Finding all minimal separators of a graph

Citation for published version (APA):

Document status and date:
Published: 01/01/1993

Publisher Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 04. Sep. 2020
Finding all minimal separators of a graph

by

T. Kloks and D. Kratsch
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

All rights reserved
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 car­

dinality, or a certain partition of the vertices, which possess a certain property.
For example the \textbf{clique number} of a graph \( G \) is the maximum cardinality of a
subset \( S \) such that \( G[S] \) is complete. Similar questions are the \textbf{independence}
number, the \textbf{domination number} or the \textbf{chromatic number}. For many of
these problems, it would be convenient if one could use a decomposition of the

graph by means of certain \textit{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 \textsc{np}-complete problems exactly because a decomposition can be made
using \textit{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 \textit{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 $O(2^k n^2)$. 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, comparability 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
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. \qed

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 \( S^* \subseteq S \cup C_b \).

**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 \( P' \subseteq P \) 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 $(\Delta, N)$ 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 $\delta$ have at least one neighbor in $D$.

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

Proof. Assume $(\Delta, N)$ 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 $(\Delta, N)$ 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 $C'_b \subseteq C'_b$, and since each vertex of $N$ has a neighbor in $C'_b$, $N \subseteq C'_b$.

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 $\Delta$ has a neighbor in $C'_b$. Let $\delta \in \Delta$. By definition, either $\delta$ 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 $x$ and $\delta$ have a neighbor in $D$. $D$ is also connected in $G[V - S^*]$ and since $x$ has a neighbor in $D$, $D \subseteq C'_b$.

Theorem 3.2 Assume $S$ is close to $b$. Let $S^* \neq S$ be a minimal $a, b$-separator. There exists a good pair $(\Delta, N)$ 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, $S^* \subseteq S \cup C_a$. Let $\Delta = S^* \cap C_a$ and $N = S \setminus S^*$. We show that $(\Delta, N)$ 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 $S^* \subseteq \Delta \cup S$, $C_a' \subseteq C_a^*$. 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 $N \subseteq C_a^*$ and that $C_a^*$ 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, $\delta$ has a neighbor $x$ in $C_a^*$. Since $\delta$ 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 $D$. \hfill \Box

\section{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 $\Delta$ 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 $\Delta$ is removed. Now let $N$ be the set of vertices of $S$ which do not have a neighbor in $C_a'$. The pair $(\Delta, N)$ 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, $\Delta$ is a minimal $x,a$-separator in $C_a(x)$ and hence every vertex of $\Delta$ has a neighbor in $C_a'$.

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

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 \((\Delta, N)\) be a good pair for \(S\). Let \(x \in N\) and let \((\Delta^*, N^*)\) 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 \((\Delta, \overline{N})\) 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 \(\Delta^*\) contains no vertices of \(C_a'\).
- \(\Delta' \subseteq \Delta^*\) since every vertex of \(\Delta'\) has a neighbor in \(C_a'\).
- \(\Delta \setminus \Delta' \subseteq C_a^*\) since every vertex of \(\Delta\) 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 \(\Delta\).
- 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 \(\overline{N} = N^*\) (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 \((\Delta, \overline{N})\) is good for \(S^*\). Notice that every vertex of \(\overline{\Delta}\) has a neighbor in \(C_a^*\), since this holds for every vertex of \(\Delta\).

Let \(\delta \in \Delta^*\) and assume that \(\delta\) has no neighbors in \(\overline{N}\). Since \(\delta \in C_a^*\), \(\delta\) has no neighbor in \(N^*\). Hence \(\delta\) has no neighbor in \(N\). Now \((\Delta, \overline{N})\) is a good pair, hence there is a vertex \(z \in N\) and a connected component \(D\) of \(G[C_a - \Delta]\) such that \(\delta\) 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 \(\delta\) both have a neighbor in this component. The following observations lead to a contradiction.

- \(N(\delta) \cap D \subseteq C_a^*\). Otherwise, since \(\Delta^* \setminus \Delta' \subseteq \overline{N}\), \(\delta\) 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 \(\delta\) both have a neighbor in this component.
- \(D\) contains no vertices of \(\Delta^*\), by the same argument.
This shows that $D \subset C_\ast^\ast$. If $z \in N^\ast$ then $z$ can have no neighbors in $D$, since $z$ has no neighbors in $C_\ast^\ast$. Hence $z \in N \setminus N^\ast$. This is a contradiction, since now there is a connected component $D$ in $G[V - \overline{S}]$ and a vertex $z \in \overline{N}$ such that $z$ and $\delta$ both have a neighbor in $D$.

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

This lemma shows that if $S$ is a minimal separator and $S^\ast$ is another minimal separator defined by a good pair for $S$, then we can get closer to $S^\ast$ 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. $(\Delta_1, N_1)$ is a minimal pair for $S_1$ and some vertex $x_1 \in N_1$.
2. For $i = 2, \ldots, t$, $(\Delta_i, N_i)$ 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 $T = \{S\}$ and $Q = \{S\}$. Then a call separators($G,a,b,T,Q$) determines a set $Q$ containing all minimal $a,b$-separators.

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

Remark 5.1 If we let $T = \{\{b\}\}$ and $Q = \emptyset$ then a call separators($G,a,b,T,Q$) 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 $C_a$. 

8
procedure separators(G,a,b,T,Q)
input: Graph $G$ and non adjacent vertices $a$ and $b$ and
sets $T$ and $S$ of minimal $a,b$-separators.
output: Set $Q$ of all minimal $a,b$-separators in $G$.
begin
  $T' := \emptyset$;
  for each $S \in T$ 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 $\Delta$;
          {$\Delta$ 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 $C'_a$}
        $S^* := (S \cup \Delta) \setminus N$;
        $T' := T' \cup \{S^*\}$
          {Add $S^*$ to $T'$ only if not yet present!}
      end for
    end for;
  $Q := Q \cup T'$;
sepators(G,a,b,T',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 \( T \), the connected component of \( G[V - S] \) that contains \( a \) has at most \( n - i \) vertices.

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

Determining \( \Delta \) takes at most \( O(n^2) \) time. Also computing \( C_n \) and \( N \) can clearly be done in \( O(n^2) \) time. We have to make sure that the new set \( T' \) 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.

\[ \square \]

**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


**In this series appeared:**

<table>
<thead>
<tr>
<th>Volume</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/02</td>
<td>R.P. Nederpelt, H.C.M. de Swart</td>
<td>Implication. A survey of the different logical analyses &quot;if...,then...&quot;, p. 26.</td>
</tr>
<tr>
<td>91/03</td>
<td>J.P. Katoen, L.A.M. Schoenmakers</td>
<td>Parallel Programs for the Recognition of $P$-invariant Segments, p. 16.</td>
</tr>
<tr>
<td>91/05</td>
<td>D. de Reus</td>
<td>An Implementation Model for GOOD, p. 18.</td>
</tr>
<tr>
<td>91/06</td>
<td>K.M. van Hee</td>
<td>SPECIFICATIEMETHODEN, een overzicht, p. 20.</td>
</tr>
<tr>
<td>91/07</td>
<td>E. Poll</td>
<td>CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.</td>
</tr>
<tr>
<td>91/12</td>
<td>E. van der Sluis</td>
<td>A parallel local search algorithm for the travelling salesman problem, p. 12.</td>
</tr>
<tr>
<td>91/14</td>
<td>P. Lemmens</td>
<td>The PDB Hypermedia Package. Why and how it was built, p. 63.</td>
</tr>
<tr>
<td>91/16</td>
<td>A.J.J.M. Marcelis</td>
<td>An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.</td>
</tr>
</tbody>
</table>
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.
             R.V. Schuwer
             W.P. de Roever
             J. Zwiers
91/23  K.M. van Hee  Z and high level Petri nets, p. 16.
             L.J. Somers
             M. Voorhoeve
91/24  A.T.M. Aerts  Formal semantics for BRM with examples, p. 25.
         D. de Reus
91/25  P. Zhou  A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52.
         J. Hooman
         R. Kuiper
91/26  P. de Bra  The GOOD based hypertext reference model, p. 12.
         G.J. Houben
         J. Paredaens
91/27  F. de Boer  Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
         C. Palamidessi
91/28  F. de Boer  A compositional proof system for dynamic process creation, p. 24.
         R. van Geldrop
91/30  J.C.M. Baeten  An Algebra for Process Creation, p. 29.
         F.W. Vaandrager
91/31  H. ten Eikelder  Some algorithms to decide the equivalence of recursive types, p. 26.
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  Asynchronous communication in process algebra, p. 20.
         J.W. Klop
         C. Palamidessi
A note on compositional refinement, p. 27.
A compositional semantics for fault tolerant real-time systems, p. 18.
Real space process algebra, p. 42.
Program derivation in acyclic graphs and related problems, p. 90.
Conservative fixpoint functions on a graph, p. 25.
Discrete time process algebra, p. 45.
The fine-structure of lambda calculus, p. 110.
On stepwise explicit substitution, p. 30.
Composition and decomposition in a CPN model, p. 55.
Demonic operators and monotype factors, p. 29.
Set theory and nominalisation, Part II, p. 22.
The total order assumption, p. 10.
A system at the cross-roads of functional and logic programming, p. 36.
Integrity checking in deductive databases; an exposition, p. 32.
Interval timed coloured Petri nets and their analysis, p. 20.
A unified approach to Type Theory through a refined lambda-calculus, p. 30.
Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
Are Types for Natural Language? P. 32.
Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
92/22  R. Nederpelt
        F.Kamareddine
        E.Klein
        M.Codish
        D.Dams
        Eyal Yardeni
        E.Poll
        T.H.W.Beelen
        W.J.J.Stut
        P.A.C. Verkoulen

92/24  Bottom-up Abstract Interpretation of Logic Programs,
       p. 33.

92/25  A Programming Logic for Fo, p. 15.

92/26  A modelling method using MOVIE and SimCon/ExSpect,
       p. 15.

92/27  A taxonomy of keyword pattern matching algorithms,
       p. 50.

93/01  Deriving the Aho-Corasick algorithms: a case study into
       the synergy of programming methods, p. 36.

93/02  A continuous version of the Prisoner's Dilemma, p. 17

93/03  Quicksort for linked lists, p. 8.

93/04  Deterministic and randomized local search, p. 78.

93/05  A congruence theorem for structured operational
       semantics with predicates, p. 18.

93/06  On the unavoidability of metastable behaviour, p. 29

93/07  Exercises in Multiprogramming, p. 97

93/08  A Formal Deterministic Scheduling Model for Hard Real-
       Time Executions in DEDOS, p. 32.

93/09  Systems Engineering: a Formal Approach

93/10  Systems Engineering: a Formal Approach
       Part II: Frameworks, p. 44.

93/11  Systems Engineering: a Formal Approach

93/12  Systems Engineering: a Formal Approach
       Part IV: Analysis Methods, p. 63.

93/13  Systems Engineering: a Formal Approach
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 Foo, 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 

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 

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 

93/14  J.C.M. Baeten  
J.A. Bergstra  
On Sequential Composition, Action Prefixes and 
Process Prefix, p. 21.
<table>
<thead>
<tr>
<th>No.</th>
<th>Authors</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/15</td>
<td>J.C.M. Baeten</td>
<td>A Real-Time Process Logic</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>J.A. Bergstra</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R.N. Bof</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93/16</td>
<td>H. Schepers</td>
<td>A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>J. Hooman</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93/17</td>
<td>D. Alstein</td>
<td>Hard Real-Time Reliable Multicast in the DEDOS system</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>P. van der Stok</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93/18</td>
<td>C. Verhoef</td>
<td>A congruence theorem for structured operational semantics with predicates and negative premises</td>
<td>22</td>
</tr>
<tr>
<td>93/19</td>
<td>G-J. Houben</td>
<td>The Design of an Online Help Facility for ExSpect</td>
<td>21</td>
</tr>
<tr>
<td>93/20</td>
<td>F.S. de Boer</td>
<td>A Process Algebra of Concurrent Constraint Programming</td>
<td>15</td>
</tr>
<tr>
<td>93/21</td>
<td>M. Codish</td>
<td>Freeness Analysis for Logic Programs - And Correctness?</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>D. Dams</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G. Filé</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M. Bruynooghe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93/22</td>
<td>E. Poll</td>
<td>A Typechecker for Bijective Pure Type Systems</td>
<td>28</td>
</tr>
<tr>
<td>93/23</td>
<td>E. de Kogel</td>
<td>Relational Algebra and Equational Proofs</td>
<td>23</td>
</tr>
<tr>
<td>93/24</td>
<td>E. Poll and Paula Severi</td>
<td>Pure Type Systems with Definitions</td>
<td></td>
</tr>
<tr>
<td>93/26</td>
<td>W.M.P. van der Aalst</td>
<td>Multi-dimensional Petri nets</td>
<td>25</td>
</tr>
</tbody>
</table>