Approximating treewidth and pathwidth of some classes of perfect graphs

T. Kloks, H. Bodlaender

RUU-CS-92-29
September 1992

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Approximating treewidth and pathwidth of some classes of perfect graphs

T. Kloks, H. Bodlaender

Technical Report RUU-CS-92-29
September 1992

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0024-3275

Approximating Treewidth and Pathwidth of some Classes of Perfect Graphs

T. Kloks * H. Bodlaender †
Department of Computer Science, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

In this paper we discuss the problem of finding approximations for the treewidth and pathwidth of cotriangulated graphs, convex graphs, permutation graphs and of cocomparability graphs. For a cotriangulated graph, of which the treewidth is at most k, the pathwidth is at most $3k+4$ and we show there exists an $O(n^2)$ algorithm finding a path-decomposition with width at most $3k+4$. For convex graphs with treewidth at most k, the pathwidth is at most $2k+1$ and we give a $O(nk)$ algorithm which computes a path-decomposition with width at most $2k+1$. If $G[\pi]$ is a permutation graph with treewidth k, then we show that the pathwidth of $G[\pi]$ is at most $2k$, and we give an algorithm which constructs a path-decomposition with width at most $2k$ in time $O(nk)$. We assume that the permutation π is given. In this paper we also discuss the problem of finding an approximation for the treewidth and pathwidth of cocomparability graphs. We show that, if the treewidth of a cocomparability graph is at most k, then the pathwidth is at most $O(k^2)$, and we give a simple algorithm finding a path-decomposition with this width. The running time of the algorithm is dominated by a coloring algorithm of the graph. Such a coloring can be found in time $O(n^3)$.

If the treewidth is bounded by some constant, these results, together with previous results, [11, 31], show that, once the approximations are given, the exact treewidth and pathwidth can be computed in linear time for all these graphs.

1 Introduction

In many recent investigations in computer science, the notions of treewidth and pathwidth play an increasingly important role. One reason for this is that many

*This author is supported by the foundation for Computer Science (S.I.O.N) of the Netherlands Organization for Scientific Research (N.W.O.), Email: ton@cs.ruu.nl.

†Email:hansb@cs.ruu.nl.

problems, including many well studied NP-complete graph problems, become solvable in polynomial and usually even linear time, when restricted to the class of graphs with bounded tree- or pathwidth, [1, 3, 5, 7, 8, 32]. Of crucial importance for these algorithms is, that a tree-decomposition or path-decomposition of the graph is given in advance. Much research has been done in finding a tree-decomposition with a reasonable small treewidth. Recent results [36] show that an $O(n \log n)$ algorithm exists to find a suitable tree-decomposition for a graph with bounded treewidth. However, the constant hidden in the 'big oh', is exponential in the treewidth, limiting the practicality of this algorithm.

For many *special classes* of graphs, it has been shown that the treewidth can be computed efficiently. In this paper we discuss the problem of finding approximate tree- and path-decompositions for cotriangulated graphs, convex graphs, permutation graphs and for cocomparability graphs. We also show that for these graphs, if the treewidth is at most k, then the pathwidth is bounded by some polynomial in k.

It has been shown that computing the pathwidth of a triangulated graph is NP-complete [27]. It is unknown whether there exist good approximations, which can be computed efficiently, for the pathwidth of a triangulated graph. Surprisingly, for cotriangulated graphs, the pathwidth and treewidth are related in a linear fashion, and there is a very simple algorithm that computes an approximate path-decomposition.

Permutation graphs have a large number of applications in scheduling problems. See for example [21], where permutation graphs are used to describe a problem concerning the memory requirements of a number of programs at a certain time (see also [26]). Permutation graphs also arise in a natural way, in the problem of sorting a permutation, using queues in parallel. In [26] it is shown that this problem is closely related with the coloring problem of permutation graphs.

We show that the pathwidth of a permutation graph is at most two times the treewidth of that graph, and we give a linear time algorithm which produces a path-decomposition which is at most two times off the optimal one. If the treewidth of the permutation graph is bounded by a constant, this result, together with earlier results show that an optimal tree- and path-decomposition can be computed in linear time. In a forthcoming paper, [10], we show that for permutation graphs, the treewidth and pathwidth are in fact equal, and we show an $O(nk^2)$ algorithm which computes the (exact) treewidth (given the permutation).

It is easy to see that computing the treewidth and pathwidth for bipartite graphs is NP-complete. Indeed, finding an approximation algorithm with a certain performance ratio for bipartite graphs is at least as difficult as finding an approximation with the same performance for graphs in general. In this paper we show that, if the bipartite graph is *convex*, with treewidth k, the pathwidth is at most $2k + 1$ and it is almost straightforward to find a path-decomposition with width $2k + 1$ in $O(nk)$ time. In a forthcoming paper, [30], we show that computing the treewidth of chordal bipartite graphs is polynomial. As the class of convex graphs is properly

contained in the class of chordal bipartite graphs, this shows that the exact treewidth of convex graphs can be computed in polynomial time. However, the running time of this algorithm is not very good ($O(e^3)$, where e is the number of edges), and we think that, especially for practical applications, the algorithm described in this paper, is of importance. Furthermore, as far as we know, computing the exact pathwidth is still an open problem.

Comparability graphs and their complements have a large number of applications (see e.g. [26, 39]). These graph classes, as well as those of the triangulated and of the cotriangulated graphs, are among the largest and most important classes of perfect graphs. Cocomparability graphs properly contain many other well known classes of perfect graphs, like interval graphs, cographs (or P_4-free graphs), permutation graphs, indifference graphs, complements of superperfect graphs etc. (see e.g. [26] or [6, pages 67–96]). Cocomparability graphs are exactly the intersection graphs of continuous functions $F_i : (0,1) \rightarrow R$ (see [40]). Comparability graphs can be recognized in polynomial time [35, 26, 25, 41]. Given a comparability graph, a transitive orientation of the edges can be found in $O(n^2)$ time [41].

We show that, if the treewidth of a cocomparability graph G is at most k, then the pathwidth is at most $O(k^2)$, and we give a simple algorithm which finds a path-decomposition with this width. Our algorithm uses a transitive orientation of the complement $\overline{G}$, and the *height function* which can be obtained from this orientation in linear time. The algorithm also uses a vertex coloring of the graph G. This vertex coloring, or clique cover in the complement, can be found in $O(n^3)$ time using a flow algorithm, as described in [26].

2 Preliminaries

In this section we start with some definitions and easy lemmas. For more information on special perfect graph classes the reader is referred to [26, 14].

One of the oldest classes of graphs known to be perfect are the triangulated graphs.

Definition 2.1 *A graph is called* triangulated *if it has no induced chordless cycle of length at least four. The complement of a triangulated graph is called* cotriangulated.

Triangulated graphs are also called chordal. There are some nice characterizations of triangulated graphs. For example, a graph is triangulated if and only if every minimal vertex separator induces a clique [20] or, a graph is triangulated if and only if it has a perfect elimination scheme [23]. Triangulated graphs are *perfect*, i.e. for every induced subgraph the chromatic number is equal to the maximum number of vertices in a clique [28]. They can be recognized in linear time, and there exist fast algorithms for many NP-complete problems (see e.g. [26]). The perfect graph theorem [33], states that a graph is perfect if and only if the complement of the graph is perfect, hence also cotriangulated graphs are perfect.

Another class of perfect graphs is the class of convex graphs (in fact all bipartite graphs are perfect).

Definition 2.2 *Let $G = (X, Y, E)$ be a bipartite graph. An ordering of X has the* adjacency property, *if for each $y \in Y$, the neighbors of y in X are consecutive in the ordering of X.*

Definition 2.3 *A bipartite graph $G = (X, Y, E)$ is* convex, *if there is an ordering of X or of Y with the adjacency property.*

A bipartite graph $G = (X, Y, E)$ is *biconvex* if there is an ordering of X *and* Y with the adjacency property. Convex graphs contain the bipartite permutation graphs. Notice that convex graphs can be recognized in linear time, using for example the PQ-tree algorithms of Booth and Lueker [13] (see also [14]).

Definition 2.4 *An undirected graph $G = (V, E)$ is called a* comparability graph, *or transitively orientable graph, if there exists an orientation (V, F) of the edges satisfying:*

$$F \cap F^{-1} = \emptyset \wedge F + F^{-1} = E \wedge F^2 \subseteq F$$

where $F^2 = \{(a, c) | \exists_{b \in V} (a, b) \in F \wedge (b, c) \in F\}$. Such an orientation is called a transitive *orientation. A* cocomparability graph *is the complement of a comparability graph.*

If a graph G is a comparability graph, then this holds for every induced subgraph of G. There exists a complete list of critical non-comparability graphs [24]. Comparability graphs, as well as triangulated graphs, are also perfect and can be recognized in polynomial time [41]. By the perfect graph theorem, also cocomparability graphs are perfect. For our algorithm we shall need the concept of a height function defined in [26]. Let F be an acyclic orientation of an undirected graph $G = (V, E)$. A *height function* h assigns a non-negative integer to each vertex as follows: $h(v) = 0$ if v is a sink; otherwise, $h(v) = 1 + \max\{h(w) \mid (v, w) \in F\}$. In other words, $h(v)$ is the maximal length of a path from v to a sink. A height function can be assigned in linear time [26], and represents a proper vertex coloring of G. If F is a transitive orientation, the coloring by h is optimal (i.e. uses the least possible number of colors) [26].

We also need a coloring of the cocomparability graph. In [26] a method is described to find an optimal coloring of a cocomparability graph by using a minimum flow algorithm. It follows that this coloring can be found in $O(n^3)$ time.

Cocomparability graphs are intersection graphs [40].

Lemma 2.1 *A graph G with n vertices is a cocomparability graph if and only if G is the intersection graph of n continuous functions $F_i : (0, 1) \rightarrow R$.*

We only use this lemma implicitly.

We think of a permutation π of the numbers $1, \ldots, n$, as the sequence $\pi = [\pi_1, \ldots, \pi_n]$. We use the notation π_i^{-1} for the position of the number i in this sequence.

Definition 2.5 *If π is a permutation of the numbers $1, \ldots, n$, we can construct an undirected graph $G[\pi] = (V, E)$ with vertex set $V = \{1, \ldots, n\}$, and edge set E:*

$$(i, j) \in E \Leftrightarrow (i - j)(\pi_i^{-1} - \pi_j^{-1}) < 0$$

An undirected graph is called a permutation graph *if there exists a permutation π such that $G \cong G[\pi]$.*

Notice that we can obtain the complement of $G[\pi]$, by reversing the sequence π. Hence the complement of a permutation graph is also a permutation graph. It is also easy to see that a permutation graph is a comparability graph. Pnueli, Lempel and Even ([35]) showed that a graph G is a permutation graph if and only if both G and $\overline{G}$ are comparability graphs. It follows that permutation graphs are *perfect*. They can be recognized in time $O(n^2)$ (see [41]). There exist fast algorithms for many NP-complete problems like CLIQUE, INDEPENDENT SET, FEEDBACK VERTEX SET and DOMINATING SET when restricted to permutation graphs [26, 22, 16, 15].

In this paper we assume that the permutation π is given, and we show some results on the pathwidth and treewidth of $G[\pi]$, which is sometimes called the *inversion graph* of π. If the permutation π is *not* given, transitive orientations of G and $\overline{G}$ can be computed in $O(n^2)$ time [41]. Given these orientations, a permutation can be computed in $O(n^2)$ time [26].

A permutation graph $G[\pi]$ is an intersection graph, which is illustrated by the *matching diagram* of π [26].

Definition 2.6 *Let π be a permutation of $1, \ldots, n$. The* matching diagram *of π can be obtained as follows. Write the number $1, \ldots, n$ horizontally from left to right. Underneath, write the numbers $\pi_1, \ldots, \pi_n$, also horizontally from left to right. Draw n straight line segments joining the two 1's, the two 2's, etc.*

An example of a matching diagram of a permutation graph $G[\pi]$ is illustrated in figure 1 (page 6). Notice that two vertices i and j of $G[\pi]$ are adjacent if and only if the corresponding line segments in the matching diagram of π intersect. Matching diagrams are often useful in visualizing certain concepts.

Definition 2.7 *A* tree-decomposition *of a graph $G = (V, E)$ is a pair $D = (S, T)$ with $T = (I, F)$ a tree and $S = \{X_i \mid i \in I\}$ a collection of subsets of vertices of G, one subset for each node in T, such that the following three conditions are satisfied:*

1. $\bigcup_{i \in I} X_i = V$.

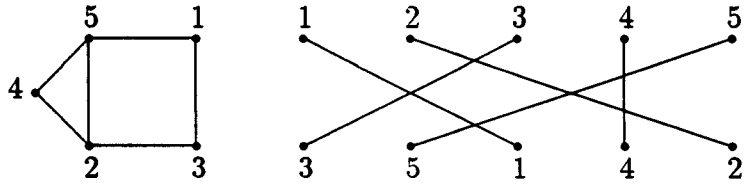

Figure 1: permutation graph and matching diagram

2. *For all edges* $(v, w) \in E$ *there is a subset* $X_i \in S$*, such that both* v *and* w *are contained in* X_i*.*

3. *For each vertex* x*, the set* $\{i \in I \mid x \in X_i\}$ *forms a connected subtree of* T*.*

A path-decomposition *of a graph* G *is a tree-decomposition* (S, T) *such that* T *is a path. We also use the notation* $(X_1, X_2, \ldots)$ *for a path-decomposition. The* width *of a tree-decomposition* (S, T)*, with* $S = \{X_i \mid i \in I\}$*, is* $\max_{i \in I} (|X_i| - 1)$*.*

Definition 2.8 *The* treewidth *of* G *is the minimum width over all tree-decompositions of* G*. The* pathwidth *of* G *is the minimum width over all path-decompositions of* G*.*

An alternative way to define the class of graphs with treewidth at most k is by means of *partial* k*-trees*. A k-tree is defined recursively as follows: A clique with $k+1$ vertices is a k-tree. Given a k-tree T_n with n vertices, a k-tree with $n+1$ vertices is constructed by making a new vertex x_{n+1} adjacent to a k-clique of T_n and nonadjacent to the $n-k$ other vertices of T_n. A *partial* k*-tree* is a subgraph of a k-tree. Notice that k-trees are triangulated, and have maximum clique size $k+1$. It can be shown that the class of graphs with treewidth at most k is exactly the class of partial k-trees (see e.g. [32]). There exist linear time algorithms for many NP-complete problems, when restricted to the class of partial k-trees for some *constant* k and when a tree-decomposition with bounded width is given [1, 5, 8, 3]. There are also results stating that large *classes* of problems can be solved in linear time, when a tree-decomposition with bounded width is given [17, 18, 19].

Determining whether the treewidth or pathwidth of a given graph is at most a given integer k is NP-complete [2]. In view of this, the results of Robertson and Seymour on minor closed classes of graph are of great interest. Robertson and Seymour proved that every minor closed class of graphs is recognizable in $O(n^3)$ time [37]. Since for every fixed k, the class of graphs with treewidth (pathwidth) at most k is minor closed, it follows that for every constant k there is a polynomial algorithm that recognizes graphs with treewidth (pathwidth) at most k. In fact, for these minor closed classes faster algorithms exist. We list some of the results.

1. For $k = 2, 3$ there is a linear time algorithm for the treewidth problem using rewrite rules [4, 34].

2. For fixed $k \geq 4$ an $O(n \log n)$ algorithm exists which constructs a tree-decomposition with width k [36, 11, 31].

3. If an (approximate) tree-decomposition with bounded width is given, the exact treewidth, and a corresponding tree-decomposition, can be computed in linear time [11]. In this case, when also the pathwidth is bounded, also the pathwidth and an optimal path-decomposition can be computed in linear time.

For an introductory overview of recent results dealing with treewidth and pathwidth, the reader is referred to [7].

3 An approximate path-decomposition for cotriangulated graphs

For triangulated graphs the treewidth is equal to the maximum clique size minus one. It follows that, for triangulated graphs, the treewidth can be computed in linear time. Computing the pathwidth of a triangulated graph is an NP-complete problem, as shown in [27].

It is unknown if the treewidth or pathwidth of cotriangulated graphs can be computed efficiently. We show how to find good approximations for the treewidth and pathwidth of cotriangulated graphs. Let G be a cotriangulated graph with treewidth at most k. We show that the pathwidth of G is at most $3k + 4$ and there exists an $O(n^2)$ algorithm that finds a path-decomposition with this width. The following lemma is easily checked. For a similar result see e.g. [38].

Lemma 3.1 *Let $G = (V, E)$ be a triangulated graph with n vertices. There is a clique C, such that every component of $G[V - C]$ has at most $\lceil \frac{1}{2}(n - |C|) \rceil$ vertices.*

The following lemma will also be useful (see also e.g. [12]).

Lemma 3.2 *For the complete bipartite graph $G = K(m, n)$, the treewidth is* $\min(m, n)$.

Proof. Assume $m \leq n$. Let V be the set of vertices of G. Let V_1 be the independent set with n vertices and let $V_2 = V \setminus V_1$. If we add all edges between vertices in V_2 (making a clique of V_2), then we obtain a m-tree. Thus the treewidth of G is at most m. Since K_{m+1} is a minor of G, the treewidth of G is at least m. □

Let C be a clique in $\overline{G}$ as mentioned in the lemma 3.1. The vertices of $V \setminus C$ can be partitioned in two set A and B such that no vertex of A is adjacent to a vertex of B, and both A and B have at most $\lceil \frac{2}{3}(n - |C|) \rceil$ vertices. Notice that the subgraph

induced by A and B has a complete bipartite subgraph in G, since every vertex of A is adjacent to every vertex of B. Since the treewidth of G is at most k and by lemma 3.2:

$$\lfloor \frac{1}{3}(n - |C|) \rfloor \leq \min(|A|, |B|) \leq k$$

Hence $|A \cup B| \leq 3(k+1)$. We can triangulate G by adding edges such that $A \cup B$ becomes a clique. Since C is a stable set in G, the result is a splitgraph. For a splitgraph, the following lemma is easily checked.

Lemma 3.3 *Let G be a splitgraph with clique number c. If G is an interval graph, the pathwidth is $c-1$, otherwise, the pathwidth is c.*

This proves the following theorem.

Theorem 3.1 *If G is a cotriangulated graph with treewidth at most k, then the pathwidth is at most $3k+4$ and there exists an $O(n^2)$ algorithm which produces a path-decomposition with this width.*

As a consequence, we have $O(n^2)$ approximation algorithms for the treewidth and for the pathwidth of cotriangulated graphs with performance ratio $3+\varepsilon$ for all $\varepsilon > 0$.

4 An approximate path-decomposition for convex graphs

Let $G = (X, Y, E)$ be a convex graph, with $|X| = m$, and assume the vertices of X have been ordered $1, 2, \ldots, m$ such that this ordering fulfills the adjacency property (i.e. for each $y \in Y$ the neighbors in X are consecutive). Let k be some integer. In this section we describe an $O(nk)$ algorithm which produces a path-decomposition with width at most $2k+1$ or shows that the treewidth of G is larger than k.

Consider the case where $k \geq m$. If we add edges such that X becomes a clique, then the result is a splitgraph with cliquesize at most $k+1$. Hence, according to lemma 3.3, we find a path-decomposition with width at most $k+1$. Hence we may assume without loss of generality that $k \leq m-1$. Let Y' be the set of vertices of Y with degree at least $k+1$. Consider the subgraph G', induced by vertices of X and the vertices of Y'. We construct a path-decomposition for G' as follows. For $i = 1, \ldots, m-k$ let Z_i be the subset with:

$$\begin{aligned} Z_i \cap X &= \{i, i+1, \ldots, i+k\} \\ Z_i \cap Y &= \{y \in Y \mid Z_i \cap X \subseteq N(y)\} \end{aligned}$$

Where $N(y)$ is the neighborhood of $y \in Y$. Notice that $Z_i \cup Y \subseteq Y'$.

Lemma 4.1 *Assume $k \leq m-1$. If the treewidth of G' is k then the pathwidth of G' is at most $2k$ and $(Z_1, \ldots, Z_{m-k})$ is a path-decomposition for G' with width at most $2k$.*

Proof. Notice that each Z_i has $k+1$ vertices of X. Each vertex $y \in Y'$ which is in Z_i is adjacent to all vertices of $Z_i \cap X$. This show there can be at most k vertices of Y in Z_i, otherwise G' has a complete bipartite subgraph $K(k+1, k+1)$, which is forbidden by lemma 3.2. Obviously, $(Z_1, \ldots, Z_{m-k})$ is indeed a path-decomposition. □

We now show how to extend this path-decomposition to a path-decomposition for G. Consider a vertex $y \in Y$, with at most k neighbors. Notice that there exists a subset Z_i containing $N(y)$. Take such a subset Z_i containing $N(y)$ with at most $2k+1$ elements. Make a new subset $Z_{i'} = Z_i \cup \{y\}$, and make a new path-decomposition $(Z_1, \ldots, Z_i, Z_{i'}, Z_{i+1}, \ldots, Z_{m-k})$. This clearly is a path-decomposition for the subgraph of G induced by the vertices $X \cup Y' \cup \{y\}$. By induction the following lemma follows.

Lemma 4.2 *Let $G = (X, Y, E)$ be convex with treewidth k. Then the pathwidth of G is at most $2k+1$.*

We now show that the algorithm described above can be implemented to run in $O(nk)$ time. We assume that the vertices of X are ordered $1, \ldots, m$ such that this ordering has the adjacency property. Assume $Y = \{y_1, \ldots, y_t\}$.

Step 1 First assume that $k \geq m$. Then make a subset $Z_y = X \cup \{y\}$ for each vertex $y \in Y$. The sequence $(Z_{y_1}, Z_{y_2}, \ldots, Z_{y_t})$, (for *any* ordering $y_1, \ldots, y_t$ of the vertices of Y) is a correct path-decomposition. Stop.

Step 2 Assume $k \leq m-1$. Check if the number of edges does not exceed $nk - \frac{1}{2}k(k+1)$. If it does, then stop; the treewidth of G is larger than k.

Step 3 Otherwise, for each $y \in Y$, determine the maximum and minimum neighbor in X, say $\min(y)$ and $\max(y)$. If a vertex y has degree 0 then we set $\min(y) = 0$ and $\max(y) = -1$. The result of this step is that the set of neighbors of $y \in Y$, with degree at least one, is $\{x \in X \mid \min(y) \leq x \leq \max(y)\}$.

Step 4 Calculate for each vertex $y \in Y$ the degree, $\max(y) - \min(y) + 1$, and make a list Y' of vertices of degree at least $k+1$.

Step 5 Initialize subsets $Z_i = \{i, \ldots, \min(m, i+k)\}$, for $i = 1, \ldots, m$. For isolated vertices of Y, we initialize $Z_0 = \emptyset$.

Step 6 For each $y \in Y'$, put y in the subsets Z_i, for $i = \min(y), \ldots, \max(y) - k$. If one of these subsets, say Z_i, gets more than $2k+1$ vertices, then stop; the treewidth of G exceeds k. The vertices of Z_i induce a $K(k+1, k+1)$.

Step 7 For each $y \in Y \setminus Y'$, make a subset $Z'_y = Z_{\min(y)} \cup \{y\}$.

Step 8 For $i = 0, \ldots, m$, let $y_1^i, y_2^i, \ldots$ be the vertices $y \in Y \setminus Y'$ with $\min(y) = i$. Then the path-decomposition is $(Z_0, Z'_{y_1^0}, Z'_{y_2^0}, \ldots, Z_1, Z'_{y_1^1}, Z'_{y_2^1}, \ldots, Z_2, \ldots)$.

The discussion above proves the following theorem.

Theorem 4.1 *Let G be a convex graph and let k be an integer. There exists an $O(nk)$ algorithm which either determines that the treewidth exceeds k, or produces a path-decomposition of G with width at most $2k+1$.*

5 An approximate path-decomposition for permutation graphs

In this section, let $G[\pi]$ be a permutation graph with n vertices and with treewidth k. We show there exists a path-decomposition of width at most $2k$, and we give a linear time algorithm to compute this. The algorithm outputs a set X_i for $1 \leq i \leq n$. A vertex j is put in all sets X_k with $\pi_j^{-1} \leq k < j$ or $j \leq k < \pi_j^{-1}$. The precise algorithm is given below.

Procedure Pathdec (input π; output X)
for $i \leftarrow 1$ **to** n **do** $X_i \leftarrow \emptyset$
for $j \leftarrow 1$ **to** n **do**
  if $\pi_j^{-1} = j$ **then** $X_j \leftarrow X_j \cup \{j\}$
  if $\pi_j^{-1} > j$ **then**
    for $k \leftarrow j$ **to** $\pi_j^{-1} - 1$ **do** $X_k \leftarrow X_k \cup \{j\}$
  if $\pi_j^{-1} < j$ **then**
    for $k \leftarrow \pi_j^{-1}$ **to** $j - 1$ **do** $X_k \leftarrow X_k \cup \{j\}$

The next lemma shows that the constructed sets form indeed a path-decomposition.

Lemma 5.1 *Let $S = \{X_i \mid 1 \leq i \leq n\}$ be the subsets of vertices constructed by the algorithm. Let $P = (1, \ldots, n)$ be the path with n vertices. Then (S, P) is a path-decomposition for the permutation graph $G[\pi]$.*

Proof. We first show that each vertex is in at least one subset of S. Consider a vertex i. If $\pi_i^{-1} \geq i$ then i is in the subset X_i. If $\pi_i^{-1} < i$ then i is in the subset X_{i-1}. Notice that the subsets containing i are consecutive. The only thing left to show is that every edge is in at least one subset. Consider again a vertex i and let j be a neighbor of i. Assume without loss of generality that $i < j$. In the matching diagram, the line segment corresponding with j must intersect the line segment of i. Since $i < j$, this implies that $\pi_j^{-1} < \pi_i^{-1}$. We consider the different orderings of i, j, π_i^{-1} and π_j^{-1}. If $i < j \leq \pi_j^{-1} < \pi_i^{-1}$, then both i and j are contained in the subset X_j. If $i \leq \pi_j^{-1} \leq j \leq \pi_i^{-1}$ then both are contained in $X_{\pi_j^{-1}}$. If $i \leq \pi_j^{-1} < \pi_i^{-1} \leq j$, then both are contained in $X_{\pi_j^{-1}}$. If $\pi_j^{-1} \leq i \leq \pi_i^{-1} \leq j$, then both are contained in X_i. Finally, if $\pi_j^{-1} < \pi_i^{-1} \leq i < j$, then both i and j must be in $X_{\pi_i^{-1}}$. □

We now show that the width of this path-decomposition is at most $2k$.

Lemma 5.2 *Each subset produced by the algorithm has at most $2k+1$ elements.*

Proof. Consider a subset X_i. Notice that $X_i \subset S_1 \cup S_2 \cup \{i\}$ where S_1 and S_2 are defined by: $S_1 = \{j \mid j \leq i < \pi_j^{-1}\}$ and $S_2 = \{j \mid \pi_j^{-1} \leq i < j\}$. Note that, as π is a permutation, there must be as many lines in the matching diagram with their upper point left of i and their lower point right of i, as lines with their upper point right of i and their lower point left of i. Hence $|S_1| = |S_2|$. Every vertex in S_1 is adjacent to every vertex in S_2, hence the subgraph induced by $S_1 \cup S_2$ contains a complete bipartite subgraph $K(m,m)$, with $m = |S_1|$. By lemma 3.2, this implies that $k \geq m$. Hence $|X_i| \leq |S_1| + |S_2| + 1 \leq 2k+1$. □

Notice that the algorithm can be implemented to run in $O(nk)$ time, since at each step one new element is put into a subset. Hence we have proved the following theorem:

Theorem 5.1 *If $G[\pi]$ is a permutation graph with treewidth at most k, then the pathwidth of $G[\pi]$ is at most $2k$, and the $O(nk)$ time algorithm Pathdec produces a path-decomposition with width at most $2k$.*

Using results of [11] this shows that, if the treewidth is bounded by a constant, an optimal path-decomposition and tree-decomposition can be computed in linear time. Also, it follows that we have $O(nk)$ approximation algorithms for pathwidth and treewidth with performance ratio 2. In a forthcoming paper [10] we show that the pathwidth and treewidth are equal, and can be computed in $O(nk^2)$ time.

6 An approximate path-decomposition for cocomparability graphs

In this section, let G be a cocomparability graph with treewidth at most k. We start with an informal discussion of the algorithm. Recall lemma 2.1. There exist n continous functions, $F_i : (0,1) \rightarrow R$, such that two vertices are adjacent if and only if the corresponding functions intersect. First make a vertex coloring of G, such that no two adjacent vertices have the same color. Since the treewidth of G is at most k, this can be done by using at most $k+1$ colors. Fix any position L at the line $x = 0$. This partitions the vertices of G into two sets: one set of vertices for which the correponding functions start below L, and one set for which the correponding functions start at position at least L. For each of these positions L we make a subset X_L as follows. For each colorclass C_t, take the top most $k+1$ functions which start below L, say $\overline{C_t}$ (take all if there are less than $k+1$). Notice that the functions correponding with a colorclass do not intersect, hence the 'topmost functions' are well defined. For functions starting at position at least L, take those which are adjacent to $k+1$ vertices of $\overline{C_t}$, say $\mathcal{F}_t$. Notice that $\mathcal{F}_t$ is empty if $|\overline{C_t}| \leq k$. Also notice that if $|\overline{C_t}| = k+1$, then each vertex of $\mathcal{F}_t$ is adjacent to all vertices of $\overline{C_t}$.

In this last case $|\mathcal{F}_t| \leq k$, otherwise there is a $K(k+1, k+1)$ subgraph. It follows that X_L has at most $(k+1)(2k+1)$ vertices, since there are at most $k+1$ color classes and for each color class C_t we have $|\overline{C_t}| + |\mathcal{F}_t| \leq 2k+1$. Notice that we only used the ordering of the functions at position $x = 0$. We can find a suitable ordering using the height function of G.

We now give the formal description of the algorithm and proof the correctness, without using the function model. We assume that a height function h of the complement $\overline{G}$ with transitive orientation F, and a coloring of G are given. Let $C_1, \ldots, C_s$ be the color classes of G, where $s = \chi(G)$ is the chromatic number of G. Since a partial k-tree can always be colored with $k+1$ colors, we may assume that the number of color classes $s \leq k+1$.

The first step of the algorithm is to renumber the vertices.

Definition 6.1 *Let $G = (V, E)$ be a cocomparability graph with n vertices and let h be a height function of the transitively oriented complement $\overline{G}$. A* height labeling *of G is a bijection $L : V \to \{1, \ldots, n\}$, such that $h(x) > h(y)$ implies that $L(x) > L(y)$.*

A height labeling clearly can be computed in $O(n)$ time, if the height function h is given.

Definition 6.2 *For $i = 1, \ldots, n$ and for $t = 1, \ldots, s$, let $C_t(i)$ be the set of vertices in color class C_t with label at most i: $C_t(i) = C_t \cap \{x \mid L(x) \leq i\}$. Write $C_t(i) = \{x_1, \ldots, x_m\}$, with $i \geq L(x_1) > L(x_2) \ldots > L(x_m)$. We define for $i = 1, \ldots, n$ and $t = 1, \ldots, s$:*

$$\overline{C_t}(i) = \begin{cases} C_t(i) & \text{if } m \leq k+1 \\ \{x_1, \ldots, x_{k+1}\} & \text{otherwise} \end{cases}$$

Notice that, since C_t is a clique in $\overline{G}$, all heights of vertices in C_t are different. It follows that $\overline{C_t}(i)$ is uniquely determined as the set of $k+1$ vertices of $C_t(i)$ with the largest heights.

Definition 6.3 *For $i = 1, \ldots, n$ and $t = 1, \ldots, s$, define*

$$\mathcal{F}_t(i) = \{x \mid L(x) > i \wedge |Adj(x) \cap C_t(i)| \geq k+1\}$$

where $Adj(x)$ is the set of neighbors of x.

We can now define the subsets of the path-decomposition:

Definition 6.4 *For $i = 1, \ldots, n$, let X_i be the following set of vertices:*

$$X_i = \bigcup_{1 \leq t \leq s} (\mathcal{F}_t(i) \cup \overline{C_t}(i))$$

In the rest of this section we prove that each subset X_i has at most $(2k+1)\chi(G)$ elements and we show that $(X_1, \ldots, X_n)$ forms indeed a path-decomposition. The following lemma is crucial.

Lemma 6.1 *Each $y \in \mathcal{F}_t(i)$ is adjacent to all vertices of $\overline{C_t}(i)$.*

Proof. Suppose not. Let $y \in \mathcal{F}_t(i)$ be not adjacent to every vertex in $\overline{C_t}(i)$. Let $C_t(i) = \{x_1, \ldots, x_m\}$ with $h(x_1) > h(x_2) > \ldots > h(x_m)$. If $m < k+1$, then by definition $\mathcal{F}_t(i) = \emptyset$. Hence we may assume $m \geq k+1$ and $\overline{C_t}(i) = \{x_1, \ldots, x_{k+1}\}$. Let $x_w \in \overline{C_t}(i)$ (with $1 \leq w \leq k+1$) be not adjacent to y. Consider the complement $\overline{G}$ with the transitive orientation F. $C_t(i)$ is a clique in $\overline{G}$ and $(x_p, x_q) \in F$ for all x_p and x_q in $C_t(i)$ with $p < q$. Since y is adjacent to x_w in $\overline{G}$, we must have $h(y) \neq h(x_w)$. Since $L(y) > L(x_w)$, it follows that $h(y) > h(x_w)$ and hence $(y, x_w) \in F$. Since F is transitive, we find that y is adjacent in $\overline{G}$ to all vertices x_p with $w \leq p \leq m$. So y can have at most $w-1$ neighbors in $C_t(i)$ in G, hence it can not be in $\mathcal{F}_t(i)$, contradiction. □

Corollary 6.1 *A vertex y with $L(y) > i$ is in $\mathcal{F}_t(i)$ if and only if it is adjacent to the vertex in $\overline{C_t}(i)$ with the smallest label.*

Theorem 6.1 *If G is a cocomparability graph with treewidth at most k, then $|\overline{C_t}(i)| \leq k+1$ and $|\mathcal{F}_t(i)| < k+1$.*

Proof. Notice that $|\overline{C_t}(i)| \leq k+1$, by definition. If $|\overline{C_t}(i)| < k+1$, then $C_t(i)$ contains less than $k+1$ elements, and then, by definition $\mathcal{F}_t(i) = \emptyset$.

Now assume that $C_t(i)$ contains at least $k+1$ vertices. Then $|\overline{C_t}(i)| = k+1$. By lemma 6.1 all vertices of $\mathcal{F}_t(i)$ are adjacent to all vertices of $\overline{C_t}(i)$.

The treewidth of G is at most k. Hence G can not have a complete bipartite subgraph $K(k+1, k+1)$. This implies that $|\mathcal{F}_t(i)| < k+1$. □

Corollary 6.2 $\forall_i \, |X_i| \leq (2k+1)\chi(G)$.

Corollary 6.2 shows that, if $(X_1, \ldots, X_n)$ is a path-decomposition of G, then the width of this path-decomposition is at most $2k^2 + 3k$. The next three lemmas show that $(X_1, \ldots, X_n)$ is indeed a path-decomposition.

Lemma 6.2 *For every vertex x of G: $x \in X_{L(x)}$.*

Proof. Let x be in the color class C_p. By definition, $x \in \overline{C_p}(L(x)) \subseteq X_{L(x)}$. □

Lemma 6.3 *Let $\{x, y\} \in E$ be an edge of G. Then there is a subset X_i such that x and y are both in X_i.*

Proof. Assume $L(x) < L(y)$. Notice that x and y are not in the same color class, since they are adjacent. Consider the color class of x, say $C_p = \{x_1, \ldots, x_m\}$, and let $L(x_1) > L(x_2) > \ldots > L(x_m)$. Let $x = x_j$ for some $1 \leq j \leq m$. Clearly, if $j \leq k+1$, then x and y are both contained in $X_{L(y)}$ since $x \in \overline{C_p}(L(y)) \subseteq X_{L(y)}$. Now assume that $j > k+1$. Consider $z = x_{j-k}$. If $L(y) < L(z)$ then x and y are both contained in $X_{L(y)}$, since $x \in \overline{C_p}(L(y))$. If $L(y) > L(z)$, then $x \in \overline{C_p}(L(z))$ and $y \in \mathcal{F}_p(L(z))$. □

Lemma 6.4 *The subsets X_i containing a given vertex x, are a consecutive subsequence of $(X_1, \ldots, X_n)$.*

Proof. Assume $L_1 < L_2$ and $x \in X_{L_1} \cap X_{L_2}$. Let $L_1 < L < L_2$. We prove that $x \in X_L$. We consider three cases:

case 1 $L(x) \geq L_2$. Since $x \in X_{L_1}$, x must be adjacent to at least $k+1$ vertices of some color class which are in X_{L_1}. Clearly, this also holds for every $L_1 \leq L < L(x)$. Hence $x \in X_L$.

case 2 $L(x) \leq L_1$. Since $x \in X_{L_2}$, x must be among the vertices in its color class with the $k+1$ largest heights which are in X_{L_2}. But, then clearly this must hold for every $L(x) \leq L \leq L_2$.

case 3 $L_1 < L(x) < L_2$. The argument of the first case shows that $x \in X_L$ for all $L_1 \leq L < L(x)$. In the second case it is shown that $x \in X_L$ for all $L(x) \leq L \leq L_2$.

□

By lemmas 6.2, 6.3, and 6.4 the sequence of subsets $(X_1, \ldots, X_n)$ is a path-decomposition for the cocomparability graph G. According to corollary 6.2, the width of this path-decomposition is at most $(2k+1)\chi(G) - 1$. In the following theorem we summarize these results.

Theorem 6.2 *Let G be a cocomparability graph. The sequence $(X_1, \ldots, X_n)$ with X_i defined in definition 6.4, is a path-decomposition for G. If the treewidth of G is at most k, then the width of this path-decomposition is at most $(2k+1)\chi(G) - 1 \leq 2k^2 + 3k$.*

Consider the time it takes to compute the sets X_i. We can sort all the color classes according to increasing labels in $O(nk)$ time. Then each set $\overline{C_t}(i)$ can be computed in $O(k)$ time. Now notice that $\mathcal{F}_t(i) \subseteq \mathcal{F}_t(i+1) \cup \{x \mid L(x) = i+1\}$. If we use an adjacency matrix to represent G, we can compute each set $\mathcal{F}_t(i)$ in time $O(k)$: An element y in $\mathcal{F}_t(i+1) \cup \{x \mid L(x) = i+1\}$ is in $\mathcal{F}_t(i)$ if and only if y is adjacent to the vertex with the smallest label in $\overline{C_t}(i)$ (corollary 6.1). The sets $\overline{C_t}(i)$ and $\mathcal{F}_t(i)$ have at most $k+1$ elements and there are at most $k+1$ of each for each i. Hence we can easily compute each set X_i in time $O(k^2)$.

Corollary 6.3 *Let G be a cocomparability graph with treewidth k. Assume a vertex coloring of G is given (with at most $k+1$ colors), and a transitive orientation F of the complement $\overline{G}$. If an adjacency matrix is used to represent G, then a path-decomposition of G with width at most $2k^2 + 3k$ can be computed in time $O(nk^2)$.*

We end this section with some remarks concerning the complexity of finding approximations for the treewidth of bipartite graphs. As all bipartite graphs are comparability graphs, the same remarks hold for comparability graphs. Given a graph G, let $S(G)$ be the subdivision graph (see [29, pages 79–80]). Clearly, $S(G)$ is bipartite. It is easily seen that the treewidth of $S(G)$ is equal to the treewidth of G. Given a tree-decomposition for $S(G)$, this can easily be transformed into a tree-decomposition for G with the same width. This shows that finding an approximation for the treewidth of bipartite graphs (within a constant factor) is as hard as finding approximations for the treewidth in general.

7 Conclusions

In this paper we described very simple and efficient algorithms to approximate pathwidth and treewidth of cotriangulated graphs, permutation graphs, convex graphs and cocomparability graphs. There are classes of graphs for which the exact pathwidth and treewidth can be computed efficiently. For example cographs [12], splitgraphs (lemma 3.3) and interval graphs. Also, in a forthcoming paper [10] we show that the pathwidth and treewidth of a permutation graph are equal and can be computed in $O(nk)$ time. In fact, in this paper we show that part of this can be generalized to cocomparability graphs; i.e. for *any* cocomparability graph the treewidth and pathwidth are equal. However, computing the exact treewidth for cocomparability graphs in polynomial time remains an open problem. The treewidth can also be computed efficiently for chordal graphs and circular arc graphs [42]. Also for chordal bipartite graphs (containing the convex graphs), the treewidth can be computed in polynomial time [30]. It would be of interest to know, if there exists a fast algorithm which computes the treewidth for cotriangulated graphs. In this respect we like to mention again the result of [27] which shows that the pathwidth of chordal graphs is NP-complete. Another interesting conclusion from this paper is that, for the considered graph classes, if the treewidth is at most k then the pathwidth is at most some polynomial of k. It would be of interest to know for which other classes of graphs such a relation between the pathwidth and treewidth exists. Finally, in [9] it is shown that there is a polynomial algorithm that finds a tree-decomposition of G with treewidth at most $O(k \log n)$, where k is the treewidth of G and n the number of vertices. We have shown that there are many graph classes for which there is a polynomial algorithm that finds a tree-decomposition with width at most some fixed polynomial of k. It would be very interesting to know if this could be generalized.

8 Acknowledgements

We like to thank D. Seese, D. Kratsch and B. Reed for valuable discussions.

References

[1] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability — A survey. *BIT* **25**, 2 – 23, 1985.

[2] S. Arnborg, D.G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.* **8**, 277 – 284, 1987.

[3] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms* **12**, 308 – 340, 1991.

[4] S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, *SIAM J. Alg. Disc. Meth.* **7**, 305 – 314, 1986.

[5] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees. *Disc. Appl. Math.* **23**, 11 – 24, 1989.

[6] C. Berge and C. Chvatal, *Topics on Perfect Graphs*, Annals of Discrete Math. **21**, 1984.

[7] H.L. Bodlaender, A tourist guide through treewidth, Technical report RUU-CS-92-12, Department of computer science, Utrecht University, Utrecht, The Netherlands, 1992. To appear in: *Proceedings 7th International Meeting of Young Computer Scientists*, Springer Verlag, Lecture Notes in Computer Science.

[8] H.L. Bodlaender, Dynamic programming algorithms on graphs with bounded treewidth, *Proceedings of the 15th International colloquium on Automata, Languages and Programming*, 105 – 119, Springer Verlag, Lecture Notes in Computer Science, vol. 317, 1988.

[9] H. Bodlaender, J. Gilbert, H. Hafsteinsson and T. Kloks, Approximating treewidth, pathwidth and minimum elimination tree height, In G. Schmidt and R. Berghammer, editors, *Proceedings 17th International Workshop on Graph-Theoretic Concepts in Computer Science WG'91*, 1 – 12, Springer Verlag, Lecture Notes in Computer Science, vol. 570, 1992.

[10] H. Bodlaender, T. Kloks and D. Kratsch, Treewidth and pathwidth of permutation graphs, Technical report RUU-CS-92-30, Department of computer science, Utrecht University, Utrecht, The Nehterlands, 1992.

[11] H. Bodlaender and T. Kloks, Better algorithms for the pathwidth and treewidth of graphs, *Proceedings of the 18th International colloquium on Automata, Languages and Programming*, 544 – 555, Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[12] H. Bodlaender and R.H. Möhring, The pathwidth and treewidth of cographs, *Proceedings 2nd Scandinavian Workshop on Algorithm Theory*, 301 – 309, Springer Verlag, Lecture Notes in Computer Science vol. 447, 1990.

[13] K. Booth and G. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity testing using PQ-tree algorithms, *J. of Computer and System Sciences* **13**, 335 – 379, 1976.

[14] A. Brandstädt, Special graph classes—A survey, Schriftenreihe des Fachbereichs Mathematik, SM-DU-199 (1991) Universiät Duisburg Gesamthochschule.

[15] A. Brandstädt and D. Kratsch, On the restriction of some NP-complete graph problems to permutation graphs, *Fundamentals of Computation Theory, proc. FCT* 1985, 53 – 62, Lecture Notes in Comp. Science vol. 199, Springer Verlag, New York, 1985.

[16] A. Brandstädt and D. Kratsch, On domination problems for permutation and other perfect graphs, *Theor. Comput. Sci.* **54**, 181 – 198, 1987.

[17] B. Courcelle, The monadic second-order logic of graphs I: Recognizable sets of finite graphs, *Information and Computation* **85**, 12 – 75, 1990.

[18] B. Courcelle, The monadic second-order logic of graphs III: Treewidth, forbidden minors and complexity issues, Report 8852, University Bordeaux 1, 1988. To appear in: *Informatique Théoretique et Applications*.

[19] B. Courcelle, Graph rewriting: an algebraic and logical approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, 192 – 242, Amsterdam, 1990. North Holland Publ. Comp.

[20] G.A. Dirac, On rigid circuit graphs, *Abh. Math. Sem. Univ. Hamburg* **25**, 71 – 76, 1961.

[21] S. Even, A. Pnueli and A. Lempel, Permutation graphs and transitive graphs, *J. Assoc. Comput. Mach.* **19**, 400 – 410, 1972.

[22] M. Farber and M. Keil, Domination in permutation graphs, *J. Algorithms* **6**, 309 – 321, 1985.

[23] D.R. Fulkerson and O.A. Gross, Incidence matrices and interval graphs, *Pacific J. Math.* **15**, 835 – 855, 1965.

[24] T. Gallai, Transitive orientierbaren Graphen, *Acta Math. Sci. Hung.* **18**, 25–66, 1967.

[25] Gilmore and Hoffman, A characterization of comparability and interval graphs, *Canad. J. Math.* **16**, 539 – 548, 1964.

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

[27] J. Gustedt, Pathwidth for chordal graphs is NP-complete. Technical report 221/1989, Technical University Berlin, Berlin, Germany, 1989. To appear in: *Discr. Appl. Math.*

[28] A. Hajnal and J. Surányi, Über die Auflösung von Graphen in vollständige Teilgraphen, *Ann. Univ. Sci. Budapest Eötvös. Sect. Math.* **1**, 113 – 121, 1958.

[29] F. Harary, *Graph Theory*, Addison-Wesley, Reading, Massachusetts, 1969.

[30] T. Kloks and D. Kratsch, Treewidth of chordal bipartite graphs, Technical report RUU-CS-92-28, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 1992.

[31] J. Lagergren and S. Arnborg, Finding minimal forbidden minors using a finite congruence, *Proceedings of the 18th International colloquium on Automata, Languages and Programming*, 532–543, Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[32] J. van Leeuwen, Graph algorithms. In *Handbook of Theoretical Computer Science, A: Algorithms an Complexity Theory*, 527–631, Amsterdam, 1990. North Holland Publ. Comp.

[33] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Math.* **2**, 253 – 267, 1972.

[34] J. Matoušek and R. Thomas, Algorithms Finding Tree-Decompositions of Graphs, *Journal of Algorithms* **12**, 1 – 22, 1991.

[35] A. Pnueli, A. Lempel, and S. Even, Transitive orientation of graphs and identification of permutation graphs, *Canad. J. Math.* **23**, 160 – 175, 1971.

[36] B. Reed, Finding approximate separators and computing treewidth quickly, *24th Annual ACM Symposium on Theory of Computing*, 221 – 228, 1992.

[37] N. Robertson and P.D. Seymour, Graph minors—A survey. In I. Anderson, editor, *Surveys in Combinatorics*, 153 – 171. Cambridge Univ. Press 1985.

[38] N. Robertson and P.D. Seymour, Graph minors II. Algorithmic aspects of treewidth. *J. Algorithms* **7**, 309 – 322, 1986.

[39] F.S. Roberts, Graph theory and its applications to problems of society, NFS-CBMS Monograph no. 29 (SIAM Publications, Philadelphia, PA. 1978).

[40] D. Rotem and J. Urrutia, Comparability graphs and intersection graphs, *Discrete Math.* **43**, 37 – 46, 1983.

[41] J. Spinrad, On comparability and permutation graphs, *SIAM J. Comp.* **14**, No. 3, August 1985.

[42] R. Sundaram, K. Sher Singh and C. Pandu Rangan, Treewidth of circular arc graphs, Manuscript 1991.