Discussiones Mathematicae
Graph Theory 26 (2006) 149–159

OPTIMAL EDGE RANKING OF COMPLETE BIPARTITE GRAPHS IN POLYNOMIAL TIME *

Ruo-Wei Hung

Department of Information Management
Nan-Kai Institute of Technology
Tsao-Tun, Nantou 542, Taiwan

e-mail: rwhung@cs.ccu.edu.tw

Abstract

An *edge ranking* of a graph is a labeling of edges using positive integers such that all paths connecting two edges with the same label visit an intermediate edge with a higher label. An edge ranking of a graph is *optimal* if the number of labels used is minimum among all edge rankings. As the problem of finding optimal edge rankings for general graphs is NP-hard [12], it is interesting to concentrate on special classes of graphs and find optimal edge rankings for them efficiently. Apart from trees and complete graphs, little has been known about special classes of graphs for which the problem can be solved in polynomial time. In this paper, we present a polynomial-time algorithm to find an optimal edge ranking for a complete bipartite graph by using the dynamic programming strategy.

Keywords: graph algorithms, edge ranking, vertex ranking, edge-separator tree, complete bipartite graphs.

2000 Mathematics Subject Classification: 05C78, 05C85, 68Q25.

1. Introduction

All graphs considered in this paper are finite and undirected, without loops or multiple edges. Let $G = (V, E)$ be a graph and let t be a positive integer.

*This research was partly supported by the National Science Council of Republic of China under grant no. NSC94-2213-E252-003.

An *edge t-ranking* of G is an edge labeling $\phi : E \longrightarrow \{1, 2, \cdots, t\}$ such that for every pair of edges e and f with $\phi(e) = \phi(f)$ and for every path between e and f there is an edge g on this path with $\phi(g) > \phi(e)$. The *edge-ranking number* of G, denoted by $\chi_e(G)$, is the smallest value t for which G admits an edge t-ranking. An edge $\chi_e(G)$-ranking of G is said to be an *optimal edge ranking* of G. The *edge ranking problem* is to find an optimal edge ranking of a graph. As revealed in [14, 20], the edge ranking problem is equivalent to the problem of finding a *minimum-height edge-separator tree* of a graph. Finding an optimal edge ranking has an interesting application in scheduling the assembly of multi-part product [10].

The vertex ranking of a graph is closely related to the edge ranking and is similarly defined, except that a labeling on vertex set is used instead of a labeling on edge set. A vertex ranking of a graph is *optimal* if the number of labels used is minimum among all vertex rankings. The *vertex ranking problem* is to find an optimal vertex ranking of a graph. The vertex ranking problem is equivalent to the problem of finding a *minimum-height elimination tree* of a graph [5, 6] and is NP-hard for general graphs [16, 17]. It remains NP-hard even for cobipartite and bipartite graphs [2]. Polynomial-time algorithms are known only for a few special classes of graphs including trees [18], cographs [15, 19], AT-free graphs [11], trapezoid graphs, permutation graphs, interval graphs [1, 7], and circular-arc graphs [7]. The vertex ranking problem on complete bipartite graphs can be easily solved in linear time since these graphs form a subclass of cographs and the vertex ranking problem on cographs can be linear solvable [15].

The edge ranking problem is NP-hard for general graphs [12]. Extensive work has been done in finding optimal edge rankings of trees [5, 21], now there exists a linear-time algorithm for trees [13]. Bodlaender et al. derived a formula to compute the edge ranking-number of a complete graph [2]. Little has been known about special classes of graphs for which the problem can be solved in polynomial time except trees and complete graphs. In this paper, we propose an $O(|E|^2)$-time algorithm to solve the edge ranking problem on complete bipartite graphs, where E is the edge set of the input complete bipartite graph.

The remainder of the paper is organized as follows: In Section 2, we give some basic definitions and notation. Section 3 presents a dynamic programming algorithm to solve the edge ranking problem on complete bipartite graphs. Finally, we make some concluding remarks and state an open problem in Section 4.

2. Preliminaries

For any positive integer n, the *complete graph* of n vertices, denoted by K_n, is a graph of n vertices in which every two vertices are adjacent. A *bipartite graph* is one whose vertices can be partitioned into two (disjoint) nonempty set V_1 and V_2, called *bipartition sets*, in such a way that every edge joins a vertex in V_1 and a vertex in V_2. In particular, there are no edges within V_1 nor V_2. A *complete bipartite graph* is a bipartite graph in which every vertex in V_1 is adjacent to all vertices in V_2. The complete bipartite graph on bipartition sets of m vertices and n vertices, respectively, is denoted by $K_{m,n}$. Properties of and optimization problems on complete bipartite graphs have been studied in [3, 8, 9] and the related literature.

Let $G = (V, E)$ be a graph with vertex set V and edge set E. For $E' \subseteq E$, $G(E')$ denotes the graph (V, E') with vertex set V and edge set E'. The *deletion* of edge set R from G, denoted by $G \setminus R$, is the graph $G(E \setminus R)$. A subset $R \subseteq E$ of a graph $G = (V, E)$ is said to be an *edge separator* if $G(E \setminus R)$ is disconnected. As shown in [14, 20], the edge ranking problem is equivalent to the problem of finding the *minimum-height edge-separator tree* of a graph. Thus, we will use edge separators as a tool to investigate edge rankings of complete bipartite graphs. We then define the *edge-separator tree* of a graph as follows.

Definition 2.1 [2]**.** Given an edge t-ranking $\phi : E \longrightarrow \{1, 2, \cdots, t\}$ of a connected graph $G = (V, E)$, we assign a rooted tree $T(\phi)$ to it by an inductive construction as follows:

(1) If no label occurs more than once in G, then $T(\phi)$ consists of a single node r (called root) and the edge set of G is assigned to r.

(2) Otherwise, let i be the largest label assigned to more than one edge by ϕ. Then, the set of edges labeled by $\{i + 1, i + 2, \cdots, t\}$ has to be an edge separator R of G. We create a root r of $T(\phi)$ and assign R to r. (The induced subgraph of G corresponding to the subtree of T rooted at r will be G itself.) Assuming that an edge-separator tree $T_i(\phi)$ with root r_i has already been defined for each connected component G_i of the graph $G(E \setminus R)$, the children of r in $T(\phi)$ will be the nodes r_i and the subtree of $T(\phi)$ rooted at r_i will be $T_i(\phi)$.

The rooted tree $T(\phi)$ is said to be an *edge-separator tree* of G. The *height* of $T(\phi)$ is defined to be the largest number of edges assigned to nodes of $T(\phi)$ on a path from a leaf to the root. An edge-separator tree

$T(\phi)$ is called the *minimum-height edge-separator tree* of G if its height is minimum among all edge-separator trees of G.

Notice that all edges of G assigned to nodes of $T(\phi)$ on a path from a leaf to the root receive different labels, and ϕ is an optimal edge ranking of G if and only if $T(\phi)$ is a minimum-height edge-separator tree of G.

3. The Edge Ranking Problem in Complete Bipartite Graphs

Bodlaender et al. [2] applied the concept of the *edge-separator tree* to derive a formula for complete graph K_n as follows: For every positive integer n, $\chi_e(K_n) = \frac{n^2+g(n)}{3}$ where $g(1) = -1, g(2n) = g(n)$, and $g(2n+1) = g(n+1) + n$. In this section, we will apply the analogous concept to solve the edge ranking problem on complete bipartite graphs.

For every edge t-ranking of a graph G, the following property holds: if i is the largest label occurring more than once, then the edges with labels $i+1, i+2, \cdots, t$ form an edge separator of G. Moreover, making an appropriate relabeling of these labels $i+1, i+2, \cdots, t$ we get a new edge t-ranking of G with the property that there is a label $j > i$ such that all edges with labels $j, j+1, \cdots, t$ form an edge separator of G which is minimal under inclusion. Therefore, it is sufficient to consider only the *minimal edge separators* to construct the edge-separator tree.

Consider two edges e_1 and e_2 that share a common endpoint to have a path between them. By definition of edge ranking, e_1 and e_2 must be labeled by different labels. Hence the following lemma clearly holds.

Lemma 3.1. *$\chi_e(K_{1,n}) = n$, for every positive integer n.*

In the following, assume $K_{m,n}$ satisfies that $m > 1$ and $n > 1$. We then give a theorem to compute $\chi_e(K_{m,n})$. The following is a key theorem in designing our algorithm.

Theorem 3.2. *Assume that $K_{m,n}$ is a complete bipartite graph with $m > 1$ and $n > 1$. Then, $\chi_e(K_{m,n}) = \min_{1 \le m_1 < m, 1 \le n_1 < n}\{m_1 n_2 + m_2 n_1 + \max\{\chi_e(K_{m_1,n_1}), \chi_e(K_{m_2,n_2})\}\}$, where $m_1 + m_2 = m$ and $n_1 + n_2 = n$.*

Proof. Consider a complete bipartite graph $K_{m,n}$ with bipartition sets V_1 and V_2. Without loss of generality, assume that $1 < m \le n$. Let ϕ be an edge t-ranking of the graph. Let $e = (u, v)$ be an edge with label i which is

the largest label occurring more than once in ϕ. Notice that if there does not exist such a label, then $t = mn$ and we can find a better edge ranking ϕ^* with the largest label less than mn. Thus there must exist such an edge e with label i. Let X_1 and X_2 be the vertex sets that can be reached from e through edges with labels less than i, where $X_1 \subseteq V_1$, $X_2 \subseteq V_2$, $u \in X_1$, and $v \in X_2$. Let $Y_1 = V_1 \setminus X_1$ and $Y_2 = V_2 \setminus X_2$. Let $|X_1| = m_1$, $|X_2| = n_1$, $|Y_1| = m_2$, and $|Y_2| = n_2$.

Now we consider the special case that either m_2 or n_2 equals 0. Note that $m_1 \neq 0$ and $n_1 \neq 0$. Assume by contradiction that $m_2 = 0$. Then, $Y_1 = \emptyset$ and $Y_2 \neq \emptyset$. Let $x \in X_1$, $y \in Y_2$, and let $\widetilde{e} = (x, y)$. If $u = x$, then edges e and $\widetilde{e}$ share a common vertex u and hence, $\widetilde{e}$ cannot have label exactly i by definition. Suppose $u \neq x$ and $\widetilde{e}$ is labeled by i. Let $e' = (v, x)$. By definition, e' receives a label less than i. Then, the path passing through edges e, e', and $\widetilde{e}$ results in a contradiction for edge ranking ϕ. Thus, $\widetilde{e}$ cannot have label exactly i. By definition of Y_2, $\widetilde{e}$ must be labeled by an integer larger than i. Therefore, the set of edges connecting two vertices of X_1 and Y_2 must be with labels larger than i. By definition, the set of edges connecting two vertices of $X_1 \setminus \{u\}$ and X_2 must be with labels less than i. Thus edge e is assigned label i which occurs exactly once in ϕ, a contradiction occurs. Thus, $m_2 \neq 0$. The case of $n_2 \neq 0$ can be proved similarly. Therefore, $m_2 \neq 0$ and $n_2 \neq 0$.

We can see that the set of edges connecting two vertices of X_1 and Y_2 or two vertices of X_2 and Y_1 must be with labels larger than i and it is a minimal edge separator that separates edge e from other edges with label i. We can label edges in this minimal edge separator with labels j, $j + 1$, $\cdots$, t, where $j = t - (m_1 n_2 + m_2 n_1) + 1$. Therefore we have that $t = (m_1 n_2 + m_2 n_1) + \max\{\chi_e(K_{m_1,n_1}), \chi_e(K_{m_2,n_2})\}$ and the corresponding edge separator has size $m_1 n_2 + m_2 n_1$. Note that the corresponding edge separator separates $K_{m,n}$ into K_{m_1,n_1} and K_{m_2,n_2}. Every edge ranking starting with the edge separator has at least $(m_1 n_2 + m_2 n_1) + \max\{\chi_e(K_{m_1,n_1}), \chi_e(K_{m_2,n_2})\}$ labels, and there is indeed one using exactly that many labels.

We can obtain an optimal edge ranking satisfying that $m_1 \neq 0$, $m_2 \neq 0$, $n_1 \neq 0$, and $n_2 \neq 0$. Therefore, $\chi_e(K_{m,n}) = \min_{1 \leq m_1 < m, 1 \leq n_1 < n}\{m_1 n_2 + m_2 n_1 + \max\{\chi_e(K_{m_1,n_1}), \chi_e(K_{m_2,n_2})\}\}$, where $m_1 + m_2 = m$ and $n_1 + n_2 = n$. This completes the proof. ∎

Definition 3.1. Let $K_{m,n}$ be a complete bipartite graph with $m > 1$ and $n > 1$. Let R be an edge separator of $K_{m,n}$ such that it separates $K_{m,n}$

into $K_{p,q}$ and $K_{m-p,n-q}$, $|R| = p(n-q) + q(m-p)$, and $\chi_e(K_{m,n}) = |R| + \max\{\chi_e(K_{p,q}), \chi_e(K_{m-p,n-q})\}$. We call p and q the *separator points* of $K_{m,n}$.

Based on Theorem 3.2, we design an efficient dynamic programming algorithm, which runs in $O(|E|^2)$ time, for finding an optimal edge ranking of a complete bipartite graph with edge set E. The algorithm is formally presented as follows:

Algorithm EdgeRank($K_{m,n}$)
Input: A complete bipartite graph $K_{m,n} = (V_1, V_2, E)$ with $1 < (m = |V_1|) \leq (n = |V_2|)$.
Output: An optimal edge ranking ϕ.
Method:

/* Phase 1: Compute $\chi_e(K_{i,j})$, for $1 \leq i \leq m$ and $i \leq j \leq n$ */
1. $\chi_e(K_{1,j}) \leftarrow j$, for $j = 1$ to n;
2. **for** $i = 2$ to m **do**
3. **for** $j = i$ to n **do**
4. $\chi_e(K_{i,j}) \leftarrow \min_{1 \leq i_1 < i, 1 \leq j_1 < j} \{i_1 j_2 + i_2 j_1 + \max\{\chi_e(K_{i_1,j_1}), \chi_e(K_{i_2,j_2})\}\}$, where $i_1 + i_2 = i$ and $j_1 + j_2 = j$;
5. let *upp* and *low* be the separator points of $K_{i,j}$;
6. $U[i][j] \leftarrow upp$; $L[i][j] \leftarrow low$;

/* Phase 2: Construct the minimum-height edge-separator tree T_e of $K_{m,n}$ */
7. $T_e \leftarrow \emptyset$;
8. call MakeTree($\emptyset, K_{m,n}$) to construct the minimum-height edge-separator tree of $K_{m,n}$; /* Note that procedure MakeTree will be presented after the algorithm */

/* Phase 3: Rank the edges of $K_{m,n}$ */
9. let $(r_1, r_2, \cdots, r_k)$ be the postorder sequence of nodes in T_e;
10. **for** $i = 1$ to k **do**
11. let R_i be the set of edges assigned to node r_i;
12. **if** r_i is a leaf, **then** label edges in R_i from 1 to $|R_i|$ arbitrarily;
13. **else**
14. let t_1 and t_2 be the largest labels of edges assigned to r_1 and r_2, respectively, where r_1 and r_2 are the children of node r_i in T_e;
15. $t \leftarrow \max\{t_1, t_2\} + 1$;
16. label edges in R_i from t to $t + |R_i| - 1$ arbitrarily;
17. **output** the labels of all edges in $K_{m,n}$, and **stop.**

We will present Procedure MakeTree to construct the minimum-height edge-separator tree T_e in the following.

Procedure MakeTree($r, K_{i,j}$)

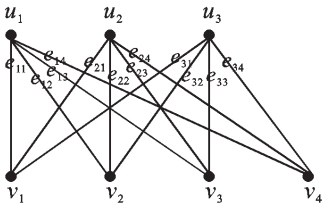
Input: Either ($\emptyset$ and $K_{m,n}$) or (an internal node r of T_e and a subgraph $K_{i,j}$ of $K_{m,n}$).

Process: Construct a minimum-height edge-separator tree T_e recursively.

Method:

1. create a node γ of T_e;
2. let R be the edge set of $K_{i,j}$ if $i = 1$ or $j = 1$; otherwise, let R be the edge separator of $K_{i,j}$ that separates $K_{i,j}$ into $K_{U[i][j],L[i][j]}$ and $K_{i-U[i][j],j-L[i][j]}$;
3. assign R to node γ;
4. **if** $r = \emptyset$, **then** let γ be the root of T_e; **else** let γ be the child of node r in T_e;
5. **if** $i \neq 1$ and $j \neq 1$, **then**
6. call MakeTree($\gamma, K_{U[i][j],L[i][j]}$); call MakeTree($\gamma, K_{i-U[i][j],j-L[i][j]}$);

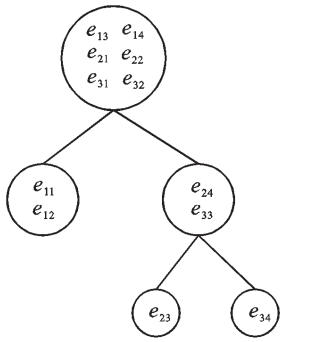
We give an example to illustrate Algorithm EdgeRank as follows. Consider the complete bipartite graph $K_{3,4} = (V_1, V_2, E)$ shown in Figure 1(a), where $V_1 = \{u_1, u_2, u_3\}$, $V_2 = \{v_1, v_2, v_3, v_4\}$, and the edge e_{ij} in E is to connect u_i with v_j. Phase 1 of Algorithm EdgeRank($K_{3,4}$) computes $\chi_e(K_{3,4})$ as follows: Initially, $\chi_e(K_{1,j}) = j$ for $1 \leq j \leq 4$. Then, it computes $\chi_e(K_{i,j})$ for $2 \leq i \leq 3$ and $i \leq j \leq 3$. Assume that $\chi_e(K_{i,j})$'s for $i \neq 3$ and $j \neq 4$ are computed. It computes $\chi_e(K_{3,4})$ below. Let R be the edge separator of $K_{3,4}$ that separates $K_{3,4}$ into $K_{1,2}$ and $K_{2,2}$, where $X_1 = \{u_1\}$, $X_2 = \{v_1, v_2\}$, $Y_1 = \{u_2, u_3\}$, and $Y_2 = \{v_3, v_4\}$. Then, $R = \{e_{13}, e_{14}, e_{21}, e_{22}, e_{31}, e_{32}\}$ and $|R| = 6$. The edge ranking starting with R uses exactly $|R| + \max\{\chi_e(K_{1,2}), \chi_e(K_{2,2})\} = 6 + 3 = 9$ labels. After considering all possible edge separators of $K_{3,4}$, it computes $\chi_e(K_{3,4}) = 9$. Then the separator points of $K_{3,4}$ are 1 and 2. During the computation of $\chi_e(K_{i,j})$, Algorithm EdgeRank records the separator points of $K_{i,j}$. The edge-ranking numbers of $K_{i,j}$ for $1 \leq i \leq 3$ and $i \leq j \leq 4$ and the separator points of $K_{i,j}$ are shown in Figure 1(b). In Phase 2, it constructs the minimum-height edge-separator tree T_e shown in Figure 1(c). Phase 3 labels the edges assigned to nodes of T_e in a postorder sequence and gets an optimal edge ranking of $K_{3,4}$ which is shown in Figure 1(d).



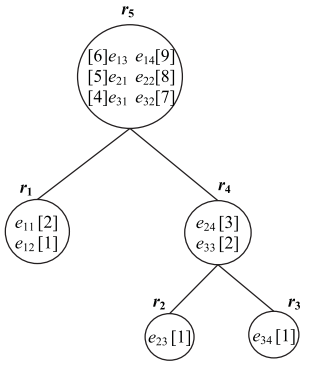
(a)

$\chi_e(K_{i,j})$ (upp_i, low_j) i \ j	1	2	3	4
1	1	2	3	4
2		3 (1, 1)	5 (1, 1)	6 (1, 2)
3			7 (1, 2)	9 (1, 2)

(b)



(c)



(d)

Figure 1. An optimal edge ranking of $K_{3,4}$, where (a) $K_{3,4}$, (b) $\chi_e(K_{i,j})$ for $1 \leq i \leq 3$ and $i \leq j \leq 4$, (c) the minimum-height edge-separator tree T_e of $K_{3,4}$, and (d) an optimal edge ranking of $K_{3,4}$. Note that the data in each cell of (b) represents the triple of $\chi_e(K_{i,j})$, upp_i, and low_j, where upp_i and low_j are the separator points of $K_{i,j}$.

Based on Theorem 3.2, Phase 1 of Algorithm EdgeRank($K_{m,n}$) computes $\chi_e(K_{m,n})$. During the computation of $\chi_e(K_{i,j})$ for $2 \leq i \leq m$ and $i \leq j \leq n$, it also puts down the separator points of $K_{i,j}$. Using the information produced by Phase 1, Phases 2 and 3 of Algorithm EdgeRank($K_{m,n}$) rank the edges of $K_{m,n}$ using exactly $\chi_e(K_{m,n})$ labels. Hence, Algorithm EdgeRank($K_{m,n}$) obtains an optimal edge ranking of $K_{m,n}$. Now we anaylze the time complexity of the algorithm. When we compute $\chi_e(K_{i,j})$, $\chi_e(K_{p,q})$'s for $1 \leq p < i$ and $1 \leq q < j$, need to be computed. Note that we use a 2-dimension array to store $\chi_e(K_{i,j})$, for $1 \leq i \leq m$ and $i \leq j \leq n$. We only compute and use the upper triangular part of the array since $\chi_e(K_{i,j}) = \chi_e(K_{j,i})$. Thus before computing $\chi_e(K_{i,j})$, $\chi_e(K_{p,q})$'s for $1 \leq p < i$ and $i \leq q < j$, have been computed. It is easy to see that the time complexity of computing $\chi_e(K_{i,j})$ will be $O(ij)$. Therefore, computing $\chi_e(K_{i,j})$'s, for

$2 \leq i \leq m$ and $i \leq j \leq n$, can be done in $O(m^2n^2)$ time. On the other hand, the separator points of $K_{i,j}$'s for $2 \leq i \leq m$ and $i \leq j \leq n$ can be found in the same time. Hence, Phase 1 of Algorithm EdgeRank runs in $O(m^2n^2)$ time. We can easily see that Phases 2 and 3 of Algorithm EdgeRank can be done in $O(mn)$ time. Hence, we conclude the following theorem:

Theorem 3.3. *Given a complete bipartite graph $K_{m,n}$, Algorithm EdgeRank solves the edge ranking problem in $O(m^2n^2)$ time.*

4. Concluding Remarks

We propose an $O(|E|^2)$-time algorithm for finding an optimal edge ranking of a complete bipartite graph $K_{m,n}$, where $|E| = mn$. Our algorithm uses the concept of edge-separator trees and is a dynamic programming algorithm. It is interesting to know whether the approach used in this paper can be applied to solve the edge ranking problem on other classes of graphs, such as cographs.

For a complete bipartite graph $K_{n,n}$ with $n = 2^k$ and $k \geq 1$, we conjecture that $\chi_e(K_{n,n}) = \frac{2n^2+1}{3}$; that is, the separator points of $K_{n,n}$ are $\frac{n}{2}$. We have made many simulations on $K_{n,n}$, for $1 \leq n \leq 1000$, to support this conjecture. But, we have not yet succeeded in proving the above conjecture. We would like to post it as an open problem to interested readers.

Acknowledgements

The author would like to thank one anonymous referee for many useful comments and suggestions which have improved the presentation of this paper.

References

[1] B. Aspvall and P. Heggernes, *Finding minimum height elimination trees for interval graphs in polynomial time*, BIT **34** (1994) 484–509.

[2] H.L. Bodlaender, J.S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller and Z. Tuza, *Rankings of graphs*, SIAM J. Discrete Math. **11** (1998) 168–181.

[3] C.C. Chou, C.M. Fu and W.C. Huang, *Decomposition of $K_{m,n}$ into short cycles*, Discrete Math. **197/198** (1999) 195–203.

[4] D.G. Corneil, H. Lerchs and L.K. Stewart, *Complement reducible graphs*, Discrete Appl. Math. **3** (1981) 163–174.

[5] P. De La Torre, R. Greenlaw and A.A. Schaffer, *Optimal edge ranking of trees in polynomial time*, Algorithmica **13** (1995) 592–618.

[6] J.S. Deogun, T. Kloks, D. Kratsch and H. Müller, *On vertex ranking for permutation and other graphs*, Lecture Notes in Comput. Sci. **775** (Springer-Verlag, 1994) 747–758.

[7] J.S. Deogun, T. Kloks, D. Kratsch and H. Müller, *On vertex ranking for trapezoid, circular-arc and other graphs*, Discrete Appl. Math. **98** (1999) 39–63.

[8] P.E. Haxell, *Partitioning complete bipartite graphs by monochromatic cycles*, J. Combin. Theory (B) **69** (1997) 210–218.

[9] A.A. Ivanov, R.A. Liebler, T. Penttila and C.E. Praeger, *Antipodal distance-transitive covers of complete bipartite graphs*, Europ. J. Combinatorics **18** (1997) 11–33.

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

[11] T. Kloks, H. Müller and C.K. Wong, *Vertex ranking of asteroidal triple-free graphs*, Inform. Process. Lett. **68** (1989) 201–206.

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

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

[14] C.E. Leiserson, *Area efficient graph layouts for VLSI*, in: Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science FOCS'80 (1980) 270–281.

[15] C.M. Liu and M.S. Yu, *An optimal parallel algorithm for node ranking of cographs*, Discrete Appl. Math. **87** (1998) 187–201.

[16] D.C. Llewellyn, C. Tovey and M. Trick, *Local optimization on graphs*, Discrete Appl. Math. **23** (1989) 157–178.

[17] A. Pothen, The complexity of optimal elimination trees, Technical Report CS-88–13 (the Pennsylvania State University, 1988).

[18] A.A. Schäffer, *Optimal node ranking of trees in linear time*, Inform. Process. Lett. **33** (1989/1990) 91–96.

[19] P. Scheffler, *Node ranking and searching on graphs* (*Abstract*), in: U. Faigle and C. Hoede (eds.), 3rd Twente Workshop on Graphs and Combinatorial Optimization, Memorandum No. 1132, Faculty of Applied Mathematics, University of Twente, The Netherlands, 1993.

[20] J.D. Ullman, Computational aspects of VLSI (Computer Science Press, Rockville, MD, 1984).

[21] X. Zhou and T. Nishizeki, *Finding optimal edge rankings of trees*, in: Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms SODA'95 (1995) 122–131.

Received 17 June 2005
Revised 20 October 2005