Finding all minimal separators of a graph

Eindhoven University of Technology

Department of Mathematics and Computing Science

Finding all minimal separators of a graph

by

T. Kloks and D. Kratsch

93/27

Computing Science Note 93/27
Eindhoven, August 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

editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Finding all minimal separators of a graph

T. Kloks *
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O.Box 513
5600 MB Eindhoven, The Netherlands

D. Kratsch
Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität
Universitätshochhaus, 07740 Jena, Germany

Abstract

In this paper we give an efficient algorithm to find all minimal vertex separators of an undirected graph. The algorithm needs polynomial time per separator that is found.

1 Introduction

Given a graph, one is often interested in finding subsets of vertices, or their cardinality, or a certain partition of the vertices, which possess a certain property. For example the CLIQUE NUMBER of a graph G is the maximum cardinality of a subset S such that $G[S]$ is complete. Similar questions are the INDEPENDENCE NUMBER, the DOMINATION NUMBER or the CHROMATIC NUMBER. For many of these problems, it would be convenient if one could use a decomposition of the graph by means of certain *separators*.

This is perhaps best illustrated by the recent results for classes of graphs with bounded treewidth. For these classes, linear time algorithms exist for many NP-complete problems *exactly* because a decomposition can be made using *separators of bounded size* [1, 2, 3, 4, 10]. A decomposition of this type can be found in linear time [5, 10], however the huge constants involved in these algorithms do not make them of much practical use. Our results show that for many classes of graphs *efficient* decomposition algorithms exist, i.e., the size of the separators has no effect on the running time.

A closely related, but somewhat different approach was surveyed in [12]. In this paper (see also [7]) it is shown that for many classes of graphs (for example

*Email: ton@win.tue.nl.

chordal graphs, clique separable graphs and edge intersection graphs of paths in a tree or EPT-graphs) a decomposition by *clique separators* is possible, and it is illustrated that such a decomposition can also be used to solve efficiently many NP-complete problems like MINIMUM FILL-IN, MAXIMUM CLIQUE, GRAPH COLORING and MAXIMUM INDEPENDENT SET. In [13] an algorithm is given for finding clique separators efficiently (the algorithm uses $O(nm)$ time to find one clique separator). Our results (combined with the result of [11]) generalize the above mentioned results in the sense that at least some of these NP-complete problems are solvable for much more graph classes, i.e., graph classes for which the number of minimal separators is polynomial bounded.

In [9] an algorithm is given which finds all, what the author calls *minimum size separators*. By this is meant that, given a graph which is k-connected, the algorithm finds all separators with k vertices. Moreover, it is shown in this paper that the number of these separators is bounded by $O(2^k \frac{n^2}{k})$. The algorithm which lists all minimum size separators runs in time $O(2^k n^3)$ time.

We call a subset of vertices S a minimal separator if there are non adjacent vertices x and y such that the removal of S separates x and y into disjoint connected components in such a way that no proper subset of S also does this (see Definition 2.1). A closely related concept which we call *inclusion minimal separators* lies more or less between the minimum size separators and the minimal separators, i.e., all minimum size separators are inclusion minimal and all inclusion minimal separators are minimal separators.

The following example shows that the minimum size separators and the inclusion minimal separators are only of limited use. Consider any graph G. Take a new vertex x and make this adjacent to all vertices of G. Take another new vertex y and make this adjacent to x. Call this new graph H. The only inclusion minimal separator of H, which is also the only minimum size separator, is $\{x\}$. However if S is some minimal separator of G, then $S \cup \{x\}$ is a minimal separator in H. Hence H has at least as many minimal separators as G.

In [6, 10, 11] it is shown that many important classes of graphs have a polynomial number of minimal vertex separators. These graph classes include permutation graphs, circular permutation graphs, trapezoid graphs, circle graphs, circular arc graphs, distance hereditary graphs, chordal bipartite graphs, cocomparability graphs of bounded dimension and weakly triangulated graphs. In [11] it is shown that if, for a certain class of graphs, all minimal separators can be computed in polynomial time, then the problems TREEWIDTH and MINIMUM FILL-IN can be solved in polynomial time for graphs in this class. In this paper we present an algorithm to compute all minimal vertex separators.

2 Preliminaries

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. For a vertex

$x \in V$ we use $N(x)$ to denote the neighborhood of x.

The following definition can be found for instance in [8].

Definition 2.1 *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 [8]. It provides an easy test whether a given set S of vertices is a minimal separator or not.

Lemma 2.1 *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 $G[V - S]$ 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. Let $x \in S$. Since S is a *minimal* a, b-separator, there is a path between a and b passing through x but using no other vertex in S. Hence x must have a neighbor in C_a and in C_b.

Now let S be a separator and let C_a and C_b be two connected components such that every vertex of S has a neighbor in C_a and in C_b. Let $a \in C_a$ and $b \in C_b$. Then clearly S is a minimal a, b-separator, for if $x \in S$, then there is a path between a and b which uses no vertices of $S \setminus \{x\}$. Hence $S \setminus \{x\}$ is not an a, b-separator. □

Notice that this also proves the following. Let S be a minimal separator and let C_1 and C_2 be two connected components of $G[V - S]$ such that every vertex of S has a neighbor in both C_1 and C_2. If a is a vertex of C_1 and b is a vertex of C_2 then S is a minimal a, b-separator.

It may be a bit surprising at first sight that it is very well possible for one minimal separator to be contained in another one. An example of this can be found in [8]. However, for minimal a, b-separators things are different, since by definition one minimal a, b-separator cannot be properly contained in another one.

We now show that at least some of the minimal separators are easy to find.

Definition 2.2 *Let a and b be non adjacent vertices. If S is a minimal a, b-separator such that $S \subseteq N(a)$ then S is called* close to a.

Lemma 2.2 *If a and b are non adjacent then there exists exactly one minimal a, b-separator close to a.*

Proof. Let S be a minimal a, b-separator close to a. We show that S is exactly the set of neighbors of a for which there is a path to b using no other neighbors of a.

For every vertex in S there is a path to b which does not use any other vertex of $N(a)$, since S is minimal. On the other hand, if x is a neighbor of a such that there is a path to b without any other vertex of $N(a)$, then x must be an element of S, otherwise there is a path between x and b which avoids S and this is a contradiction since x is in the component of $G[V - S]$ that contains a. □

Notice that a minimal separator close to a can easily be computed as follows. Start with $S = N(a)$. Clearly, since a and b are non adjacent S separates a and b. Let C_b be the connected component of $G[V - S]$ containing b. Let $S' \subseteq S$ be the set of those vertices of S which have at least one neighbor in C_b. S' is a minimal a, b-separator by Lemma 2.1, and since it only contains neighbors of a, it is close to a.

Lemma 2.3 *Let S be the minimal a, b-separator close to a and let C_a and C_b be the connected components containing a and b respectively. Let $S^* \neq S$ be another minimal a, b-separator. Then $S^* \subset S \cup C_b$.*

Proof. First assume S^* has a vertex $x \notin S \cup C_a \cup C_b$. Then there is a path P from a to b passing through x but using no other vertices of S^*. But then P has to pass through S at least twice. Clearly P can be shortened since S contains only neighbors of a.

Now assume S^* has a vertex $x \in C_a$. $S^* \setminus \{x\}$ does not separate a and b hence there is a path P between a and b using x but no other vertex of S^*. Since S is a minimal separator, P goes through a vertex $y \in S$. Since S is close to a, y is adjacent to a. Hence there is a path $P' \subset P$ between a and b that does not contain x. Then P' contains no vertex of S^*. □

In the next two sections we show how to obtain new minimal a, b-separators from a given one using so called minimal pairs. A minimal pair is in some sense the smallest step to go from one minimal a, b-separator to the next one. The main difficulty is to prove that we indeed obtain all minimal separators by using small steps only.

In section 5 we describe an algorithm that computes, for a given pair of non adjacent vertices a and b, all minimal a, b-separators in a breadth-first-search manner (Figure 1, page 9), we prove that it is correct and we analyse its time complexity. We end with some concluding remarks and some open problems.

3 Good pairs

Let $G = (V, E)$ be a graph and let a and b be non adjacent vertices in G. Let S be a minimal a, b-separator and let C_a and C_b be the connected components containing a and b respectively.

Definition 3.1 *Let $\Delta \subseteq C_a \setminus \{a\}$ and let C'_a be the connected component of $G[C_a - \Delta]$ that contains a. Let $N \subseteq S$ be the set of vertices in S that do not have a neighbor in C'_a. The pair (Δ, N) is called* good *for S if the following conditions are satisfied.*

1. *$N \neq \emptyset$.*
2. *Each $\delta \in \Delta$ has at least one neighbor in C'_a.*
3. *Each $\delta \in \Delta$ either has a neighbor in N or there exists a vertex $x \in N$ and a connected component D of $G[C_a - \Delta]$ such that both x and δ have at least one neighbor in D.*

Lemma 3.1 *If S is close to a then there is no good pair.*

Proof. Assume (Δ, N) is a good pair. Hence $\Delta \subseteq C_a \setminus \{a\}$. Let C'_a be the connected component of $G[C_a - \Delta]$ that contains a. The set N is defined as the subset of S that does not contain any neighbor in C'_a. Then $N = \emptyset$ since S contains only neighbors of a. But by definition $N \neq \emptyset$. □

In Theorem 3.1 we show that a good pair defines a new separator. In Theorem 3.2 we show that *each* minimal a,b-separator can be obtained by a good pair for the separator that is close to b. In section 4 we show that only a restricted type of good pairs, called minimal pairs, have to be considered.

Theorem 3.1 *Let (Δ, N) be a good pair. Define $S^* = (S \cup \Delta) \setminus N$. Then S^* is a minimal a,b-separator.*

Proof. Let C'_a be the connected component of $G[C_a - \Delta]$ that contains a. Clearly, S^* separates a and b, since vertices of N do not have neighbors in C'_a. Let C'_b be the connected component of $G[V - S^*]$ that contains b. Notice that $C_b \subset C'_b$, and since each vertex of N has a neighbor in C_b, $N \subset C'_b$.

Each vertex of S^* has at least one neighbor in C'_a by definition of a good pair, and each vertex of $S^* \setminus \Delta$ has at least one neighbor in C'_b since it has at least one neighbor in C_b. The only thing left to show is that each vertex of Δ has a neighbor in C'_b. Let $\delta \in \Delta$. By definition, either δ has a neighbor in N (and hence in C'_b) or there is a vertex $x \in N$ and a connected component D of $G[C_a - \Delta]$ such that both δ and x have a neighbor in D. D is also connected in $G[V - S^*]$ and since x has a neighbor in D, $D \subset C'_b$. □

Theorem 3.2 *Assume S is close to b. Let $S^* \neq S$ be a minimal a,b-separator. There exists a good pair (Δ, N) such that $S^* = (S \cup \Delta) \setminus N$.*

Proof. Let C^*_a and C^*_b be the connected components of $G[V - S^*]$ containing a and b respectively.

Since S is close to b, by Lemma 2.3, $S^* \subset S \cup C_a$. Let $\Delta = S^* \cap C_a$ and $N = S \setminus S^*$. We show that (Δ, N) is a good pair.

Since $S^* \neq S$ and both are minimal a,b-separators: $N \neq \emptyset$.

Let C'_a be the connected component of $G[C_a - \Delta]$ containing a. We show that N is exactly the set of vertices in S which do not have a neighbor in C'_a. In order to do this we claim that $C'_a = C^*_a$. Since C'_a is a connected component of $G[V - (\Delta \cup S)]$ and since $S^* \subset \Delta \cup S$, $C'_a \subseteq C^*_a$. Now assume there is a vertex $x \in N$ which has a neighbor $y \in C'_a$. Since S is close to b, x is a neighbor of b. This is a contradiction since there would be a path between a and b which does not use any vertex of S^*. This shows that $C'_a = C^*_a$. Since S^* is minimal, N is exactly the set of vertices in S that do not have a neighbor in C'_a. It now also follows that every vertex of $\Delta \cup (S \setminus N)$ has at least one neighbor in C'_a.

To prove the last item first notice that $N \subset C^*_b$ and that C^*_b contains exactly those connected components D of $G[C_a - \Delta]$ for which there is a vertex $y \in N$ which has a neighbor in D. Now let $\delta \in \Delta$. Since S^* is minimal, δ has a neighbor x in C^*_b. Since δ only has neighbors in $C_a \cup S$, x must be an element of N or of some component D of $G[C_a - \Delta]$. In this second case, there must also be a vertex $y \in N$ which has a neighbor in D. □

4 Minimal pairs

Again let $G = (V, E)$ be a graph and let a and b be non adjacent vertices in G. Let S be a minimal a,b-separator and let C_a and C_b be the connected components of $G[V - S]$ containing a and b respectively. In this section we show how to find some good pairs.

Definition 4.1 *Let $x \in S$ be non adjacent to a. Let $C_a(x)$ be the subgraph induced by $C_a \cup \{x\}$. Let Δ be the minimal x,a-separator in $C_a(x)$ close to x, and let C'_a be the connected component containing a in $C_a(x)$ when Δ is removed. Now let N be the set of vertices of S which do not have a neighbor in C'_a. The pair (Δ, N) is called the* minimal pair *for S and x.*

Lemma 4.1 *A minimal pair is good.*

Proof. Notice that $x \in N$, hence $N \neq \emptyset$.

Now, Δ is a minimal x,a-separator in $C_a(x)$ and hence every vertex of Δ has a neighbor in C'_a.

Finally, if $\delta \in \Delta$ then δ is adjacent to x since Δ is close to x. Hence each vertex of Δ has a neighbor in N. □

We want to prove that we can find every minimal a,b-separator by starting with the minimal a,b-separator that is close to b and by recursively using minimal pairs. The following rather technical lemma proves this.

Lemma 4.2 *Let (Δ, N) be a good pair for S. Let $x \in N$ and let (Δ^*, N^*) be the minimal pair for S and x. Let $S^* = (S \cup \Delta^*) \setminus N^*$. Define $\overline{\Delta} = \Delta \setminus \Delta^*$ and $\overline{N} = (N \setminus N^*) \cup (\Delta^* \setminus \Delta)$. Then:*

1. *if $\overline{N} = \emptyset$ then $(S \cup \Delta) \setminus N = S^*$, and if*
2. *$\overline{N} \neq \emptyset$ then $(\overline{\Delta}, \overline{N})$ is a good pair for S^* and $(S \cup \Delta) \setminus N = (S^* \cup \overline{\Delta}) \setminus \overline{N}$.*

Proof. We start with some easy observations. Let C'_a be the connected component of $G[C_a - \Delta]$ that contains a and let C^*_a be the connected component of $G[C_a - \Delta^*]$ that contains a. Let $\Delta' = N(x) \cap \Delta$.

- $C'_a \subseteq C^*_a$ since Δ^* contains no vertices of C'_a.
- $\Delta' \subseteq \Delta^*$ since every vertex of Δ' has a neighbor in C'_a.
- $\Delta \setminus \Delta' \subseteq C^*_a$ since every vertex of Δ has a neighbor in C'_a.
- $N^* \subseteq N$, since $C'_a \subseteq C^*_a$.
- C'_a is exactly the connected component of $G[C^*_a - (\Delta \setminus \Delta')]$ containing a since $C^*_a - (\Delta \setminus \Delta')$ contains all vertices of C'_a but no vertex of Δ.
- The set of vertices in S^* without a neighbor in C'_a is exactly $\overline{N}$, which is easy to check.

Assume $\overline{N} = \emptyset$. Then $\Delta^* \subseteq \Delta$ and $N = N^*$ (since $N^* \subseteq N$). Now clearly, $\Delta^* = \Delta$ holds, otherwise S^* and $(S \cup \Delta) \setminus N$ are two minimal a, b-separators of which one is properly contained in the other which is impossible by definition. Hence $S^* = (S \cup \Delta) \setminus N$.

Now assume $\overline{N} \neq \emptyset$. We show that $(\overline{\Delta}, \overline{N})$ is good for S^*. Notice that every vertex of $\overline{\Delta}$ has a neighbor in C'_a, since this holds for every vertex of Δ.

Let $\delta \in \overline{\Delta}$ and assume that δ has no neighbors in $\overline{N}$. Since $\delta \in C^*_a$, δ has no neighbor in N^*. Hence δ has no neighbor in N. Now (Δ, N) is a good pair, hence there is a vertex $z \in N$ and a connected component D of $G[C_a - \Delta]$ such that δ and z have a neighbor in D.

Suppose that for no vertex of $\overline{N}$ there is a connected component in $G[C^*_a - \overline{\Delta}]$ such that this vertex and δ both have a neighbor in this component. The following observations lead to a contradiction.

- $N(\delta) \cap D \subseteq C^*_a$. Otherwise, since $\Delta^* \setminus \Delta' \subset \overline{N}$, δ has a neighbor in $\overline{N}$.
- $G[D \setminus \Delta^*]$ is connected. Since otherwise every connected component has a vertex with a neighbor in $\Delta^* \setminus \Delta$, and hence there is a connected component and some vertex in $\overline{N}$ such that this vertex and δ both have a neighbor in this component.
- D contains no vertices of Δ^*, by the same argument.

This shows that $D \subset C_a^*$. If $z \in N^*$ then z can have no neighbors in D, since z has no neighbors in C_a^*. Hence $z \in N \setminus N^*$. This is a contradiction, since now there is a connected component D in $G[C_a^* - \overline{\Delta}]$ and a vertex $z \in \overline{N}$ such that z and δ both have a neighbor in D.

The fact that $(S \cup \Delta) \setminus N = (S^* \cup \overline{\Delta}) \setminus \overline{N}$ is obvious. □

This lemma shows that if S is a minimal separator and S^* is another minimal separator defined by a good pair for S, then we can get closer to S^* by choosing a suitable minimal pair. By Lemma 4.2 and Theorem 3.2 we obtain the following result.

Corollary 4.1 *Let S be a minimal a,b-separator and let S_1 be the minimal a,b-separator close to b. There exists a sequence $(\Delta_1, N_1), \ldots, (\Delta_t, N_t)$ such that*

1. *(Δ_1, N_1) is a minimal pair for S_1 and some vertex $x_1 \in N_1$.*
2. *For $i = 2, \ldots, t$, (Δ_i, N_i) is a minimal pair for $S_i = (S_{i-1} \cup \Delta_{i-1}) \setminus N_{i-1}$ and some vertex $x_i \in N_i$.*
3. *$S = (S_t \cup \Delta_t) \setminus N_t$.*

5 An algorithm finding minimal separators

In this section we give an algorithm that, given a graph G and two non adjacent vertices a and b finds all minimal a,b-separators. This algorithm is displayed in figure 1 on page 9.

Theorem 5.1 *Let S be the minimal a,b-separator that is close to b and let $\mathcal{T} = \{S\}$ and $\mathcal{Q} = \{S\}$. Then a call separators$(G, a, b, \mathcal{T}, \mathcal{Q})$ determines a set $\mathcal{Q}$ containing all minimal a,b-separators.*

Proof. By Corollary 4.1 the set $\mathcal{Q}$ contains all minimal a,b-separators. By Lemma 4.1 and Theorem 3.1 all sets in $\mathcal{Q}$ are minimal separators. □

Remark 5.1 *If we let $\mathcal{T} = \{\{b\}\}$ and $\mathcal{Q} = \emptyset$ then a call separators$(G, a, b, \mathcal{T}, \mathcal{Q})$ has the same result.*

Theorem 5.2 *Let R be the number of minimal a,b-separators (for non adjacent vertices a and b). The algorithm to determine all minimal a,b-separators can be implemented to run in time $O(n^4 R)$.*

Proof. Assume that the graph is given with an adjacency matrix. The minimal separator S that is close to b can easily be found in $O(n^2)$ time as follows. Initialize $S = N(b)$. Determine the connected component C_a of $G[V - S]$. Remove vertices from S that do not have a neighbor in C_a.

procedure separators($G, a, b, \mathcal{T}, \mathcal{Q}$)
input: Graph G and non adjacent vertices a and b and
sets $\mathcal{T}$ and $\mathcal{S}$ of minimal a, b-separators.
output: Set $\mathcal{Q}$ of all minimal a, b-separators in G.
begin
$\mathcal{T}' := \emptyset$;
for each $S \in \mathcal{T}$ **do**
begin
Determine C_a;
{C_a is the connected component of $G[V - S]$ that contains a}
for each $x \in S$ which is not adjacent to a **do**
begin
Determine Δ;
{Δ is the minimal x, a-separator in $C_a(x)$ that is close to x}
Determine C'_a;
{C'_a is the connected component of $G[C_a - \Delta]$ that contains a}
Determine N;
{N is the set of vertices in S that do not have a neighbor in C'_a}
$S^* := (S \cup \Delta) \setminus N$;
$\mathcal{T}' := \mathcal{T}' \cup \{S^*\}$
{Add S^* to $\mathcal{T}'$ only if not yet present!}
end for
end for;
$\mathcal{Q} := \mathcal{Q} \cup \mathcal{T}'$;
separators($G, a, b, \mathcal{T}', \mathcal{Q}$)
end.

Figure 1: Algorithm finding minimal a, b-separators

First we show that the outermost loop of the procedure separators is executed at most n times. If the outer loop is executed for the i^{th} time, for each separator S in the set $\mathcal{T}$, the connected component of $G[V - S]$ that contains a has at most $n - i$ vertices.

Since the set $\mathcal{T}$ contains only different minimal separators, the second loop is executed at most R time. Clearly, each separator has at most $O(n)$ elements.

Determining Δ takes at most $O(n^2)$ time. Also computing C'_a and N can clearly be done in $O(n^2)$ time. We have to make sure that the new set $\mathcal{T}'$ contains no duplicate separators. We can do this by keeping it in a suitable data structure, allowing an update in $O(n \log R) = O(n^2)$ time.

This shows that the algorithm can be implemented to run in $O(n^4 R)$ time. □

Corollary 5.1 *The set of all minimal separators of a graph can be found in $O(n^6 R)$ time, where n is the number of vertices in the graph and R is the total number of minimal separators.*

6 Conclusions

In this paper we have presented an algorithm to determine a list of all minimal vertex separators of a graph. The algorithm needs only polynomial time per separator that is found. We like to mention some open problems.

First of all, we feel that it should be possible to improve the running time of the algorithm presented here.

A related concept is that of an *inclusion minimal separator*. This is a minimal separator with the additional constraint that no proper subset is also a minimal separator. The following lemma shows that our algorithm can be used to find all inclusion minimal separators. However, the example given in the introduction illustrates that this may not be the most efficient way to do this.

Lemma 6.1 *A separator S of a graph $G = (V, E)$ is inclusion minimal if and only if every vertex of S has a neighbor in every connected component of $G[V - S]$.*

It follows that a list of all inclusion minimal separators can easily be obtained from the list of all minimal separators. Until now, we have not been able to find an efficient algorithm which finds all inclusion minimal separators.

7 Acknowledgement

We thank B. Monien for drawing our attention to this important problem and A. Kaldewaij for his careful reading of the manuscript.

References

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

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

[3] Arnborg, S. and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, *Disc. Appl. Math.* **23**, (1989), pp. 305–314.

[4] Bodlaender, H., A tourist guide through treewidth, Technical report RUU-CS-92-12, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, (1992).

[5] Bodlaender, H., A linear time algorithm for finding tree-decompositions of small treewidth, *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, (1993), pp. 226–234.

[6] 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*, Springer-Verlag, Lecture Notes in Computer Science 700, (1993), pp. 114–125.

[7] Gavril, F., Algorithms on clique separable graphs, *Discrete Math.* **19** (1977), pp. 159–165.

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

[9] Kanevsky, A., On the number of minimum size separating vertex sets in a graph and how to find all of them, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 411–421, (1990).

[10] Kloks, T., *Treewidth*, Ph.D. Thesis, Utrecht University, The Netherlands, 1993.

[11] Kloks, T., H. Bodlaender, H. Müller and D. Kratsch, Computing treewidth and minimum fill-in: all you need are the minimal separators. To appear in *Proceedings of the First Annual European Symposium on Algorithms*, (1993).

[12] Tarjan, R. E., Decomposition by clique separators, *Discrete Mathematics* **55** (1985), pp. 221–232.

[13] Whitesides, S. H., An Algorithm for finding clique cut-sets, *Information Processing Letters* **12** (1981), pp. 31–32.

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.

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.
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.