Technical Report #99008

On minimum edge ranking spanning trees

Kazuhisa Makino, [†] Yushi Uno [§] and Toshihide Ibaraki [‡]

April 1999

Department of Applied Mathematics and Physics
Kyoto University, Kyoto, Japan 606-8501

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.

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

[§]Department of Mathematics and Information Sciences, Faculty of Integrated Arts and Sciences, Osaka Prefecture University, Sakai, 599-8531 Japan (uno@mi.cias.osakafu-u.ac.jp)

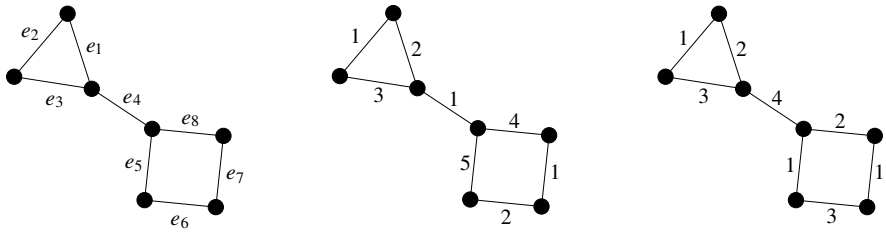
[‡]Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, 606 Japan. (ibaraki@i.kyoto-u.ac.jp)

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 $\mathrm{rank}(G) \leq k$)?

In the affirmative case, such a $\mathrm{rank}(G)$-edge ranking is also required. Figure 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 $G = (V, E)$ (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, 15]. It is known that MER is in general NP-hard [10], but it can be solved in polynomial time when the graph is a tree [2, 9, 15].

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 as *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 $\mathrm{rank}(T) \leq k$)?

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. Figure 2 gives an example of a minimum edge ranking spanning tree of the graph G in Figure 1 (a), together with its edge ranking. Figure 3 demonstrates the contraction steps defined by the T.

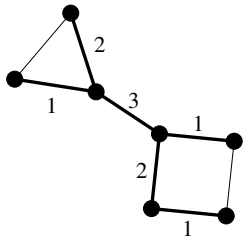


Figure 2: A minimum edge ranking spanning tree $T = (V, E_T)$ of the graph $G = (V, E)$ in Figure 1 (a).

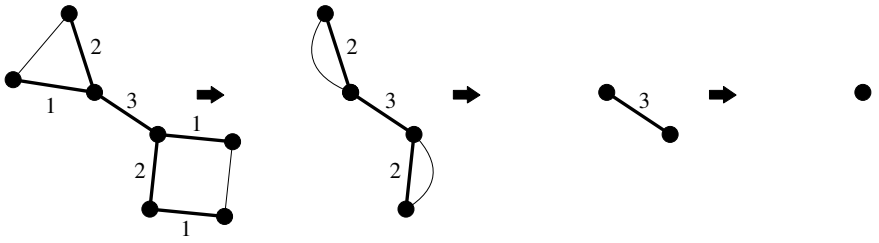


Figure 3: Edge contraction of the graph in Figure 1 (a) according to the minimum edge ranking spanning tree in Figure 2.

Problem MERST can be found in many practical applications.

Let us first consider a network of web sites, where the data are stored in different sites. In order to derive some new information from them, we want to merge a set of relevant data into one site. During the merging process, the pertaining data have to be locked to avoid the intervention from others such as being overwritten, and therefore no data can participate in more than one merging process at the same time. However, pairs of data can be merged simultaneously if they do not share any data. It is now not difficult to see that the minimization of the completion time of this merging process is formulated as MERST.

Another example is found in relational database theory. Let us consider the "query graph (join graph)" [6, 14], where its vertex set corresponds to 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 come into a single relational table. The whole process can again be modeled as MERST.

One more example [2, 7] is scheduling the parallel assembly of a multi-part product from its components. Given a set of connections between components to be combined in order to complete a product, we aim to minimize the number of steps of combining components in parallel under the condition that any component can not used in more than one combining step at a time.

In this paper, we show that MERST is NP-hard, and present an approximation algorithm for MERST, with 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. 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.

The rest of the paper is organized as follows. Section 2 shows the NP-hardness of MERST, and then Section 3 presents a simple approximation algorithm for MERST. In Section 4, we analyze our algorithm by deriving lower and upper bounds on the edge ranking of trees. Finally, Section 5 summarizes the paper.

2 NP-hardness of MERST

In this section, we show the NP-completeness of MERST, after showing several lemmas on the edge ranking. 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 $\mathrm{rank}(G) = k$ has at most 2^k vertices.*

Proof. Given a k-edge ranking r of a graph $G = (V, E)$, let $E_i = \{r(e) = i \mid e \in E\}$, $i = 1, 2, \ldots, k$ and $G_i = (V, E_1 \cup E_2 \cup \cdots \cup E_i)$, $i = 0, 1, \ldots, k$. By definition, G_0 has no edge and $G_k = G$.

Since G_i is obtained from G by removing $\sum_{j=i+1}^{k} |E_j|$ edges, G_i consists of at most $\sum_{j=i+1}^{k} |E_j| + 1$ connected components. By the definition of edge ranking, each connected component in G_i has at most one edge e with $r(e) = i$. Therefore, $|E_i|$ is bounded from above by

$$|E_i| \leq \sum_{j=i+1}^{k} |E_j| + 1,$$

where $|E^k| = 1$. By induction, we easily derive $|E_i| \leq 2^{k-i}$ from this equality. Thus, we have

$$\sum_{i=1}^{k} |E_i| \leq \sum_{i=1}^{k} 2^{k-i} = 2^k - 1.$$

A connected graph with at most $2^k - 1$ edges can have at most 2^k vertices. □

By the proof of Lemma 1, $\mathrm{rank}(G) = k$ holds for a graph $G = (V, E)$ with $|V| = 2^k$ only if $|E| = 2^k - 1$ holds; i.e., G is a tree (See Figure 4).

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 term is referred to as k-partition in [3])

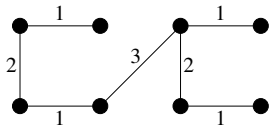


Figure 4: A tree T with $\mathrm{rank}(T) = 3$ and $2^3 = 8$ vertices.

of V is a $(|V|/k)$-tuple $(V_1, V_2, \ldots, V_{|V|/k})$ and $V = V_1 \cup V_2 \cup \ldots \cup V_{|V|/k}$, $V_i \cap V_j = \emptyset$ 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.*

Proof. To show the only-if part, let us assume that $G = (V, E)$ has a k-edge rankable spanning tree $T = (V, E_T)$. If $k = 0$ (i.e., $|V| = 1$), then G clearly has a nested partition. If $k \geq 1$, $\mathrm{rank}(T) = k$ holds by Lemma 1, since otherwise $(\mathrm{rank}(T) < k)$, we have $|V| < 2^k$, a contradiction. The spanning tree T has exactly one edge e with the highest rank k. Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be the subtrees obtained from T by removing e. Since T_1 and T_2 are both $(k-1)$-edge rankable, $|V_1|, |V_2| \leq 2^{k-1}$ holds by Lemma 1. Furthermore, since $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$, we have $|V_1| = |V_2| = 2^{k-1}$. Thus G has a connected size-2^{k-1}-partition (V_1, V_2). By recursively applying this argument to T_1 and T_2, we can show that G has a nested partition.

Let us next show the if part. Assume that $G = (V, E)$ has a nested partition. If $k = 0$ (i.e., $|V| = 1$), $T = (V, \emptyset)$ is clearly a 0-edge rankable spanning tree. Assume that the if part holds for $k = \ell$, and consider the case of $k = \ell + 1$. By definition, V has a connected size-2^ℓ-partition (V_1, V_2) such that the graphs $G[V_1]$ and $G[V_2]$ have a nested partition. By the induction hypothesis, each $G[V_i]$ has an ℓ-edge rankable spanning tree $T_i = (V_i, E_i)$, for $i = 1, 2$. Let r_i be such an edge ranking of T_i. We now define a spanning tree $T = (V, E_T)$ of G and a mapping $r \colon E_T \to Z^+$ by

$$E_T = E_1 \cup E_2 \cup \{e^*\}$$

$$r(e) = \begin{cases} r_1(e) & \text{if } e \in E_1 \\ r_2(e) & \text{if } e \in E_2 \\ k & \text{if } e = e^*, \end{cases}$$

where $e^* = (v, w)$ be an arbitrary edge in E such that $v \in V_1$ and $w \in V_2$. It is easy to see that this r is a k-edge ranking of T. □

We now show that MERST is intractable. The idea of our proof is based on the NP-hardness proof of the connected size-k-partition problem for planar bipartite graphs [3].

Theorem 1 *MERST is NP-complete.*

Proof. Given a spanning tree $T = (V, E_T)$, we can check whether T is k-edge rankable or not in polynomial time [9]. Hence, MERST belongs to NP.

To prove the completeness, we reduce the following NP-complete problem [5] to MERST.

3-Dimensional Matching (3DM)
Input: Disjoint sets $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, $Z = \{z_1, z_2, \ldots, z_n\}$ and a set of triples $S = \{s_j = (s_{j1}, s_{j2}, s_{j3}), \, | \, s_{j1} \in X, s_{j2} \in Y, s_{j3} \in Z, \ j = 1, 2, \ldots, m\}$.
Question: Does S contain a matching M? (i.e., $|M| = n$ and every element of $W = X \cup Y \cup Z$ is covered by exactly one triple of M?)

An instance I of 3DM is depicted in Figure 5 as a bipartite graph with vertex sets S and W, where $|S| = m$ and $|W| = 3n$. It is asked to find a set of n vertices in S that covers all the vertices in W exactly once.

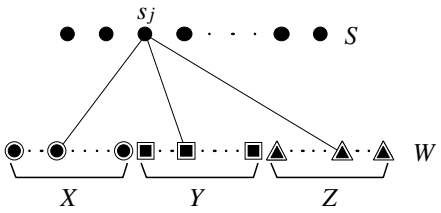


Figure 5: An instance of 3DM.

Without loss of generality, we can assume that

$$n + m = 2^\ell \tag{1}$$

holds for some integer ℓ. Otherwise, let ℓ be the smallest integer such that $2^\ell - (n + m) \geq 2$, and let α and β be nonnegative integers such that $2\alpha + \beta = 2^\ell - (n + m)$ and $\beta \in \{0, 1\}$. Then, define the instance I' of 3DM by

$$\begin{aligned}
X' &= X \cup \{x_{n+1}, x_{n+2}, \ldots, x_{n+\alpha}\} \\
Y' &= X \cup \{y_{n+1}, y_{n+2}, \ldots, y_{n+\alpha}\} \\
Z' &= X \cup \{z_{n+1}, z_{n+2}, \ldots, z_{n+\alpha}\} \\
S' &= \begin{cases} S \cup \{S_{m+j} = \{x_{n+j}, y_{n+j}, z_{n+j}\} \, | \, j = 1, 2, \ldots, \alpha\} & \text{if } \beta = 0 \\ S \cup \{S_{m+j} = \{x_{n+j}, y_{n+j}, z_{n+j}\} \, | \, j = 1, 2, \ldots, \alpha\} \cup \{x_{n+1}, y_{n+1}, z_1\} & \text{if } \beta = 1 \end{cases}
\end{aligned}$$

It is easy to see that S has a matching M if and only if S' has a matching $M' = M \cup \{S_{m+j} \, | \, j = 1, 2, \ldots, \alpha\}$. Thus the assumption (1) does not affect the NP-hardness of 3DM. (This intermediate instance is illustrated as I' in Figure 7.)

Let us now construct the instance of MERST from an instance of 3DM (i.e., X, Y, Z and S). Let $X^* = \{x_i^{(q)} \mid x_i \in X, \ q = 1, 2, 3\}$, $Y^* = \{y_i^{(q)} \mid y_i \in Y, \ q = 1, 2, 3\}$, $Z^* = \{z_i^{(q)} \mid z_i \in Z, \ q = 1, 2\}$, and $S^* = \{s_j^{(p)} \mid s_j \in S, \ p = 1, 2, \ldots, 8\}$. Define an input graph $G = (V, E)$ by

$$\begin{aligned}
V &= X^* \cup Y^* \cup Z^* \cup S^* \\
E &= \bigcup_{h=1}^{6} E_h,
\end{aligned} \tag{2}$$

where

$$E_1 = \{(x_i^{(1)}, x_i^{(2)}),\ (x_i^{(2)}, x_i^{(3)}) \mid x_i \in X\}$$
$$E_2 = \{(y_i^{(1)}, y_i^{(2)}),\ (y_i^{(2)}, y_i^{(3)}) \mid y_i \in Y\}$$
$$E_3 = \{(z_i^{(1)}, z_i^{(2)}) \mid z_i \in Z\}$$
$$E_4 = \{(s_j^{(p)}, s_j^{(p+1)}) \mid s_j \in S,\ p = 1, 2, \ldots, 7\}$$
$$E_5 = \{(s_j^{(1)}, s_{j1}^{(1)}),\ (s_j^{(2)}, s_{j2}^{(1)}),\ (s_j^{(3)}, s_{j3}^{(1)}) \mid s_j = (s_{j1}, s_{j2}, s_{j3}) \in S\}$$
$$E_6 = \{(s_j^{(5)}, s_{j'}^{(5)}) \mid s_j, s_{j'} \in S\}.$$

Recall that, in the instance of 3DM in Figure 5, a triple $s_j = (s_{j1}, s_{j2}, s_{j3}) \in S$ forms a *claw* including three vertices s_{j1}, s_{j2} and $s_{j3} \in W$, where s_{j1}, s_{j2} and s_{j3} are respectively represented by a doubled circle, a doubled square and a doubled triangle. We refer to these three vertices in $X \cup Y \cup Z$ as *matching vertices* and the last vertex $s_j \in S$ as a *tuple vertex*. In our transformation of (2), we replace each claw $s_j = (s_{j1}, s_{j2}, s_{j3})$ into a *comb* as shown in the right of Figure 6. A comb consists of sixteen vertices, where $s_{j1}^{(q)}$, $q = 1, 2, 3$, $s_{j2}^{(q)}$, $q = 1, 2, 3$, and $s_{j3}^{(q)}$, $q = 1, 2$, correspond to the matching vertices s_{j1}, s_{j2} and s_{j3} in a claw, respectively, and the remaining eight vertices $s_j^{(p)}$, $p = 1, 2, \ldots, 8$, correspond to the tuple vertex s_j in a claw. Note that the comb corresponding to s_j may share the vertices $s_{jh}^{(q)}$ with those corresponding to $s_{j'}$ if $s_{jh} = s_{j'h}$ holds. All combs are completely connected by edges in E_6. Also note that

$$|V| = 8m + 8n \ = \ 2^3(m+n) \ = \ 2^{\ell+3}$$

holds by assumption (1). Thus, we can let $k = \ell + 3$ (See Figure 7).

Then, we only need to prove the following property for the NP-completeness proof.

(i) S has a matching M if and only if the resulting graph G has a k-edge rankable spanning tree T.

To prove (i), we shall show below that

(ia) S has a matching M if and only if the graph G has a connected size-4-partition, and
(ib) the graph G has a connected size-4-partition if and only if it has a nested partition,

since (ia) and (ib) imply claim (i), according to Lemma 2.

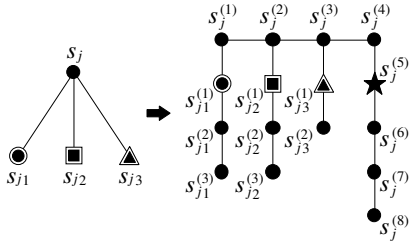


Figure 6: Transformation from a claw in 3DM to a comb in MERST.

Let us first show (ia). Suppose that S has a matching M. For a triple $s_j = (s_{j1}, s_{j2}, s_{j3}) \in M$, let

$$V_{j1} = \{s_j^{(1)}, s_{j1}^{(1)}, s_{j1}^{(2)}, s_{j1}^{(3)}\}$$

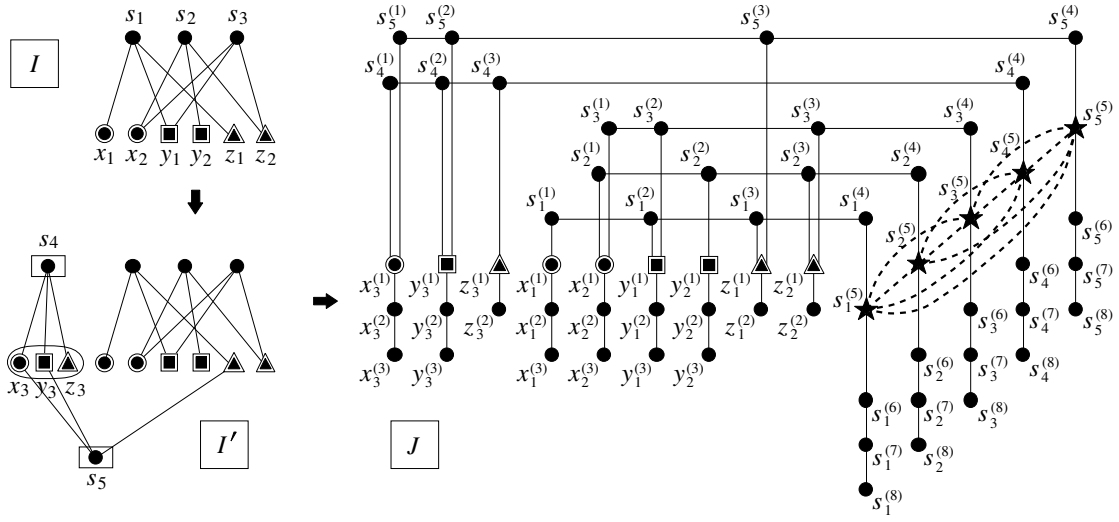


Figure 7: Transformation from set of triples $S = \{s_1, s_2, s_3\}$ of 3DM to the input graph G of MERST.

$$
\begin{aligned}
V_{j2} &= \{s_j^{(2)}, s_{j2}^{(1)}, s_{j2}^{(2)}, s_{j2}^{(3)}\} \\
V_{j3} &= \{s_j^{(4)}, s_j^{(3)}, s_{j3}^{(1)}, s_{j3}^{(2)}\} \\
V_{j4} &= \{s_j^{(5)}, s_j^{(6)}, s_j^{(7)}, s_j^{(8)}\},
\end{aligned} \tag{3}
$$

and for a triple $s_j = (s_{j1}, s_{j2}, s_{j3}) \in S \setminus M$, let

$$
\begin{aligned}
V_{j1} &= \{s_j^{(1)}, s_j^{(2)}, s_j^{(3)}, s_j^{(4)}\} \\
V_{j2} &= \{s_j^{(5)}, s_j^{(6)}, s_j^{(7)}, s_j^{(8)}\}.
\end{aligned} \tag{4}
$$

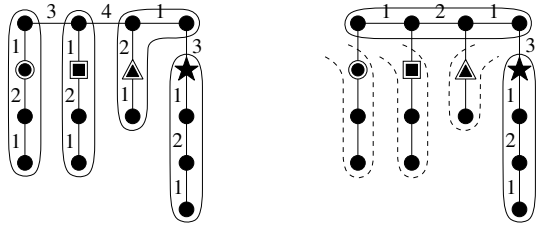
Figure 8 (a) and (b) show the situations (3) and (4), respectively. It is easy to see that each V_{jq} is connected and satisfies $|V_{jq}| = 4$. Furthermore, since M is a matching of S,

$$
\begin{aligned}
V &= \bigcup_{j,q} V_{jq} \\
V_{jq} \cap V_{j'q'} &= \emptyset \quad \text{for all } (j,q) \neq (j',q').
\end{aligned}
$$

Thus the only-if part holds.

Conversely, assuming that V has a connected size-4-partition, we show that each element of the partition is represented by either (3) or (4).

Let us consider vertices $x_i^{(3)}, x_i^{(2)}$ and $x_i^{(1)} \in X^*$. Since the degrees of $x_i^{(3)}$ and $x_i^{(2)}$ are 1 and 2, respectively, and $(x_i^{(3)}, x_i^{(2)}), (x_i^{(2)}, x_i^{(1)}) \in E$, vertices $x_i^{(3)}, x_i^{(2)}$ and $x_i^{(1)}$ must be contained in the same element. For every i, $x_i^{(1)}$ is adjacent to $x_i^{(2)}$ and all vertices $s_j^{(1)}$ such that $s_{j1} = x_i$. Thus, there exists an index j such that $s_{j1} = x_i$ and $V_{x_i} = \{x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, s_j^{(1)}\}$ is an element of the partition. Similarly, for every i, there exists an index j such that $s_{j2} = y_i$ and $V_{y_i} = \{y_i^{(1)}, y_i^{(2)}, y_i^{(3)}, s_j^{(2)}\}$ is an element of the partition. Also for every j, $V_{s_j} = \{s_j^{(5)}, s_j^{(6)}, s_j^{(7)}, s_j^{(8)}\}$ is an element of the partition. Moreover, we can see that $z_i^{(2)}$ and $z_i^{(1)}$ must be contained in the same element, and that $s_j^{(3)}$ and $s_j^{(4)}$ must be contained in the same element (note that the degree of $s_j^{(4)}$ is 1 in the graph $G[V \setminus (\bigcup_{j=1}^{m} V_{s_j})]$). These imply that,



(a) In case of $s_i = (s_{i1}, s_{i2}, s_{i3}) \in M$ (b) In case of $s_i = (s_{i1}, s_{i2}, s_{i3}) \in S \setminus M$

Figure 8: Two possible patterns of connected size-4-partition and their edge ranks.

for every i, there exists an index j such that $s_{j3} = z_i$ and $V_{z_i} = \{z_i^{(1)}, z_i^{(2)}, s_j^{(3)}, s_j^{(4)}\}$ is an element of the partition. By the definitions of V_{x_i}, V_{y_i} and V_{z_i}, the following statements are equivalent.

(I) $s_j^{(1)}$ is contained in element $V_{s_{j1}}$.
(II) $s_j^{(2)}$ is contained in element $V_{s_{j2}}$.
(III) $s_j^{(3)}$ is contained in element $V_{s_{j3}}$.

Therefore, if an index j satisfies these conditions (I), (II) and (III), then elements $V_{s_{jq}}$, $q = 1, 2, 3$, and V_{s_j} correspond to the elements in (3); otherwise, V_{s_j} and $V'_{s_j} = \{s_j^{(1)}, s_j^{(2)}, s_j^{(3)}, s_j^{(4)}\}$ correspond to the elements in (4). In other words, all s_j of type (3) define a matching M, proving the if part.

Figure 9 illustrates a connected size-4-partition of the graph G, which is transformed from S in Figure 7. It is easy to see that this connected size-4-partition gives to a matching $M = \{s_1, s_2\}$ of S.

Let us finally show (ib). Clearly we only need to prove the only-if part. By the above discussion, each element of a connected size-4-partition corresponds to either (3) or (4). Since each element V_{jq} induces a path of four vertices, $G[V_{jq}]$ has a nested partition.

For all $h = 3, 4, \ldots, k$, we now construct a connected size-2^h-partition of V, in which each element induces a graph with a nested partition. We assume without loss of generality that the combs representing s_j, $j = 1, 2, \ldots, n$, are divided as (3) (i.e., $M = \{s_j \mid j = 1, 2, \ldots, n\}$), and those representing s_j, $j = n+1, n+2, \ldots, m$, are divided as (4).

First, consider the case of $h = 3$. For every $j = 1, 2, \ldots, n$, let $W_{j1} = V_{j1} \cup V_{j2}$ and $W_{j2} = V_{j3} \cup V_{j4}$ (see (3)), and for every $j = n+1, n+2, \ldots, m$, let $W_{j1} = V_{j1} \cup V_{j2}$ (see (4)). Then these W_{j1} and W_{j2} for $j = 1, 2, \ldots, m$ form a connected size-2^3-partition of V having a desired property, since all $G[V_{jq}]$ have nested partitions.

Next consider the case of $h = 4$. For every $j = 1, 2, \ldots, n$, let $U_j = W_{j1} \cup W_{j2}$, and for every $j = n+1, n+3, \ldots, m-1$, let $U_{j,j+1} = W_{j1} \cup W_{j+1,1}$. Since $m - n$ is even by $m + n = 2^\ell$ (even), $U_{j,j+1}$ is well-defined for all j. These U_j and $U_{j,j+1}$ form a connected size-2^4-partition of V having a desired property, since all $G[W_{j1}]$ and $G[W_{j2}]$ have nested partitions. Since all pairs of U_j and $U_{j',j'+1}$ are connected by the edges in E_6, by applying a similar argument to the cases $h = 5, 6, \ldots, k$, we can construct connected size-2^h-partitions of V having the property of nested partitions. This proves the only-if part. □

In Figure 9, we actually show a k-edge rankable spanning tree ($k = 6$) by bold edges. The edge ranking of combs in Figure 8, together with the edge ranking of the remaining edges in 9 give the obtained k-edge ranking of the spanning tree.

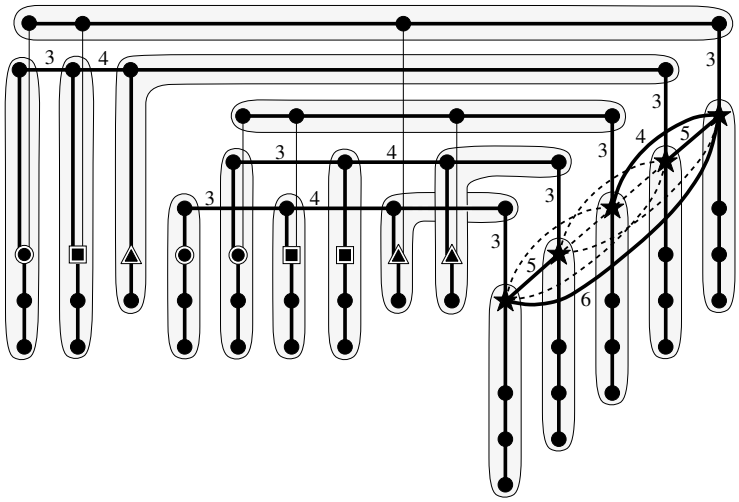


Figure 9: Edge ranking of a spanning tree (bold edges) of the graph G in Figure 7, where the edge ranking of individual combs is given in Figure 8.

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 Section 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

$$\Delta^* \leq \Delta_T \ \leq \ \Delta^* + 1 \ (\leq \Delta_G). \tag{5}$$

Our approximation algorithm for MERST first computes a spanning tree T_{Approx} of G satisfying (5) (by using the algorithm in [4]), and then compute its minimum edge ranking. Recall that MERT is polynomially solvable (see, e.g., [9]). Thus, our algorithm can be executed in polynomial time. Formally it can be described as follows.

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_{Approx} of G satisfying (5).

Step 2: Compute a minimum edge ranking r of T_{Approx}.
Step 3: Output $T = T_{\mathrm{Approx}}$ 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 graph G. Then, the approximation ratio of algorithm* Approx_MERST *can be bounded from above as*

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

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

In the next section, we prove the stated approximation ratio by deriving lower and upper bounds on the edge ranking of trees.

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 the edge ranks $\mathrm{rank}(T)$ 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 Bound on the Edge Rank of Trees

First of all, we can derive a simple lower bound on the edge rank $\mathrm{rank}(T)$ of a tree T.

Lemma 3 *For any tree $T = (V, E_T)$,*

$$\mathrm{rank}(T) \geq \max\{\Delta_T, \lceil \log n \rceil\} \tag{6}$$

holds, where Δ_T is the maximum degree of vertices in T and $n = |V|$.

Proof. By definition, no two edges with the same rank can be adjacent. Thus, T requires at least Δ_T different ranks. Moreover, since a tree T has at most $2^{\mathrm{rank}(T)}$ vertices by Lemma 1, we have $2^{\mathrm{rank}(T)} \geq n$, i.e., $\mathrm{rank}(T) \geq \lceil \log n \rceil$. □

It is easy to see that 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 $\mathrm{rank}(T_2) = \lceil \log n \rceil$. For example, a star graph $K_{1,n-1}$ and a line graph P_n give such examples of T_1 and T_2, respectively.

4.2 Upper Bound on the Edge Rank of Trees

The edge rank $\mathrm{rank}(T)$ of a tree T has the following upper bound.

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

$$\mathrm{rank}(T) = \lceil \log n \rceil \qquad \text{if } \Delta_T = 0, 1, 2 \tag{7}$$

$$\mathrm{rank}(T) \leq \frac{(\Delta_T - 2)\log n}{\log \Delta_T - 1} \qquad \text{if } \Delta_T \geq 3. \tag{8}$$

Recall that a tree T is a line when $\Delta_T = 0, 1, 2$. Therefore, equality (7) of Lemma 4 is clear since, as noted earlier, we have $\mathrm{rank}(T) = \lceil \log n \rceil$ in these cases. To prove inequality (8), we need to show that $\dfrac{\Delta_T - 2}{\log \Delta_T - 1}$ is monotone nondecreasing in Δ_T for $\Delta_T \geq 3$ as well as we shall show inequality (8) itself. The proof of monotonicity can be seen in Lemma 11 in Appendix, and the proof of the upper bound will be given in Subsections 4.2.1 and 4.2.2.

4.2.1 Edge Ranking of a Tree by Top-down Approach

For the purpose of evaluating the upper bound on $\mathrm{rank}(T)$ in Lemma 4, we present an algorithm which gives a consistent (but not always minimum) edge ranking of trees.

All the existing exact algorithms for MERT [2, 9, 15] 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 single connected component 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 Figure 10. We give the same rank to all the edges removed in the same depth of the partition tree; the highest rank k in depth 0, rank $k-1$ in depth 1, and so on, where k is the height of the partition tree.

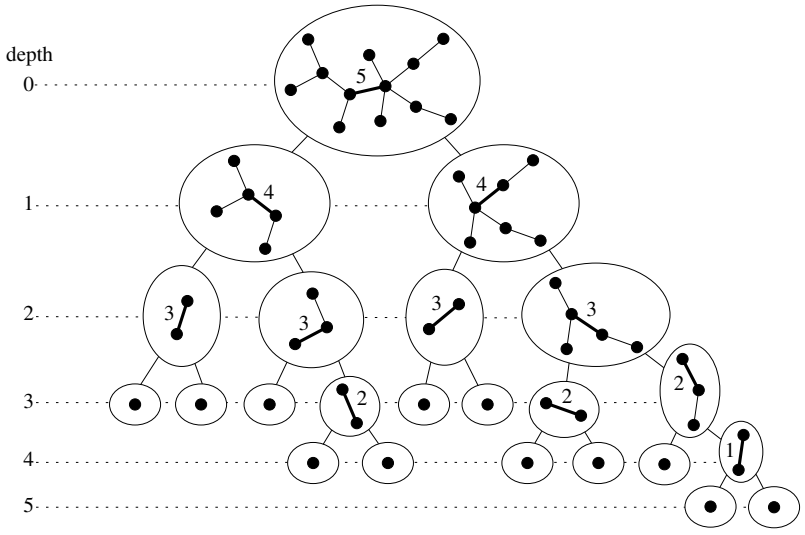


Figure 10: Partition tree constructed by top-down approach.

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 only of T.

Step 2: If there is a subtree T', which has more than one vertex and is located in a leaf position of the 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 current partition tree. The two subtrees resulting from the removal of e become the children of T' in the partition tree. If there is no subtree satisfying the above condition, then go to Step 3; otherwise return to Step 2.

Step 3: Let h be the height of the resulting partition tree, and define the ranks of 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 1 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 $\big|\,|V_1| - |V_2|\,\big|$ is minimum. An edge is called *optimal* if its removal produces an optimal connected partition.

In order to characterize optimal edges, we define some more definitions on trees.

Definition 2 Given a tree $T = (V, E_T)$, 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*.

We show in Figure 11 two example trees and weights of vertices. The centroid vertices are indicated by doubled circles.

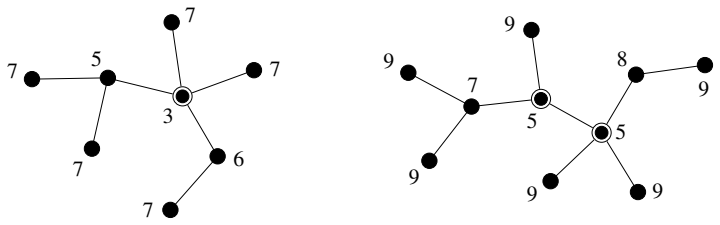


Figure 11: Weights of vertices and centroid.

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). This is illustrated in Figure 12. By definition, $|T_1| \geq |T_2| \geq \cdots \geq |T_b|$, and hence $w(v_0) = |T_1|$ holds.

The following lemmas characterize a centroid vertex v.

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.*

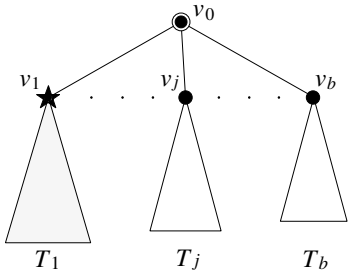


Figure 12: A tree T rooted at a vertex v_0.

Proof. For a vertex $v_0 \in V$, let T_j $(j = 1, 2, \ldots, b)$ be defined as above. For the only-if part, let us assume that $w(v_0) = |T_1| > n/2$ holds. In this case, we regard v_1 as the new root. Then, the subtree $\{v_0\} \cup T_2 \cup T_3 \cup \cdots \cup T_b$ has the size $n - |T_1| (< n/2)$. Since the sizes of other subtrees of v_1 (contained in T_1) are less than $|T_1|$, we can conclude that v_0 is not a centroid vertex.

Conversely, if $w(v_0) = |T_1| \leq n/2$, then $w(v_j^*) \geq n - |T_j| \geq n/2$ holds for all vertices v_j^* in T_j, where $j = 1, 2, \ldots, b$. Thus, v_0 is a centroid vertex of T. □

Lemma 6 *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.*

Proof. Suppose that v_0 is a centroid vertex of T, and let $v_1, v_2, \ldots, v_b, T_1, T_2, \ldots, T_b$ be defined as above. By Lemma 5, $w(v_0) = |T_1| \leq n/2$ holds. Note that no $v \in \bigcup_{j=2}^{b} T_j$ is a centroid vertex, since its weight $w(v)$ is at least $|T_1| + 1 > w(v_0)$. Furthermore, no $v \in T_1 \setminus \{v_1\}$ is a centroid vertex, because its weight $w(v)$ is also greater than $n - |T_1| \geq n/2$. Thus, there remain only the following two cases: v_0 is the only centroid vertex, or v_0 and v_1 (which are adjacent) are centroid vertices. If both v_0 and v_1 are centroid vertices (i.e., $w(v_0) = w(v_1)$), by Lemma 5, we have $w(v_0) = |T_1| \leq n/2$ and $w(v_1) \geq n - |T_1| \geq n/2$. Then $w(v_0) = w(v_1)$ implies $w(v_0) = w(v_1) = n/2$. □

Figure 11 shows the two possible cases in Lemma 6. 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. The following lemma immediately follows from Lemmas 5 and 6.

Lemma 7 *For a tree T, let v_0 and v_1 be defined as above. Then (v_0, v_1) is an optimal edge of T.*

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.

4.2.2 Analysis of Algorithm ERT

Let

$$\gamma = \frac{(\Delta_T - 2) \log n}{\log \Delta_T - 1}.$$

In this subsection, we prove the inequality (8) of Lemma 4 holds; i.e., Algorithm ERT always gives a γ-edge ranking to a tree $T = (V, E_T)$, where $n = |V|$ and we assume $\Delta_T \geq 3$.

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. Recall here again that $\frac{\Delta_{T'}-2}{\log \Delta_{T'}-1} \leq \frac{\Delta_T-2}{\log \Delta_T-1}$ holds for $\Delta_{T'} \leq \Delta_T$ by its monotonicity in Δ_T.

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

$$|T^{(i,0)}| \geq |T^{(i,1)}| \tag{9}$$

for all i. This partition process of T ($= T^{(0,0)}$) can be illustrated as the binary tree of Figure 13. Of course, this constitutes a part of the partition tree of ERT, as shown in Figure 10.

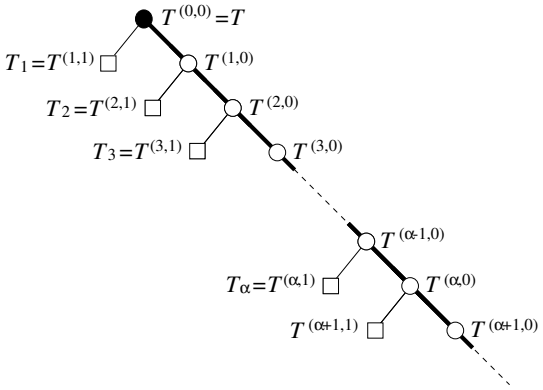


Figure 13: Partition of T.

In the 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 $\alpha \leq \deg(v_0)$, where $\deg(v)$ denotes the degree of v in T. 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

$$\alpha \leq \deg(v_0) - 1 \quad (\leq \Delta_T - 1). \tag{10}$$

Lemma 8 *$2|T^{(\alpha,1)}| \geq |T^{(\alpha,0)}|$ holds.*

Proof. Let us consider the case of $\alpha = \alpha_1$, since otherwise (i.e., $\alpha = \alpha_2$), the lemma clearly holds. Recall the definitions of $v_1, v_2, \ldots, v_{\deg(v_0)}$ and $T_1, T_2, \ldots, T_{\deg(v_0)}$ for v_0 in T. We can view that $v_i = v_1^{(i)}$ and $T_i = T^{(i,1)}$ for all $i = 1, 2, \ldots, \alpha$. Note that $T_{\alpha+1}$ is well-defined by (10), and hence $|T^{(\alpha,0)}| \geq |T_{\alpha+1}|+1 \geq 2$. Since this implies $v_0^{(\alpha)} \neq v_0^{(\alpha+1)}$ (i.e., $v_0^{(\alpha+1)}$ is well-defined), we have $|T_{\alpha+1}| > |T^{(\alpha,0)}|/2$ by Lemma 5. Therefore, $2|T^{(\alpha,1)}| > |T^{(\alpha,0)}|$ holds by $|T^{(\alpha,1)}| (= |T_\alpha|) \geq |T_{\alpha+1}|$. □

Now we prove the next claim.

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 $(v_0^{(i)}, v_1^{(i)})$, $i = 1, 2, \ldots, \alpha$ (as obvious if we consider the depths of these edges in the partition tree of ERT), and hence T has a γ-edge ranking.

The following lemma estimates the sizes of $T^{(i,j)}$ from above in terms of $|T|$.

Lemma 9 *Let v_0, $T^{(i,1)}$, $i = 1, 2, \ldots, \alpha$ and $T^{(\alpha,0)}$ be defined as above. Then it holds that*

$$|T^{(i,1)}| \leq \frac{|T|}{i+2} \qquad i = 1, 2, \ldots, \alpha - 1 \tag{11}$$

$$|T^{(\alpha,1)}| \leq \frac{|T|}{\alpha+1} \tag{12}$$

$$|T^{(\alpha,0)}| \leq \frac{2|T|}{\alpha+2}. \tag{13}$$

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

$$|T^{(\alpha,0)}| \leq \frac{|T| + \alpha}{\alpha+1}. \tag{14}$$

Proof. First, recall again that

$$|T^{(1,1)}| \ \geq \ |T^{(2,1)}| \ \geq \ \cdots \ \geq \ |T^{(\alpha,1)}| \tag{15}$$

hold by definition.

(11): By the definition of α, we have $|T^{(i,0)}| > 2|T^{(i,1)}|$ for $i = 1, 2, \ldots, \alpha - 1$. Combining this with (15), it implies that $|T| = |T^{(1,1)}| + |T^{(2,1)}| + \cdots + |T^{(i,1)}| + |T^{(i,0)}| \geq (i+2)|T^{(i,1)}|$, i.e., (11) holds.

(12): Combining $|T^{(\alpha,0)}| \geq |T^{(\alpha,1)}|$ (see (9)) and (15), we have $|T| \geq (\alpha+1)|T^{(\alpha,1)}|$ which is equivalent to (12).

(13): First, $|T^{(\alpha,1)}| \geq |T^{(\alpha,0)}|/2$ holds as noted before Lemma 8. Furthermore, by (15), we have $|T| = \sum_{i=1}^{\alpha} |T^{(i,1)}| + |T^{(\alpha,0)}| \geq \alpha|T^{(\alpha,0)}|/2 + |T^{(\alpha,0)}|$, and hence (13) holds.

(14): If $\alpha = \deg(v_0) - 1$, then $|T^{(\alpha,0)}| = |T_{\alpha+1}| + 1$ and hence $|T^{(\alpha-1,0)}| = |T^{(\alpha,1)}| + |T_{\alpha+1}| + 1$ hold. Moreover, $2|T^{(\alpha-1,1)}| < |T^{(\alpha-1,0)}|$ holds by the definition of α. Since $|T^{(\alpha-1,1)}| \geq |T^{(\alpha,1)}| \geq |T_{\alpha+1}|$, we have $|T^{(\alpha-1,1)}| = |T^{(\alpha,1)}| = |T_{\alpha+1}|$. Thus this, together with (15), implies

$$|T| = \sum_{i=1}^{\alpha} |T^{(i,1)}| + |T^{(\alpha,0)}| \geq \alpha|T^{(\alpha,1)}| + |T^{(\alpha,0)}| \geq (\alpha+1)(|T^{(\alpha,0)}| - 1) + 1,$$

which is (14). □

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 9, $|T^{(i,1)}| \leq |T|/(i+1)$ holds for all $i = 1, 2, \ldots, \alpha$. By the induction hypothesis, we shall show $\gamma - i \geq \frac{(\Delta_{T^{(i,1)}} - 2)\log(n/(i+1))}{\log \Delta_{T^{(i,1)}} - 1}$. Since $\frac{\Delta_T - 2}{\log \Delta_T - 1} \geq \frac{\Delta_{T^{(i,1)}} - 2}{\log \Delta_{T^{(i,1)}} - 1}$ holds by its monotonicity in Lemma 11, it suffices to show $\gamma - i \geq \frac{(\Delta_T - 2)\log(n/(i+1))}{\log \Delta_T - 1}$, which is equivalent to

$$f_1(i) \stackrel{\text{def}}{=} (\Delta_T - 2)\log(i+1) - i(\log \Delta_T - 1) \ \geq \ 0. \tag{16}$$

Since $1 \leq i \leq \Delta_T - 1$ holds by (10), 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 (10), we divide it into two cases: (a) $\alpha \leq \Delta_T - 2$ and (b) $\alpha = \Delta_T - 1$.

Case $\alpha \leq \Delta_T - 2$: By Lemma 9, we have $|T^{(\alpha,0)}| \leq 2|T|/(\alpha+2)$. By the induction hypothesis, we shall show $\gamma - \alpha \geq \frac{(\Delta_{T^{(\alpha,0)}} - 2)\log(2n/(\alpha+2))}{\log \Delta_{T^{(\alpha,0)}} - 1}$. Since $\frac{\Delta_T - 2}{\log \Delta_T - 1} \geq \frac{\Delta_{T^{(\alpha,0)}} - 2}{\log \Delta_{T^{(\alpha,0)}} - 1}$ holds by its monotonicity in Lemma 11, it suffices to show $\gamma - \alpha \geq \frac{(\Delta_T - 2)\log(2n/(\alpha+2))}{\log \Delta_T - 1}$, which is equivalent to

$$f_2(\alpha) \stackrel{\text{def}}{=} (\Delta_T - 2)(\log(\alpha+2) - 1) - \alpha(\log \Delta_T - 1) \ \geq \ 0.$$

Since $1 \leq \alpha \leq \Delta_T - 2$, and its derivative $f_2'(\alpha) = (\Delta_T - 2)\log e/(\alpha+2) - (\log \Delta_T - 1)$ is monotone decreasing in this range, it is sufficient to show $f_2(1) \geq 0$ and $f_2(\Delta_T - 2) \geq 0$. Since $f_2(1) = (\Delta_T - 2)(\log 3 - 1) - \log \Delta_T + 1$ is monotone increasing in Δ_T (≥ 3), we have

$$f_2(1) \geq (3-2)(\log 3 - 1) - \log 3 + 1 \ = \ 0.$$

Similarly, we have

$$f_2(\Delta_T - 2) \geq (\Delta_T - 2)(\log \Delta_T - 1) - (\Delta_T - 2)(\log \Delta_T - 1) \ = \ 0.$$

Case $\alpha = \Delta_T - 1$: $|T^{(\alpha,0)}| \geq 2$ holds by (10).

If $|T^{(\alpha,0)}| = 2$, then Algorithm ERT clearly gives a 1-edge ranking for $T^{(\alpha,0)}$. Thus we have only to show $\gamma - \alpha \geq 1$, which is equivalent to

$$(\Delta_T - 2)\log n - (\log \Delta_T - 1)(\alpha + 1) \geq 0. \tag{17}$$

Since $\alpha = \Delta_T - 1$ and $n \geq \Delta_T + 1$ by (14), it is sufficient to show

$$f_3(\Delta_T) \stackrel{\text{def}}{=} (\Delta_T - 2)\log(\Delta_T + 1) - \Delta_T(\log \Delta_T - 1) \ \geq \ 0.$$

This is proved as Lemma 12 in Appendix.

Similarly, if $|T^{(\alpha,0)}| = 3$, then Algorithm ERT clearly gives a 2-edge ranking for $T^{(\alpha,0)}$. Thus we have only to show $\gamma - \alpha \geq 2$, which is equivalent to

$$(\Delta_T - 2)\log n - (\log \Delta_T - 1)(\alpha + 2) \geq 0. \tag{18}$$

Since $\alpha = \Delta_T - 1$ and $n = |T| \geq 2\Delta_T + 1 \geq 2\Delta_T$ by (14), it is sufficient to show

$$\begin{aligned} f_4(\Delta_T) &\stackrel{\text{def}}{=} (\Delta_T - 2)\log 2\Delta_T - (\Delta_T + 1)(\log \Delta_T - 1) \\ &= 2\Delta_T - 3\log \Delta_T - 1 \ \geq \ 0. \end{aligned}$$

Since $f_4(\Delta_T)$ is monotone increasing in Δ_T for $\Delta_T \geq 3$, and $f_4(3) = 5 - 3\log 3 > 0$, we have $f_4(\Delta_T) \geq 0$ if $\Delta_T \geq 3$.

Finally we consider the case of $|T^{(\alpha,0)}| \geq 4$. Note that in this case, we have $|T^{(\alpha,0)}| \leq (|T|+\alpha)/(\alpha+1)$ by (14). Thus, by the induction hypothesis, we shall show $\gamma - \alpha \geq \frac{(\Delta_{T^{(\alpha,0)}}-2)\log((n+\alpha)/(\alpha+1))}{\log \Delta_{T^{(\alpha,0)}}-1}$. By Lemma 11, it suffices to show that $\gamma - \alpha \geq \frac{(\Delta_T-2)\log((n+\alpha)/(\alpha+1))}{\log \Delta_T - 1}$. Since $\alpha = \Delta_T - 1$, this is equivalent to

$$\begin{aligned} f_5(\Delta_T) &\stackrel{\text{def}}{=} (\Delta_T - 2)\log n - (\Delta_T - 1)(\log \Delta_T - 1) - (\Delta_T - 2)(\log(n + \Delta_T - 1) - \log \Delta_T) \\ &= (\Delta_T - 2)(\log n - \log(n + \Delta_T - 1)) + \Delta_T - \log \Delta_T - 1 \ \geq \ 0. \end{aligned}$$

Since $n = |T| \geq 3\Delta_T + 1$ by (14), this is proved as Lemma 13 in Appendix.

Consequently, the proof of the claim is completed.

4.2.3 Tight Examples of Algorithm ERT

In this subsection, we present a family of trees T which shows that the upper bound of Corollary 4 in Subsection 4.2 is tight. 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. We regard v_0 as the root of $T_{(d,h)}$; the height of $T_{(d,h)}$ is h. For example, Figure 14 shows a tree $T_{(4,5)}$.

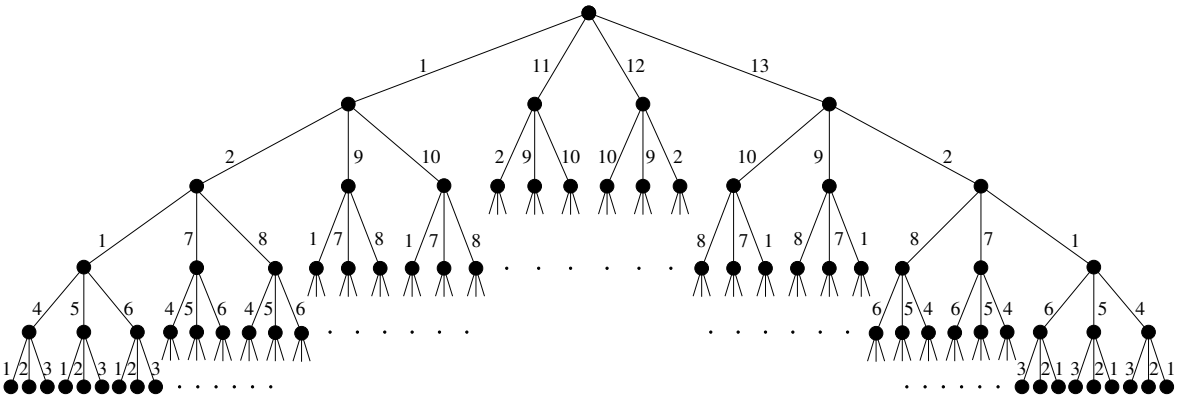


Figure 14: $T_{(4,5)}$ and its minimum edge ranking.

Lemma 10 *Let d and h be integers such that $d \geq 3$ and $h \geq 2$. Then, $T_{(d,h)}$ satisfies*

$$\mathrm{rank}(T_{(d,h)}) \geq \frac{(d-2)\log n}{\log(d-1)}.$$

Proof. Let $\mathrm{dep}(v)$ denote the depth of vertex v in $T_{(d,h)}$. For a vertex v with $\mathrm{dep}(v) = k$, let $E(v)$ denote set of edges (v, w) such that $\mathrm{dep}(w) = k+1$. Then, $|E(v)| = d-1$ if $\mathrm{dep}(v) = 1, 2, \ldots, h-1$, $|E(v)| = d$ if $\mathrm{dep}(v) = 0$, and $|E(v)| = 0$ if $\mathrm{dep}(v) = h$. Let us consider the minimum edge ranking r of $T_{(d,h)}$, which is

computed by the bottom-up procedure proposed in [9]. Since it is the bottom-up procedure, all vertices v in the same depth have the same set of ranks,

$$r(E(v)) = \{r(e) \,|\, e \in E(v)\}. \tag{19}$$

Furthermore, we can see that

$$\begin{aligned} r(E(v)) &= \{1, 2, \ldots, d-1\} && \text{if } \mathrm{dep}(v) = h-1 \\ r(E(v)) &= \{d+1, d+2, \ldots, 2d-2\} && \text{if } \mathrm{dep}(v) = h-2, \end{aligned}$$

and for every vertex v with $\mathrm{dep}(v) = k \leq h-3$, there always exist $|E(v)|-1$ edges $e_1, e_2, \ldots, e_{|E(v)|-1} \in E(v)$ such that

$$r(e_1), r(e_2), \ldots, r(e_{|E(v)|-1}) \;>\; \max\{r(e) \,|\, e \in E(w), \mathrm{dep}(w) = k+1\}$$

by the way of the bottom-up edge ranking and its optimality shown in [9]. Therefore, we have

$$\mathrm{rank}(T_{(d,h)}) \geq 2(d-1) + (h-3)(d-2) + d - 1 = (d-2)h + 3.$$

Since $n \equiv |T_{(d,h)}|$ is bounded from above by

$$n \;=\; \frac{d(d-1)^h - 2}{d-2} \;\leq\; \frac{d(d-1)^h}{d-2},$$

we have

$$h \geq \frac{\log n}{\log(d-1)} - \frac{\log d - \log(d-2)}{\log(d-1)} \geq \frac{\log n}{\log(d-1)} - \frac{2\log e}{(d-2)\log(d-1)},$$

where the second inequality follows from the inequality

$$\ln(a+b) - \ln a \geq b/(a+b)$$

for $a, b > 0$ with $a = d-2$ and $b = 2$. Thus,

$$\begin{aligned} \mathrm{rank}(T_{(d,h)}) &\geq (d-2)h + 3 \\ &\geq \frac{(d-2)\log n}{\log(d-1)} - \frac{2\log e}{\log(d-1)} + 3 \\ &\geq \frac{(d-2)\log n}{\log(d-1)} \qquad \text{(by substituting } d = 3) \end{aligned}$$

which completes the proof. □

This lemma implies that the upper bound in Corollary 4 is tight for the family of trees $T_{(d,h)}$.

4.3 Proof of Theorem 2

Finally, we arrives at the proof of Theorem 2, which gives the approximation ratio of algorithm Approx_MERST.

Proof of Theorem 2: By the algorithm of Fürer and Raghavachari [4], we can find a spanning tree T of G such that $\Delta^* \geq \Delta_T \leq \Delta^* + 1$ in the first step of Approx_MERST. Furthermore, since any tree T has

a $(\Delta_T - 2)\log n/(\log \Delta_T - 1)$-edge ranking by Corollary 4, we can obtain $(\Delta^* - 1)\log n/(\log(\Delta^* + 1) - 1)$-ranking of T by APPROX_MERST. On the other hand, every tree T satisfies $\mathrm{rank}(T) \geq \max\{\Delta_T, \lceil \log n \rceil\}$ by Lemma 3. Therefore, the approximation ratio of APPROX_MERST

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

can be obtained. □

5 Conclusion

This paper 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 [11].

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.

Acknowledgment

This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan under Grant No. 09750460. We deeply express our appreciation to Prof. Tiko Kameda at Simon Fraser University for his valuable comments.

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 $O(n\sqrt{\log n})$ time, *Technical Report TR-97-01*, University of Hong Kong, 1997.

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

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

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

[12] Perl, Y. and Zaks, S., On the complexity of edge labelings for trees, *Theoretical Computer Science*, 19, 1–16, 1982.

[13] Ullman, J. D., *Principles of Database and Knowledge-Base Systems, Vol. 1*, Computer Science Press, 1990.

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

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

Appendix: Proofs

Let ln and log denote the logarithms whose bases are e and 2, respectively. The following inequalities are well known. For $a, b > 0$,

$$\ln(a+b) - \ln a \leq b/a \tag{20}$$

$$\ln(a+b) - \ln a \geq b/(a+b). \tag{21}$$

Lemma 11 *Let* $f_6(x) \stackrel{\text{def}}{=} \frac{x-2}{\log x - 1}$. *Then* $f_6'(x) \geq 0$ *for all* $x \geq 3$, *where* $f_6'(x)$ *is the derivative of* $f_6(x)$.

Proof.

$$f_6'(x) = \frac{-(x-2)\log e/x + \log x - 1}{(\log x - 1)^2}.$$

Since its denominator is positive for $x \geq 3$, we consider its numerator $g_1(x) \stackrel{\text{def}}{=} -(x-2)\log e/x + \log x - 1$.

$$\begin{aligned} g_1(x) &= (\ln x - 1 + 2/x)\log e - 1 \\ &\geq ((x-e)/x + 2/x)\log e - 1 \quad \text{(apply (21) by } a = e, b = x - e) \\ &= (x - e + 2)\log e/x - 1. \end{aligned}$$

Since $g_2(x) \stackrel{\text{def}}{=} (x - e + 2)\log e/x - 1$ is monotone increasing when $x > 0$, we have

$$\begin{aligned} g_2(3) &\geq (5-e)\log e/3 - 1 \\ &\geq 1.44(5 - 2.8)/3 - 1 \ > \ 0. \quad \text{(by } e < 2.8 \text{ and } \log e > 1.44) \end{aligned}$$

□

Lemma 12 *Let* $f_3(\Delta_T) \stackrel{\text{def}}{=} (\Delta_T - 2)\log(\Delta_T + 1) - \Delta_T(\log \Delta_T - 1)$. *Then* $f_3(\Delta_T) \geq 0$ *holds for* $\Delta_T \geq 3$.

Proof.

$$\begin{aligned} f_3(\Delta_T) &= \Delta_T \log e(\ln(\Delta_T+1) - \ln \Delta_T) + \Delta_T - 2\log(\Delta_T+1) \\ &\geq \Delta_T \log e/(\Delta_T+1) + \Delta_T - 2\log(\Delta_T+1) && \text{(apply (21) by } a=\Delta_T, b=1) \\ &\geq 3\log e/4 + \Delta_T - 2\log(\Delta_T+1). && \text{(by } \Delta_T \geq 3) \end{aligned}$$

Let $g(\Delta_T) \stackrel{\text{def}}{=} 3\log e/4 + \Delta_T - 2\log(\Delta_T+1)$. Then, since

$$g'(\Delta_T) = 2 - 2\log e/\Delta_T \ > \ 0, \quad \text{for} \ \ \Delta_T \geq 3,$$

and $g(3) = 3\log e/4 - 1 \geq 0$ by $\log e > 1.44$, we have $f_3(\Delta_T) \geq g(\Delta_T) \geq 0$ for $\Delta_T \geq 3$. □

Lemma 13 *Let* $f_5(\Delta_T) \stackrel{\text{def}}{=} (\Delta_T - 2)(\log n - \log(n+\Delta_T-1)) + \Delta_T - \log \Delta_T - 1$. *If* $n \geq 3\Delta_T + 1$, *then* $f_5(\Delta_T) \geq 0$ *holds for* $\Delta_T \geq 3$.

Proof.

$$\begin{aligned} f_5(\Delta_T) &= (\Delta_T - 2)\log e(\ln n - \ln(n+\Delta_T-1)) + \Delta_T - \log \Delta_T - 1 \\ &\geq -(\Delta_T-2)(\Delta_T-1)\log e/n + \Delta_T - \log \Delta_T - 1 && \text{(apply (20) by } a=n, b=\Delta_T-1) \\ &\geq -(\Delta_T-2)(\Delta_T-1)\log e/(3\Delta_T+1) + \Delta_T - \log \Delta_T - 1 && \text{(by } n \geq 3\Delta_T+1). \end{aligned}$$

Then $f_5(3) \geq 2 - \log e/5 - \log 3 > 0$ holds by $\log e < 1.45$ and $\log 3 < 1.59$. For $\Delta_T \geq 4$, we consider $g(\Delta_T) \stackrel{\text{def}}{=} -(\Delta_T-2)\log e/3 + \Delta_T - \log \Delta_T - 1$. $f_5(\Delta_T) \geq g(\Delta_T)$ clearly holds when $\Delta_T \geq 4$. Moreover, since $g'(\Delta_T) = 1 - \log e/3 - \log e/\Delta_T > 0$ if $\Delta_T \geq 4$, and $g(4) = 1 - 2\log e/3 \geq 0$, we have $f_5(\Delta_T) \geq g(\Delta_T) \geq 0$ for $\Delta_T \geq 4$. Therefore, $f_5(\Delta_T) \geq 0$ holds for all $\Delta_T \geq 3$. □