_T - 1). \tag{5}$$

Claim: Algorithm ERT can give at most a $(\gamma - i)$-edge ranking for $T^{(i,1)}\,(= T_i)$, $i = 1,2,\ldots,\alpha$, and a $(\gamma - \alpha)$-edge ranking for $T^{(\alpha,0)}$.

Note that this will complete our induction, since Algorithm ERT then gives the ranks, which are at most $\gamma + 1 - i$, to the remaining edges $(v_0^{(i)}, v_1^{(i)})$, $i = 1,2,\ldots,\alpha$. Before showing our claim, we require the lemma which estimates the sizes of $T^{(i,j)}$.

Lemma 6 *Let v_0, $T^{(i,1)}$ $i = 1,2,\ldots,\alpha$ and $T^{(\alpha,0)}$ be defined as above. Then*

$$|T^{(i,1)}| \leq |T|/(i+2) \qquad i = 1,2,\ldots,\alpha - 1 \tag{6}$$
$$|T^{(\alpha,1)}| \leq |T|/(\alpha+1) \tag{7}$$
$$|T^{(\alpha,0)}| \leq 2|T|/(\alpha+2). \tag{8}$$

hold. Furthermore, if $\alpha = \deg(v_0) - 1$, we have

$$|T^{(\alpha,0)}| \leq (|T| + \alpha)/(\alpha+1). \tag{9}$$

Now we are ready to prove the above claim.

First, let us consider the edge ranking of $T^{(i,1)}$, $i = 1,2,\ldots,\alpha$. By Lemma 6, $|T^{(i,1)}| \leq |T|/(i+1)$ holds for all $i = 1,2,\ldots,\alpha$. By the induction hypothesis, Claim is proved by showing $\gamma - i \geq \frac{(\Delta_{T(i,1)}-2)\log(n/(i+1))}{\log\Delta_{T(i,1)}-1}$. Since $\frac{\Delta_T-2}{\log\Delta_T-1} \geq \frac{\Delta_{T(i,1)}-2}{\log\Delta_{T(i,1)}-1}$ holds by the monotonicity of γ in Δ_T, it suffices to show $\gamma - i \geq \frac{(\Delta_T-2)\log(n/(i+1))}{\log\Delta_T-1}$. which is equivalent to $f_1(i) \stackrel{\text{def}}{=} (\Delta_T - 2)\log(i+1) - i(\log\Delta_T - 1) \geq 0$. Since $1 \leq i \leq \Delta_T - 1$ holds by (5), and its derivative $f_1'(i) = (\Delta_T - 2)\log e/(i+1) - (\log\Delta_T - 1)$ is monotone decreasing in this range, it is sufficient to show that $f_1(1) \geq 0$ and $f_1(\Delta_T - 1) \geq 0$. Actually, $f_1(1) = f_1(\Delta_T - 1) = \Delta_T - \log\Delta_T - 1 \geq 2 - \log 3 > 0$, and thus Algorithm ERT gives a $(\gamma - i)$-edge ranking for $T^{(i,1)}$, $i = 1,2,\ldots,\alpha$.

Let us next consider the edge ranking of $T^{(\alpha,0)}$. Since $\alpha \leq \Delta_T - 1$ by (5), we divide it into two cases: (a) $\alpha \leq \Delta_T - 2$ and (b) $\alpha = \Delta_T - 1$.

Case (a): we have $|T^{(\alpha,0)}| \leq 2|T|/(\alpha+2)$. By the induction hypothesis, we shall show $\gamma - \alpha \geq \frac{(\Delta_{T(\alpha,0)}-2)\log(2n/(\alpha+2))}{\log\Delta_{T(\alpha,0)}-1}$. By the monotonicity of α, it suffices to show $\gamma - \alpha \geq \frac{(\Delta_T-2)\log(2n/(\alpha+2))}{\log\Delta_T-1}$. Case (b): $|T^{(\alpha,0)}| \geq 2$ holds by (5). If $|T^{(\alpha,0)}| = 2$ or 3, then Algorithm ERT gives 1- or 2-edge ranking of $T^{(\alpha,0)}$, respectively. If $|T^{(\alpha,0)}| \geq 4$, it follows from (9) that $|T^{(\alpha,0)}| \leq (|T| + \alpha)/(\alpha+1)$ holds, and by the induction hypothesis, we shall show $\gamma - \alpha \geq \frac{(\Delta_{T(\alpha,0)}-2)\log((n+\alpha)/(\alpha+1))}{\log\Delta_{T(\alpha,0)}-1}$. In every case, we can prove Claim by a direct case analysis in a manner similar to the case of $T^{(i,1)}$.

Consequently, the proof of the claim is completed.

4.2 Tight Examples of Algorithm ERT

Let $T_{(d,h)}$ denote a tree in which all the inner vertices have the same degree d and there exists a vertex v_0 such that the distances between v_0 and all the leaves are exactly h. This $T_{(d,h)}$ attains the upper bound of Lemma 4.

Lemma 7 *Let d and h be integers such that $d \geq 3$ and $h \geq 2$. Then, $T_{(d,h)}$ satisfies* $\mathrm{rank}(T_{(d,h)}) \geq \dfrac{(d-2)\log n}{\log(d-1)}$.

5 Conclusion

In this paper, we introduced the minimum edge ranking spanning tree problem (MERST), and proved that MERST is NP-hard, but it has a simple approximation algorithm, whose approximation ratio is $\min\{(\Delta^*-1)\log n/\Delta^*, \Delta^*-1\}/(\log(\Delta^*+1)-1)$, where n is the number of vertices in a given graph and Δ^* is the maximum degree of a spanning tree whose maximum degree is minimum.

Some issues remain for future work. One issue is finding special classes of graphs, in which MERST is polynomially solvable. Indeed, we could show that MERST is polynomially solvable for the class of threshold graphs [10].

Another issue is to consider the minimum vertex ranking spanning tree problem (MVRST), which is the vertex version of our problem. Although this problem seems to be as hard as MERST, its complexity is still open.

References

[1] Bodlaender, H. L., Deogun, J. S., Jansen, K., Kloks, T., Kratsch, D., Müller, H. and Tuza, Zs., Ranking of graphs, *SIAM J. Discrete Mathematics*, 11, 1, 168–181, 1998.

[2] de la Torre, P., Greenlaw, R. and Schäffer, A. A., Optimal edge ranking of trees in polynomial time, *Algorithmica*, 13, 592–618, 1995.

[3] Dyer, M. E. and Frieze, A. M., On the complexity of partitioning graphs into connected subgraphs, *Discrete Applied Mathematics*, 10, 139–153, 1985.

[4] Fürer, M. and Raghavachari, B., Approximating the minimum-degree Steiner tree to within one of optimal, *Journal of Algorithms*, 17, 409–423, 1994.

[5] Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[6] Ibaraki, T. and Kameda, T., On the optimal nesting order for computing N-relational joins, *ACM Transactions on Database Systems*, 9, 3, 482–502, 1984.

[7] Iyer, A. V., Ratliff, H. D. and Vijayan, G., On an edge-ranking problem of trees and graphs, *Discrete Applied Mathematics*, 30, 43–52, 1991.

[8] Lam, T. W. and Yue, F. L., Optimal edge ranking of trees in linear time, *Proc. 9th ACM-SIAM SODA*, 436–445, 1998.

[9] Lam, T. W. and Yue, F. L., Edge ranking of graphs is hard, *Discrete Applied Mathematics*, 85, 71–86, 1998.

[10] Makino, K., Uno, Y. and Ibaraki, T., Minimum edge ranking spanning trees of threshold graphs, Working paper.

[11] Uno, Y. and Ibaraki, T., Complexity of the optimum join order problem in relational databases, *IEICE Transactions*, E74, 7, 2067–2075, 1991.

[12] Zhou, X., Kashem, M. A. and Nishizeki. T., Generalized edge-rankings of trees, *IEICE Transactions*, E81-A, 2, 310–319, 1998.