DIMACS Technical Report 2003-27
August 2003

Minimum Edge Ranking Spanning Trees of Split Graphs

by

Kazuhisa Makino
Division of Mathematical Science for Social Systems,
Graduate School of Engineering Science, Osaka University,
Toyonaka, Osaka, 560-8531 Japan.
makino@sys.es.osaka-u.ac.jp

Yushi Uno
Department of Mathematics and Information Sciences,
College of Integrated Arts and Sciences,
Osaka Prefecture University, Sakai, 599-8531 Japan.
uno@mi.cias.osakafu-u.ac.jp

Toshihide Ibaraki
Department of Applied Mathematics and Physics,
Graduate School of Informatics,
Kyoto University, Kyoto, 606-8501 Japan.
ibaraki@i.kyoto-u.ac.jp

ABSTRACT

Given a graph G, the minimum edge ranking spanning tree problem (MERST) is to find a spanning tree of G whose edge ranking is minimum. However, this problem is known to be NP-hard for general graphs. In this paper, we show that the problem MERST has a polynomial time algorithm for split graphs, which have useful applications in practice. The result is also significant in the sense that this is a first non-trivial graph class for which the problem MERST is found to be polynomially solvable. We also show that the problem MERST for threshold graphs can be solved in linear time, where threshold graphs are known to be split.

1 Introduction

Let $G = (V, E)$ be an undirected graph that is simple and connected. An *edge ranking* of a graph G is a labeling $r \colon E \to \boldsymbol{Z}_+$, with the property that every path between two edges with the same label i contains an intermediate edge with label $j > i$. Here $\boldsymbol{Z}_+$ denotes the set of all non-negative integers. An edge ranking is *minimum* if it uses the least number of distinct labels among all edge rankings of G. The *minimum edge ranking spanning tree problem* (MERST) is to find a spanning tree whose edge ranking is minimum. The problem MERST has a number of practical applications in many fields such as parallel processing, integration of distributed data, and so on [14]. It is known [12, 14] that the problem MERST is in general NP-hard and linearly solvable if a given graph is a tree. However, the computational complexity of the problem MERST has been still unknown even when a given graph is restricted to be chordal, split, partial k-tree, bipartite or threshold, for example.

In this paper, we consider the problem MERST for split graphs. This is motivated by efficient parallel joins in a relational database (see Section 2 for the details). Split graphs have been extensively studied (e.g., [4, 5, 7]). For example, it is known that split graphs are perfect and the recognition can be done in linear time. We show that the problem MERST for split graphs can be solved in polynomial time. This is a first result that finds a non-trivial polynomially solvable class for the problem MERST. We also show that the problem MERST for threshold graphs can be solved in linear time. Here threshold graphs were introduced by Chvátal and Hammer [2], and have also been well-studied (e.g., [5, 6, 13]). It is known that threshold graphs are split (and hence perfect) and the recognition can be done in linear time.

The rest of the paper is organized as follows. In Section 2, we first give a formal definition of our problem and several notations required, and then overview properties of split and threshold graphs with emphasis on applications. Section 3 shows that any split graph G has a spanning caterpillar whose edge ranking is minimum among all spanning trees of G. We also characterize optimal split and threshold graphs $G = (V, E)$ with $|V| = 2^k$. Note that optimal graphs are those graphs that have a spanning tree whose minimum edge rank is k, since it is known that any connected graph needs at least $\lceil \log |V| \rceil$ edge ranks. (We assume that the base of the logarithm is 2 throughout the paper.) In Section 4, we reveal several properties of the edge ranking of caterpillars as spanning trees of a split graph. In Section 5, we present a polynomial time algorithm that solves the problem MERST for split graphs, and Section 6 presents a linear time algorithm that solves the problem MERST for threshold graphs.

2 Preliminaries

2.1 Definitions and Notations

Throughout the subsequent sections, we assume that a graph $G = (V, E)$ is simple, undirected and connected. As usual, $V(G)$ and $E(G)$ are used to denote the vertex and edge set of G, respectively. For an edge set $F \subseteq E$, we denote by $V(F)$ the set of all the end vertices of the edges in F. Similarly, for an vertex set $U \subseteq V$, we denote by $E(U)$ the set of all the edges that are incident to the vertices in U. Moreover, for two vertex sets U and W, we use notation $E(U, W) = \{(u, w) \in E \mid u \in U, w \in W\}$. For simplicity, we sometimes write $V(e)$, $E(u)$ and $E(u, W)$ for $V(\{e\})$, $E(\{u\})$ and $E(\{u\}, W)$, for example. Let $G[U]$ (resp., $G[F]$) denote the subgraph induced by a vertex set $U \subseteq V$ (resp., an edge set $F \subseteq E$). If the resulting subgraph induced by U (resp., F) is a tree, we sometimes denote it by $T[U]$ (resp., $T[F]$). For a vertex $u \in V$ and an edge set $F \subseteq E$, the *degree* of u in $G[F]$ is the number of edges in F that are incident to u, and is denoted by $\deg_F(u)$, i.e., $\deg_F(u) = |\{e \mid e = (u, v) \in F\}|$.

A mapping (labeling) $r : E \to \boldsymbol{Z}_+$ is called an *edge ranking* if r satisfies that every path between two edges with the same label i contains an intermediate edge with label $j > i$. An integer given to an edge by an edge ranking is called a *rank* of the edge. 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, and such k is called the *minimum edge rank* of G and is denoted by $\mathrm{rank}(G)$. For an edge ranking r of G, we call the multiple set of ranks $\{r(e) \mid e \in E\}$ the *rank set* of r, and denote it by $RS(r)$. Note that $|RS(r)| = |E(G)|$ holds. The *minimum edge ranking problem* (MER) asks to compute a minimum edge ranking r of a given graph G.

As an example, let us consider a graph $G = (V, E)$ in Fig. 1 (a), where $E = \{e_1, e_2, \ldots, e_8\}$. Fig. 1 (b) provides a 4-edge ranking r_1 of G, where $r_1(e_2) = r_1(e_5) = r_1(e_7) = 1$, $r_1(e_1) = r_1(e_6) = 2$, $r_1(e_3) = r_1(e_6) = 3$ and $r_1(e_4) = 4$. Note that r_1 is in fact a minimum edge ranking of G, i.e., $\mathrm{rank}(G) = 4$. For this r_1, we have $RS(r_1) = \{1, 1, 1, 2, 2, 3, 3, 4\}$.

The problem MER for example has applications in the context of assembling a multipart product from its components in the smallest number of parallel integration stages, and has been extensively studied (e.g., [1, 3, 9, 11, 12]). De la Torre, Greenlaw and Schäffer [3] first proposed a polynomial time algorithm for the problem MER if G is a tree, and Lam and Yue [12] finally showed that it can be solved in linear time. On the other hand, Lam and Yue [11] also showed that MER is NP-hard for general graphs.

A spanning tree T of a graph G is called a *minimum edge ranking spanning tree* if it has the minimum $\mathrm{rank}(T)$ among all spanning trees T of G. The minimum edge ranking spanning tree problem (MERST) can be described as follows [14]:

MERST (The minimum edge ranking spanning tree problem)	
Input:	A graph $G = (V, E)$.
Output:	A minimum edge ranking spanning tree T together with a minimum edge ranking of T.

For example, Fig. 1 (c) shows a minimum edge ranking spanning tree $T = (V, E_T)$ of G in Fig. 1 (a), together with a minimum edge ranking r_2 of T.

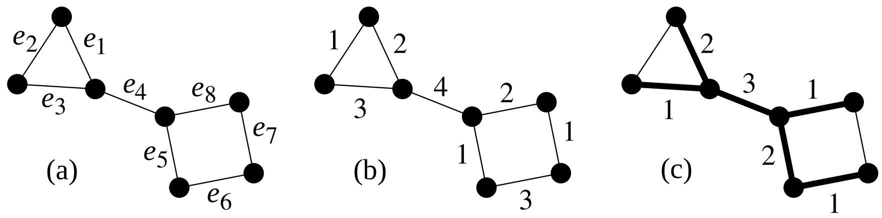

Figure 1: A graph and edge rankings. (a) a graph $G = (V, E)$, (b) a minimum edge ranking r_1 of G, and (c) a minimum edge ranking spanning tree $T = (V, E_T)$ of G and a minimum edge ranking r_2 of T.

Note that the problem MERST coincides with the problem MER when G is a tree. ¿From a graph theoretical viewpoint, MERST is the problem of minimizing the number of edge contraction steps until G becomes a single vertex (which may have self-loops), under the condition that the edges contracted simultaneously do not share any of their end vertices (i.e., such edges always form a matching) [14], while MER is to minimize the number of edge contraction steps until G becomes a single vertex with no edge, under the same condition but regarding the edge contraction of a self-loop e as the removal of e.

MERST is known to have quite natural practical interpretations, such as join operations in relational databases and scheduling the parallel assembly of a multi-part product [14, 16]. In Section 2.2, we briefly discuss one of the applications of MERST for threshold graphs.

Unfortunately, MERST is known to be NP-hard for general graphs. However, it has a simple approximation algorithm 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 [14]. In this paper we present a polynomial time algorithm for MERST, when a given graph is split.

2.2 Split Graphs, Threshold Graphs and their Applications

A graph $G = (V, E)$ is *split* if the vertex set V can be partitioned into a clique K and a stable set S. There is no restriction on edges between vertices of K and vertices of S. A graph $G = (V, E)$ is *threshold* if there exist a weight $a : V \to \boldsymbol{Z}_+$ and a threshold $t \in \boldsymbol{Z}_+$ such that for any subset S of V,

$$S \text{ is stable} \iff \sum_{u \in S} a(u) \le t$$

Fig. 2 (a) gives a threshold graph with weights to vertices and a threshold $t = 10$.

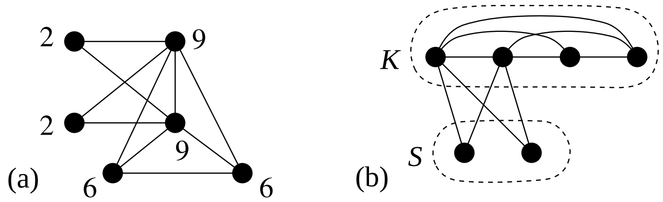

Figure 2: (a) A threshold graph with vertex weights and a threshold $t = 10$, and (b) a vertex partition into a stable set S and a clique K.

It is known [5, 6, 13] that threshold graphs can be characterized as follows.

Lemma 1 *For a graph $G = (V, E)$, the following three statements are equivalent.*
(i) *G is threshold.*
(ii) *There exists a weight $a : V \to \boldsymbol{Z}_+$ and a threshold t such that for any $u, v \in V$ with $u \neq v$,*

$$(u, v) \in E \iff a(u) + a(v) > t.$$

(iii) *The vertex set V can be partitioned into a clique $K = \{u_1, \ldots, u_{|K|}\}$ and a stable set $S = \{v_1, \ldots, v_{|S|}\}$ such that*

$$(u_i, v_j) \in E \implies (u_{i'}, v_{j'}) \in E \quad (i' \leq i,\ j' \leq j). \tag{1}$$

∎

Fig. 2 (b) shows the vertex partition of the graph G in Fig. 2 (a). It follows from (iii) in Lemma 1 that threshold graphs are split.

As an example of MERST for threshold graphs, let us consider to compute the join of n relations $R_1, \ldots, R_n$ in a relational database [8, 16]. The situation is depicted by a join graph with n vertices corresponding to the n relations, in which each edge represents the join operation of the two relations of its end vertices. To obtain the join of n relations, we select a spanning tree T and perform $n - 1$ join operations specified by the edges of T. If these operations are done in parallel, their order can be exactly described by an edge ranking r of T and the number of stages of parallel joins is equal to the rank of r.

In the above process, we may also consider the sizes of intermediate relations formed by join operations. Let relation R_i have N_i tuples, and let m_i be the number of columns used in the join of n relations. Then it can be argued that the number of tuples in the relation resulting from the join of R_i and R_j is approximately proportional to $(N_i/m_i)(N_j/m_j)$. Therefore, it may be reasonable to consider only those edges having small expected numbers of tuples. For this, let $P_i = m_i/N_i$ for all i. Then $(N_i/m_i)(N_j/m_j) < K$ is equivalent to $P_iP_j > 1/K$, i.e., $\log P_i + \log P_j > \log(1/K)$. Therefore, giving weights $a_i = \log P_i$ to vertices R_i, it amounts to consider only those edges satisfying

$$a_i + a_j > t,$$

where $t = \log 1/K$ is the threshold.[1]

In this way, we obtain a threshold graph by Lemma 1 (ii), and for our purpose, the remaining task is to solve MERST for the graph.

For example, assume that N_i and m_i for each relation R_i are given in a pair of (N_i, m_i) as shown in Fig. 3 (b). Then computing the label $a_i = \log P_i + A$ $(A = 12)$, and setting the threshold $t = 5$ $(= -7 + A)$ lead a threshold graph in Fig. 3 (c), where A is introduced to make a_i positive integers.

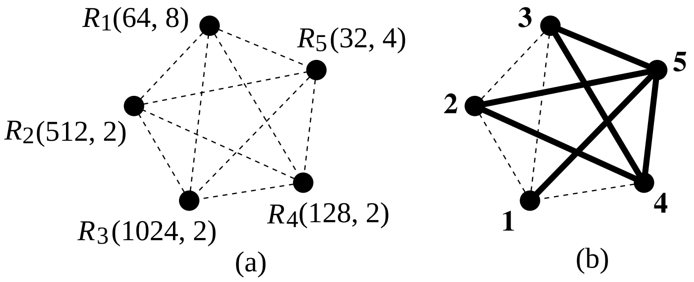

Figure 3: (a) Five relations with their size and column information, and (b) a graph representing edges that should be processed earlier.

3 MERST for Split Graphs

This section discusses several properties of the edge ranking of spanning tress in split graphs (and hence threshold graphs). Especially, we show that any split graph G has a spanning caterpillar whose edge ranking is minimum among all spanning trees of G. We also characterize split graphs and threshold graphs $G = (V, E)$ with $|V| = 2^k$ having a spanning tree of minimum edge rank k.

3.1 Split Graphs and Their Spanning Caterpillars

A tree $T = (V, E)$ is called a *caterpillar* if V can be partitioned into B and L (i.e., $B \cup L = V$ and $B \cap L = \emptyset$) such that $T[B]$ is a path and L is stable. A vertex in B (resp., L) is called a *body vertex* (resp., a *leg vertex*) of a caterpillar T. By definition, each leg vertex u is adjacent to exactly one body vertex v, where (u, v) is called a *leg* of T. Note that the partition (B, L) of V may not be unique (see Fig 4 (a) and (b)), and hence the names body and/or leg can be used when the partition is specified. We sometimes write a caterpillar $T = (V, E)$ with a vertex partition (B, L) as $T = (B \cup L, E)$.

We now show two properties of the edge ranking of a spanning tree in a split graph.

[1] If we are concerned with the space to store the relation instead of the number of tuples in it, we can use $P_i = m_i/N_iM_i$, since the space to store the relation resulting from the join of R_i and R_j is approximately proportional to $(N_iM_i/m_i)(N_jM_j/m_j)$. Here M_i denotes the number of columns in R_i.

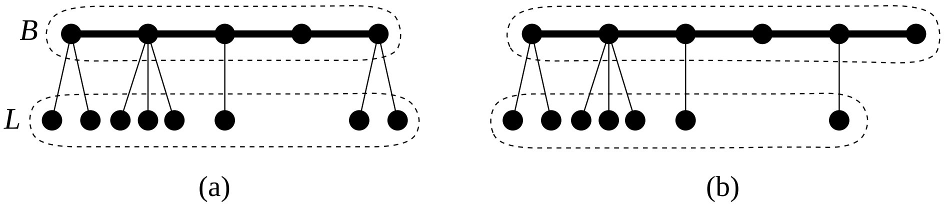

Figure 4: (a) A caterpillar T and a vertex partition (B, L) of V, (b) another partition for the same caterpillar.

Lemma 2 *Let $G = (V, E)$ be a split graph with a vertex partition (K, S), where K and S are clique and stable set, respectively. Let T be a spanning tree of G, and let r be an edge ranking of T. Then G has a spanning tree T^* with an edge ranking r^* such that $T^*[K]$ is connected and $RS(r^*) = RS(r)$ holds.*

Proof. Let $T = (V, E_T)$, and for $u \in V$ let

$$E_u = \{(u, v) \in E_T\}.$$

Assume that $T[K]$ is not connected. Then there exists a vertex $u \in S\,(= V - K)$ such that $|E_u| \geq 2$. For such a vertex u, let (u, v) be an edge in E_u with the largest rank $r((u, v))$, and w be a vertex such that $w \neq v$ and $(u, w) \in E_u$. Note that $v, w \in K$ holds since $T[S]$ is stable, and $r(e) < r((u, v))$ for all $e \in E_u - \{(u, v)\}$ since r is an edge ranking of T. We construct from T a new spanning tree $T' = (V, E_{T'})$ of G and an edge labeling r' of T' from T by exchanging (u, v) with (v, w) (see Fig. 5 (a) and (b)), i.e.,

$$E_{T'} = (E_T - \{(u, v)\}) \cup \{(v, w)\}$$
$$r'(e) = \begin{cases} r((u, v)), & e = (v, w) \\ r(e), & \text{otherwise.} \end{cases}$$

Now no edge $e' \in T[K]$ incident to w satisfies $r(e') = r((u, v))$, since otherwise edges e' and (u, v) do not satisfy the condition of edge ranking r. Also let P be a path in T containing (u, v). Then the corresponding path P' in T' must contain (v, w). Since $r((u, v))$ is the largest among $e \in E_u$, this implies that r' is an edge ranking of T' satisfying $RS(r') = RS(r)$.

By applying this modification to T' repeatedly, we shall eventually have a desired spanning tree T^* and an edge ranking r^*. ∎

Lemma 3 *Let $G = (V, E)$ be a split graph with a vertex partition (K, S), where K and S are clique and stable set, respectively. Let T be a spanning tree of G such that $T[K]$ is connected, and let r be an edge ranking of T. Then G has a spanning tree T^* with an edge ranking r^* such that $T^*[K]$ is a path and $RS(r^*) = RS(r)$ holds.*

Proof. For a tree $T = (V, E_T)$, let

$$\delta_K(T) = \sum_{u \in K} \max\{\deg_{E_T[K]}(u) - 2, 0\}.$$

Assume that $T[K]$ is not a path. Then $\delta_K(T) \geq 1$ holds since $T[K]$ is connected. This means that there exists a $W \subseteq K$ and a $x \in W$ such that $T[W]$ is a path, $\deg_{E_{T[K]}}(x) > 2$ and $E_T(W, K - W) = E_T(x, K - W)$. Let y and z be the vertices in W that are adjacent to x in T, and assume $r((x, y)) > r((x, z))$ without loss of generality. Similarly to the proof of Lemma 3, we construct a new spanning tree $T' = (V, E_{T'})$ of G and an edge labeling r' of T' from T by exchanging (x, y) with (y, z). It is not difficult to see that this r' is also an edge ranking of T' satisfying $RS(r') = RS(r)$. Furthermore, we see that either $\delta_K(T') < \delta_K(T)$ or $|W'| < |W|$ holds, where $W' = W - \{x\}$ plays the role of the above W for the new tree T'. In the second case, we repeatedly apply this modification to T', but it cannot continue indefinitely since W is a finite set. Therefore we will have a spanning tree T'' with an edge ranking r'' such that $\delta_K(T'') < \delta_K(T)$ and $RS(r) = RS(r'')$. Now, by applying this procedure until $\delta_K(T)$ becomes 0, we will obtain a desired spanning tree T^* and an edge ranking r^* of T^* (See Fig. 5 (b) and (c)). ∎

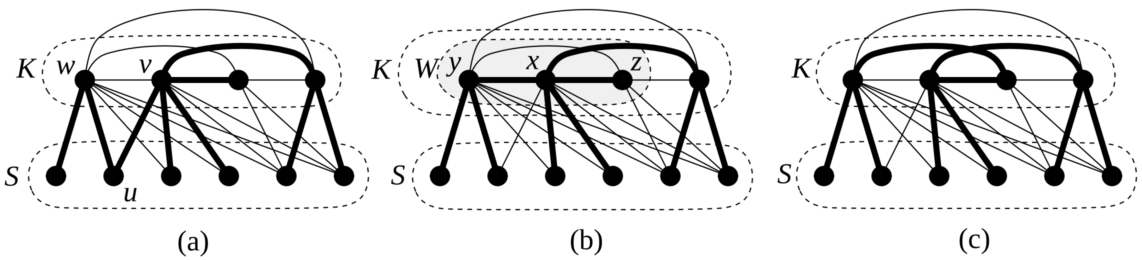

Figure 5: Illustrations for proofs of Lemmas 2 and 3. (a) A split (threshold) graph and a spanning tree (bold edges) that is not a caterpillar $T = (K \cup S, E_T)$, (b) a spanning tree where $T[K]$ is connected, and (c) a spanning caterpillar.

Lemmas 2 and 3 immediately imply the following corollary.

Corollary 4 *Let $G = (V, E)$ be a split graph with a vertex partition (K, S), where K and S are clique and stable set, respectively. Let T be a spanning tree of G, and let r be an edge ranking of T. Then G has a spanning caterpillar $T^* = (B \cup L, E_{T^*})$ with an edge ranking r^* such that $B = K$, $L = S$ and $RS(r^*) = RS(r)$.* ∎

3.2 Optimal Split and Threshold Graphs

It is known that $\text{rank}(G) \geq \lceil \log |V| \rceil$ holds for any connected graph $G = (V, E)$. This section characterizes optimal split and threshold graphs G satisfying $\text{rank}(G) = \log |V|$ when $|V| = 2^k$.

Let us introduce some additional notions. For a graph $G = (V, E)$ and a positive integer k, a *size-k partition* [14] of V is a $(|V|/k)$-tuple $(V_1, V_2, \ldots, V_{|V|/k})$ such that $V = \bigcup_{i=1}^{|V|/k} V_i$, $V_i \cap V_j = \emptyset$ for $i \neq j$, and $|V_i| = k$ for $i = 1, 2, \ldots, |V|/k$. 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.

The following lemma characterizes optimal edge rankings by nested partitions.

Lemma 5 ([14]) *Let $G = (V, E)$ be a graph with $|V| = 2^k$ $(k \geq 0)$. Then G is optimal (i.e., G has a k-edge ranking spanning tree) if and only if it has a nested partition.* ∎

We are now ready to characterize optimal graphs.

Theorem 6 *Let $G = (V, E)$ be a split graph with $|V| = 2^k$ $(k \geq 0)$ and a vertex partition (K, S), where K and S are clique and stable set, respectively. Then there exists a spanning tree T of G with* $\mathrm{rank}(T) = k$ *if and only if the bipartite graph $G(K, S) = (K, S, E(K, S))$ has a matching of size $|S|$, where $E(K, S) = \{(u, v) \in E \mid u \in K, v \in S\}$.*

Proof. We first show the only-if part. It follows from Corollary 4 that G has a k-edge ranking spanning caterpillar $T = (K \cup S, E_T)$. If some $u \in K$ satisfies $|\{(u, v) \in E_T \mid v \in S\}| \geq 2$, then T has no nested partition. This contradicts Lemma 5, and hence each $u \in K$ satisfies $|\{(u, v) \in E_T \mid v \in S\}| < 2$, implying $E(K, S) \cap E_T$ forms a matching of size $|S|$. This completes the only-if part.

For the if part, let $E' = \{(u_i, v_i) \mid u_i \in K, v_i \in S, i = 1, 2, \ldots, |S|\}$ be a matching of $G(K, S)$. Then G has a spanning caterpillar $T = (K \cup S, E_T)$ such that

$$E_T = \{(u_i, u_{i+1}) \mid i = 1, 2, \ldots, |K| - 1\} \cup E'$$

Here u_i, $i = |S| + 1, \ldots, |K|$ are vertices in $K \setminus \{u_1, u_2, \ldots, u_{|S|}\}$ that can be arbitrarily ordered. We can easily see that this T has a nested partition, and thus by Lemma 5, we have $\mathrm{rank}(T) = k$. ∎

Theorem 7 *Let $G = (V, E)$ be a threshold graph with $|V| = 2^k$ $(k \geq 0)$ and a vertex partition $(K = \{u_1, \ldots, u_{|K|}\}, S = \{v_1, \ldots, v_{|S|}\})$ satisfying Lemma 1* (iii)*. Then there exists a spanning tree T of G with* $\mathrm{rank}(T) = k$ *if and only if $|K| \geq |S|$ and $(u_{1+j}, v_{|S|-j}) \in E$ for $j = 0, 1, \ldots, |S| - 1$.*

Proof. It follows from Theorem 6 that G has a spanning tree T with $\mathrm{rank}(T) = k$ if and only if the bipartite graph $G(K, S)$ has a matching of size $|S|$. Since G is threshold, the latter condition is equivalent to that $|K| \geq |S|$ and $(u_{1+j}, v_{|S|-j}) \in E$ for $j = 0, 1, \ldots, |S| - 1$ (Fig. 6). ∎

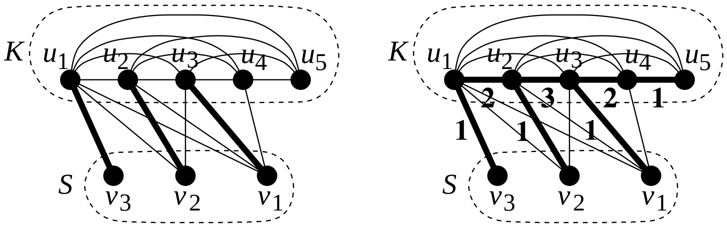

Figure 6: (a) A threshold graph G with $2^3 = 8$ vertices and an edge set E', and (b) its minimum edge ranking spanning tree T, where $\mathrm{rank}(T) = 3$.

4 Minimum Edge Rankings of Caterpillars

By Theorem 4, we only consider caterpillars $T = (K \cup S, E_T)$ as minimum edge ranking spanning trees of split graphs. This section investigates some properties of minimum edge rankings of caterpillars. In particular, we show that it is sufficient to consider only normalized rankings and that a balanced caterpillar $T^* = (K \cup S, E_{T^*})$ has the minimum rank among all caterpillars $T = (K \cup S, E_T)$ (that is, caterpillars having the same vertex partition).

Let $T = (B \cup L, E)$ be a caterpillar. We assume that the vertices in B are ordered in such a way that $(u_i, u_{i+1}) \in E$ holds for $i = 1, \ldots, |B| - 1$. Given a k-edge ranking r of a caterpillar $T = (V, E)$, let $T_p = (V, E_p)$ $(p = 0, 1, \ldots, k)$ denote the spanning subgraph of T consisting of all edges $e \in E$ with $r(e) \leq p$. By definition, T_0 has no edge and $T_k = T$. Furthermore, let $T_{pq} = (V_{pq}, E_{pq})$ $(q = 1, \ldots, c_p)$ be the connected components of T_p, where c_p is the number of connected components in T_p. An edge ranking r of a caterpillar $T = (B \cup L, E)$ is *BL-connected* if

$$|V_{pq} \cap B| \geq 2 \implies E(V_{pq} \cap B, L) \subseteq E_{pq}. \tag{2}$$

In other words, if a connected component T_{pq} contains at least two body vertices, then it contains all the legs that are incident to some body vertex in V_{pq}. One can easily see that an edge ranking r is BL-connected if and only if

$$r((u_i, v)) < r((u_{i-1}, u_i)) \;\text{ and }\; r((u_i, v)) < r((u_i, u_{i+1})) \tag{3}$$

hold for any leg (u_i, v) with $u_i \in B$ and $v \in L$.

Lemma 8 *For any edge ranking r of a caterpillar T, there exists a BL-connected edge ranking r^* of T such that $RS(r^*) = RS(r)$.*

Proof. If r is not BL-connected, then it follows from the discussion above that there exists a leg $(u_i, v) \in E$ such that $u_i \in B$, $v \in L$ and either $r((u_i, v)) > r((u_{i-1}, u_i))$ or $r((u_i, v)) > r((u_i, u_{i+1}))$ holds. We construct a new edge labeling r' from r by exchanging $r((u_i, v))$ with $r((u_{i-1}, u_i))$ if $r((u_i, v)) > r((u_{i-1}, u_i))$; otherwise, with $r((u_i, u_{i+1}))$. Since v

is a leaf, it is easy to show that this r' is also an edge ranking. By repeating this argument, we can construct a desired BL-connected edge ranking r^*. ∎

Note that, for each $u_i \in B$, legs in $E(u_i, L)$ cannot have the same rank. Therefore, any BL-connected edge ranking can be transformed into a "better" BL-connected edge ranking in the sense that the legs in $E(u_i, L)$ $(i = 1, \ldots, |K|)$ receive their ranks from 1 to $|E(u_i, L)|$. We call such a ranking *normalized*. Fig. 7 shows a normalized edge ranking of a caterpillar.

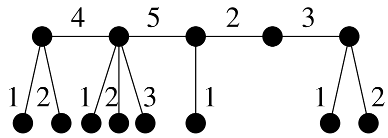

Figure 7: An normalized (and minimum) edge ranking of a caterpillar originally given in Fig. 4 (a).

In order to compare different edge rankings, we define a *lexicographic order* $\preceq$ on rank sets as follows: for any two rank sets $RS(r)$ and $RS(r')$,

$$RS(r) \preceq RS(r') \iff \Big|\{e \in E \mid r(e) \geq p\}\Big| \leq \Big|\{e \in E \mid r(e) \geq p\}\Big| \quad \text{for all } p = 1, 2, \ldots.$$

By combining the discussion above with Lemma 8, we have the following lemma.

Lemma 9 *For any edge ranking r of a caterpillar T, there exists a normalized edge ranking r^* of T such that $RS(r^*) \preceq RS(r)$.* ∎

This lemma shows that, for a caterpillar T, we can always find a minimum edge ranking among normalized ones. Therefore, we can restrict our attention to normalized rankings.

We next consider all the edge rankings of a caterpillar $T = (B \cup L, E)$. The *degree sequence* $\alpha = (\alpha_1, \ldots, \alpha_{|B|})$ of a caterpillar T is the sequence of the numbers of legs incident to $u_i \in B$ (i.e., $\alpha_i = |E(u_i, L)|$) for $i = 1, 2, \ldots, |B|$. Note that the degree sequence α uniquely defines a caterpillar, denoted by T^α. As noticed before, partition (B, L) of V for T is not always unique, and hence a caterpillar can be represented by different degree sequences. For example, for the caterpillar T shown in Fig. 4, there are partitions (B, L) depicted in (a) and (b), which have degree sequences $\alpha = (2, 3, 1, 0, 2)$ and $\beta = (2, 3, 1, 0, 1, 0)$, respectively.

Let α and β be respectively degree sequences of caterpillars T^α and T^β with the same vertex partition (B, L). Then T^β is called the *left-shifted* caterpillar of T^α if β is obtained from α by sorting α in the non-increasing order. For example, Fig. 8 shows the left-shifted caterpillar T^β of T^α.

The next lemma shows that the minimum edge rank of the left-shifted caterpillar cannot be worse than that of of the original caterpillar.

Lemma 10 *Let r be a normalized edge ranking of a caterpillar T. Then there exists an edge ranking r^* of the left-shifted caterpillar of T such that $RS(r^*) = RS(r)$.*

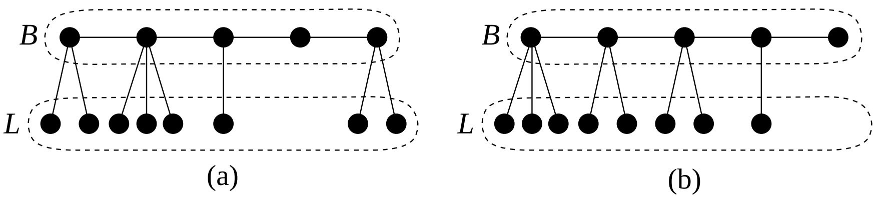

Figure 8: (a) A caterpillar T^α, and (b) the left-shifted caterpillar T^β of T^α. Note that the vertex partition (B, L) is common.

Proof. Let $\alpha = (\alpha_1, \ldots, \alpha_{|B|})$ and $\beta = (\beta_1, \ldots, \beta_{|B|})$ be degree sequences of caterpillars T and the left-shifted caterpillar of T, respectively. Let us assume that $\alpha_i = \beta_i$ for $i = 1, \ldots, k$, $\alpha_{k+1} \neq \beta_{k+1}$ ($i = 0$ if $\alpha_1 \neq \beta_1$), and $\alpha_\ell = \beta_{k+1}$ for $\ell > k+1$. Let $p = \beta_{k+1} (= \alpha_\ell)$ and consider the spanning subgraph $T_p = (V, E_p)$ of T. Since r is normalized, E_p contains all the edges (u_i, v) with $i \geq k+1$ for all $v \in L$, and $(u_k, u_{k+1}), (u_{\ell-1}, u_\ell), (u_\ell, u_{\ell+1}) \notin E_p$. Let T_{p1} and T_{p2} be connected components of T_p containing u_{k+1} and u_ℓ, respectively. Let T^γ be a caterpillar obtained from T by exchanging T_{p1} with T_{p2} (see Fig. 9), with its degree sequence γ. Then it is easy to see that r is also an edge ranking of T^γ with $\gamma_i = \beta_i$ for $i = 1, \ldots, k+1$. Therefore, by applying this argument at most $|B|$ times, we can construct the left-shifted caterpillar of T having an edge ranking r^* with $RS(r^*) = RS(r)$. ∎

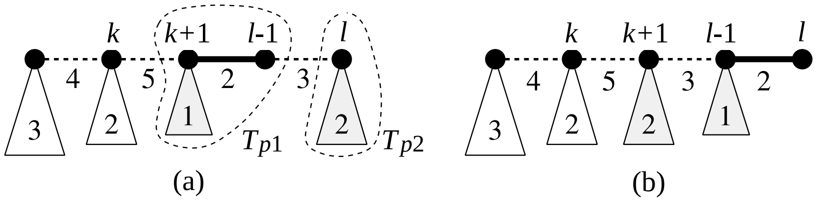

Figure 9: (a) Spanning subgraph T_2 of a caterpillar T with degree sequence $\alpha = \{3, 2, 1, 0, 2\}$, and (b) the left-shifted caterpillar of T and its edge ranking.

We then compare the minimum edge rankings of the left-shifted caterpillars in order to find the best one among those. Similarly to the lexicographic order $\preceq$ on rank sets, we define a lexicographic order $\preceq$ on non-increasing degree sequences $\alpha = (\alpha_1, \ldots, \alpha_{|B|})$ as follows: for $\alpha = (\alpha_1, \ldots, \alpha_{|B|})$ and $\beta = (\beta_1, \ldots, \beta_{|B|})$,

$$\alpha \preceq \beta \iff \sum_{i=1}^{k} \alpha_i \leq \sum_{i=1}^{k} \beta_i \quad \text{for all } k = 1, \ldots, |B|.$$

Lemma 11 *Let α and β respectively be two non-increasing degree sequences of caterpillars T^α and T^β with a common vertex partition (B, L), and let r be a normalized edge ranking of T^β. If $\alpha \preceq \beta$, then there exists an edge ranking r' of T^α such that $RS(r') \preceq RS(r)$.* ∎

Proof. Let us assume that $\sum_{i=1}^{k}\alpha_i = \sum_{i=1}^{k}\beta_i$ for $k \le \ell - 1$, and $\sum_{i=1}^{\ell}\alpha_i < \sum_{i=1}^{\ell}\beta_i$. Let h be the largest integer such that $\beta_h = \beta_\ell$, and let j be the smallest integer such that $\beta_j \le \beta_\ell - 2$. Since T^α and T^β have a common vertex partition (B, L), such j is well defined. By definition, we have $\beta_\ell = \cdots = \beta_h$ and $\beta_{h+1} = \cdots = \beta_{j-1}\,(= \beta_\ell - 1)$. We construct a non-increasing degree sequence γ by

$$\gamma_i = \begin{cases} \beta_i - 1 & \text{if } i = h, \\ \beta_i + 1 & \text{if } i = j, \\ \beta_i & \text{otherwise.} \end{cases}$$

We then have $\alpha \preceq \gamma \prec \beta$. Therefore, it is enough to show that T^γ has an edge ranking r^* such that $RS(r^*) \preceq RS(r)$.

Let us define r^* by

$$r^*((u_i, u_{i+1})) = \begin{cases} \beta_h & \text{if } i = h, \\ r((u_{i-1}, u_i)) & \text{if } i = h+1, \ldots, j-1, \\ \max\{r((u_{i-1}, u_i)), r((u_i, u_{i+1}))\} & \text{if } i = j, \\ r((u_i, u_{i+1})) & \text{otherwise.} \end{cases}$$

Fig. 10 (a) illustrates a part of T^β and its edge ranking r, and (b) illustrates the corresponding T^γ and its edge labeling r^*. Since r is normalized, r^* is also normalized and is an edge ranking of T^γ. Moreover, since $RS(r^*) \preceq RS(r)$, the proof is completed. ∎

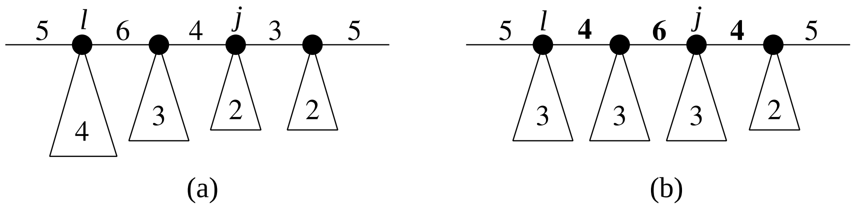

Figure 10: Replacement of edge ranks in case of degree sequences are comparable.

Let $\alpha^* = (\alpha_1^*, \ldots, \alpha_{|B|}^*)$ be the lexicographically minimum non-increasing degree sequence for a partition (B, L), that is,

$$\alpha_i^* = \begin{cases} \lfloor |L|/|B| \rfloor + 1, & i = 1, \ldots, p \\ \lfloor |L|/|B| \rfloor, & i = p+1, \ldots, |B|, \end{cases}$$

where $p = |L| \bmod |B|$ $(0 \le p < |B|)$.

From Lemmas 9, 10 and 11, we have the following result.

Theorem 12 *Let $\mathcal{T}$ be the set of all caterpillars $T = (V, E)$ with a vertex partition (B, L), and let α^* be defined above. Then for any edge ranking r of T, there exists an edge ranking r^* of T^{α^*} such that $RS(r^*) \preceq RS(r)$. In particular,* $\mathrm{rank}(T^{\alpha^*})$ *is the minimum among all* $\mathrm{rank}(T)$, $T \in \mathcal{T}$. ∎

5 A Polynomial Time Algorithm to Solve MERST for Split Graphs

Let $G = (V, E)$ be a split graph with a vertex partition (K, S), where K and S are clique and stable set, respectively. It follows from Corollary 4, Lemmas 9 and 10 that MERST for a split graph G can be solved by considering the minimum normalized edge ranking of the spanning left-shifted caterpillar $T = (K \cup S, E_T)$ of G. We note here that a spanning caterpillar $T = (K \cup S, E_T)$ of a split graph G can always be transformed into the left-shifted caterpillar, since K is a clique and hence an arbitrary ordering of vertices in K always forms a path. Moreover, if G contains a spanning left-shifted caterpillar $T^\alpha = (K \cup S, E_{T^\alpha})$ with the minimum α (with respect to $\preceq$ defined for degree sequences), this T^α is a minimum edge ranking spanning tree by Lemma 11. We show that this is always the case, and the minimum α can be computed in polynomial time.

The problem of finding the minimum α has already been studied under the name of *uniform load balancing problem* [10]. Let $G = (V_1, V_2, E)$ be a bipartite graph, where V_1 and V_2 are regarded as the set of machines and jobs, respectively. An edge set $F \subseteq E$ is called an *assignment* of V_2 to V_1 if each vertex v in V_2 has exactly one edge in F that is incident to v. Kleinberg, Rabani, and Tardos considered the problem of computing a fair assignment of V_2 (jobs) to V_1 (machines), i.e., an assignment F with the minimum degree sequence of V_1 after sorting from larger to smaller. They showed the following result.

Lemma 13 ([10, Theorem 2.1]) *Given a bipartite graph $G = (V_1, V_2, E)$, a fair assignment of V_2 to V_1 always exists and can be computed in polynomial time.* ∎

For the completeness of our paper, we give a sketch of their algorithm.

Let F denote a partial assignment of V_2 to V_1 (i.e., some vertices are allowed not to be assigned). We say that F is a partial assignment of maximum degree d if the maximum number of vertices assigned to a vertex in V_1 is d, and F is a maximum assignment of degree at most d if the number of unassigned vertices is the least among all partial assignments of degree at most d. Given any partial assignment F of degree at most d, we can use augmenting paths to find a maximum assignment of degree at most d. This process is referred as $\textsc{Augment}(F, d)$. The algorithm starts with $F_0 = \emptyset$, and computes $F_d = \textsc{Augment}(F, d-1)$ iteratively for $d = 1, 2, \ldots$ until all vertices in V_2 are assigned. The assignment at termination is then returned. Although we skip the details, the procedure above computes a fair assignment in polynomial time. Therefore, we have the following result.

Theorem 14 *A minimum edge ranking spanning tree of a split graph G can be computed in polynomial time.* ∎

6 A Linear Time Algorithm to Solve MERST for Threshold Graphs

In this section, we present a linear time algorithm that solves MERST for threshold graphs [15].

6.1 Spanning Caterpillars with the Minimum Degree Sequence

Let $G = (V, E)$ be a threshold graph with a vertex partition (K, S) satisfying Lemma 1 (iii). As mentioned in Section 5, we try to find a spanning left-shifted caterpillar $T^\alpha = (K \cup S, E_{T^\alpha})$ with the minimum α. Note that $K = \{u_1, \ldots, u_{|K|}\}$ and $S = \{v_1, \ldots, v_{|S|}\}$ satisfy condition (1) of Lemma 1, and hence we can restrict spanning caterpillars $T = (K \cup S, E_T)$ to those T that satisfy $E_T(K) = \{(u_i, u_{i+1}) \mid i = 1, \ldots, |K| - 1\}$. In other words, the ordering of the body vertices is the same as in Lemma 1 (iii). This is shown in Fig. 11.

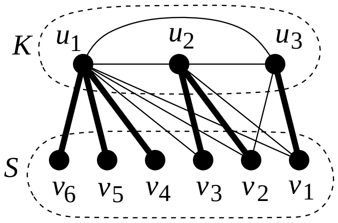

Figure 11: A threshold graph with a vertex partition ($K = \{u_1, u_2, u_3\}$, $S = \{v_1, \ldots, v_6\}$). Bold edges express the minimum degree sequence $\alpha = (3, 2, 1)$ that can be realized in this graph.

Let $d_i = |E(u_i, S)|$ for $i = 1, \ldots, |K|$. From condition (1), we have $d_1 \geq d_2 \geq \ldots \geq d_{|K|}$. The following procedure computes the minimum degree sequence of (K, S) that can be realized by a spanning caterpillar in G.

Procedure Minimum_DS
Input: $d = (d_1, \ldots, d_{|K|})$.
Output: $\alpha = (\alpha_1, \ldots, \alpha_{|K|})$.
begin
  for $i = |K|$ **downto** 1 **do**

$$\alpha_i := \min\left\{\left\lfloor (d_{i-\ell+1} - \textstyle\sum_{j=i+1}^{|K|} \alpha_j)/\ell \right\rfloor \mid \ell = 1, \ldots, i\right\} \tag{4}$$

end.

Lemma 15 *Procedure* Minimum_DS *correctly computes the minimum degree sequence that can be realized by a spanning caterpillar in a threshold graph $G = (K \cup S, E)$.*

Proof. We separately show the correctness of Procedure Minimum_DS as follows:
(i) $\sum_{i=1}^{|K|} \alpha_i = |S|$,

(ii) α is a non-increasing sequence,
(iii) α is realizable by a spanning caterpillar of G, and
(iv) α is the minimum degree sequence with respect to the ordering $\preceq$.

Here, note that statement (ii) together with $\alpha_{|K|} \geq 0$ implies that α_i is non-negative for all $i = 1, \ldots, |K|$.

First, we can easily see that α_i computed by Procedure MINIMUM_DS is obtained by solving the following integer programming problem from $i = |K|$ down to 1:

$$\begin{array}{ll} \text{Problem} & \text{IP}(i) \\ \text{maximize} & \alpha_i \\ \text{s.t.} & \ell\alpha_i + \sum_{j=i+1}^{|K|} \alpha_j \leq d_{i-\ell+1},\ \ell = 1, \ldots, i \\ & \alpha_i\text{: an integer.} \end{array} \tag{5}$$

Therefore, we show (i) $\sim$ (iv) by using IP(i).

For (i), we have $\sum_{i=1}^{|K|} \alpha_i = d_1$ by IP(1). Furthermore, since G is a connected threshold graph, $d_1 = |S|$ holds.

For (ii), notice that α_{i+1} is the maximum integer that satisfies

$$\ell\alpha_{i+1} + \sum_{j=i+1}^{|K|} \alpha_j \leq d_{i-\ell+1},\ \ell = 0, 1, \ldots, i.$$

Therefore, α_{i+1} is a feasible solution of the α_i in IP(i). This means that $\alpha_i \geq \alpha_{i+1}$.

For (iii), let us consider the caterpillar $T = (K \cup S, E_T)$ by

$$E_T = \Big\{(u_i, u_{i+1}) \mid i = 1, \ldots, |K|-1\Big\} \cup \Big\{(u_i, v_{j_i}) \mid i = 1, \ldots, |K|,\ j_i = \sum_{j=i+1}^{|K|} \alpha_j + 1, \ldots, \sum_{j=i}^{|K|} \alpha_j\Big\}. \tag{6}$$

Note that T is a caterpillar defined by α, i.e., $T = T^{\alpha}$. Moreover, from the constraints (5) for $\ell = 1$, we have $\sum_{j=i}^{|K|} \alpha_j \leq d_i$, $i = 1, 2, \ldots, |K|$. This together with Lemma 1 (iii) implies $E_T \subseteq E$, which completes the proof of (iii).

For (iv), suppose that there exists a degree sequence β of (K, S) such that $\beta \not\succeq \alpha$ and β is realizable by a spanning caterpillar of G. Then some k ($2 \leq k \leq |K| - 1$) satisfies $\sum_{i=1}^{k-1} \alpha_i > \sum_{i=1}^{k-1} \beta_i$, i.e.,

$$\sum_{i=k}^{|K|} \alpha_i < \sum_{i=k}^{|K|} \beta_i. \tag{7}$$

Let k be the largest integer satisfying (7). By definition, we have $\alpha_k < \beta_k$. We can see that β also satisfies (5) (since otherwise non-increasing sequence β is not realizable by a spanning caterpillar of G). In particular, we have $\ell\beta_k + \sum_{j=k+1}^{|K|} \beta_j \leq d_{k-\ell+1}$ ($\ell = 1, \ldots, k$). This means

$$\sum_{j=k}^{|K|} \beta_j \leq d_{k-\ell+1} - (\ell - 1)\beta_k, \quad \ell = 1, \ldots, k. \tag{8}$$

On the other hand, since α_k is the maximum integer satisfying (5) for $i = k$, some inequality for $\ell = \ell^*$ must be tight for α_k, i.e., $\sum_{j=k}^{|K|} \alpha_j = d_{k-\ell^*+1} - (\ell^* - 1)\alpha_k$. Since $\alpha_k < \beta_k$, we have $\sum_{j=k}^{|K|} \alpha_j \geq \sum_{i=k}^{|K|} \beta_i$, a contradiction to (7) ∎

For example, by applying this procedure to a threshold graph in Fig. 11, we obtain the minimum degree sequence $\alpha = (3, 2, 1)$, which is also shown in Fig. 11 as bold lines.

6.2 An Algorithm and its Complexity

Finally we can describe an algorithm that computes a minimum edge ranking spanning tree for a given threshold graph.

Algorithm MERST_THRESHOLD
Input: A threshold graph $G = (K \cup S, E)$.
Output: A minimum edge ranking spanning tree $T = (V, E_T)$ of G with a minimum edge ranking $r : E_T \to \boldsymbol{Z}_+$.
begin
1. Compute the minimum degree sequence α by procedure MINIMUM_DS.
2. Construct a spanning caterpillar $T^\alpha = (K \cup S, E_T)$ by (6).
3. Compute a minimum edge ranking r of T^α.

end.

For a threshold graph in Fig. 11, we obtain a minimum edge ranking spanning tree as shown in Fig. 12.

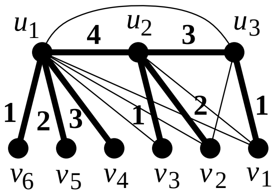

Figure 12: A threshold graph and a minimum edge ranking spanning tree.

Theorem 16 *Algorithm* MERST_THRESHOLD *correctly computes a minimum edge ranking spanning tree of a threshold graph G in $O(|V| + |E|)$ time.*

Proof. Since the correctness of the algorithm is clear from the discussion so far, we only examine its time complexity. Note that a vertex partition (K, S) and $d = (d_1, \ldots, d_{|K|})$ can be computed in $O(|V|+|E|)$ time. Procedure MINIMUM_DS computes the minimum degree sequence α from (K, S) and d in $O(|K|^2) = O(|E|)$ time. More careful implementation shows that it can be done in $O(|V|)$ time. Step 2 can be done in $O(|V|)$, and Step 3 is also possible in $O(|V|)$ time [12]. ∎

Note that the proof of Theorem 16 implies that MERST for threshold graphs is solvable in $O(|V|)$ time if $d = (d_1, \ldots, d_{|K|})$ is given in advance.

7 Concluding Remarks

This paper showed that the minimum edge ranking spanning tree problem (MERST), which is NP-hard in general, has a polynomial time algorithm for split graphs by exploiting several properties of edge rankings of caterpillars. Especially, it shoed that there exists a linear time algorithm for threshold graphs. These results are significant in the sense that these are the first non trivial graph classes that turn out to have polynomial time algorithms for MERST.

Some issues remain for future work. One direction is to find other graph classes in which MERST is polynomially solvable and characterize those classes. Another issue is to investigate the minimum edge ranking problem (MER) itself. MER certainly is important since MERST is originated from it, however, no graph classes other than trees are known to have polynomial time algorithms.

We have already mentioned the vertex version of our problem, i.e., the minimum vertex ranking spanning tree problem (MVRST) [14]. This is another issue to be dealt with, but its complexity is still open in general.

Acknowledgment

This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports, Culture and Technology of Japan. We deeply express our appreciation to Prof. Tiko Kameda at Simon Fraser University for his valuable comments.

References

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

[2] V. Chvátal and P. L. Hammer, Aggregation of inequalities in integer programming, *Annals of Discrete Mathematics*, **1** (1977), 145–162.

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

[4] S. Földes and P. L. Hammer, Split graphs, *Congres. Numer.*, **19** (1977), 311–315.

[5] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[6] P. L. Hammer, T. Ibaraki and B. Simeone, Threshold sequences, *SIAM Journal on Algebraic and Discrete Mathematics*, **2**, 39–49, 1981.

[7] P. L. Hammer and B. Simeone, The splittance of a graph, *Combinatorica*, **1** (1981), 275–284.

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

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

[10] J. Kleinberg, Y. Rabani and É. Tardos, Fairness in routing and load balancing, *Journal of Computer and System Sciences*, **63** (2001), 2–20.

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

[12] T. W. Lam and F. L. Yue, Optimal edge ranking of trees in linear time, *Algorithmica*, **30** (2001), 12–33.

[13] N. V. R. Mahadev and U. N. Peled, *Threshold Graphs and Related Topics*, Ann. Disc. Math., 56, North-Holland, 1995.

[14] K. Makino, Y. Uno and T. Ibaraki, On minimum edge ranking spanning trees, *Journal of Algorithms*, **38** (2001), 411–437.

[15] K. Makino, Y. Uno and T. Ibaraki, Minimum edge rankings spanning trees of threshold graphs, in *Proceedings of the 13th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science 2518 (P. Bose and P. Morin, eds.), (2002), 428–440.

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