2003－ＡＬ－92　(6)
2003／11／7

置換グラフ上における最小節点ランキング全域木問題を解くアルゴリズム

中山　慎一 †　　　増山　繁 ‡

† 徳島大学総合科学部自然システム学科数理科学
‡ 豊橋技術科学大学　知識情報工学系

要 旨

最小節点ランキング全域木問題とは，与えられたグラフ G 上において，節点ランキングが最小となる全域木を求める問題である．本論文では，置換グラフ上における最小節点ランキング全域木問題を解く $O(n^3)$ 時間アルゴリズムを提案する．

An algorithm for solving the minimum vertex ranking spanning tree problem on permutation graphs.

Shin-ichi Nakayama†　　Shigeru Masuyama‡

†Department of Mathematical Sciences, Faculty of Integrated Arts and Sciences, The University of Tokushima,
‡Department of Knowledge-Based Information Engineering, Toyohashi University of Technology,

Abstract

The minimum vertex ranking spanning tree problem is to find a spanning tree of G whose vertex ranking is minimum. This paper proposes an $O(n^3)$ time algorithm for solving the minimum vertex ranking spanning tree problem on a permutation graph.

1 Introduction

Consider a simple connected undirected graph $G = (V, E)$. A vertex ranking of G is labeling r from the vertices of G to the positive integers such that for each path between any two vertices u and v, $u \neq v$, with $r(u) = r(v)$, there exists at least one vertex w on the path with $r(w) > r(u) = r(v)$. The value $r(v)$ of a vertex v is called the rank of vertex v. A vertex ranking r of G is minimum if the largest rank k assigned by r is the smallest among all rankings of G. Such rank k is called the vertex ranking number of G, denoted by $\chi(G)$. The vertex ranking problem is to find a minimum ranking of given graph G. The vertex ranking problem has interesting applications to e.g., communication network design, planning efficient assembly of products in manufacturing systems [19], and VLSI layout design[18].

As for the complexity, this problem is NP-complete even when restricted to cobipartite graphs [13] and bipartite graphs [2], and a number of polynomial time algorithms for this problem have been developed on several subclasses of graphs. Much work has been done in finding the minimum vertex ranking of a tree; a linear time algorithm for trees is proposed in [16]. The problem is trivial on split graphs and is solvable in linear time on cographs [17]. Concerning to interval graphs, Deogun et al has given an $O(n^3)$ time algorithm recently[5], which outperforms the previously known $O(n^4)$ time algorithm [1] where n is the number of vertices. They also presented $O(n^6)$ time algorithms on permutation graphs and on trapezoid graphs, respectively, and showed that a polynomial time algorithm on d-trapezoid graphs exists [5]. Moreover, a polynomial time algorithm on graphs with treewidth at most k was developed [3].

The problem described above is the ranking with respect to vertices, while a ranking with respect to edges is similarly defined as follows. An edge ranking of G is labeling r_e from the edges of G to the positive integers such that for each path between any two edges e_u and e_v, $e_u \neq e_v$, with $r(e_u) = r(e_v)$, there exists at least one edge e_w on the path with $r(e_w) > r(e_u) = r(e_v)$. The value $r(e_v)$ of an edge e_v is called the rank of edge e_v. An edge ranking of G is minimum if the largest rank k assigned is the smallest among all rankings of G. Such rank k is called the edge ranking number of G, denoted by $\chi_e(G)$. The edge ranking problem is to find a minimum edge ranking of given graph G. Before the proof of this problem to be NP-complete was given, an $O(n^3)$ time algorithm for trees was known [19]. By now, a linear time algorithm for trees is shown in [9]. Recently, it has finally been shown that this problem on general graphs is NP-complete [8].

Makino et al. introduced a minimum edge ranking spanning tree problem which is related to the minimum edge ranking problem but is essentially different [11]. The minimum edge ranking spanning tree problem is to find a spanning tree of G whose edge ranking is minimum. They proved that this problem is NP-complete and presented an approximation algorithm for this problem. This problem has interesting applications, e.g., scheduling the parallel assembly of a multipart product from its components and the relational database [11].

In this paper, we consider the vertex version of this problem, i.e., the minimum vertex ranking spanning tree problem. The minimum vertex ranking spanning tree problem is to find a spanning tree of G whose vertex ranking is minimum. We recently proved that this problem is NP-complete[10] and developed an $O(n^3)$ time algorithm when an input graph is an interval graph [12]. We show that, in this paper, an $O(n^3)$ time algorithm for the minimum vertex ranking spanning tree exists when an input graph is a permutation graph. It is interesting that, for permutation graphs, the minimum vertex ranking spanning tree problem is solved in $O(n^3)$ time, although the time complexity of known algorithm for the minimum vertex ranking problem is $O(n^6)$.

2 Permutation graph

Let $V = \{v_1, v_2, \cdots, v_n\}$ and $\pi = [\pi[1], \pi[2], \cdots, \pi[n]]$ be a permutation on V. We construct an undirected graph $G(\pi) = (V, E)$ such that $\{v_i, v_j\} \in E$ iff $(i - j)(\pi^{-1}[i] - \pi^{-1}[j]) < 0$, where $\pi^{-1}[i]$ denotes the position of vertex v_i in π. An undirected graph G is a *permutation graph* if there exists a π such that G is isomorphic to $G(\pi)$ [6]. Pnueli et al.[14] describe an $O(n^3)$ algorithm for testing if a given undirected graph is a permutation graph. This result was improved to $O(n^2)$ by Spinrad [15], whose algorithm produces the corresponding permutation if the graph is a permutation graph.

A permutation graph can also be visualized by its corresponding *permutation diagram.* The permutation diagram consists of two horizontal parallel channels, named the top channel and the bottom channel, respectively. Put the index $1, 2, \cdots, n$ of vertices on the top channel, in the order from left to right, and put the index of vertex in $\pi[1], \pi[2], \cdots, \pi[n]$ on the bottom channel in the same way. Finally, for each i, draw a straight line joining the two i's, one on the top channel and the other on the bottom channel, respectively [6]. The index number i of vertex v_i is same as that of the corresponding line l_i. Note that line l_i intersects line l_j in the diagram iff l_i and l_j appear in the reversed order in π. That is, lines l_i and l_j intersect iff vertices v_i and v_j of the corresponding permutation graph are adjacent. The reader is encouraged to draw the permutation diagram for given π's since they are sometimes quite useful in visualizing the properties of the original permutation graphs.

Permutation graphs are a useful discrete mathematical structure for modeling practical problems [6]. Moreover, permutation graphs construct an important class of perfect graphs and many problems that are NP-complete on arbitrary graphs are shown to admit polynomial time algorithms on this class [6].

3 The basic idea of the algorithm

The basic idea of our algorithm is as follows: First find a shortest path P^* of G between a certain pair of vertices, then construct a spanning tree with the minimum vertex ranking by joining each vertex $v \in V - V(P^*)$ to a vertex of P^* using an edge of G, based on the fact, to be proven in this paper, that, for permutation graphs, $v \in V - V(P^*)$ not included in P^* is adjacent to some vertex on P^*. For preparation, we introduce a known result on the vertex ranking of paths.

Lemma 1 (17) *The ranking $\chi(P)$ of a path $P = x_1, x_2, \cdots, x_n$ is $\lfloor \log n \rfloor^1 + 1$.* □

In the following, we clarify what kind of shortest path P^* is selected and how each vertex in $V - V(P^*)$ should be joined to some vertex on P^* in order to construct a minimum vertex ranking spanning tree.

A shortest path to be selected in our algorithm is a shortest path between a vertex corresponding to the rightmost line on the diagram and a vertex corresponding to the leftmost line on the diagram. Namely, denoting the vertex corresponding to a line whose position is 1 and n on the top (resp. bottom) channel by v_1^t (resp. v_1^b) and v_n^t (resp. v_n^b), respectively, we select a path whose length is shortest among four shortest paths from v_1^t to v_n^t, from v_1^t to v_n^b, from v_1^b to v_n^t and from v_1^b to v_n^b. Note here that the length of each edge is 1. Let P^* be the selected shortest path. On a spanning tree T of permutation graph G, as the length of a diameter of T is equal to or greater than that of P^*, for the minimum ranking $\chi(P^*)$ of P^* on G, $\chi(P^*) \leq \chi(T)$.

Our algorithm first finds the shortest path P^* described above and then constructs a spanning tree by joining each vertex in $V - V(P^*)$ to a vertex on P^* using an edge of G. Now, we show that, for permutation graph G, each vertex in $V - V(P^*)$ is adjacent to some vertices on P^*.

Lemma 2 *Let a shortest path selected by the above process be $P^* = v_1, v_2, \ldots, v_l$. For permutation graphs $G = (V, E)$, each vertex in $V - V(P^*)$ is adjacent to some vertex on P^* in G.*

[1] Throughout this paper, log denotes $\log_2$.

(Proof) We consider lines $l_1, l_2, \ldots, l_l$ corresponding to vertices $v_1, v_2, \ldots, v_l$, respectively. If a vertex v is not adjacent to any vertex on P^*, none of lines $l_1, l_2, \ldots, l_l$ intersects the line l_v corresponding to v. Hence, l_v is to the left of l_1 or is to the right of l_l. However, by the definition of P^*, as v_1 (resp. v_l) corresponds to the leftmost (resp. rightmost) line on the diagram, a line setting on the left (resp. right) position of l_1 (resp. l_l) but not intersecting l_1 (resp. l_l) does not exist. Thus, $v \in V - V(P^*)$ is adjacent to a vertex on P^*. □

We now consider how each vertex in $V - V(P^*)$ should be joined to a vertex on P^* in order to construct a minimum vertex ranking spanning tree. Let a vertex set $V - V(P^*)$ be V'. By lemma 2, each vertex $v' \in V'$ is adjacent to a vertex on P^*. Then, our algorithm finds a path P^* of G and joins each vertex in V' to a vertex on P^* using an edge of G.

By Lemma 2, the relation of connections between $v' \in V'$ and vertices on P^* are classified into the following three cases.

(1) $v' \in V'$ is adjacent to only one vertex on P^*.
(2) $v' \in V'$ is adjacent to two consecutive vertices v_j, v_{j+1} on P^* or three consecutive vertices v_j, v_{j+1}, v_{j+2} on P^*.
(3) $v' \in V'$ is not adjacent to consecutive vertices on P^* but adjacent to two vertices v_j, v_{j+2} having one skip on P^*.

Note: As P^* is the shortest path, $v' \in V'$ is adjacent to neither more than three consecutive vertices on P^* in the case (2) nor two vertices which have more than one skip on P^* in the case (3).

Let V_1' denote a subset of V' that contains vertices in V' each of which is adjacent to only one vertex on P^*, let V_2' denote a subset of V' that contains vertices in V' each of which is adjacent to two or three consecutive vertices on P^* and let V_3' denote a subset of V' that contains vertices in V' each of which is adjacent to two vertices v_j, v_{j+2} having one skip on P^*.

We first consider $v'' \in V_2'$ adjacent to two or three consecutive vertices on P^*. As for $v'' \in V_2'$ adjacent to at least two vertices on P^*, we can select a vertex on P^* to be joined to v'' in order to construct a spanning tree. Then, let us consider to which vertex of P^* $v'' \in V_2'$ should be joined. After finding the minimum vertex ranking of P^*, for consecutive vertices v_i, v_{i+1} on P^*, either $r(v_i) > r(v_{i+1})$ or $r(v_i) < r(v_{i+1})$ holds by the definition of the vertex ranking. As $v'' \in V_2'$ is adjacent to at least two consecutive vertices on P^*, v'' is adjacent to a vertex v on P^* whose rank is at least 2. Then, joining v'' to v and assigning rank 1 to v'', we can construct a spanning tree T with $\chi(T) = \chi(P^*)$, without changing the rank of vertices on P^*.

Next, we consider $v' \in V_1'$ adjacent to only one vertex on P^* and $v' \in V_3'$ which is adjacent to two vertices having one skip on P^*. In this case, depending on the result of vertex ranking of P^*, v' may be adjacent to a vertex v on P^* with rank 1. Then, when selecting the edge (v', v) in order to construct a spanning tree, we must modify the rank of v for satisfying the vertex ranking. Moreover, G may not have a spanning tree T such that $\chi(T) = \chi(P^*)$. Fortunately, for permutation graphs, the upper bound on $\chi(T)$ is determined as shown in the following lemma.

Lemma 3 *For permutation graph G, the ranking $\chi(T)$ of a spanning tree T satisfies the following inequality:* $\chi(T) \leq \chi(P^*) + 1$.

(Proof) By lemma 2, any vertex v not included in P^* is adjacent to some vertex on P^*. We assume that each vertex on P^* is given a rank such that the ranking of P^* is minimum. For each vertex v on P^*, the rank $r(v) + 1$ is newly assigned to v, that is, $r(v) \leftarrow r(v) + 1$. Each rank $r(v')$ of $v' \in V'$ is set to 1. Then, the ranking of a tree constructed by P^* and $v' \in V'$ satisfies the condition of vertex ranking. Therefore, $\chi(T) \leq \chi(P^*) + 1$. □

By lemma 3, the ranking of spanning tree $\chi(T)$ is either $\chi(P^*)$ or $\chi(P^*) + 1$. Therefore, our algorithm tries to construct a spanning tree T with rank $\chi(P^*)$. As a result, if we can not construct a spanning tree T with rank $\chi(P^*)$, we construct a spanning tree T with rank $\chi(P^*) + 1$.

After assigning ranks to vertices on P^* with a minimum ranking, if the rank of a vertex v_j on P^* adjacent to $v' \in V_1'$ is 1, a spanning tree satisfying the ranking condition can not be constructed by joining v' to v_j by this assignment. Similarly, if each rank of vertices v_j, v_{j+2} on P^* adjacent to $v' \in V_3'$ is 1, a spanning tree satisfying the ranking condition can not be constructed by joining v' to v_j or v_{j+2}. In these cases, we may get a spanning tree satisfying the ranking condition either by changing the rank of v_j (or v_{j+2}) to become greater than 1 or by joining v' to a vertex in V'. Then, our algorithm classifies each vertex $v' \in V_1' \cup V_3'$ according to the connection between v' and vertices on P^* and selects an edge to join v'.

For illustration, we now consider the minimum vertex ranking of trees. A tree is divided into more than one components $T_1, T_2, \cdots, T_l$ by removing a vertex v other than a leaf. A path from a vertex of T_i to a vertex of T_j $(i \neq j)$ obviously go through v. Then, by assigning the largest rank max$\{ \chi(T_1), \chi(T_2), \cdots, \chi(T_l)\} + 1$ to v, the condition of vertex ranking of the tree is satisfied. However, the resulting vertex ranking is not necessarily the minimum one. Based on this observation, we develop an algorithm as sketched below. We assign the largest rank $\chi(P^*)(= \lfloor \log |P^*| \rfloor + 1)$ to a vertex v_i on $P^*(= v_1, \cdots, v_l)$. (Here $|P^*|$ denotes the number of vertex on P^*.) Then, we pay attention to two subgraphs $G_{v_i}^1$, $G_{v_i}^2$ of G such that $G_{v_i}^1$ is induced by path $v_1, v_2, \cdots, v_{i-1}$ and vertices in V' $(= V - V(P^*))$ adjacent to $v_1, v_2, \cdots, v_{i-1}$ and $G_{v_i}^2$ is induced by path $v_{i+1}, v_{i+2}, \cdots, v_l$ and vertices in V' adjacent to $v_{i+1}, v_{i+2}, \cdots, v_l$, respectively. As will be described in detail later, the case when $G_{v_i}^1$ and $G_{v_i}^2$ share a common vertex v^* of V' needs to be treated separately. Then, we find a minimum vertex ranking spanning tree T_1 in $G_{v_i}^1$ and T_2 in $G_{v_i}^2$, respectively. If both of minimum vertex rankings of T_1 and T_2 are not greater than $\lfloor \log |P^*| \rfloor$, a spanning tree with ranking $\chi(P^*)(= \lfloor \log |P^*| \rfloor + 1)$ can be constructed by joining T_1, T_2 via v_i. Even when a spanning tree with ranking $\lfloor \log |P^*| \rfloor + 1$ can not be constructed, by using some other vertex on P^* instead of v_i, a spanning tree with ranking $\lfloor \log |P^*| \rfloor + 1$ may be constructed. Hence, we check whether each of $G_{v_i}^1$ and $G_{v_i}^2$ has a spanning tree with ranking at most $\lfloor \log |P^*| \rfloor$ for each v_i, $i = 2, \cdots, l-1$, with the largest rank. For this purpose, we use the dynamic programming. We check whether a subgraph induced by k consecutive vertices v_j, $\cdots$, v_{j+k} on P^*, $(j = 1, \cdots, l,\ k = 0, \cdots, l-j)$, and vertices in V' adjacent to v_j, $\cdots$, v_{j+k} has a spanning tree with ranking $\lfloor \log |P^*_{v_j v_{j+k}}| \rfloor + 1$. (Note that $P^*_{v_i v_j}$ denotes a subpath v_i, $\cdots$, v_j on P^*.) Therefore, we now consider a spanning tree on a subgraph induced by consecutive vertices v_j, $\cdots$, v_{j+k} on P^* and vertices in V' adjacent to v_j, $\cdots$, v_{j+k}.

Let define some terms needed to explain the algorithm in the following. As for consecutive vertices $v_j, \cdots, v_k$ on P^*, a subgraph of G induced by $v_j, \cdots, v_k$ and vertices in $V^{'}$ adjacent to $v_j, \cdots, v_k$ is called a *subgraph regarding* v_j, $\cdots$, v_k and denoted by $G[v_j, v_k]$. For $G[v_j, v_k]$, if we can construct a spanning tree such that each rank of vertices in $G[v_j, v_k]$ is at most $\lfloor \log |P^*_{v_j v_k}| \rfloor + 1 (= \chi(P^*_{v_j v_k}))$, we say that $G[v_j, v_k]$ is *minimum-rankable*.

Note: For a subgraph $G[v_i, v_i]$ regarding one consecutive sequence of vertices, as we can always construct spanning tree T with ranking at most 2 by assigning rank 2 to v_i and rank 1 to vertices adjacent to v_i. Then, we say that each subgraph $G[v_i, v_i]$ regarding one consecutive vertices is minimum-rankable.

Using these terms, what we are going to do in the dynamic programming is as follows: Let subpaths of P^* selected in the first step be $P_1^* = v_i, \cdots, v_{j-1}$ and $P_2^* = v_{j+1}, \cdots, v_k$, respectively. We check whether $G[v_i, v_{j-1}]$, $G[v_{j+1}, v_k]$ are minimum-rankable or not. If each of $G[v_i, v_{j-1}]$, $G[v_{j+1}, v_k]$ is minimum-rankable, the subgraph $G[v_i, v_k]$ regarding $v_i, \cdots, v_k$ is minimum-rankable by assigning $\lfloor \log |P^*_{v_j v_k}| \rfloor + 1$ to v_j. However, when $G[v_i, v_{j-1}]$ and $G[v_{j+1}, v_k]$ share a common vertex, even if these are not minimum-rankable, we need to check some conditions, to be described later, because $G[v_i, v_k]$ may be minimum-rankable. If either of $G[v_i, v_{j-1}]$ or $G[v_{j+1}, v_k]$ is not minimum-rankable and do not share a common vertex, $G[v_i, v_k]$ is not minimum-rankable.

As mentioned above, for constructing a minimum vertex ranking spanning tree, our algorithm first check whether subgraphs $G[v_i, v_{i+1}]$, for $i = 1, \cdots, l-1$, regarding two consecutive vertices on P^* is minimum-rankable, and then check whether subgraphs $G[v_i, v_{i+2}]$, for $i = 1, \cdots, l-2$, regarding three consecutive vertices on P^* is minimum-rankable. Concerning subgraphs $G[v_i, v_{i+k}]$, $k \leq 3$, regarding more than three consecutive vertices on P^*, using known information about subgraphs, we check whether $G[v_i, v_{i+k}]$ is minimum-rankable by using the dynamic programming.

We then consider the way to check whether a subgraph regarding consecutive vertices is minimum-rankable. We classify each vertex $v^{'} \in V_1^{'} \cup V_3^{'}$ according to the connection between $v^{'}$ and vertices on P^* and investigate whether each case is minimum-rankable or not.

3.1 Subgraph regarding two consecutive vertices

We consider whether a subgraph $G[v_j, v_{j+1}]$ regarding two consecutive vertices v_j, v_{j+1} on P^* is minimum-rankable or not. That is, we examine whether we can construct a spanning tree such that each rank of vertices in $G[v_j, v_{j+1}]$ is at most $\lfloor \log |P^*_{v_j v_{j+1}}| \rfloor + 1$ $(= \chi(P^*_{v_j v_{j+1}}) = 2)$. We classify the cases by connection between $v^{'} \in V_1^{'} \cup V_3^{'}$ and a vertex of P^*. However, we do not consider, for brevity, the cases which can be treated by discussions similar to some other cases due to symmetry. The proof of each case is omitted due to the space limit.

Case 1: $v^{'} \in V_1^{'}$ is adjacent to only one vertex on P^*.

Case 1-1: If each of v_j and v_{j+1} is adjacent to a vertex in V_1' whose degree is 1, $G[v_j, v_{j+1}]$ is not minimum-rankable. However, if either of v_j and v_{j+1} is adjacent to a vertex in V_1' whose degree is 1, $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 1-2: v_j is adjacent to $v_j' \in V_1'$ whose degree is at least 2, or v_{j+1} is adjacent to $v_{j+1}' \in V_1'$ whose degree is at least 2.

Case 1-2-1: If v_j, v_{j+1} are adjacent to $v_j', v_{j+1}' \in V_1'$, respectively, and v_j' and v_{j+1}' are only adjacent to each other, $G[v_j, v_{j+1}]$ is not minimum-rankable.

Case 1-2-2: If v_j, v_{j+1} are adjacent to $v_j', v_{j+1}' \in V_1'$, respectively, and v_j' and v_{j+1}' are adjacent to a vertex $v^{''} \in V_2^{'}$, $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 1-2-3: If v_j, v_{j+1} are adjacent to $v_j', v_{j+1}' \in V_1'$, respectively, and v_{j+1}' is adjacent to a vertex $v^* \in V_2^{'}$ adjacent to v_{j+2}', then $G[v_j, v_{j+1}]$ is minimum-rankable. (By symmetry, the case where v_j' is adjacent to a vertex $v^* \in V_2^{'}$ adjacent to v_{j-1}', can be discussed in a similar way.)

Case 2: $v^{'''} \in V_3^{'}$ is adjacent to not consecutive vertices on P^* but two vertices v_j, v_{j+2} having one skip on P^*.

Case 2-1: If $v^{'''} \in V_3^{'}$ is adjacent to only two vertices v_j and v_{j+2}, then $G[v_j, v_{j+1}]$ is minimum-rankable. (By symmetry, the case where $v^{'''} \in V_3^{'}$ is only adjacent to v_{j-1} and v_{j+1}, can be discussed in a similar way.)

Case 2-2: If $v''' \in V_3'$ is adjacent to only two vertices v_{j+1} and v_{j+3}, then $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 3: Both vertices in V_1' and in V_3' exist in $G[v_j, v_{j+1}]$.

Case 3-1: A vertex in V_1' and a vertex in V_3' share a common vertex on P^*.

Case 3-1-1: $v''' \in V_3'$ is adjacent to v_j and v_{j+2} on P^* and either v_j or v_{j+1} is adjacent to vertices in V_1'.

Case 3-1-2: If $v''' \in V_3'$ is adjacent to v_{j+1} and v_{j+3} on P^* and v_{j+1} is adjacent to a vertex in V_1', $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 3-2: Vertices in V_1' and these in V_3' do not share a common vertex: $v''' \in V_3'$ is adjacent to v_{j+1}, v_{j+3} on P^*, the degree of v''' is 2 and a vertex in V_1' with degree 1 is adjacent to v_j.

When a vertex in V_1' is adjacent to v_j, v''' can not be joined to v_{j+1} for $G[v_j, v_{j+1}]$ to be minimum-rankable. Then, in this case, whether $G[v_j, v_{j+1}]$ is minimum-rankable or not depends on the rank of v_{j+3} in a subgraph $G[v_{j+3}, *]$ regarding v_{j+3}, v_{j+4}, $\cdots$. If the rank of v_{j+3} is greater than 1, we can join v''' to v_{j+3}. Therefore, in this case, we decide whether $G[v_j, v_{j+1}]$ is minimum-rankable or not when connecting a spanning tree in $G[v_j, v_{j+1}]$ and one in $G[v_{j+3}, *]$ via v_{j+2}.

In the following, we call a vertex like v''' a *suspension vertex* and if $G[v_j, v_{j+1}]$ has a suspension vertex, we say that $G[v_j, v_{j+1}]$ is not minimum-rankable by a suspension vertex.

3.2 Subgraph regarding three consecutive vertices

We consider whether a subgraph $G[v_j, v_{j+2}]$ regarding three consecutive vertices v_j, v_{j+1}, v_{j+2} on P^* is minimum-rankable or not. We classify the cases with respect to connection between $v' \in V_1' \cup V_3'$ and a vertex of P^*. However, we eliminate the cases which can be treated in a manner similar to some other cases due to symmetry. The proof of each case is omitted due to the space limit.

Case 4: $v' \in V_1'$ is adjacent to only one vertex on P^*.

Case 4-1: If v_j is an articulation (1-cut) vertex in G and $v_j' \in V_1'$ adjacent to v_j is not adjacent to a vertex adjacent to v_{j-1}, then $G[v_j, v_{j+2}]$ is not minimum-rankable. Note that an *articulation vertex* is a vertex of a connected graph whose deletion disconnects the graph. (By symmetry, the case where v_{j+2} is an articulation vertex and $v_{j+2}' \in V_1'$ adjacent to v_{j+2} is not adjacent to a vertex adjacent to v_{j+3}, can be discussed in a similar way.)

Case 4-2: If $v_j' \in V_1'$ adjacent to v_j is adjacent to v_{j-1}' adjacent to v_{j-1}, $G[v_j, v_{j+2}]$ is minimum-rankable. (By symmetry, the case where $v_{j+2}' \in V_1'$ adjacent to v_{j+2} is adjacent to v_{j+3}' adjacent to v_{j+3}, can be discussed in a similar way.)

Case 4-3: v_j and v_{j+2} are not articulation vertices: Whereas $v_j' \in V_1'$ is adjacent to v_j, if $v_j^* \in V_2' \cup V_3'$ that is adjacent to v_{j-1} and v_{j+1} exists, $G[v_j, v_{j+2}]$ is minimum-rankable. (As for v_{j+2}, we can discuss in a similar way.)

Case 5: $v''' \in V_3'$ is adjacent to not consecutive vertices on P^* but adjacent to two vertices v_j, v_{j+2} having one skip on P^*.

Case 5-1: If $v''' \in V_3'$ is adjacent to only two v_j and v_{j+2} on P^* and v_j and v_{j+2} are articulation vertices, then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 5-2: If $v''' \in V_3'$ is adjacent to two vertices v_j, v_{j+2} and $v_{j+3}' \in V'$ is adjacent to v_{j+3}, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 5-3: If $v''' \in V_3'$ is adjacent to both v_j and v_{j+2} on P^* and $v^* \in V_2' \cup V_3'$ that is adjacent to both v_{j+1} and v_{j+3} exits, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 5-4: If $v''' \in V_3'$ is adjacent to two vertices v_{j+1}, v_{j+3} on P^*, then $G[v_j, v_{j+2}]$ is minimum-rankable. (By symmetry, the case where $v''' \in V_3'$ is adjacent to two vertices v_{j-1}, v_{j+1} on P^*, can be discussed in a similar way.)

Case 5-5: If $v''' \in V_3'$ is adjacent to only two vertices v_{j+2} and v_{j+4} on P^* and v_{j+2} and v_{j+4} are articulation vertices, then $G[v_j, v_{j+2}]$ is not minimum-rankable by a suspension vertex. (By symmetry, the case where $v''' \in V_3'$ is adjacent to only two vertices v_j and v_{j-2} on P^*, and the fact that v_j and v_{j-2} are articulation vertices can be discussed in a similar way.)

Case 5-6: If $v''' \in V_3'$ is adjacent to v_{j+2}, v_{j+4} on P^* and is adjacent to $v_{j+3}' \in V'$ adjacent to v_{j+3}, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 5-7: If $v''' \in V_3'$ is adjacent to v_{j+2}, v_{j+4} on P^* and $v^* \in V_2' \cup V_3'$ that is adjacent to v_{j+1} and v_{j+3} exits, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6: Both vertices in $V_1^{'}$ and in $V_3^{'}$ exists in $G[v_j, v_{j+2}]$.

Case 6-1: If a vertex in $V_3^{'}$ is adjacent to two vertices v_j, v_{j+2}, $v_j^{'} \in V_1^{'}$ (resp. $v_{j+2}^{'} \in V_1^{'}$) is adjacent to v_j (resp. v_{j+2}) and v_j (resp. v_{j+2}) is articulation vertices, then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 6-2: A vertex $v^{'''} \in V_3^{'}$ is adjacent to two vertices v_j, v_{j+2}, $v_{j+2}^{'} \in V_1^{'}$ (resp. $v_j^{'} \in V_1^{'}$) is adjacent to v_{j+2} (resp. v_j) and a vertex $v_{j+3}^{'} \in V^{'}$ adjacent to v_{j+3} is adjacent to $v^{'''}$ or $v_{j+2}^{'}$.

Case 6-2-1: If $v_{j+3}^{'} \in V^{'}$ is adjacent to $v^{'''} \in V_3^{'}$, then $G[v_j, v_{j+2}]$ is minimum-rankable,

Case 6-2-2: If $v_{j+3}^{'} \in V^{'}$ is adjacent to $v_{j+2}^{'} \in V_1^{'}$ but not adjacent to $v^{'''}$, then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 6-3: A vertex $v^{'''} \in V_3^{'}$ is adjacent to two vertices v_j, v_{j+2}, $v_{j+2}^{'} \in V_1^{'}$ (resp. $v_j^{'} \in V_1^{'}$) is adjacent to v_{j+2} (resp. v_j) and a vertex $v^* \in V_2^{'} \cup V_3^{'}$ is adjacent to v_{j+1} and v_{j+3}. In this case, $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6-4: If a vertex $v^{'''} \in V_3^{'}$ is adjacent to two vertices v_{j+2}, v_{j+4} and $v_{j+2}^{'} \in V_1^{'}$ adjacent to v_{j+2} is adjacent to $v^{'''}$, then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 6-5: If a vertex $v^{'''} \in V_3^{'}$ is adjacent to two vertices v_{j+2}, v_{j+4}, $v_{j+2}^{'} \in V_1^{'}$ adjacent to v_{j+2} is adjacent $v^{'''}$ and $v^{'''}$ is adjacent to $v_{j+3}^{'} \in V^{'}$ adjacent to v_{j+3}, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6-6: If a vertex $v^{'''} \in V_3^{'}$ is adjacent to two vertices v_{j+2}, v_{j+4}, $v_{j+2}^{'} \in V_1^{'}$ adjacent to v_{j+2} is adjacent $v^{'''}$ and a vertex $v^* \in V_2^{'} \cup V_3^{'}$ is adjacent to v_{j+1} and v_{j+3}. In this case, $G[v_j, v_{j+2}]$ is minimum-rankable.

4 An algorithm for solving the minimum vertex ranking spanning tree problem

Following the above explanations given in sections 3.1 and 3.2, we can check whether spanning trees with rank 2 can be constructed in subgraphs regarding two consecutive vertices and subgraphs regarding three consecutive vertices, respectively.

Using the dynamic programming, we then check whether spanning trees with rank $\chi(P^*_{v_j v_{j+3}})(= \lfloor \log |P^*_{v_j v_{j+3}}| \rfloor + 1 = 3)$ can be constructed in subgraphs regarding four consecutive vertices v_j, $\cdots$, v_{j+3} on P^* and spanning trees with rank $\chi(P^*_{v_j v_{j+4}})(= \lfloor \log |P^*_{v_j v_{j+4}}| \rfloor + 1 = 3)$ can be constructed in subgraphs regarding five consecutive vertices and so on. Namely, for example, if each of $G[v_i, v_i]$, $G[v_{i+2}, v_{i+3}]$ is minimum-rankable, the subgraph $G[v_i, v_{i+3}]$ regarding four consecutive vertices $v_i, \cdots, v_{i+3}$ is minimum-rankable by assigning rank $\lfloor \log |P^*_{v_i v_{i+3}}| \rfloor + 1 (= 3)$ to v_{i+1} or if each of $G[v_i, v_{i+1}]$, $G[v_{i+3}, v_{i+3}]$ is minimum-rankable, the subgraph $G[v_i, v_{i+3}]$ is minimum-rankable by assigning rank 3 to v_{i+2}. Thus, if a pair of $G[v_i, v_{j-1}]$ and $G[v_{j+1}, v_k]$ which are minimum-rankable exists, $G[v_i, v_k]$ is minimum-rankable, as otherwise, $G[v_i, v_k]$ is not minimum-rankable.

Our algorithm is described as follows. In the algorithm, we use an array $R[v_i, v_j]$, for i, $j = 1, \cdots, l$. If $G[v_i, v_j]$ is minimum-rankable, 'OK' is assigned to $R[v_i, v_j]$.

Procedure Find_Minimum_Ranking_Spanning_Tree

begin

Step 1. Find a path $P^*(= v_1, v_2, \ldots, v_l)$ whose length is shortest among four shortest paths from v_1^t to v_n^t, from v_1^t to v_n^b, from v_1^b to v_n^t and from v_1^b to v_n^b.

Step 2. For $V - V(P^*)$, find vertex sets $V_1^{'}$, $V_2^{'}$ and $V_3^{'}$.

Step 3. If every vertex in $V - V(P^*)$ is in $V_2^{'}$, a spanning tree with $\chi(T) = \lfloor \log |P^*| \rfloor + 1$ can be constructed. Stop.

Step 4. For $i, j = 1$ to l, $R[v_i, v_j] \leftarrow$ 'null'
For $k = 1$ to l, $R[v_k, v_k] \leftarrow$ 'OK'.

Step 5. For subgraph $G[v_j, v_{j+1}]$ regarding two consecutive vertices v_j, v_{j+1}, $j = 1, \cdots, l-1$, on P^*, check whether $G[v_j, v_{j+1}]$ is minimum-rankable. If $G[v_j, v_{j+1}]$ is minimum-rankable, $R[v_j, v_{j+1}] \leftarrow$ 'OK'.

Step 6. For subgraph $G[v_j, v_{j+2}]$ regarding three consecutive vertices v_j, v_{j+1}, v_{j+2}, $j = 1, \cdots, l-2$, on P^*, check whether $G[v_j, v_{j+2}]$ is minimum-rankable. If $G[v_j, v_{j+2}]$ is minimum-rankable, $R[v_j, v_{j+2}] \leftarrow$ 'OK'.

Step 7. For the pairs of vertices on P^* whose distance is greater than 3, sort $R[v_i, v_k]$'s in increasing order according to value of the distance between v_i and v_k.

Step 8. Compute $R[v_i, v_k]$'s in the order of step 7 as follows :

for each j such that $i < j < k$ *do*

begin

If $G[v_i, v_{j-1}]$ is not minimum-rankable by a suspension vertex v''', we check whether the rank of v_{j+1} adjacent to v''' in $G[v_{j+1}, v_k]$ is 1. If the rank of v_{j+1} is not 1, as a suspension vertex v''' can be joined to v_{j+1} in $G[v_{j+1}, v_k]$ for $G[v_i, v_{j-1}]$ to be minimum-rankable, then $R[v_i, v_{j-1}] \leftarrow$ 'OK'.

If $G[v_{j+1}, v_k]$ is not minimum-rankable by a suspension vertex v''', we check whether the rank of v_{j-1} adjacent to v''' in $G[v_i, v_{j-1}]$ is 1. If the rank of v_{j-1} is not 1, as a suspension vertex v''' can be joined to v_{j-1} in $G[v_i, v_{j-1}]$ for $G[v_{j+1}, v_k]$ to be minimum-rankable, then $R[v_{j+1}, v_k] \leftarrow$ 'OK'.

If the value of $R[v_i, v_{j-1}]$ is 'OK', that of $R[v_{j+1}, v_k]$ is 'OK' and $\max\{\lfloor \log |P^*_{v_i v_{j-1}}| \rfloor + 1, \lfloor \log |P^*_{v_{j+1} v_k}| \rfloor + 1\} \leq \lfloor \log |P^*_{v_i v_k}| \rfloor$ then, $R[v_i, v_k] \leftarrow$ 'OK'.

end

Step 9. If the value of $R[1, l]$ is 'OK', a spanning tree with $\chi(T) = \lfloor \log |P^*| \rfloor + 1$ can be constructed. Otherwise, a spanning tree with $\chi(T) = \lfloor \log |P^*| \rfloor + 1 + 1 (= \chi(P^*) + 1)$ can be constructed.

end.

Theorem 1 *Procedure Find_Minimum_Ranking_Spanning_Tree solves the minimum vertex ranking spanning tree problem in $O(n^3)$ time.*

The proof is lengthy and is omitted due to the space limit.

5 Conclusion

In this paper, we proposed an $O(n^3)$ time algorithm for solving the minimum vertex ranking spanning tree problem, when an input graph is a permutation graph. It is interesting that, for permutation graphs, the minimum vertex ranking spanning tree problem is solved in $O(n^3)$ time, although the time complexity of known algorithm for the minimum vertex ranking problem is $O(n^6)$.

References

[1] B. Aspvall, P. Heggernes : "Finding minimum height elimination tree for interval graphs in polynomial time", *BIT*, **34**, pp. 484-509, 1994.

[2] H. L. Bodlaender, J. S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, Z. Tuza, "Rankings of graphs", *Lecture Notes in Computer Science*, vol. 903, Springer, Berlin, pp.292-304, 1996.

[3] H. Bodlaender, J. R. Gilbert, H. Hafsteinsson, T. Kloks, D. Kratsch, H. Müller, Z. Tuza, "Rankings of Graphs", *SIAM J. Discrete Math*, **11** pp.168-181, 1998.

[4] J.A.Bondy and U.S.R.Murty : "Graph Theory with Applications", North-Holland, 1976.

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

[6] M. C. Golumbic, "Algorithmic graph theory and perfect graphs", *Academic Press*, New York, 1980.

[7] D.E.Knuth, "The Art of Computer Programming, Vol. III: Sorting and Searching", *Addison-Wesley*, Reading Mass., 1973.

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

[9] T. W. Lam, F. L. Yue, "Optimal edge ranking of trees in linear time", Proceeding of the ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.436-445, 1998.

[10] K. Miyata, S. Masuyama, S. Nakayama, "Computational complexity of the minimum vertex ranking spanning tree problem", to be reported in IEICE Technical Report COMP2003, Nov. 2003.

[11] K. Makino, Y. Uno, T, Ibaraki, "On minimum edge ranking spanning trees", *J. Algorithms*, **38**, pp.411-437, 2001.

[12] S. Nakayama, S. Masuyama, "An algorithm for solving the minimum vertex ranking spanning tree problem on interval graphs", *IEICE Trans, Fundamentals*, Vol.E86-A, No.5, pp.1019-1026, 2003.

[13] A. Pothen, "The complexity of optimal elimination trees", Technical Report CS-88-13, Pennsylvania State University, USA, 1988.

[14] A. Pnueli, A. Lempel, S. Even, "Transitive orientation of graphs and identification of permutation graphs", *Can. J. Math.*, **23**, pp.160-175, 1971.

[15] J. Spinrad, "On comparability and permutation graphs", *SIAM J. Computing*, **14**, pp.658-670, 1985.

[16] A. A. Schäffer, "Optimal node ranking of trees in linear time", *Information Process. Lett.*, **33**, pp.91-96, 1989/1990.

[17] P. Scheffler, "Node ranking and searching on graphs (Abstract)", in: U. Faigle, C.Hoede (Eds.), Third Twente Workshop on Graphs and Combinatorial Optimization, Memorandum No. 1132, The Netherlands, 1993.

[18] A. Sen, H Deng, S. Guha, "On a graph partition problem with application to VLSI layout", *Information Process. Lett.*, **43**, pp.87-94, 1992.

[19] P. de la Torre, R. Greenlaw, A. A. Schäffer, "Optimal edge ranking of trees in polynomial time, *Algorithmica*, **13**, pp.592-618, 1995.