Computing the toughness and the scattering number for interval and other graphs

Dieter Kratsch, T. Klots, H. Müller

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Computing the toughness and the scattering number for interval and other graphs

D. Kratsch, T. Kloks, H. Müller

N° 2237

Mars 1994

PROGRAMME 1

Architectures parallèles,

bases de données,

réseaux et systèmes distribués

ISSN 0249-6399

Computing the toughness and the scattering number for interval and other graphs

D. Kratsch *, T. Kloks **, H. Müller ***

Programme 1 — Architectures parallèles, bases de données, réseaux
et systèmes distribués
Projet Pampa

Rapport de recherche n ° 2237 — Mars 1994 — 22 pages

Abstract: We show that the scattering number and the toughness of a graph, two graph parameters strongly related to hamiltonian properties of graphs, can be computed in polynomial time on interval graphs, circular-arc graphs, permutation graphs, circular permutation graphs, trapezoid graphs and co-comparability graphs of bounded dimension. This leads to new algorithms deciding whether a graph has a hamiltonian circuit and a hamiltonian path, respectively, for some of these classes.

(Résumé : tsvp)

* IRISA, Campus de Beaulieu, 35042 Rennes, France.
** Department of Mathematics and Computing Science, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands.
*** Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität, 07740 Jena, Germany.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Calcul du nombre de coriacité et de dispersion pour les graphes d'intervalles et d'autres graphes

Résumé : Nous montrons que le nombre de coriacité et de dispersion d'un graphe peuvent être calculés en temps polynomial pour les graphes d'intervalles, les graphes d'arcs circulaires, les graphes de permutations, les graphes trapézoïdaux et les graphes de cocomparabiliés de dimension bornée. Comme conséquence, nous obtenons de nouveaux algorithmes permettant de décider si un graphe admet un circuit hamiltonien ou un chemin hamiltonien pour certaines des classes précédement citées.

Computing the toughness and the scattering number for interval and other graphs

Contents

1 Introduction

In this paper we present polynomial time algorithms computing the scattering number and the toughness of graphs when the input graphs are restricted to various classes of well-structured graphs. The problems are of particular interest because of two reasons. On one hand, toughness is considered to be a reasonable measure for the reliability of a network in case of node failures [1, 27]. On the other hand, both parameters are strongly related to hamiltonian properties of graphs. If a graph has a hamiltonian circuit then its scattering number is at most zero and its toughness is at least one. Nevertheless both conditions are not sufficient for hamiltonicity of graphs in general. Fortunately both conditions are necessary and sufficient for some graph classes. This was shown for cographs by Jung [17] and recently extended to cocomparability graphs by Steiner [26].

Hamiltonian properties of graphs are one of the most intensively studied subjects of graph theory (see e.g. [6]). Unfortunately the HAMILTONIAN CIRCUIT problem 'Given a graph G, decide whether it has a hamiltonian circuit' as well as the HAMILTONIAN PATH problem 'Given a graph G, decide whether it has a hamiltonian path' are **NP**-complete, like many other algorithmic decision problems related to well-studied graph problems [13].

The toughness of a graph is an extensively studied graph parameter [2, 3, 8], which is mainly due to the famous conjecture of Chvátal that every graph G is necessarily hamiltonian if its toughness $t(G) \geq 2$ [9]. However not much is known about the algorithmic properties of the TOUGHNESS problem 'Given a graph G and a rational $t > 0$, decide whether $t(G) \geq t$'. Bauer, Hakimi and Schmeichel have shown that the problem is **coNP**-complete [2]. Even the problem 'Given a bipartite graph G, decide whether $t(G) = 1$' remains **coNP**-complete [21]. Corneil, Lerchs and Stewart have given a linear time algorithm computing the scattering number of cographs which leads to linear time algorithms deciding whether a cograph has a hamiltonian circuit and whether a cograph has a hamiltonian path [10]. There is a $O(\sqrt{n}\, m)$ algorithm deciding whether a split graph has toughness less than one [21].

Many papers were written in the last decades dealing with the algorithmic complexity of the hamiltonian circuit problem when restricted to certain classes of well-structured graphs [16, 24, 25]. The following are the best known

timebounds of polynomial time hamiltonian circuit algorithms for the graph classes considered in this paper: $O(n+m)$ [18] and $O(n \log n)$ [23] for interval graphs, $O(n^2 \log n)$ for circular-arc graphs [28] and $O(n^3)$ for cocomparability graphs [12].

For definitions and properties of classes of well-structured graphs not given here we refer to [7, 15].

In this paper we illustrate a method which enables the design of polynomial time algorithms for computing the scattering number and the toughness of interval graphs, circular-arc graphs, permutation graphs, circular permutation graphs, trapezoid graphs and cocomparability graphs of bounded dimension.

2 Preliminaries

Throughout the paper we consider only finite, undirected graphs $G = (V, E)$ without loops and without multiple edges. The vertex set of the graph is denoted by V and the edge set of the graph is denoted by E. We always denote the number of vertices of a graph by n and the number of edges of a graph by m.

Let $G = (V, E)$ be a graph. Let W be a subset of V. Then $G[W]$ denotes the graph *induced* by the vertex set W, i.e., $G[W]$ has vertex set W and two vertices of W are adjacent in $G[W]$ if and only if they are adjacent in G. We denote the number of connected components of a graph G by $c(G)$. A set $S \subseteq V$ is said to be a *separator* (or *cutset*) of G if $c(G[V \setminus S]) > 1$. Note that $S = \emptyset$ is a separator of G iff G is disconnected.

Now we give some definitions and important facts on toughness, scattering number, hamiltonicity and traceability of graphs. The *toughness* $t(G)$ of a graph G was defined by Chvátal in [9].

Definition 2.1 *The* toughness *of a complete graph K_n is $t(K_n) = \infty$. If G is not complete, then*

$$t(G) = \min \left\{ \frac{|\, S \,|}{c(G[V \setminus S])} : \ S \text{ separator of } G \right\}.$$

A graph G is said to be t-tough *if $t(G) \geq t$ holds, i.e., $|\, S \,| \geq t \cdot c(G[V \setminus S])$ for every separator S of G.*

The *scattering number* $sc(G)$ of a graph G was defined by Jung in [17].

Definition 2.2 *The scattering number of a complete graph K_n is $sc(K_n) = -\infty$. If G is not complete, then*

$$sc(G) = \max\{c(G[V \setminus S]) - |S| : \ S \text{ separator of } G\}.$$

The following is an immediate consequence of the definitions.

Lemma 2.3 *For every graph G holds $t(G) \geq 1$ iff $sc(G) \leq 0$.*

The **NP**-completeness of the problem 'Given a bipartite graph G, decide whether $t(G) < 1$' [21] and Lemma 2.3 immediately imply

Theorem 2.4 *The problem 'Given a bipartite graph $G = (V, E)$, decide whether $sc(G) > 0$' is* **NP***-complete. Hence the* SCATTERING NUMBER *problem 'Given a graph G and an integer k, decide whether $sc(G) \geq k$' is* **NP***-complete.*

Definition 2.5 *A* hamiltonian circuit *of a graph $G = (V, E)$ is a simple cycle containing all vertices of G. A graph is said to be* hamiltonian *if it has a hamiltonian circuit. A* hamiltonian path *of a graph $G = (V, E)$ is a simple path containing all vertices of G. A graph is said to be* traceable *if it has a hamiltonian path.*

The following theorem states well-known necessary conditions for hamiltonicity and traceability [6, 9].

Theorem 2.6 *If $G = (V, E)$ is a hamiltonian graph then $c(G[V \setminus S]) \leq |S|$ for every nonempty proper subset S of V.*
If $G = (V, E)$ is a traceable graph then $c(G[V \setminus S]) \leq |S|+1$ for every nonempty proper subset S of V.

The following corollary given in [9, 17] is an immediate consequence of Theorem 2.6, Definition 2.1 and 2.2.

Corollary 2.7 *If a graph G is hamiltonian then $t(G) \geq 1$ and $sc(G) \leq 0$. If a graph G is traceable then $sc(G) \leq 1$.*

Fortunately there are classes of well-structured graphs for which the conditions of Corollary 2.7 are sufficient as well.

Definition 2.8 *Let $\mathcal{G}$ be a hereditary class of graphs. $\mathcal{G}$ is said to be* cycle-tough *if for every graph G of $\mathcal{G}$ holds: G is hamiltonian if and only if G is 1-tough. $\mathcal{G}$ is said to be* path-tough *if for every graph G of $\mathcal{G}$ holds: G is traceable if and only if $sc(G) \leq 1$.*

Cocomparability graphs are cycle-tough [26] and path-tough [22].

Theorem 2.9 *A cocomparability graph $G = (V, E)$ is hamiltonian if and only if G is 1-tough. A cocomparability graph $G = (V, E)$ is traceable if and only if $sc(G) \leq 1$.*

Hence interval, permutation, trapezoid and cocomparability graphs of dimension at most d for any fixed $d \geq 1$, as proper subclasses of the class of cocomparability graphs, are also cycle-tough and path-tough. The fact that interval graphs are cycle-tough is implicitly mentioned in [18, 23].

There is a lot of evidence that HAMILTONIAN CIRCUIT when restricted to a cycle-tough graph class can not be **NP**-complete, as shown by the following theorem given in [21].

Theorem 2.10 *Let $\mathcal{G}$ be a graph class whose recognition problem belongs to* **NP**$\cap$**coNP**. *If $\mathcal{G}$ is cycle-tough then the* HAMILTONIAN CIRCUIT *problem, when restricted to $\mathcal{G}$, cannot be* **NP**-*complete unless* **NP**=**coNP**.

Notice that the first assumption of Theorem 2.10 is obviously fulfilled for every graph class having a polynomial time recognition algorithm.

A similar theorem holds for path-tough graph classes and the HAMILTONIAN PATH problem [21].

3 Minimal separators and scattering number

Our algorithms, computing the scattering number and the toughness for interval graphs, circular-arc graphs, permutation graphs and other graph classes, will heavily use minimal separators of graphs and a 'dynamic programming on pieces' similar to the technique introduced in [4, 11].

Definition 3.1 *A subset $S \subseteq V$ is an a,b-separator for nonadjacent vertices a and b of a graph $G = (V, E)$, 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 *S is a set of vertices such that S is a minimal a,b-separator for some nonadjacent vertices a and b.*

The following lemma was given in [19, 20]. It provides an easy algorithm to recognize minimal separators.

Lemma 3.2 *Let S be a separator of the graph $G = (V,E)$. Then S is a minimal separator of G if and only if there are at least two different connected components of $G[V \setminus S]$ for which every vertex of S has a neighbor in both of these components.*

Our main tool for designing efficient algorithms computing the scattering number of a graph on certain classes of well-structured graphs is given in the next theorem.

Theorem 3.3 *Let $G = (V,E)$ be a graph which is not complete. Then*

$$sc(G) = \max_S \left(\sum_{i=1}^{k} \max(sc(G[C_i]), 1) \; - \; |S| \right),$$

where the maximum is taken over all minimal separators S of the graph G and $C_1, C_2, \ldots, C_k$ are the connected components of $G[V \setminus S]$.

Proof. First let X be a separator of $G = (V,E)$ such that $sc(G) = c(G[V \setminus X]) - |X|$. Let S be a minimal separator of G that is a subset of X and let $C_1, C_2, \ldots, C_k$ be the connected components of $G[V \setminus S]$.

We consider the sets $X_i = X \cap C_i$, $i \in \{1, 2, \ldots, k\}$. Assume $X_i = \emptyset$. Then C_i is also a component of $G[V \setminus X]$. Consequently $c(G[C_i \setminus X_i]) - |X_i| = 1$. Assume $X_i \neq \emptyset$. Suppose X_i would not be a separator of $G[C_i]$. Then we had $c(G[V \setminus (X \setminus X_i)]) - |X \setminus X_i| = c(G[V \setminus X]) - |X| + |X_i| > c(G[V \setminus X]) - |X| = sc(G)$, a contradiction. Hence $X_i \neq \emptyset$ implies that X_i is a separator of $G[C_i]$, thus $sc(G[C_i]) \geq c(G[C_i \setminus X_i]) - |X_i|$.

Computing $c(G[V \setminus X]) - |X|$ by essentially summing up the values of $c(G[C_i \setminus X_i]) - |X_i|$ over all components C_i of $G[V \setminus S]$ will complete the proof of the first direction.

$$sc(G) = c(G[V \setminus X]) - |X| = \sum_{i=1}^{k} \{c(G[C_i \setminus X_i]) - |X_i|\} \; - \; |S| \leq$$

$$\leq \sum_{i=1}^{k} \max(sc(G[C_i]), 1) \; - \; |S|.$$

Now let S be a minimal separator of $G = (V, E)$. Furthermore let $C_1, C_2, \ldots, C_k$ be the connected components of $G[V \setminus S]$. We construct a separator X of G such that $X = S \cup \bigcup_{i=1}^{k} X_i$ with $X_i \subseteq C_i$ for every $i \in \{1, 2, \ldots, k\}$.

For $i \in \{1, 2, \ldots, k\}$ we set $X_i = \emptyset$ if $sc(G[C_i]) \leq 1$. Otherwise, if $sc(G[C_i]) > 1$, we choose a separator X_i of $G[C_i]$ such that $sc(G[C_i]) = c(G[C_i \setminus X_i]) - |X_i|$. Then $X \supseteq S$ is a separator of G and we have

$$sc(G) \geq c(G[V \setminus X]) - |X| = \sum_{i=1}^{k} \{c(G[C_i \setminus X_i]) - |X_i|\} \; - \; |S|.$$

W.l.o.g. let $C_1, C_2, \ldots, C_r$, $0 \leq r \leq k$, be the components of G with $sc(G[C_i]) \leq 1$. Consequently

$$sc(G) = \sum_{i=1}^{k} \{c(G[C_i \setminus X_i]) - |X_i|\} \; - \; |S| =$$

$$\sum_{i=1}^{r} 1 + \sum_{i=r+1}^{k} sc(G[C_i]) \; - \; |S| = \sum_{i=1}^{k} \max(sc(G[C_i]), 1) \; - \; |S|.$$

This completes the proof. □

4 Toughness and minimal separators

Contrary to the vertex ranking problem and the computation of the scattering number there seems to be no formula concerning toughness of the type of Theorem 3.3, i.e., a formula $t(G) = \min_S \; f\{t(G[C_1]), t(G[C_2]), \ldots, t(G[C_k])\}$ where the minimum is taken over all minimal separators S of the graph G, $C_1, C_2, \ldots, C_k$ are the connected components of $G[V \setminus S]$ and f is an 'efficiently computable function'. Fortunately there is a possibility to overcome this difficulty which requires to compute auxiliary data.

Definition 4.1 *Let $G = (V, E)$ be a graph. For $i \in \{0, 1, \ldots, n\}$ we define $c_i(G)$ to be the largest number of components the graph G can get after the removal of exactly i vertices; i.e., $c_n(G) = 0$ and for $i < n$;*

$$c_i(G) = \max \{c(G[V \setminus S]) : \ S \subseteq V, \ |S| = i\}.$$

Remark 4.2 *Let $G = (V, E)$ be a graph which is not complete. Then*

$$t(G) = \min \left\{ \frac{i}{c_i(G)} : c_i(G) > 1 \right\}.$$

The following theorem shows how to compute $c_i(G)$ for all $i \in \{0, 1, \ldots, n\}$.

Theorem 4.3 *Let $G = (V, E)$ be a graph which is not complete and let $i \in \{0, 1, \ldots, n\}$. If $c_i(G) > 1$ then*

$$c_i(G) = \max_{|S| \leq i} \max_{0 \leq r_1, r_2, \ldots, r_k \leq n} \left\{ \sum_{j=1}^{k} c_{r_j}(G[C_j]) : \ \sum_{j=1}^{k} r_j = i - |S| \right\},$$

where the maximum is taken over all minimal separators S of G and over all nonnegative integer vectors $(r_1, r_2, \ldots, r_k)$. Furthermore $C_1, C_2, \ldots, C_k$ are the connected components of $G[V \setminus S]$.

Proof. Let $G = (V, E)$ be a graph which is not complete and let $i \in \{0, 1, \ldots, n\}$ such that $c_i(G) > 1$.

First let X be a subset of V with $|X| = i$ and $c(G[V \setminus X]) = c_i(G) > 1$. Hence X is a separator of G. Let S be a minimal separator of G with $S \subseteq X$. Let $C_1, C_2, \ldots, C_k$ be the connected components of $G[V \setminus S]$. We set $X_j = X \cap C_j$ for every $j \in \{1, 2, \ldots, k\}$. Then

$$c_i(G) = c(G[V \setminus X]) = \sum_{j=1}^{k} c(G[C_j \setminus X_j]) \leq \sum_{j=1}^{k} c_{|X_j|}(G[C_j]) \leq$$

$$\max_{|S| \leq i} \max_{0 \leq r_1, r_2, \ldots, r_k \leq n} \left\{ \sum_{j=1}^{k} c_{r_j}(G[C_j]) : \ \sum_{j=1}^{k} r_j = i - |S| \right\}.$$

Now let S be a minimal separator of G with $|S| \leq i$ and let $(r_1, r_2, \ldots, r_k)$ be a vector realizing the maximum of the right-hand side of the formula above.

For every $j \in \{1, 2, \ldots, k\}$ we choose a subset $X_j \subseteq C_j$ such that $|X_j| = r_j$ and $c_{r_j}(G[C_j]) = c(G[C_j \setminus X_j])$. Hence $X = S \cup \bigcup_{j=1}^{k} X_j$ is a subset of V and $|X| = |S| + \sum_{j=1}^{k} |X_j| = |S| + \sum_{j=1}^{k} r_j = i$. Consequently

$$\max_{|S| \leq i} \max_{0 \leq r_1, r_2, \ldots, r_k \leq n} \left\{ \sum_{j=1}^{k} c_{r_j}(G[C_j]) : \sum_{j=1}^{k} r_j = i - |S| \right\} =$$

$$\sum_{j=1}^{k} c(G[C_j \setminus X_j]) = c(G[V \setminus X]) \leq c_i(G).$$

This completes the proof. □

Remark 4.4 *For every graph $G = (V, E)$ holds $c_i(G) = 1$ if and only if $i = n - 1$ or $i < \kappa(G)$, where $\kappa(G)$ denotes the vertex connectivity of G.*

How to handle all minimal separators will be shown in the following sections. The toughness formula (Theorem 4.3) is obviously harder to compute than the scattering number formula (Theorem 3.3). The next lemma shows that the difference is not too large in the sense of efficient computations.

Lemma 4.5 *Let $G = (V, E)$ be a graph which is not complete, let S be a minimal separator of G and let $C_1, C_2, \ldots, C_k$ be the connected components of $G[V \setminus S]$. For every $j \in \{1, \ldots, k\}$ let the list H_j be $(c_0(G[C_j]), c_1(G[C_j]), \ldots, c_{|C_j|}(G[C_j]))$. There is an algorithm computing*

$$\max_{0 \leq r_1, r_2, \ldots, r_k \leq n} \left\{ \sum_{j=1}^{k} c_{r_j}(G[C_j]) : \sum_{j=1}^{k} r_j = i - |S| \right\}$$

for every $i \geq |S|$ from the lists $H_1, H_2, \ldots, H_k$ in time $O(n^3)$.

Proof. Let S be a minimal separator of the graph G and let $C_1, C_2, \ldots, C_k$ be the components of $G[V \setminus S]$. We define $c_i^{(r)}(G[V \setminus S])$, $1 \leq r \leq k$, as the largest number of components of the graph $G[\bigcup_{j=1}^{r} C_j]$ after the removal of $i - |S|$ vertices of $\bigcup_{j=1}^{r} C_j$. Hence $c_i^{(k)}(G[V \setminus S])$ is exactly the largest number

of components $G[\bigcup_{j=1}^{k} C_j]$ can have after the removal of $i - |S|$ vertices of $\bigcup_{j=1}^{k} C_j$ which is exactly

$$\max_{0 \leq r_1, r_2, \ldots, r_k \leq n} \left\{ \sum_{j=1}^{k} c_{r_j}(G[C_j]) : \sum_{j=1}^{k} r_j = i - |S| \right\}.$$

Let the list $L^{(r)}$ be $(c_0^{(r)}(G[V \setminus S]), c_1^{(r)}(G[V \setminus S]), \ldots, c_{|\bigcup_{i=1}^{r} C_i|}^{(r)}(G[V \setminus S]))$, $1 \leq r \leq k$. Then $L^{(1)} = H_1$.

The algorithm iteratively computes for $r = 2, 3, \ldots, k$ the list $L^{(r)}$ from $L^{(r-1)}$ and H_r by using

$$c_i^{(r)}(G[V \setminus S]) = \max \left\{ c_p^{(r-1)}(G[V \setminus S]) + c_q(G[C_r]) : p + q = i \right\}.$$

The computation of one entry $c_i^{(r)}(G[V \setminus S])$ can be done in time $O(n)$. Hence all $O(n^2)$ entries are computed in time $O(n^3)$ by the algorithm. □

In the next section we demonstrate how a 'dynamic programming on pieces' leads to polynomial time algorithms for computing toughness and scattering number on interval graphs.

5 Interval graphs

A graph $G = (V, E)$ is an *interval graph* if there is a one-to-one correspondence between the vertex set V and a family of intervals of the real line such that two vertices of G are joined by an edge iff the corresponding intervals have non-empty intersection. Interval graphs are a well-known family of perfect graphs with plenty of nice structural properties (cf. [7, 15]). We will use the following characterization given by Gilmore and Hoffman [14].

Lemma 5.1 *A graph $G = (V, E)$ is an interval graph if and only if the maximal cliques of G can be linearly ordered, such that, for every vertex v of G, the maximal cliques containing v occur consecutively.*

Such a linear ordering of the maximal cliques of an interval graph is said to be a *consecutive clique arrangement*. Notice that triangulated graphs and

thus interval graphs have at most n maximal cliques [15]. There is a linear time recognition algorithm for interval graphs by Booth and Lueker which also computes a consecutive clique arrangement of the input graph if it is an interval graph [5].

Using the characterization of Lemma 5.1, we can easily identify the minimal separators of an interval graph [20].

Lemma 5.2 *Let G be an interval graph and let $A_1, \ldots, A_t$ be a consecutive clique arrangement of G. Then the set of all minimal separators of G consists of the sets $S_p = A_p \cap A_{p+1}$ for $p \in \{1, 2, \ldots, t-1\}$.*

Hence an interval graph $G = (V, E)$ has at most n minimal separators.

Now we show how to compute the scattering number and the toughness of an interval graph by using the formulae of Theorem 3.3 and 4.3. We compute $sc(G[P])$ in the scattering number algorithm and $c_i(G[P])$, $i \in \{0, 1, \ldots, |P|\}$, in the toughness algorithm only for pieces $P \subseteq V$ of the given interval graph $G = (V, E)$.

Definition 5.3 *Let G be an interval graph with consecutive clique arrangement $A_1, \ldots, A_t$. We define $A_0 = A_{t+1} = \emptyset$. For all l, r with $1 \leq l \leq r \leq t$ we define $\mathcal{P}(l, r) = (\bigcup_{i=l}^{r} A_i) \setminus (A_{l-1} \cup A_{r+1})$. A set $\mathcal{P}(l, r)$, $1 \leq l \leq r \leq t$, is said to be a* piece *of G if $\mathcal{P}(l, r) \neq \emptyset$ and $G[\mathcal{P}(l, r)]$ is connected. Furthermore $V = \mathcal{P}(1, t)$ is a piece of G (even if G is disconnected).*

The following lemma justifies the dynamic programming on pieces for interval graphs.

Lemma 5.4 *Let S be a minimal separator of $G[\mathcal{P}(l, r)]$, $1 \leq l < r \leq t$. Then there exists a minimal separator S_p of G, $l \leq p < r$, such that $S = S_p \cap \mathcal{P}(l, r) = S_p \setminus (A_{l-1} \cup A_{r+1})$. Moreover, every connected component of $G[\mathcal{P}(l, r) \setminus S]$ is a piece of G.*

Proof. $G[\mathcal{P}(l, r)]$ is an interval graph and the linear arrangement $A_l \setminus (A_{l-1} \cup A_{r+1}), A_{l+1} \setminus (A_{l-1} \cup A_{r+1}), \ldots, A_r \setminus (A_{l-1} \cup A_{r+1})$ has all properties of a consecutive clique arrangement for $G[\mathcal{P}(l, r)]$, except that cliques may occur more than once. Therefore we can still apply Lemma 5.2 implying that all minimal separators of $G[\mathcal{P}(l, r)]$ are sets of the form $(A_p \setminus (A_{l-1} \cup A_{r+1})) \cap$

$(A_{p+1} \setminus (A_{l-1} \cup A_{r+1})) = (A_p \cap A_{p+1}) \setminus (A_{l-1} \cup A_{r+1}) = S_p \cap \mathcal{P}(l,r)$, $p \in \{l, l+1, \ldots, r-1\}$.

For every $v \in V$ we define $l(v) = \min\{k : v \in A_k\}$ and $r(v) = \max\{k : v \in A_k\}$. Then for all l, r with $1 \leq l \leq r \leq t$ and for every component C of $G[\mathcal{P}(l,r)]$ holds $C = \mathcal{P}(l(C), r(C))$ with $l(C) = \min\{l(v) : v \in C\}$ and $r(C) = \max\{r(v) : v \in C\}$, i.e., C is a piece.

Now let $S = S_p \cap \mathcal{P}(l,r)$ be a minimal separator of $G[\mathcal{P}(l,r)]$, $1 \leq l \leq p < r \leq t$. The graph $G[\mathcal{P}(l,r) \setminus S] = G[\mathcal{P}(l,r) \setminus S_p]$ is either the disjoint union of $G[\mathcal{P}(l,p)]$ and $G[\mathcal{P}(p+1,r)]$ or equal to one of them (in case that $\mathcal{P}(l,p) = \emptyset$ or $\mathcal{P}(p+1,r) = \emptyset$). Hence the set of components of $G[\mathcal{P}(l,r) \setminus S_p]$ is equal to the union of the set of components of $G[\mathcal{P}(l,p)]$ and of the set of components of $G[\mathcal{P}(p+1,r)]$ or it is equal to one of these sets. Hence all components of $G[\mathcal{P}(l,r) \setminus S_p]$ are pieces.

□

There are essentially two different types of pieces of an interval graph. A piece is called *complete* if it induces a complete graph and it is called *noncomplete* otherwise. The pieces $\mathcal{P}(l,l)$, $l \in \{1, 2, \ldots, t\}$, are complete. Furthermore a piece $\mathcal{P}(l,r)$, $l < r$, may also be complete. For every complete piece $\mathcal{P}(l,r)$, $l \leq r$, holds

$$sc(G[\mathcal{P}(l,r)]) = -\infty \tag{1}$$

$$c_i(G[\mathcal{P}(l,r)]) = \begin{cases} 1 & \text{if } i \in \{0, 1, \ldots, |\mathcal{P}(l,r)| - 1\} \\ 0 & \text{if } i = |\mathcal{P}(l,r)| \end{cases} \tag{2}$$

The subgraphs induced by noncomplete pieces $\mathcal{P}(l,r)$, $1 \leq l < r \leq t$, have minimal separators. Hence Theorem 3.3, Theorem 4.3 and Lemma 5.4 lead to the following formulae. For every noncomplete piece $\mathcal{P}(l,r)$, $1 \leq l < r \leq t$, holds

$$sc(G[\mathcal{P}(l,r)]) = \max\left\{\sum_{i=1}^{k} \max(sc(G[P_i]), 1) \; - \; |S_p \cap \mathcal{P}(l,r)|\right\} \tag{3}$$

where the maximum is taken over all $S_p \cap \mathcal{P}(l,r)$, $p \in \{l, l+1, \ldots, r-1\}$, that are minimal separators of $G[\mathcal{P}(l,r)]$.

For every noncomplete piece $\mathcal{P}(l,r)$, $1 \leq l < r \leq t$, and every $i \in \{\kappa(G[\mathcal{P}(l,r)]), \ldots, |\mathcal{P}(l,r)| - 2\}$ holds

$$c_i(G[\mathcal{P}(l,r)]) = \max \sum_{j=1}^{k} c_{r_j}(G[P_j]) \tag{4}$$

where the maximum is taken over all $S_p \cap \mathcal{P}(l,r)$, $p \in \{l, l+1, \ldots, r-1\}$, that are minimal separators of $G[\mathcal{P}(l,r)]$ fulfilling the condition $|S_p \cap \mathcal{P}(l,r)| \leq i$ and over all nonnegative integer vectors $(r_1, r_2, \ldots, r_k)$ fulfilling the condition $\sum_{j=1}^{k} r_j = i - |S_j \cap \mathcal{P}(l,r)|$. In Eq. 3 and in Eq. 4 $P_1, P_2, \ldots, P_k$ are the connected components of $G[\mathcal{P}(l,r) \setminus S_p]$.

Let $G = (V,E)$ be an interval graph. If G is a complete graph then $sc(G) = -\infty$ and $t(G) = \infty$. Otherwise the 'dynamic programming on pieces' works as follows:

1. Compute a consecutive clique arrangement $A_1, A_2, \ldots, A_t$ of G, compute $l(v) = \min\{k : v \in A_k\}$ and $r(v) = \max\{k : v \in A_k\}$ for every $v \in V$, compute all minimal separators $S_p = A_p \cap A_{p+1}$, $p \in \{1, 2, \ldots, t-1\}$.

2. For all l, r with $1 \leq l \leq r \leq t$ compute the vertex set $\mathcal{P}(l,r)$, mark (l,r) 'empty' if $\mathcal{P}(l,r) = \emptyset$ and mark (l,r) 'complete' if $\mathcal{P}(l,r) \neq \emptyset$ and $G[\mathcal{P}(l,r)]$ is a complete graph.

3. For all nonmarked tuples (l,r) check whether $G[\mathcal{P}(l,r)]$ is connected. If so mark (l,r) 'noncomplete'. Else mark (l,r) 'disconnected', compute the components $P_j = \mathcal{P}(l_j, r_j)$, $1 \leq j \leq k$, of $G[\mathcal{P}(l,r)]$ and store $(l_1, r_1), (l_2, r_2), \ldots, (l_k, r_k)$ in a linked list with a pointer from (l,r) to the head of this list.

4. For all (l,r), $1 \leq l < r \leq t$, marked 'noncomplete' and for every $p \in \{l, l+1 \ldots, r-1\}$ compute the components $P_j = \mathcal{P}(l_j, r_j)$, $1 \leq j \leq k$, of $G[\mathcal{P}(l,r) \setminus S_p]$, check whether $S_p \cap \mathcal{P}(l,r)$ is a minimal separator of $G[\mathcal{P}(l,r)]$ and if so mark (p,l,r) 'minimal', store $(l_1, r_1), (l_2, r_2), \ldots, (l_k, r_k)$ in a linked list with a pointer from (p,l,r) to the head of this list and compute $|S_p \cap \mathcal{P}(l,r)|$.
   (Compute $\kappa(G[\mathcal{P}(l,r)]) = \min\{|S_p \cap \mathcal{P}(l,r)| : (p,l,r)$ marked 'minimal'$\}$.)

5. For every pair (l,r) marked 'complete' compute $sc(G[\mathcal{P}(l,r)])$ according to Eq. 1
(resp. compute $c_i(G[\mathcal{P}(l,r)])$ for every $i \in \{0,1,\ldots,|\mathcal{P}(l,r)|\}$ according to Eq. 2).

6. For $d := 1$ to t and for $l := 1$ to $t-d$ compute if $(l,l+d)$ is marked 'noncomplete' $sc(G[\mathcal{P}(l,l+d)])$ according to Eq. 3.
(resp. $c_i(G[\mathcal{P}(l,l+d)])$ for every $i \in \{\kappa(G[\mathcal{P}(l,l+d)]),\ldots,|\mathcal{P}(l,l+d)|-2\}$ according to Eq. 4. Set $c_i(G[\mathcal{P}(l,l+d)]) = 0$ for $i = |\mathcal{P}(l,l+d)|$ and set $c_i(G[\mathcal{P}(l,l+d)]) = 1$ for $i < \kappa(G[\mathcal{P}(l,l+d)])$ or $i = |\mathcal{P}(l,l+d)|-1$.)

7. Output $sc(G) = sc(G[\mathcal{P}(1,t)])$ (resp. $t(G) = \min\{i/c_i(G[\mathcal{P}(1,t)]) : c_i(G[\mathcal{P}(1,t)]) > 1\}$).

Theorem 5.5 *There is a $O(n^4)$ algorithm computing the scattering number for interval graphs and there is a $O(n^6)$ algorithm computing the toughness for interval graphs.*

Proof. The correctness of both algorithms follows from Theorem 3.3, Theorem 4.3 and Lemma 5.4.

For the time analysis let us first consider the scattering number algorithm. Step 1, 2, 5 and 7 can be done in time $O(n^4)$ in a straightforward manner. In step 3 testing connectedness and computing the components is done by a $O(n+m)$ algorithm for at most n^2 graphs $G[\mathcal{P}(l,r)]$. If $G[\mathcal{P}(l,r)]$ is disconnected and P_j is a component then $P_j = \mathcal{P}(l_j,r_j)$ with $l_j = \min\{l(v) : v \in P_j\}$ and $r_j = \max\{l(v) : v \in P_j\}$ which can be computed in time $O(n)$. Hence step 3 can be done in time $O(n^4)$.

Step 4 has to be executed for at most n^3 triples (p,l,r) with $l \leq p < r$. If $\mathcal{P}(l,r) \setminus S_p \neq \emptyset$ then the components of $G[\mathcal{P}(l,r) \setminus S_p]$ are computed as indicated in the proof of Lemma 5.4 by using the marks of (l,p) and $(p+1,r)$, namely if the mark is 'complete' or 'noncomplete' then (l,p) and $(p+1,r)$, respectively, are stored and if the mark is 'disconnected' then the corresponding linked list is added. Thus the linked list of (p,l,r) can be computed in time $O(n)$. $S_p \cap \mathcal{P}(l,r)$ is a minimal separator of $G[\mathcal{P}(l,r) \setminus S_p]$ iff there are at least two components in the list of (p,l,r) such that every vertex of $S_p \cap \mathcal{P}(l,r)$ has a neighbour in them (see Lemma 3.2). Because of the properties of a

consecutive clique arrangement it suffices to check the two components P_j of $G[\mathcal{P}(l,p)]$ with the two largest values of r_j and the two components P_j of $G[\mathcal{P}(p+1,r)]$ with the two smallest values of l_j (if they exist). This can be done in time $O(n)$. Hence step 4 needs time $O(n^4)$.

Step 6 requires the evaluation of the right-hand side of Eq. 3 for at most n^2 pairs $(l,l+d)$. For every p with $l \leq p < l+d$ and $(p,l,l+d)$ marked 'minimal' the components P_i of $G[\mathcal{P}(l,l+d) \setminus S_p]$ can be obtained in time $O(n)$ from the linked list of $(p,l,l+d)$. Each of the at most n values $sc(G[P_i])$ can be determined in constant time by table look-up since the scattering numbers of smaller pieces are already known. Thus $\sum_{i=1}^{k} \max(sc(G[P_i]),1) - |S_p \cap \mathcal{P}(l,l+d)|$ can be evaluated in time $O(n)$. Consequently step 6 of the scattering number algorithm can be done in time $O(n^4)$.

The toughness algorithm differs from the scattering number algorithm essentially in step 6 only when the right-hand side of Eq. 4 has to be evaluated for every $i \in \{\kappa(G[\mathcal{P}(l,l+d)]),\ldots,|\mathcal{P}(l,l+d)|-2\}$. The list $H_j = (c_0(P_j),c_1(P_j),\ldots,c_{|P_j|}(P_j))$, $j \in \{1,2,\ldots,k\}$, for each component $P_j = (l_j,r_j)$ of $G[\mathcal{P}(l,l+d) \setminus S_p]$ can be determined by table look-up in time $O(n^2)$ since these lists have already been computed for the smaller pieces. Now there is a $O(n^3)$ algorithm computing

$$\max_{0 \leq r_1,r_2,\ldots,r_k \leq n} \left\{ \sum_{j=1}^{k} c_{r_j}(G[P_j]) : \sum_{j=1}^{k} r_j = i - |S_p \cap \mathcal{P}(l,l+d)| \right\}$$

for a fixed minimal separator $S_p \cap \mathcal{P}(l,l+d)$ and for every i with $|S_p \cap \mathcal{P}(l,l+d)| \leq i$ in time $O(n^3)$ given in Lemma 4.5. Thus the running time of the toughness algorithm is $O(n^6)$. □

The algorithms compute the scattering number $sc(G)$ and the toughness $t(G)$ of the given interval graph $G=(V,E)$. However, one can modify the algorithms to compute within the same timebound a witness, i.e., a separator $S \subseteq V$ with $c(G[V \setminus S]) - |S| = sc(G)$, a separator $S' \subseteq V$ with $|S'|/c(G[V \setminus S']) = t(G)$ or even sets $S_i \subseteq V$ for every $i \in \{0,1,\ldots,n\}$ such that $|S_i| = i$ and $c_i(G) = c(G[V \setminus S_i])$.

We demonstrate this for the scattering number algorithm. For computing such a set S during the execution of the scattering number algorithm we compute for every noncomplete piece $\mathcal{P}(l,r)$ a separator $S(l,r)$ of $G[\mathcal{P}(l,r)]$

such that $sc(G[\mathcal{P}(l,r)]) = c(G[\mathcal{P}(l,r) \setminus S(l,r)]) - |S(l,r)|$. To this purpose we set $S(l,r) = \emptyset$ for every complete piece $\mathcal{P}(l,r)$ in step 4 of the algorithm. During step 5 we compute $S(l,r)$ for every noncomplete piece immediately after the computation of $sc(G[\mathcal{P}(l,r)])$. Let $p \in \{l, l+1, \ldots, r-1\}$ be a value realizing the maximum of the right-hand side of Eq. 3 and let $\mathcal{P}(l_1,r_1), \mathcal{P}(l_2,r_2), \ldots, \mathcal{P}(l_k,r_k)$ be the components of $G[\mathcal{P}(l,r) \setminus S_p]$. Then we set $S(l,r) = \bigcup \{S(l_i,r_i) : sc(G[\mathcal{P}(l_i,r_i)]) > 1\}$. The correctness of this procedure follows from the proof of Theorem 3.3.

6 Other classes of well-structured graphs

We described algorithms to compute the scattering number and the toughness of interval graphs.

A similar 'dynamic programming on pieces' was used in [11] to design polynomial time algorithms for the vertex ranking problem. Permutation graphs were considered in detail there and these considerations apply also to the toughness and scattering number problem. However slight modifications are necessary (e.g. in this paper a piece is nonempty and induces a connected subgraph).

The demonstrated approach does not rely much on the structure of interval graphs. There are more classes of well-structured graphs having the properties necessary to design polynomial time algorithms using a 'dynamic programming on pieces'. This leads to the following results.

The scattering number can be computed in time $O(n^6(n+m))$ for permutation graphs, in time $O(n^6(n+m))$ for trapezoid graphs, in time $O(n^4)$ for circular-arc graphs and in time $O(n^6(n+m))$ for circular permutation graphs. The toughness can be computed in time $O(n^9)$ for permutation graphs, in time $O(n^9)$ for trapezoid graphs, in time $O(n^6)$ for circular-arc graphs and in time $O(n^9)$ for circular permutation graphs. We can also show that there is a $O(n^{3d}(n+m))$ algorithm computing the scattering number and a $O(n^{3d+3})$ algorithm computing the toughness for cocomparability graphs of dimension at most d, if an intersection model is part of the input. We refer to [15, 20] for more details on cocomparability graphs of bounded dimension.

A scattering number algorithm for interval graphs, permutation graphs, trapezoid graphs and cocomparability graphs of bounded dimension can also

be used to decide traceability and hamiltonicity for these graph classes within the same timebounds (see Theorem 2.9).

7 Conclusions

We like to mention some open problems. The HAMILTONIAN CIRCUIT problem on split graphs is **NP**-complete [15, 16]. On the other hand, since every 3/2-tough split graph is hamiltonian [21], a polynomial time algorithm computing the toughness of split graphs would allow to recognize a subclass of the hamiltonian split graphs. Does such a polynomial time algorithm exist?
Is the SCATTERING NUMBER problem and the TOUGHNESS problem, respectively, **NP**-complete on cocomparability graphs?

8 Acknowledgements

We like to thank Annelies Jacobs for doing a splendid bibliographical research.

References

[1] C. A. Barefoot, R. Entringer and H. Swart, Vulnerability in graphs — a comparative survey, *Journal of Combinatorial Mathematics and Combinatorial Computing* **1** (1987), 13–22.

[2] D. Bauer, S.L. Hakimi and E. Schmeichel, Recognizing tough graphs is NP-hard, *Discrete Applied Mathematics* **28** (1990), 191–195.

[3] D. Bauer, E. Schmeichel, H.J. Veldman, Cycles in tough graphs — Updating the last four years, Kalamazoo Conference, 1992.

[4] H. Bodlaender, T. Kloks and D. Kratsch, Treewidth and pathwidth of permutation graphs, *Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, Springer-Verlag, Lecture Notes in Computer Science 700, 1993, pp. 114–125.

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

[6] J.A. Bondy, U.S.R. Murty, *Graph Theory with Applications*, New York, 1976.

[7] A. Brandstädt, Special graph classes — a survey, Schriftenreihe des Fachbereichs Mathematik, SM-DU-199, Universität Duisburg Gesamthochschule, 1991.

[8] H.J. Broersma, J. van den Heuvel, H.J. Veldman, (eds.), *Updated contributions to the Twente Workshop on Hamiltonian graph theory*, Memorandum No. 1076, University of Twente, The Netherlands, 1992.

[9] V. Chvátal, Tough graphs and hamiltonian circuits, *Discrete Mathematics* **5** (1973), 215–228.

[10] D.G. Corneil, H. Lerchs and L. Stewart Burlingham, Complement reducible graphs, *Discrete Applied Mathematics* **3** (1981), 163-174.

[11] J.S. Deogun, T. Kloks, D. Kratsch and H. Müller, On vertex ranking for permutation and other graphs, to appear in: *Proceedings of the 11^{th} Annual Symposium on Theoretical Aspects of Computer Science*, 1994.

[12] J.S. Deogun, G. Steiner, Hamiltonian cycle is polynomial on cocomparability graphs, to appear in *SIAM Journal on Computing.*

[13] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, San Francisco, 1979.

[14] P.C. Gilmore and A.J. Hoffman, A characterization of comparability graphs and of interval graphs, *Canadian Journal of Mathematics* **16** (1964), 539–548.

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

[16] D.S. Johnson, The NP-completeness column: an ongoing guide, *Journal of Algorithms* **6** (1985), 434–451.

[17] H.A. Jung, On a class of posets and the corresponding comparability graphs, *Journal on Combinatorial Theory, Series B* **24** (1978), 125–133.

[18] M. Keil, Finding hamiltonian circuits in interval graphs, *Information Processing Letters* **20** (1985), 201–206.

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

[20] T. Kloks, D. Kratsch and J. Spinrad, Treewidth and pathwidth of cocomparability graphs of bounded dimension, Computing Science Note 93/46, Eindhoven University of Technology, Eindhoven, The Netherlands, 1993.

[21] D. Kratsch, J. Lehel and H. Müller, Toughness, hamiltonicity and split graphs, Forschungsergebnisse Math/93/5, F.-Schiller-Universität Jena, Germany, 1993.

[22] J. Lehel. The path partition of cocomparability graphs, manuscript, 1991.

[23] G.K. Manacher, T.A. Mankus, C.J. Smith, An optimum $\Theta(n \log n)$ algorithm for finding a canonical Hamiltonian path and a canonical Hamiltonian circuit in a set of intervals, *Information Processing Letters* **35** (1990), 205–211.

[24] H. Müller, Hamiltonian circuits on chordal bipartite graphs, manuscript.

[25] C.St.J.A. Nash-Williams, Hamiltonian circuits in graphs and digraphs, *The many facets of graph theory*, G.Chartrand, S.G.Kapoor, (eds.), Berlin 1969, 237–243.

[26] G. Steiner, private communication.

[27] D.R. Shier, *Network Reliability and Algebraic Structures*, Oxford University Press, 1991.

[28] W.-K. Shih, T.C. Chern and W.-L. Hsu, An $O(n^2 \log n)$ algorithm for the hamiltonian cycle problem on circular-arc graphs, *SIAM Journal on Computing* **21** (1992), 1026–1046.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399