Treewidth and pathwidth of cocomparability graphs of bounded dimension

Eindhoven University of Technology

Department of Mathematics and Computing Science

Treewidth and pathwidth of cocomparability graphs of bounded dimension

by

T. Kloks, D. Kratsch and J. Spinrad

93/46

Computing Science Note 93/46
Eindhoven, December 1993

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

Treewidth and pathwidth of cocomparability graphs of bounded dimension

T. Kloks[1] *, D. Kratsch[2] ** and J. Spinrad[3]

[1] Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O.Box 513
5600 MB Eindhoven, The Netherlands
[2] IRISA
Campus de Beaulieu
35042 Rennes, France
[3] Department of Computer Science
Vanderbilt University
Nashville TN37235, USA

Abstract. In this paper we describe a polynomial time algorithm computing the treewidth of a cocomparability graph of bounded dimension. We do not assume that an intersection model of the graph is part of the input.

1 Introduction

The TREEWIDTH problem is the problem of finding a triangulated graph H with smallest maximum clique size having the given graph G as spanning subgraph. The PATHWIDTH problem is the problem of finding an interval graph H with smallest maximum clique size having the given graph G as spanning subgraph.

The problem 'Given a graph $G = (V, E)$ and an integer k, is the treewidth of G at most k' is NP-complete, even when only complements of bipartite graphs G are allowed as input graphs [2] and it also remains NP-complete on bipartite graphs [15]. The problem 'Given a graph $G = (V, E)$ and an integer k, is the pathwidth of G at most k' is NP-complete on cobipartite graphs [2], bipartite graphs [15] and triangulated graphs [12].

The treewidth can be computed in polynomial time on triangulated graphs (trivially), cographs [5], circular arc graphs [21], chordal bipartite graphs [18], permutation graphs [4], circle graphs [16] and distance hereditary graphs [1]. Since many NP-complete problems remain NP-complete when restricted to some of these classes, it is of great importance to be able to use the algorithms for graphs of small treewidth for these problems. We want to mention here that the algorithm presented in [17] appears to be wrong. At this moment we do not know whether such a general result is possible.

It was independently shown in [4] and [13] that for every cocomparability graph the treewidth and pathwidth coincide. Cocomparability graphs are a subclass of the

* Email: ton@win.tue.nl
** On leave from Friedrich-Schiller-Universität Jena, Germany.

perfect graphs containing permutation graphs, interval graphs and trapezoid graphs. R. Möhring showed that this result is extendable to AT-free graphs (asteroidal triple-free graphs). Thus, on AT-free graphs, which contain cocomparability graphs as a proper subclass while they are no longer a subclass of perfect graphs, treewidth and pathwidth still coincide [20].

In this paper we show that for cocomparability graphs of bounded dimension, the treewidth and pathwidth can be computed in polynomial time. In [4] this was shown under the assumption that an intersection model is part of the input. However, it is well-known that the problem 'Given a poset P and an positive integer d, is the dimension of P at most d?', is NP-complete for every fixed $d \geq 3$ [22]. This problem is equivalent to the recognition problem of cocomparability graphs of dimension at most d. It follows that recognition of cocomparability graphs of dimension at most d is NP-complete for every fixed $d \geq 3$ and that finding an optimal intersection model is intractable [11]. In this paper we give a polynomial algorithm which does not require such an intersection model as part of the input. However, the input has to be a cocomparability graph of dimension at most d (although we can not check this efficiently), at least to guarantee the time bound $O(n^{3d+3})$. The algorithm will work correctly as soon as the input graph is a cocomparability graph.

2 Preliminaries

In this section we start with some necessary definitions and results. We consider only finite, undirected and simple graphs $G = (V, E)$. We always denote the number of vertices of G by n. For definitions and properties of graph classes not given here we refer to [6, 10, 14, 15].

If $G = (V, E)$ is a graph and $W \subseteq V$ a subset of vertices then we use $G[W]$ as a notation for the subgraph of G *induced* by the vertices of W.

Definition 1. A graph H is *triangulated* (or *chordal*) if it does not contain a chordless cycle of length at least four as an induced subgraph. A *triangulation* of a graph G is a graph H with the same vertex set as G such that H is triangulated and G is a subgraph of H. In that case we say that G is *triangulated into* H.

Definition 2. Given a graph $G = (V, E)$ and two non adjacent vertices a and b, a subset $S \subset V$ is an *a, b-separator* if the removal of S separates a and b in distinct connected components. If no proper subset of S is an a, b-separator then S is a *minimal a, b-separator*. A *minimal separator* is a set of vertices S for which there exist non adjacent vertices a and b such that S is a minimal a, b-separator.

The following lemma appears for example as an exercise in [10]. It provides in an easy algorithm to recognize minimal separators.

Lemma 3. *Let S be a separator of the graph $G = (V, E)$. Then S is a minimal separator if and only if there are two different connected components of $G[V \setminus S]$ such that every vertex of S has a neighbor in both of these components.*

Proof. Let S be a minimal a, b-separator and let C_a and C_b be the connected components containing a and b respectively. Assume $s \in S$ has no neighbors in

C_a. Since S is minimal there is a path from a to b going through s but using no other vertices of S. Hence s must have at least one neighbor in C_a and at least one in C_b.

Now let S be a separator and let C_a and C_b be components such that each vertex of S has at least one neighbor in C_a and C_b. Let $s \in S$. Then there is a path from a to b using s but no other vertices of S. Hence S is a minimal a,b-separator. □

The following lemma describes a useful property of minimal separators.

Lemma 4. *Let $G = (V, E)$ be a graph and S a minimal separator and a clique of G. Let C be a connected component of $G[V \setminus S]$ and let x and y be non adjacent vertices of $S \cup C$. Then every minimal x, y-separator S^* of G is a proper subset of $S \cup C$.*

Proof. Let S^* be a minimal x, y-separator of G and let C_x and C_y be the components of $G[V \setminus S^*]$ containing x and y, respectively.

C_x and C_y can not both have non-empty intersection with S since S is a clique. W.l.o.g. let $C_x \cap S = \emptyset$. Since S is a minimal separator of G, C_x is connected and $x \in C$ belongs to C_x, we get $C_x \subseteq C$. A vertex $s \in V \setminus (S \cup C)$ belongs to a component of $G[V \setminus S]$ different from C and can not have a neighbour in C_x. Hence, $s \notin S^*$. Finally, $S^* \subset S \cup C$ since $x, y \notin S^*$. □

One of the main tools in our algorithm is the fact that all minimal separators of a graph can be computed in time polynomial times the number of minimal separators. This was shown in [19].

Theorem 5. *Let R be the number of minimal separators of a graph G. There exists an algorithm which list all minimal separators in G in time $O(n^6 R)$.*

We use Dirac's characterization of triangulated graphs [8].

Lemma 6. *A graph G is triangulated if and only if every minimal separator is a clique.*

Proof. Assume G is triangulated. Let S be a minimal a,b-separator. Assume S has non adjacent vertices x and y. Since S is a minimal separator, x and y both have a neighbor in C_a and C_b. These components are connected, hence it follows that the graph has a chordless cycle of length at least four.

Let C be a chordless cycle of length at least four. Let a and b be nonadjacent vertices of C. Every minimal a,b-separator must have a vertex of each of the two paths between a and b. Since C is a chordless cycle, these vertices are non adjacent. Hence a minimal a,b-separator cannot be a clique. □

The following two theorems show how to restrict the triangulations to be considered.

Definition 7. A *minimal triangulation* H of a graph $G = (V, E)$ is a triangulation such that the following two conditions are satisfied.

1. If a and b are non adjacent in H then every minimal a,b-separator in H is also a minimal a,b-separator in G.
2. If S is a minimal separator in H and C is the vertex set of a connected component of $H[V \setminus S]$ then $G[C]$ is also connected.

In [4] the following theorem is shown.

Theorem 8. *Let H be a triangulation of a graph G. There exists a minimal triangulation H' of $G = (V, E)$ such that H' is a subgraph of H.*

Proof. Let W be a minimal a,b-separator of H such that either W induces no minimal a,b-separator in G or the connected components of $H[V \setminus W]$ are different from those of $G[V \setminus W]$. Let $S \subseteq W$ be a minimal a,b-separator in G and let $C_1, \ldots, C_t$ be the connected components of $G[V \setminus S]$.

Make a triangulated graph H' as follows. For each $1 \leq i \leq t$ take the triangulated subgraph of $C_i \cup S$ of H. Since S is a clique in H, this gives a triangulated subgraph H' of H. The vertex sets of the connected components of $H'[V \setminus S]$ are the same as those of $G[V \setminus S]$. We claim that the number of edges of H' is smaller than the number of edges of H, which, by induction, proves the theorem. Clearly H' is a subgraph of H.

First assume $S \neq W$ and let $x \in W \setminus S$. In H, x has a neighbor in the component containing a and a neighbor in the component containing b by Lemma 3. Not both these edges can be present in H'.

Now assume $S = W$. By assumption the vertex sets of the components of $H'[V \setminus S]$ are different from those of $H[V \setminus S]$. Then there must be some connected component in $H[V \setminus S]$ containing two connected components of $H'[V \setminus S]$. This can only be the case if there is some edge between these components in $H[V \setminus S]$. This proves the theorem. □

To illustrate that minimal triangulations are not very restrictive, notice that a clique is a minimal triangulation of G. We now show that we can restrict the triangulations to be considered somewhat more.

Definition 9. Let Δ be the set of all minimal separators of a graph $G = (V, E)$. For a subset $\mathcal{C} \subseteq \Delta$ let $G_{\mathcal{C}}$ be the graph obtained from G by adding edges between vertices contained in the same set $C \in \mathcal{C}$. If the graph $G_{\mathcal{C}}$ is a minimal triangulation of G such that $\mathcal{C}$ is exactly the set of all minimal separators of $G_{\mathcal{C}}$, then $G_{\mathcal{C}}$ is called an *efficient* triangulation.

Notice that for each $C \in \mathcal{C}$, the induced subgraph $G_{\mathcal{C}}[C]$ is a clique.

Theorem 10. *Let H be a triangulation of a graph G. There exists an efficient triangulation $G_{\mathcal{C}}$ of G which is a subgraph of H.*

Proof. Take a minimal triangulation H' which is a subgraph of H such that the number of edges of H' is minimal (theorem 8). We claim that H' is efficient. Let $\mathcal{C}$ be the set of minimal vertex separators of H'. We prove that $G_{\mathcal{C}} = H'$.

Since every minimal separator in a triangulated graph is a clique, it follows that $G_{\mathcal{C}}$ is a subgraph of H'. Consider a pair of vertices a and b which are adjacent in

H' but not adjacent in G. Remove the edge from the graph H'. Call the resulting graph H^*. Since the number of edges of H' is minimal, it follows that H^* has a chordless cycle. Clearly this cycle must have length four. Let $\{x, y, a, b\}$ be the vertices of this square. Then x and y are non adjacent in H'. But then a and b are contained in every minimal x, y-separator in H'. It follows that a and b are also adjacent in G_C. □

The treewidth is a graph parameter which can be defined using triangulations.

Definition 11. The *treewidth* of a graph $G = (V, E)$, denoted by $tw(G)$, is the smallest maximum clique size of all triangulations H of G decreased by one.

The TREEWIDTH problem is 'Given a graph $G = (V, E)$ and a positive integer k, decide whether $tw(G) \leq k$ holds'. The problem is NP-complete even when restricted to bipartite or to cobipartite graphs [2]. Consequently, the TREEWIDTH problem is NP-complete when restricted to cocomparability graphs, which contain the cobipartite graphs as a proper subclass. Thus, a bound on the dimension is necessary (except P=NP) and quite natural to enable the design of a polynomial time treewidth algorithm for cocomparability graphs. Fortunately, much work was done on efficient treewidth algorithms for classes of well-structured graphs in the last years [5, 3, 21, 18, 4, 16, 1, 15].

The following is an immediate consequence of Theorem 10.

Corollary 12. *Every triangulation with a minimal number of edges is efficient.*

Corollary 13. *There exists an efficient triangulation such that the maximum clique has a number of vertices equal to the treewidth of the graph plus one.*

Definition 14. A comparability graph is a graph which admits a transitive orientation of its edges. A cocomparability graph is a graph of which the complement is a comparability graph. A permutation graph is an intersection graph of straight line segments between two parallel horizontal line.

Permutation graphs can be characterized as being exactly the graphs which are comparability and cocomparability graphs and they are exactly the comparability graphs of poset dimension at most two. In [11] it is shown that cocomparability graphs are the intersection graphs of a concatenation of permutation diagrams. The minimal number of permutation graphs needed plus one is called the dimension of the cocomparability graph (in fact, this is equal to the dimension of the poset corresponding to the complement). Notice that a permutation graph is a cocomparability graph of dimension two. The following lemma was shown in [4].

Lemma 15. *A cocomparability graph of dimension d has at most $(n+1)^d$ minimal separators.*

We use the characterization of interval graphs discovered by Gilmore and Hoffman [9].

Lemma 16. *A graph G is an interval graph if and only if the maximal cliques of G can be ordered in such a way that for every vertex the maximal cliques containing it occur consecutively.*

We call this ordering of the maximal cliques a *consecutive clique arrangement* of G. Using this characterization, we can easily identify the minimal vertex separators in an interval graph.

Lemma 17. *Let G be an interval graph and let $C_1, C_2, \ldots, C_t$ be a consecutive clique arrangement of G. The minimal separators of G are the sets $C_i \cap C_{i+1}$, $(i = 1, \ldots, t-1)$.*

Proof. Since each C_i is a maximal clique, we have that for each $1 \leq i < t$: $C_i \setminus C_{i+1} \neq \emptyset$ and $C_{i+1} \setminus C_i \neq \emptyset$. Let $x \in C_i \setminus C_{i+1}$ and $y \in C_{i+1} \setminus C_i$. Then clearly $C_i \cap C_{i+1}$ is a minimal x, y-separator.

Now consider nonadjacent vertices a and b and let S be a minimal a, b-separator. Assume a appears before b in the clique arrangement. Let C_i be the last clique that contains a and let C_j be the first clique that contains b. If S contains not all vertices of $C_i \cap C_{i+1}$, then there is a path from a to all vertices of $C_{i+1} \setminus S$ without using vertices of S. Continuing in this way we either find a path from a to b or some $i \leq k < j$ such that $C_k \cap C_{k+1} \subseteq S$. □

The pathwidth problem is concerned with finding a triangulation of a graph into an interval graph such that the clique size is minimized. In general the pathwidth of a graph is at least equal to the treewidth of the graph. Determining the pathwidth of a graph is NP-complete, even when restricted to chordal graphs [12]. However, for cocomparabiltity graphs the measures treewidth and pathwidth coincide [13, 4].

Theorem 18. *For a cocomparability the pathwidth and treewidth are equal. Moreover, there exists an efficient triangulation of the graph into an interval graph.*

3 Pieces and realizers

In this section we assume that $G = (V, E)$ is a connected cocomparability graph with n vertices and of dimension d.

Definition 19. Two minimal separators S_1 and S_2 are *non-crossing* if all vertices of $S_1 \setminus S_2$ are contained in the same connected component of $G[V \setminus S_2]$ and all vertices of $S_2 \setminus S_1$ are contained in the same connected component of $G[V \setminus S_1]$.

Lemma 20. *Let H be a chordal graph. Then every pair of minimal separators in H is non-crossing.*

Proof. Let S_1 and S_2 be minimal separators. Since the graph is chordal, the subgraphs induced by these separators are cliques. Then clearly, $S_1 \setminus S_2$ must be contained in one connected component of $H[V \setminus S_2]$. □

Lemma 21. *Let G_C be an efficient triangulation of G and let S_1, S_2 be minimal separators in G_C. Then S_1 and S_2 are non-crossing separators in G.*

Proof. S_1 and S_2 are non-crossing in G_C by Lemma 20. Since G_C is efficient, S_1 and S_2 are minimal separators in G. The vertex sets of the connected components of $G_C[V \setminus S_i]$ are the same as those of $G[V \setminus S_i]$ $(i = 1, 2)$. It follows that S_1 and S_2 are also non-crossing in G. □

Definition 22. Let S_1 and S_2 be two non-crossing separators in G. Consider a connected component D of $G[V \setminus (S_1 \cup S_2)]$. The component D is called *between* S_1 and S_2, if $S_2 \setminus S_1$ and D are in the same connected component of $G[V \setminus S_1]$ and $S_1 \setminus S_2$ and D are in the same connected component of $G[V \setminus S_2]$.

We adopt the convention that if $S_2 \subseteq S_1$, then *every* connected component of $G[V \setminus (S_1 \cup S_2)]$ is in the same connected component of $G[V \setminus S_1]$ as $S_2 \setminus S_1$.

Definition 23. Let S_1 and S_2 be non-crossing separators in G. The *piece* $P = \mathcal{P}(S_1, S_2)$ is the set of vertices of S_1, S_2 and of all connected components of $G[V \setminus (S_1 \cup S_2)]$ that are between S_1 and S_2.

For example, notice that if $S_1 = S_2$ then the piece $\mathcal{P}(S_1, S_2) = V$. If $S_1 \subset S_2$, the piece consists of S_1 and the vertices of the connected component of $G[V \setminus S_1]$ that contain the vertices of $S_2 \setminus S_1$.

Lemma 24. *Let S_1 and S_2 be non-crossing separators in G. Let $G_{\mathcal{C}}$ be an efficient triangulation of G such that $S_1, S_2 \in \mathcal{C}$. Then the pieces of S_1 and S_2 in G and in $G_{\mathcal{C}}$ are equal.*

Proof. Clearly, the piece in G is a subset of the piece in $G_{\mathcal{C}}$. Let D be a connected component of $G_{\mathcal{C}}[V \setminus (S_1 \cup S_2)]$ that is in the piece in $G_{\mathcal{C}}$. Since $G_{\mathcal{C}}$ is efficient, the vertex sets of the connected components of $G_{\mathcal{C}}[V \setminus S_1]$ and $G[V \setminus S_1]$ are the same. Hence D is also contained in the same connected component as $S_2 \setminus S_1$ in $G[V \setminus S_1]$. In the same manner it follows that D and $S_1 \setminus S_2$ are contained in the same connected component of $G[V \setminus S_2]$. It follows that D is contained in the piece in G. □

We have shown that the pieces of S_1 and S_2 in G and in $G_{\mathcal{C}}$ are equal. On the other hand, in general, it is *not* true that the vertex sets of the connected components of $G[V \setminus (S_1 \cup S_2)]$ and $G_{\mathcal{C}}[V \setminus (S_1 \cup S_2)]$ are equal.

Definition 25. Let $P = \mathcal{P}(S_1, S_2)$ be a piece. The *realizer* $R(P)$ of P is the graph obtained from $G[P]$ by adding all edges between nonadjacent vertices of S_1 and all edges between non adjacent vertices in S_2.

Hence in the realizer, both subsets S_i are cliques.

4 Decomposing pieces $\mathcal{P}(S_1, S_2)$ with $S_1 \not\subseteq S_2$ and $S_2 \not\subseteq S_1$

Consider a piece $P = \mathcal{P}(S_1, S_2)$ with realizer $R(P)$. Assume there is an efficient triangulation $G_{\mathcal{C}}$ with $S_1, S_2 \in \mathcal{C}$ which is an interval graph. In this section we assume that $S_1 \not\subseteq S_2$ and $S_2 \not\subseteq S_1$.

Assume $G_{\mathcal{C}}[P]$ is not a clique, and let x and y be non adjacent vertices in $G_{\mathcal{C}}[P]$. There is a minimal x, y-separator S^* in $G_{\mathcal{C}}$.

In this section we show that S^* decomposes the piece P into smaller pieces and blocks (which are pieces with one separator contained in another one) and S_1, S_2. Blocks are treated in the next section.

Lemma 26. *Using the notation described above:*

1. $S^* \subset P$.
2. S_1 *and* S^* *(and also* S_2 *and* S^**) are non-crossing in* G.
3. $S^* \neq S_1$ *and* $S^* \neq S_2$.

Proof. By Lemma 24 the pieces of S_1 and S_2 in G and G_C are equal which enables us to analyze G_C instead G. By assumption $S_2 \setminus S_1 \neq \emptyset$. Consider the connected component C of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$. Then x and y are both contained in $S_1 \cup C = P$. It follows by Lemma 4 that S^* is also contained in $S_1 \cup C$. Hence $S^* \setminus (S_1 \cup S_2)$ and $S_2 \setminus S_1$ are in the same connected component. In the same way it follows that $S^* \setminus (S_1 \cup S_2)$ and $S_1 \setminus S_2$ are in the same connected component of $G_C[V \setminus S_2]$ It follows that $S^* \subset P$.

Since S_1, S_2 and S^* are minimal separators in G_C they are pairwise non-crossing in G by Lemma 21.

Let C be the connected component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$. Then x and y are both contained in $S_1 \cup C$. It follows that S_1 cannot be a minimal x, y-separator. Hence $S_1 \neq S^*$. □

Lemma 27. *Assume* $S_1 \not\subseteq S^*$ *and* $S_2 \not\subseteq S^*$. *Then* S^* *separates* $S_1 \setminus S^*$ *and* $S_2 \setminus S^*$ *in* G_C.

Proof. Since G_C is an interval graph, there is an consecutive clique arrangement of G_C, say $C_1, \ldots, C_t$. By Lemma 17 there are indices i and j such that $S_1 = C_i \cap C_{i+1}$ and $S_2 = C_j \cap C_{j+1}$. Assume $i < j$. Then the piece of S_1 and S_2 is contained in $\bigcup_{k=i+1}^{j} C_k$. The vertices x and y are in this piece. Hence there is an index $i < k < j$ such that $S^* = C_k \cap C_{k+1}$. Consequently, $S_1 \not\subseteq S^*$ and $S_2 \not\subseteq S^*$ implies that S^* separates $S_1 \setminus S^*$ and $S_2 \setminus S^*$. □

Lemma 28. *Assume* $S_1 \subset S^*$ *and* $S_2 \subset S^*$. *There exist connected components* $D_1, \ldots, D_t$ *of* $G_C[V \setminus S^*]$ *which partition* $P \setminus S^*$.

Proof. Let D be a connected component of $G_C[V \setminus S^*]$. We claim that either $D \subset P$ or $D \cap P = \emptyset$. Indeed, notice that D is connected in $G_C[V \setminus (S_1 \cup S_2)]$. □

Lemma 29. *Let* $S_1 \subset S^*$ *and* $S_2 \setminus S^* \neq \emptyset$. *Then there are connected components* $D_1, \ldots, D_t$ *of* $G_C[V \setminus S^*]$ *such that* $P \setminus S^*$ *can be partitioned into* $\mathcal{P}(S_2, S*)$ *and* $D_1, \ldots, D_t$.

Proof. Consider the connected components of $G_C[V \setminus S^*]$. One of these, say A, contains $S_2 \setminus S^*$. All other components are either completely contained in P or disjoint from it.

Let $z \in P \cap A$. We show that $z \in \mathcal{P}(S_2, S^*)$. If $z \in S_2 \setminus S^*$ this is clear, hence assume $z \in A \setminus S_2$. Since $z \in P$, it is contained in the connected component of $G_C[V \setminus S_2]$ that contains $S_1 \setminus S_2$. But $S^* \subset P$, hence also $S^* \setminus S_2$ is in this component. Hence z is in the component of $G_C[V \setminus S_2]$ that contains $S^* \setminus S_2$. Since z is also in the component of $G_C[V \setminus S^*]$ that contains $S_2 \setminus S^*$, it follows that $z \in \mathcal{P}(S_2, S^*)$.

Finally we have to show that $\mathcal{P}(S^*, S_2) \subseteq P$. Let $z \in \mathcal{P}(S^*, S_2)$. If $z \in S^* \cup S_2$ then clearly $z \in P$. Hence assume $z \notin S^* \cup S_2$. Then $z \in A$. Since $S_1 \subset S^*$, A is contained in the connected component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$.

Also, z is in the connected components of $G_C[V \setminus S_2]$ that contains $S^* \setminus S_2$. This component also contains $S_1 \setminus S_2$. Consequently, $z \in \mathcal{P}(S_1, S_2)$ holds. □

Lemma 30. *Assume $S_1 \setminus S^* \neq \emptyset$ and $S_2 \setminus S^* \neq \emptyset$. Then $S_1 \setminus S^*$ and $S_2 \setminus S^*$ are contained in different connected components of $G_C[V \setminus S^*]$.*

Proof. See the remark above: by the consecutive clique arrangement, S^* separates $S_1 \setminus S^*$ and $S_2 \setminus S^*$. □

Lemma 31. *Assume $S^* \subset S_1$ and $S^* \subset S_2$. Then $P = S_1 \cup S_2$.*

Proof. For $i = 1, 2$ let D_i be the connected component of $G_C[V \setminus S^*]$ that contains $S_i \setminus S^*$. Notice that the connected component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$ is just D_2. Hence $P \setminus (S_1 \cup S_2) = D_1 \cap D_2 = \emptyset$. □

Lemma 32. *Assume $S^* \subset S_1$, $S^* \not\subset S_2$ and $S_2 \not\subset S^*$. Then $P = S_1 \cup \mathcal{P}(S_2, S*)$.*

Proof. For $i = 1, 2$ let D_i be the connected component of $G_C[V \setminus S^*]$ that contains $S_i \setminus S^*$. The connected component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$ is D_2. It follows that $P \setminus S_1 \subseteq D_2$.

Let $z \in P \setminus S_1$. We show that $z \in \mathcal{P}(S^*, S_2)$. By definition z is in the component of $G_C[V \setminus S_2]$ that contains $S_1 \setminus S_2$. This component also contains $S^* \setminus S_2$. Since $z \in D_2$ it follows that $z \in \mathcal{P}(S^*, S_2)$.

It remains to show that $\mathcal{P}(S^*, S_2) \subset P$. Let $z \in \mathcal{P}(S^*, S_2)$. Then z is in D_2. Hence z is in the component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$.

Furthermore, z belongs to the connected component of $G_C[V \setminus S_2]$ that contains $S^* \setminus S_2$. Since $S^* \setminus S_2 \neq \emptyset$, this component is uniquely determined and contains also $S_1 \setminus S_2$, since S_1 is a clique containing S^*. Consequently, $z \in \mathcal{P}(S_1, S_2)$ holds. □

Lemma 33. *Assume for $i = 1, 2$ $S_i \not\subset S^*$ and $S^* \not\subset S_i$. Then there are connected components $D_1, \ldots, D_t$ of $G_C[V \setminus S^*]$ such that P is partitioned into $\mathcal{P}(S_1, S^*)$, $\mathcal{P}(S_2, S^*)$ and $D_1, \ldots, D_t$.*

Proof. Let A and B be the connected components of $G_C[V \setminus S^*]$ which contain $S_1 \setminus S^*$ and $S_2 \setminus S^*$, respectively. Then every other connected component of $G_C[V \setminus S^*]$ is either a subset of P or disjoint from P.

Now let $z \in A \cap P$. We show that $z \in \mathcal{P}(S_1, S^*)$. Since z and $S^* \setminus S_1$ are both in P, and since $S^* \setminus S_1 \neq \emptyset$, it follows that z and $S^* \setminus S_1$ are contained in the same connected component of $G_C[V \setminus S_1]$. Since also $z \in A$ it follows that $z \in \mathcal{P}(S_1, S^*)$.

We now show that $\mathcal{P}(S_1, S^*) \subset P$. Let $z \in \mathcal{P}(S_1, S^*)$. Since $S_1 \setminus S^* \neq \emptyset$ it follows that $z \in S^* \cup A$. If $z \in S^* \cup S_1$ then $z \in P$. Hence we may assume that $z \in A \setminus S_1$.

Now z and $S^* \setminus S_1$ are in the same connected component of $G_C[V \setminus S_1]$ since $z \in \mathcal{P}(S_1, S^*)$. Since $S^* \subset P$ and $S^* \setminus S_1 \neq \emptyset$, this component also contains $S_2 \setminus S_1$. It follows that z and $S_2 \setminus S_1$ are in the same connected component of $G_C[V \setminus S_1]$. This show that $z \in P$ holds. It follows that $A \cap P = \mathcal{P}(S^*, S_1) \setminus S^*$.

$B \cap P = \mathcal{P}(S_2, S^*) \setminus S^*$ can be shown analogously. □

Notice that in all cases, the partition is such that the constituents are (strictly) smaller than the original piece.

5 Decomposing pieces $\mathcal{P}(S_1, S_2)$ with $S_1 \subseteq S_2$

In this section let S_1 and S_2 be non-crossing minimal separators in G with $S_1 \subseteq S_2$. Consider the piece $P = \mathcal{P}(S_1, S_2)$ and the realizer $R(P)$. We show how to compute the treewidth of the realizer.

First assume $S_1 \neq S_2$. Then the piece consists of a minimal separator S_1 and the connected component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$. Notice that in this case, the piece can be partitioned into connected components of $G_C[V \setminus S_2]$.

Now we consider the case $S_1 = S_2$ and denote $S_1 = S_2$ by S. In this case the piece is equal to the total vertex set and the realizer is obtained from G by making a clique of S.

Definition 34. A *block* is a pair $B = (S, C)$, where S is a minimal separator of G and C is a connected component of $G[V \setminus S]$. The graph obtained from $G[S \cup C]$ by making a clique of S is called the *realizer* of the block and is denoted by $R(B)$.

Clearly if we can find the treewidth of all realizers of blocks, then this gives us the treewidth of the total graph: assign to each minimal separator a weight which is the maximum treewidth over all realizers incident with this separators. The treewidth of the graph is equal to the minimum weight over all minimal separators.

Let G_C be an efficient triangulation and let $B = (S, C)$ be a block with realizer R with $S \in C$. Let x and y be non adjacent vertices in $G_C[S \cup C]$. Let S^* be a minimal x, y-separator in G_C, thus S is also a clique in G_C. Then $S^* \subset S \cup C$ by Lemma 4.

Lemma 35. *$S^* \neq S$ and if $S^* \subset S$ then S^* separates $S \setminus S^*$ and C in G_C.*

Proof. Assume $S^* \subseteq S$. If x and y are both contained in C then S^* cannot be a minimal x, y-separator, since C is connected in $G_C[V \setminus S*]$. Thus, w.l.o.g. x is contained in $S \setminus S^*$ and y is contained in C. It follows that $S^* \neq S$.

Now assume some vertex $z \in S \setminus S^*$ has a neighbor in C. Then there is a path from x to y which avoides S^*. □

Lemma 36. *If $S \subset S^*$, then there are connected components $C_1, \ldots, C_t$ of $G_C[V \setminus S^*]$ which partition $C \setminus S^*$.*

Proof. Obvious. □

Lemma 37. *Assume $S \not\subset S^*$ and $S^* \not\subset S$. Then there exist connected components $C_1, \ldots, C_t$ of $G_C[V \setminus S^*]$ such that $S \cup C$ can be partitioned into $\mathcal{P}(S, S^*)$ and $C_1, \ldots C_t$.*

Proof. First we show that $\mathcal{P}(S, S^*) \subset S \cup C$. Let $z \in \mathcal{P}(S, S^*)$. We may assume $z \notin S$. Then z and $S^* \setminus S$ are in the same connected component of $G_C[V \setminus S]$. Since $S^* \setminus S \neq \emptyset$, and since $S^* \subset S \cup C$, it follows that $z \in C$. Since $\mathcal{P}(S, S^*)$ cannot both contain x and y (since $S \setminus S^* \neq \emptyset$), it follows that $\mathcal{P}(S, S^*) \neq S \cup C$.

Notice that in all cases, the partition is such that the constituents are (strictly) smaller than the original piece.

5 Decomposing pieces $\mathcal{P}(S_1, S_2)$ with $S_1 \subseteq S_2$

In this section let S_1 and S_2 be non-crossing minimal separators in G with $S_1 \subseteq S_2$. Consider the piece $P = \mathcal{P}(S_1, S_2)$ and the realizer $R(P)$. We show how to compute the treewidth of the realizer.

First assume $S_1 \neq S_2$. Then the piece consists of a minimal separator S_1 and the connected component of $G_C[V \setminus S_1]$ that contains $S_2 \setminus S_1$. Notice that in this case, the piece can be partitioned into connected components of $G_C[V \setminus S_2]$. on S_2

Now we consider the case $S_1 = S_2$ and denote $S_1 = S_2$ by S. In this case the piece is equal to the total vertex set and the realizer is obtained from G by making a clique of S.

Definition 34. A *block* is a pair $B = (S, C)$, where S is a minimal separator of G and C is a connected component of $G[V \setminus S]$. The graph obtained from $G[S \cup C]$ by making a clique of S is called the *realizer* of the block and is denoted by $R(B)$.

Clearly if we can find the treewidth of all realizers of blocks, then this gives us the treewidth of the total graph: assign to each minimal separator a weight which is the maximum treewidth over all realizers incident with this separators. The treewidth of the graph is equal to the minimum weight over all minimal separators.

Let G_C be an efficient triangulation and let $B = (S, C)$ be a block with realizer R with $S \in C$. Let x and y be non adjacent vertices in $G_C[S \cup C]$. Let S^* be a minimal x,y-separator in G_C, thus S is also a clique in G_C. Then $S^* \subset S \cup C$ by Lemma 4.

Lemma 35. *$S^* \neq S$ and if $S^* \subset S$ then S^* separates $S \setminus S^*$ and C in G_C.*

Proof. Assume $S^* \subseteq S$. If x and y are both contained in C then S^* cannot be a minimal x,y-separator. since C is connected in $G_{\mathcal{C}}[V \setminus S*]$. Thus, w.l.o.g. x is contained in $S \setminus S^*$ and y is contained in C. It follows that $S^* \neq S$.

Now assume some vertex $z \in S \setminus S^*$ has a neighbor in C. Then there is a path from x to y which avoides S^*. □

Lemma 36. *If $S \subset S^*$, then there are connected components $C_1, \dots, C_t$ of $G_C[V \setminus S^*]$ which partition $C \setminus S^*$.*

Proof. Obvious. □

Lemma 37. *Assume $S \not\subset S^*$ and $S^* \not\subset S$. Then there exist connected components $C_1, \dots, C_t$ of $G_C[V \setminus S^*]$ such that $S \cup C$ can be partitioned into $\mathcal{P}(S, S^*)$ and $C_1, \dots C_t$.*

Proof. First we show that $\mathcal{P}(S, S^*) \subset S \cup C$. Let $z \in \mathcal{P}(S, S^*)$. We may assume $z \notin S$. Then z and $S^* \setminus S$ are in the same connected component of $G_C[V \setminus S]$. Since $S^* \setminus S \neq \emptyset$, and since $S^* \subset S \cup C$, it follows that $z \in C$. Since $\mathcal{P}(S, S^*)$ cannot both contain x and y (since $S \setminus S^* \neq \emptyset$), it follows that $\mathcal{P}(S, S^*) \neq S \cup C$.

Since $S \not\subset S^*$ there is exactly one connected components of $G_C[V \setminus S^*]$ that contains $S \setminus S^*$. Moreover, $\mathcal{P}(S, S^*) \setminus S^*$ is contained in that component. Let A be a component of $G_C[C \setminus S^*]$. Assume that A is contained in a connected component B of $G_C[V \setminus S^*]$. If $A \neq B$, then B must contain vertices of $S \setminus S^*$. In that case however, $A \subset \mathcal{P}(S, S^*)$. □

6 The algorithm

Using the result of [19] we can find all minimal separators in the graph. For each pair of non-crossing separators, we can compute the piece. We also compute the blocks for every separator. We sort the pieces and the blocks according to increasing number of vertices. Blocks and pieces with the same number of vertices are ordered as follows. Blocks appear in the ordering before pieces with the same number of vertices. If two blocks have the same number of vertices, the block with the largest number of vertices in the separator appears before the block with the smaller number of vertices in the separator. Blocks with the same number of vertices in total and the same number of vertices in the separator can be ordered arbitrarily.

For each block and piece in turn, we compute the treewidth of the realizer, by trying all possible separators that are contained in it, using the results of sections 4 and 5. If the treewidth of each piece is determined, we look for the piece with vertex set V, with minimum treewidth. This is equal to the treewidth of the graph.

Theorem 38. *For each constant d there exists a polynomial time algorithm that computes the treewidth and pathwidth of cocomparability graphs of dimension at most d.*

Proof. Let R be the number of separators. In [4] it is shown that $R \leq (n+1)^d$. Moreover, in [19] it is shown that the set of all separators can be computed in $O(n^6 R)$ time. There are at most R^2 pieces, since these are fully characterized by two minimal separators. For each of these pieces, we can try all minimal separators to split up the piece. For each smaller piece we can look up its treewidth in $O(n^2)$ time. It follows that we can find the treewidth of a piece in $O(Rn^3)$ time. □

7 Conclusions

We believe that our approach which could be called the *minimal separator approach* is not at all restricted to the problems considered in this paper. In fact, the approach has already been used for designing polynomial time algorithms solving the vertex ranking problem (which is equivalent to the minimum elimination tree height problem) [7].

On the other hand, the PATHWIDTH problem which is closely related to the TREEWIDTH problem remains intractable even when restricted to graph classes with a polynomially bounded number of minimal separators.

Fortunately, Möhring has shown that for every AT-free graph G the treewidth of G is equal to the pathwidth of G and also the minimum fill-in of G is equal to the interval completion of G [20]. Hence, on subclasses of AT-free graphs as e.g.

cocomparability graphs and trapezoid graphs the algorithmic complexities of the corresponding problems coincides and a treewidth (resp. minimum fill-in) algorithm on the class is at the same time a pathwidth (resp. interval completion) algorithm.

8 Acknowledgements

One of the authors thanks L. Bijlsma for pointing out some errors in the original manuscript.

References

1. Anand, R., H. Balakrishnan and C. Pandu Rangan, Treewidth of distance hereditary graphs, To appear.
2. Arnborg, S., D. G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.* 8, (1987), pp. 277–284.
3. Bodlaender, H., A linear time algorithm for finding tree-decompositions of small treewidth, In *Proceedings of the* 25^{th} *Annual ACM Symposium on Theory of Computing*, 1993, pp. 226–234.
4. Bodlaender, H., T. Kloks and D. Kratsch, Treewidth and pathwidth of permutation graphs, *Proceedings of the* 20^{th} *International Colloquium on Automata, Languages and Programming*, pp. 114–125, Springer Verlag, Lecture Notes in Computer Science, vol. 700, 1993.
5. Bodlaender, H. and R. H. Möhring, The pathwidth and treewidth of cographs, In *Proceedings* 2^{nd} *Scandinavian Workshop on Algorithm Theory*, Springer Verlag, Lecture Notes in Computer Science 447, (1990), pp. 301–309.
6. Brandstädt, A., Special graph classes — a survey, Schriftenreihe des Fachbereichs Mathematik, SM-DU-199 (1991), Universität-Gesamthochschule Duisburg.
7. Deogun, J. S., T. Kloks, D. Kratsch and H. Müller, On vertex ranking for permutation and other graphs, Computing Science Notes 93/30, Eindhoven University of Technology, Eindhoven, The Netherlands, (1993). To appear in: Proceedings of 11^{th} *Annual Symposium on Theoretical Aspects of Computer Science*.
8. Dirac, G. A., On rigid circuit graphs, *Abh. Math. Sem. Univ. Hamburg* **25**, (1961), pp. 71–76.
9. Gilmore, P. C. and A. J. Hoffman, A characterization of comparability graphs and interval graphs, *Canad. J. Math.* **16**, (1964), pp. 539–548.
10. Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
11. Golumbic, M. C., D. Rotem, J. Urrutia, Comparability graphs and intersection graphs, *Discrete Mathematics* **43**, (1983), pp. 37–46.
12. Gustedt, J., On the pathwidth of chordal graphs, To appear in *Discrete Applied Mathematics*.
13. Habib, M. and R. H. Möhring, Treewidth of cocomparability graphs and a new order-theoretic parameter, Technical Report 336/1992, Technische Universität Berlin, 1992.
14. Johnson, D. S., The NP-completeness column: An ongoing guide, *J. Algorithms* **6**, (1985), pp. 434–451.
15. Kloks, T., *Treewidth*, Ph.D. Thesis, Utrecht University, The Netherlands, 1993.
16. Kloks, T., Treewidth of circle graphs, Technical Report RUU-CS-93-12, Department of Computer Science, Utrecht University, 1993. To appear in: Proc. ISAAC'93.

17. Kloks, T., H. Bodlaender, H. Müller and D. Kratsch, Computing treewidth and minimum fill-in: all you need are the minimal separators, *Proc. of the First Annual European Symposium on Algorithms*, pp. 260–271, Springer Verlag, Lecture Notes in Computer Science, vol. 726 (1993).
18. Kloks, T. and D. Kratsch, Treewidth of chordal bipartite graphs, *10^{th} Annual Symposium on Theoretical Aspects of Computer Science*, Springer-Verlag, Lecture Notes in Computer Science 665, (1993), pp. 80–89.
19. Kloks, T. and D. Kratsch, Finding all minimal separators of a graph, Computing Science Note, 93/27, Eindhoven University of Technology, Eindhoven, The Netherlands, (1993). To appear in Proc. STACS'94.
20. Möhring, R. H., private communication.
21. Sundaram, R., K. Sher Singh and C. Pandu Rangan, Treewidth of circular arc graphs, To appear in *SIAM Journal on Discrete Mathematics*.
22. Yannakakis, M., The complexity of the partial order dimension problem, *SIAM J. Alg. Discrete Methods* **3**, (1982), pp. 351–358.

This article was processed using the LaTeX macro package with LLNCS style

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.

91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.
91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.

92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.
92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.

93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness?, p. 24.
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.
93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.

93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.