REPORTS
IN
INFORMATICS

ISSN 0333-3590

Exponential time algorithms for the minimum dominating set problem on some graph classes

Serge Gaspers, Dieter Kratsch and Mathieu Liedloff

REPORT NO 322 **April 2006**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL
http://www.ii.uib.no/publikasjoner/texrap/pdf/2006-322.pdf

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Exponential time algorithms for the minimum dominating set problem on some graph classes

Serge Gaspers* Dieter Kratsch† Mathieu Liedloff†

21st April 2006

Abstract

The Minimum Dominating Set problem remains NP-hard when restricted to chordal graphs, circle graphs and c-dense graphs (i.e. $|E| \geq cn^2$ for a constant c, $0 < c < 1/2$). For each of these three graph classes we present an exponential time algorithm solving the Minimum Dominating Set problem. The running times of those algorithms are $O(1.4173^n)$ for chordal graphs, $O(1.4956^n)$ for circle graphs, and $O(1.2303^{(1+\sqrt{1-2c})n})$ for c-dense graphs.

1 Introduction

During the last years there has been a growing interest in the design of exact exponential time algorithms. Woeginger has written a nice survey on the subject [19] emphasizing the major techniques used to design exact exponential time algorithms. We also refer the reader to the recent survey of Fomin et al. [9] discussing some new techniques in the design of exponential time algorithms. In particular they discuss treewidth based techniques, Measure & Conquer and memorization.

Known results. A set $D \subseteq V$ of a graph $G = (V, E)$ is dominating if every vertex of $V \setminus D$ has a neighbor in D. The Minimum Dominating Set problem (`MDS`) asks to compute a dominating set of the input graph of minimum cardinality.

Exact exponential time algorithms for the Minimum Dominating Set problem have not been studied until recently. By now there is a large interest in this particular problem. In 2004 three papers with exact algorithms for `MDS` have been published. In [10] Fomin et al. presented an $O(1.9379^n)$ time algorithm for general graphs and algorithms for split graphs, bipartite graphs and graphs of maximum degree three with running time $O(1.4143^n)$, $O(1.7321^n)$, $O(1.5144^n)$, respectively. Exact algorithms for `MDS` on general graphs have also been given by Randerath and Schiermeyer [16] and by Grandoni [12]. Their running times are $O(1.8899^n)$ and $O(1.8026^n)$, respectively.

These algorithms have been significantly improved by Fomin et al. in [8] where the authors obtain the currently fastest exact algorithm for `MDS`. Their search tree algorithm is based on the so-called Measure & Conquer approach, and the upper bounds

*Department of Informatics, University of Bergen, N-5020 Bergen. serge.gaspers@ii.uib.no

†Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. (kratsch|liedloff)@univ-metz.fr

on the worst case running times are established by the use of non standard measures. The MDS algorithm has running time $O(1.5263^n)$ and needs polynomial space. Using memorization one can speed up the running time to $O(1.5137^n)$ needing exponential space then. Both variants are based on algorithms for the minimum set cover problem where the input consists of a universe $\mathcal{U}$ and a collection $\mathcal{S}$ of subsets of $\mathcal{U}$. These algorithms need running time $O(1.2354^{|\mathcal{U}|+|\mathcal{S}|})$ and polynomial space, or running time $O(1.2303^{|\mathcal{U}|+|\mathcal{S}|})$ and exponential space [8].

Finally, Fomin and Høie used a treewidth based approach to establish an algorithm to compute a minimum dominating set for graphs of maximum degree three [7] with running time $O(1.2010^n)$.

It is known that the problem MDS is NP-hard when restricted to chordal graphs [5], and circle graphs [13]. Furthermore it is not hard to show that MDS is NP-hard for c-dense graphs.

Our results. In this paper we study the Minimum Dominating Set problem on three graph classes and we obtain algorithms with a running time $O(\alpha^n)$ better than the best known running time for an algorithm solving MDS on general graphs, i.e. $O(1.5137^n)$.

In Section 3 we present an exact algorithm solving the MDS problem on chordal graphs in time $O(1.4173^n)$. In Section 4 an $O(1.4956^n)$ time algorithm to compute a minimum dominating set for circle graphs is established. In Section 5 we give an $O(1.2303^{n(1+\sqrt{1-2c})})$ time algorithm for c-dense graphs, i.e. for all graphs with at least cn^2 edges, where c is a constant with $0 < c < 1/2$.

Our algorithms rely heavily on the minimum set cover algorithms of Fomin et al. [8]. Furthermore the algorithms for chordal graphs and for circle graphs are treewidth based. Both of them use different algorithms for graphs of small treewidth, i.e. at most tn, and for graphs of large treewidth, i.e. larger than tn, where t is chosen to balance the running times of those two algorithms.

The algorithm for circle graphs relies on an upper bound of the treewidth of circle graphs in terms of the maximum degree which is interesting in its own. A related result for graphs of small chordality is provided in [4]. We are not aware of any previous result of this type for circle graphs.

2 Preliminaries

Let $G = (V, E)$ be an undirected and simple graph. For a vertex $v \in V$ we denote by $N(v)$ the neighborhood of v and by $N[v] = N(v) \cup \{v\}$ the closed neighborhood of v. For a given subset of vertices $S \subseteq V$, $G[S]$ denotes the subgraph of G induced by S. The maximum degree of a graph G is denoted by $\Delta(G)$ or by Δ if it is clear from the context which graph is meant.

A clique is a set $C \subseteq V$ of pairwise adjacent vertices. The maximum cardinality of a clique in a graph G is denoted by $\omega(G)$. A dominating set D of a graph $G = (V, E)$ is a subset of vertices such that every vertex of $V - D$ has at least one neighbor in D. The minimum cardinality of a dominating set of G is the domination number of G, and it is denoted by $\gamma(G)$.

Major tools of our paper are tree decompositions and treewidth of graphs. The notions have been introduced by Robertson and Seymour in [17].

Definition 1 (Tree decomposition). Let $G = (V, E)$ be a graph. A *tree decomposition* of G is a pair $(\{X_i : i \in I\}, T)$ where each X_i, $i \in I$, is a subset of V and T is a tree with elements of I as nodes such that we have the following properties :

1. $\cup_{i \in I} X_i = V$;
2. $\forall \{u, v\} \in E$, $\exists i \in I$ s.t. $\{u, v\} \subseteq X_i$;
3. $\forall i, j, k \in I$, if j is on the path from i to k in T then $X_i \cap X_k \subseteq X_j$.

The width of a tree decomposition is equal to $\max_{i \in I} |X_i| - 1$.

Definition 2 (Treewidth). The *treewidth* of a graph G is the minimum width over all its tree decompositions and it is denoted by $tw(G)$.

A tree decomposition is called *optimal* if its width is $tw(G)$.

Definition 3 (Nice tree decomposition). A *nice tree decomposition* $(\{X_i : i \in I\}, T)$ is a tree decomposition satisfying the following properties:

1. every node of T has at most two children;
2. If a node i has two children j and k, then $X_i = X_j = X_k$ (i is a Join Node);
3. If a node i has one child j, then either
   (a) $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ (i is a Insert Node);
   (b) $|X_i| = |X_j| - 1$ and $X_i \subset X_j$ (i is a Forget Node).

Lemma 4 ([14]). *For a constant k, given a tree decomposition of a graph G of width k and $O(n)$ nodes, where n is the number of vertices of G, one can find a nice tree decomposition of G of width k and with at most $4n = O(n)$ nodes in $O(n)$ time.*

Structural and algorithmic properties of graph classes will be mentioned in the corresponding sections. For definitions and properties of graph classes not given in this paper we refer to [6, 11].

3 Domination on chordal graphs

In this section we present an exponential time algorithm for the minimum dominating set problem on chordal graphs.

A graph is *chordal* if it has no chordless cycle of length greater than 3. Chordal graphs are a well-known graph class with its own chapter in Golumbic's monograph [11]. Split graphs, strongly chordal graphs and undirected path graphs are well-studied subclasses of chordal graphs.

We shall use the clique tree representation of chordal graphs that we view as a tree decomposition of the graph. A tree T is as *clique tree* of a chordal graph $G = (V, E)$ if there is a bijection between the maximal cliques of G and the nodes of T such that for each $v \in V$ the cliques containing v induce a subtree of T. It is well-known that $tw(G) \geq \omega(G) - 1$ for all graphs. Furthermore the clique tree of a chordal graph G is an optimal tree decomposition of G, i.e. its width is $\omega(G) - 1$.

Lemma 5. *There is an $O^*(3^{tw(G)})$ time algorithm to compute a minimum dominating set on chordal graphs.* [1]

Proof. Alber et al. have shown in [1] that a minimum dominating set of a graph can be computed in time $O(4^l n)$ if a tree decomposition of width l of the input graph is known. Their algorithm uses a nice tree decomposition of the input graph and a standard bottom up dynamic programming on the tree decomposition. The crucial idea is to assign three different "colors" to the vertices of a bag:

- "black", meaning that the vertex belongs to the dominating set,
- "white", meaning that the vertex is already dominated,
- "gray", meaning that the vertex is not yet dominated.

Now let us assume that the input graph is chordal. A clique tree T of G can be computed in linear time [3]. By Lemma 4, a nice optimal tree decomposition of G can be computed from the optimal tree decomposition T in time $O(n)$ and it has at most $4n$ nodes. Since G is chordal every bag in the nice tree decomposition is a clique. Therefore no bag can have both a black vertex and a gray vertex. Due to this restriction there are at most $2^{|X|}$ possible so-called vector colorings of a bag X (instead of $3^{|X|}$ for general graphs).

Consequently the running time of a modification of the algorithm of Alber et al. to chordal graphs is $O^*(3^{tw(G)})$, where the only modification is to restrict allowed vector colorings of a bag such that black and gray vertices simultaneously are forbidden.

□

The following theorem shows that graphs with sufficiently many vertices of high degree allow to speed up the MDS algorithm for general graphs.

Theorem 6. *Let $t > 0$ be a fixed integer. Then there is a $O(1.2303^{2n-t})$ time algorithm to solve the* MDS *problem if the input graph fulfills the condition $|\{v \in V : d(v) \geq t-2\}| \geq t$.*

Proof. Let $t > 0$ be an integer and $G = (V, E)$ a graph fulfilling the conditions of the theorem. Let $T = \{v \in V : d(v) \geq t-2\}$; thus $|T| \geq t$. Notice that for each minimum dominating set D of G either at least one vertex of T belongs to D, or $T \cap D = \emptyset$.

This allows to find a minimum dominating set of G by the following branching in two types of subproblems: "$v \in D$" for all $v \in T$, and "$T \cap D = \emptyset$". In both cases we shall apply the minimum set cover algorithm of [8] to solve the subproblem. Recall that the minimum set cover instance corresponding to the MDS problem for G has universe $\mathcal{U} = V$ and $\mathcal{S} = \{N[u] : u \in V\}$, and thus $|\mathcal{U}| + |\mathcal{S}| = 2n$ [8]. Consequently the running time for a subproblem will be $O(1.2303^{2n-x})$, where x is the number of vertices plus the number of subsets eliminated from the original minimum set cover problem for the graph G.

Now let us consider the two types of subproblems. For every vertex $v \in T$, we choose v in the minimum dominating set and we execute the Minimum Set Cover algorithm presented in [8] on an instance of size at most $2n-(d(v)+1)-1 \leq 2n-t$. Indeed, we remove from the universe $\mathcal{U}$ the elements of $N[v]$ and we remove from $\mathcal{S}$

[1] Modified big-Oh notation suppresses polynomially bounded factors.

the set corresponding to v. And we branch in the case "discard T": In this case we have an instance of set cover of size at most $2n - |T| = 2n - t$ since for every $v \in T$ we remove from $\mathcal{S}$ the set corresponding to each v.

□

Corollary 7. *There is an algorithm taking as input a graph G and a clique C of G and solving the* MDS *problem in time $O(1.2303^{2n-|C|})$.*

Proof. Note that every vertex in C has degree at least $|C| - 1$.

□

Our algorithm on chordal graphs works as follow: If the graph has a large treewidth then it necessarily has a large clique and we apply Corollary 7. Otherwise the graph has a small treewidth and we use Lemma 5.

Theorem 8. *There is an $O(1.4173^n)$ time algorithm to solve the* MDS *problem on chordal graphs.*

Proof. If $tw(G) \leq 0.3174n$, by Lemma 5, MDS is solvable in time $O(3^{0.3174n}) = O(1.4173^n)$. Otherwise, $tw(G) > 0.3174n$ and using Corollary 7 we obtain an $O(1.2303^{2n-0.3174n}) = O(1.4173^n)$ time algorithm.

□

4 Domination on circle graphs

In this section, we present an exponential time algorithm for MDS on circle graphs in a treewidth based approach. For a survey on treewidth based exponential time algorithms we refer to [9].

Definition 9. A *circle graph* is an intersection graph of chords in a circle. More precisely, G is a circle graph, if there is a circle with a collection of chords, such that one can associate in a one-to-one manner to each vertex a chord such that two vertices are adjacent if and only if the corresponding chords have a nonempty intersection. The circle and all its chords are called a *circle model* of the graph.

Our algorithm heavily relies on a linear upper bound on the treewidth of circle graphs in terms of the maximum degree: $tw(G) \leq 4\Delta(G) - 1$. This bound is interesting in its own and it is likely that such bounds for circle graphs or other graph classes can be used to construct exponential time algorithms for NP-hard problems on special graph classes in a way similar to our approach for domination on circle graphs.

The algorithm uses the treewidth to branch into two different approaches: one for "small treewidth" and one for "high treewidth". If there are many vertices of high degree in the input graph, Theorem 6 is used to continue, and if not, the treewidth is "small" and we use an $O^*(4^{tw(G)})$ algorithm to compute a minimum dominating set.

Theorem 10 ([1]). *Suppose the graph $G = (V, E)$ and a tree decomposition of width ℓ of G are given. Then there is an $O(4^\ell N)$ time algorithm to compute a minimum dominating set of G, where N is the number of nodes of the tree decomposition.*

We start with a brief summary of Kloks' algorithm to compute the treewidth of a circle graph [15]. Consider the circle model of a circle graph G. Go around the circle and place a new point (so-called *scanpoints*) between every two consecutive end points of chords. The treewidth of a circle graph can be computed by considering all possible triangulations of the polygon $\mathcal{P}$ formed by the convex hull of these scanpoints. The weight of a triangle in this triangulation is the number of chords in the circle model that cross this triangle. The weight of the triangulation $\mathcal{T}$ is the maximum weight of the triangles in $\mathcal{T}$. The treewidth of the graph is the minimum weight minus one over all triangulations of $\mathcal{P}$.

Theorem 11 ([15]). *There exists an $O(n^3)$ algorithm to compute the treewidth of circle graphs, that also computes an optimal tree decomposition.*

We rely on the following technical definitions in our construction of a tree decomposition of width at most $4\Delta(G) - 1$ for each circle graph G. The construction will be given in the proof of Theorem 15.

Definition 12. A *scanline* $\tilde{s} = \langle \tilde{a}, \tilde{b} \rangle$ is a line segment connecting two scanpoints $\tilde{a}$ and $\tilde{b}$.

To emphasize the difference between scanlines and chords we use different notations: A chord v connecting two end points c and d in the circle model of the graph is denoted $v = [c, d]$. We also use the following convention: two scanlines with empty intersection or intersecting in exactly one scanpoint are said to be *non-crossing*.

Definition 13. Let $\tilde{s}_1$ and $\tilde{s}_2$ be two non-crossing scanlines. A scanline $\tilde{s}$ is *between* $\tilde{s}_1$ and $\tilde{s}_2$ if every path from a scanpoint of $\tilde{s}_1$ to a scanpoint of $\tilde{s}_2$ along the circle passes through a scanpoint of $\tilde{s}$.

Definition 14. A set S of *parallel* scanlines is a set of scanlines respecting

(i) $|S| \leq 2$ and the scanlines of S are non-crossing, or

(ii) $|S| > 2$ and for every subset of three scanlines in S, one of these scanlines is between the other two.

The following theorem is one of the main results of this paper. It shows that the treewidth $tw(G)$ of circle graphs can be upper bounded by a linear function of the maximum degree $\Delta(G)$ of the graph G. Surprisingly, no linear bound seems to have been known prior to our work.

Theorem 15. *For every circle graph G holds $tw(G) \leq 4\Delta(G) - 1$.*

Proof. We construct a triangulation of the polygon $\mathcal{P}$ such that every triangle has weight at most 4Δ, i.e. it intersects at most 4Δ chords, and therefore the corresponding tree decomposition has width at most $4\Delta - 1$.

Notice that by the definition of a circle graph, every chord intersects at most Δ other chords. The triangulation of the polygon $\mathcal{P}$ is obtained by constructing the corresponding set of scanlines S which is explained by the following procedures. Having described our algorithm, we will analyze the number of chords that cross each triangle and show that it is less than or equal to 4Δ.

1. Description of the algorithm

• **FirstCut().** Start with $S = \emptyset$. Choose a chord v in the circle model of the graph G. Call **ScanChord(**S, v**)**. Call **ParaCuts(**S**)**.

• **ScanChord(**$S, v = [a, b]$**).** Let $\tilde{c}$ and $\tilde{c}'$ (resp. $\tilde{d}$ and $\tilde{d}'$) be the two scanpoints closest to a (resp. b) on the circle such that the order of the points on the circle is $\tilde{c}, a, \tilde{c}', \tilde{d}', b$ and $\tilde{d}$. Now the algorithm adds the following three scanlines to S: $\tilde{s}_1 = \langle \tilde{c}, \tilde{d} \rangle, \tilde{s}_2 = \langle \tilde{c}', \tilde{d}' \rangle$ and $\tilde{s}_3 = \langle \tilde{c}, \tilde{d}' \rangle$. If $\tilde{c} = \tilde{d}$ (or $\tilde{c}' = \tilde{d}'$) then we add only the scanline $\tilde{s}_2$ (or $\tilde{s}_1$).

• **ParaCuts(**S**).** While S is not a maximal (by inclusion) parallel set of scanlines in $\mathcal{P}$, choose a chord v such that S remains parallel when calling **ScanChord(**S, v**)**. Call **ScanChord(**S, v**)**. If S is maximal parallel, every polygon inside $\mathcal{P}$ is delimited by one or two scanlines. We call the polygons that are delimited by one scanline *outer polygons*, and those that are delimited by two scanlines *inner polygons* (see Fig. 1). There are exactly two outer polygons now, one delimited by $\tilde{s}_1$ and the other one by $\tilde{s}_2$. Call **TriangOuter(**$S, \tilde{s}_1$**)** and **TriangOuter(**$S, \tilde{s}_2$**)**. For every inner polygon, call **TriangInner(**$S, \tilde{t}_1, \tilde{t}_2$**)** where $\tilde{t}_1$ and $\tilde{t}_2$ are the two scanlines delimiting this polygon.

• **TriangOuter(**$S, \tilde{s} = \langle \tilde{a}, \tilde{b} \rangle$**).** The scanline $\tilde{s}$ divides the polygon $\mathcal{P}$ into two parts. Call $\mathcal{P}_{\tilde{s}}$ the polygon delimited by $\tilde{s}$ and the part of $\mathcal{P}$ that does not contain any scanlines. Add a scanline between $\tilde{a}$ and every scanpoint of $\mathcal{P}_{\tilde{s}}$ except $\tilde{a}$ and $\tilde{b}$ to S.

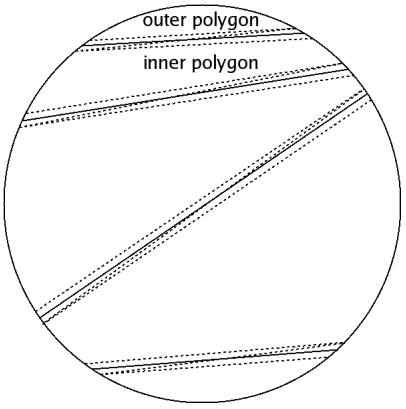

Figure 1: ParaCuts

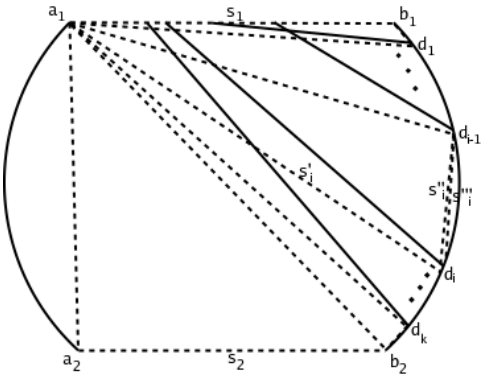

Figure 2: TriangInner

• **TriangInner(**$S, \tilde{s}_1 = \langle \tilde{a}_1, \tilde{b}_1 \rangle, \tilde{s}_2 = \langle \tilde{a}_2, \tilde{b}_2 \rangle$**).** Let the end points of $\tilde{s}_1$ and $\tilde{s}_2$ be ordered $\tilde{a}_1, \tilde{b}_1, \tilde{b}_2, \tilde{a}_2$ around the circle. W.l.o.g. assume that fewer chords cross the line $\tilde{a}_1, \tilde{a}_2$ than the line $\tilde{b}_1, \tilde{b}_2$. Now add a new scanline $\tilde{t} = \langle \tilde{a}_1, \tilde{a}_2 \rangle$ to S. Call **OuterParaCuts(**$S, \tilde{t}$**)**. Go around the circle from $\tilde{b}_1$ to $\tilde{b}_2$ (without passing through $\tilde{a}_1$ and $\tilde{a}_2$). Every time one passes through an end point e_i, $i = 1, ..., k$, (where k is the number of chords that cross $\tilde{s}_1$ and $\tilde{b}_1, \tilde{b}_2$) of a chord v_i that crosses $\tilde{s}_1$, add the following scanlines to S:

- $\tilde{s}'_i = \langle \tilde{a}_1, \tilde{d}_i \rangle$ with $\tilde{d}_i$ being the scanpoint immediately following e_i
- $\tilde{s}''_i = \langle \tilde{d}_i, \tilde{d}_{i-1} \rangle$ with $\tilde{d}_0 = \tilde{b}_1$
- $\tilde{s}'''_i = \langle \tilde{d}_{i-1}, \tilde{d}'_i \rangle$ with $\tilde{d}'_i$ being the scanpoint just before $\tilde{d}_i$.

To triangulate the part of the polygon $\mathcal{P}$ delimited by $\tilde{s}_i'''$ that does not intersect any scanlines, execute **OuterParaCuts(**$S, \tilde{s}_i'''$**)**. Finally, add the scanlines $\tilde{s}_3 = \langle \tilde{d}_k, \tilde{b}_2 \rangle$ and $\tilde{s}_4 = \langle \tilde{b}_2, \tilde{a}_1 \rangle$ to S (see Fig. 2). Execute **OuterParaCuts(**$S, \tilde{s}_3$**)**.

• **OuterParaCuts(**$S, \tilde{s} = \langle \tilde{a}, \tilde{b} \rangle$**).** This procedure is similar to **ParaCuts** on the outer polygon delimited by $\tilde{s}$. Call $\mathcal{P}_{\tilde{s}}$ the polygon delimited by $\tilde{s}$ that does not contain any scanlines. Create a new set of scanlines $S' = \{\tilde{s}\}$. While S' is not a maximal (by inclusion) parallel set of scanlines for $\mathcal{P}_{\tilde{s}}$, choose a chord v in $\mathcal{P}_{\tilde{s}}$ such that S' remains parallel when calling **ScanChord(**S', v**)**. Call **ScanChord(**S', v**)**. After that there is exactly one outer polygon in $\mathcal{P}_{\tilde{s}}$, delimited by a scanline $\tilde{t}$. Call **TriangOuter(**$S', \tilde{t}$**)**. For every inner polygon in $\mathcal{P}_{\tilde{s}}$, call **TriangInner(**$S', \tilde{t}_1, \tilde{t}_2$**)** where $\tilde{t}_1$ and $\tilde{t}_2$ are the two scanlines delimiting this polygon. Add the set of new scanlines S' to S.

2. Analysis of the algorithm

In the main procedure, **FirstCut**, no scanlines are directly added to S.

Every time **ScanChord** is executed, one or three scanlines are added to S. They form at most two triangles: $\tilde{c}, \tilde{d}, \tilde{d}'$ and $\tilde{c}, \tilde{d}', \tilde{c}'$. Each of them intersects at most $\Delta + 1$ chords: v and the chords crossing v. Furthermore, at most Δ chords cross $\tilde{s}'$ and $\tilde{s}''$, precisely the chords that cross v.

In the procedure **ParaCuts**, no scanlines are directly added to S. Moreover, when it calls the procedures **TriangOuter** and **TriangInner**, the set S is maximal parallel, which is a necessary condition for these procedures.

When **TriangOuter** is called, two conditions are always respected:
(i) S is maximal parallel by inclusion, and
(ii) at most 2Δ chords cross $\tilde{s}$.
The condition (i) implies that every chord that intersects $\mathcal{P}_{\tilde{s}}$ crosses $\tilde{s}$. Together with condition (ii) we obtain that at most 2Δ chords intersect $\mathcal{P}_{\tilde{s}}$. So any triangulation of $\mathcal{P}_{\tilde{s}}$ produces triangles with weight at most 2Δ.

When **TriangInner** is called, three conditions are always respected:
(i) S is a maximal parallel set of scanlines, and
(ii) at most Δ chords cross one of the scanlines; suppose this is $\tilde{s}_2$
(iii) at most 2Δ chords cross $\tilde{s}_1$.
There are at most 3Δ chords inside the quadrilateral $\tilde{a}_1, \tilde{b}_1, \tilde{b}_2, \tilde{a}_2$ since there is no chord crossing both the lines $\tilde{a}_1, \tilde{a}_2$ and $\tilde{b}_1, \tilde{b}_2$ (because S is maximal parallel). As fewer chords cross $\tilde{a}_1, \tilde{a}_2$ than $\tilde{b}_1, \tilde{b}_2$, at most $3/2\Delta$ chords cross the new scanline $\tilde{t} = \langle \tilde{a}_1, \tilde{a}_2 \rangle$. So, when we call **OuterParaCuts(**$S, \tilde{t}$**)** the condition that $\tilde{t}$ intersects at most 2Δ chords is respected. For every end point e_i of a chord v_i that crosses $\tilde{s}_1$, we create two triangles: $\tilde{a}_1, \tilde{d}_{i-1}, \tilde{d}_i$ and $\tilde{d}_i, \tilde{d}_{i-1}, \tilde{d}_i'$. The first triangle intersects at most 4Δ chords: at most 2Δ chords that cross $\tilde{s}_1$ (but neither v_i nor v_{i-1}), at most Δ chords that cross v_{i-1} and at most Δ chords that cross v_i. Moreover, there are at most $2\Delta + 1$ chords that intersect $\tilde{s}_i''$ and at most 2Δ chords intersect $\tilde{s}_i'''$. So, the weight of the triangle $\tilde{d}_i, \tilde{d}_{i-1}, \tilde{d}_i'$ is at most $2\Delta + 1$ and when we call **OuterParaCuts(**$S, \tilde{s}_i'''$**)** we respect the condition that the second parameter of the procedure is a scanline that crosses at most 2Δ chords.
After adding the scanlines $\tilde{s}_3$ and $\tilde{s}_4$ we obtain two more triangles: $\tilde{a}_1, \tilde{d}_k, \tilde{b}_2$ and $\tilde{a}_1, \tilde{b}_2, \tilde{a}_2$. The first one intersects at most $7/2\Delta$ chords: at most 2Δ that cross $\tilde{s}_1$, at most Δ that cross v_k and at most Δ that cross $\tilde{s}_2$ of which we have already counted $1/2\Delta$ crossing $\tilde{s}_1$. At most $5/2\Delta$ chords intersect the triangle $\tilde{a}_1, \tilde{b}_2, \tilde{a}_2$: at most 2Δ that intersect $\tilde{s}_1$ and at most Δ that intersect $\tilde{s}_2$ of which we have already counted

$1/2\Delta$ crossing $\tilde{s}_1$. Moreover at most 2Δ chords cross $\tilde{s}_3$, so **OuterParaCuts(**$S, \tilde{s}_3$**)** has valid parameters.

In the procedure **OuterParaCuts**, no scanlines are directly added to S. The following condition is always respected:
(i) at most 2Δ chords cross $\tilde{s}$.
During this procedure, we consider only the polygon $\mathcal{P}_{\tilde{s}}$. A new set of scanlines $S' = \{\tilde{s}\}$ is created and is made maximal parallel by inclusion by calling **ScanChord**. If $\{\tilde{s}\}$ is already maximal parallel, then **TriangOuter(**$S', \tilde{s}$**)** is called and the two conditions of that procedure are respected. If other scanlines had to be added to S' to make it maximal parallel, the procedure **TriangOuter(**$S', \tilde{t}$**)** is called for the outer polygon where $\tilde{t}$ is a scanline intersecting at most Δ chords. Moreover, the procedure **TriangInner(**$S, \tilde{t}_1, \tilde{t}_2$**)** is called for the inner polygons. Every scanline delimiting the inner polygons intersects at most Δ chords, except $\tilde{s}$ that can intersect up to 2Δ chords. So, we respect the condition for **TriangInner** that one scanline intersects at most Δ chords and the other one at most 2Δ chords. Finally, S' is added to S which does not create any new triangles.

We have provided a recursive algorithm to triangulate the polygon $\mathcal{P}$ and have shown that the obtained triangulation does not contain triangles intersecting more than 4Δ chords. Thus the corresponding tree decomposition of G has width at most $4\Delta - 1$. □

Linear upper bounds for the treewidth in terms of the maximum degree seem to have an immediate use in the design of treewidth based exact algorithms. Using Theorem 16 we obtain an algorithm to compute a minimum dominating set for circle graphs in time $O(1.4956^n)$. The algorithm **DS-circle** is simple and also based on the algorithms of Theorem 10 and Theorem 6.

Algorithm DS-circle(circle graph $G = (V, E)$**; circle model of** G**)**
Input: A circle graph G and its circle model.
Output: The domination number $\gamma(G)$ of G.
  $\lambda \leftarrow 0.2322$
  $X \leftarrow \emptyset$
  Compute the treewidth $tw(G)$ of G using theorem 11
  while $tw(G - X) \geq \lambda n$ **do**
    $X \leftarrow X \cup \{u\}$ where u is a vertex of $G - X$ of highest degree
  if $|X| \geq \lambda n/4$ **then**
    use the algorithm of Theorem 6 and return the result
  else
    use the algorithm of Theorem 10 and return the result

Theorem 16. *Given a circle graph* $G = (V, E)$*, algorithm* **DS-circle** *computes a minimum dominating set of* G *in time* $O(1.4956^n)$.

Proof. The algorithm constructs a vertex set $X = \{x_1, x_2, ..., x_k\}$ starting from an empty set by adding maximum degree vertices of the remaining graph to the set X until $tw(G - X) < \lambda n$.

When the vertex x_i is added to $X = \{x_1, x_2, ..., x_{i-1}\}$, we have $tw(G - X) \geq \lambda n$. The vertex $x_i \in V - X$ is of highest degree in $G - X$, i.e. $d(x_i) = \Delta(G - X)$. We have $d(x_i) > tw(G - X)/4$ by Theorem 15. Now, $d(x_i) > \lambda n/4$ because $tw(G - X) \geq \lambda n$. So, $\forall x_i \in X, d(x_i) > \lambda n/4$.

In the case $|X| \geq \lambda n/4$, we have a subset $X \subseteq V$ such that $\forall v \in X, d(v) > \lambda n/4$. So, according to Theorem 6, a minimum dominating set can be found in time $O(1.2303^{2n-\lambda n/4}) = O(1.4956^n)$.

In the other case, $|X| < \lambda n/4$ and $tw(G - X) < \lambda n$. As adding one vertex to a graph increases its treewidth at most by one, $tw(G) < \lambda n + \lambda n/4$. Using the algorithm of Theorem 10, a minimum dominating set is determined in time $O^*(4^{tw(G)}) = O(4^{(5\lambda/4)n}) = O(1.4956^n)$.

□

5 Domination on dense graphs

It is known that problems like Independent Set, Hamiltonian Circuit and Hamiltonian Path remain NP-complete when restricted to graphs having a large number of edges [18]. It is not hard to show that the MDS problem on c-dense graph is also NP-complete. A proof is given in the Appendix. In this section we present an exponential time algorithm for MDS problem on c-dense graphs.

Definition 17. A graph $G = (V, E)$ is said to be *c-dense* (or simply dense if there is no ambiguity), if $|E| \geq cn^2$ where c is a constant with $0 < c < 1/2$.

The main idea of our algorithm is to find a large subset of vertices of large degree. Despite the approach of the previous sections, neither clique trees nor tree decompositions will be used here.

Lemma 18. *For some fixed $1 \leq t \leq n$, $1 \leq t' \leq n - 1$, any graph $G = (V, E)$ with $|E| \geq 1 + \dfrac{(t-1)(n-1) + (n-t+1)(t'-1)}{2}$ has a subset $T \subseteq V$ such that*

(i) $|T| \geq t$,

(ii) $\forall v \in T, d(v) \geq t'$.

Proof. Let $1 \leq t \leq n$, $1 \leq t' \leq n - 1$, and a graph $G = (V, E)$ such that there is no subset T with the previous properties. Then for any subset $T \subseteq V$ of size at least t, $\exists v \in T$ such that $d(v) < t'$. Then a such graph can only have at most $k = k_1 + k_2$ edges where : $k_1 = (t-1)(n-1)/2$ which corresponds to $t - 1$ vertices of degree $n-1$ and $k_2 = (n-t+1)(t'-1)$ which corresponds to $n-(t-1)$ vertices of degree $t' - 1$. Observe that if one of the $n - (t - 1)$ vertices has a degree greater than $t' - 1$ then the graph has a subset T with the required properties, a contradiction.

□

Lemma 19. *Every c-dense graph $G = (V, E)$ has a set $T \subseteq V$ fulfilling*

(i) $|T| \geq n - \dfrac{\sqrt{9 - 4n + 4n^2 - 8cn^2} - 3}{2}$,

(ii) $\forall v \in T, d(v) \geq n - \dfrac{\sqrt{9 - 4n + 4n^2 - 8cn^2} + 1}{2}$.

Proof. We apply Lemma 18 with $t' = t - 2$. Since we have a dense graph, $|E| \geq cn^2$. Using inequality $1 + ((t-1)(n-1) + (n-t+1)(t-3))/2 \geq cn^2$ we obtain that in a dense graph the value of t in Lemma 18 is such that $n + \frac{3-\sqrt{9-4n+4n^2-8cn^2}}{2} \leq t \leq n \leq n + \frac{3+\sqrt{9-4n+4n^2-8cn^2}}{2}$.

□

Theorem 20. *For any c with $0 < c < 1/2$, there is a $O(1.2303^{n(1+\sqrt{1-2c})})$ time algorithm to solve the* MDS *problem on c-dense graphs.*

Proof. Combining Theorem 6 and Lemma 19 we obtain an algorithm for solving the Minimum Dominating Set problem in time

$$1.2303^{2n-(n-\frac{\sqrt{9-4n+4n^2-8cn^2}-3}{2})} = O(1.2303^{n(1+\sqrt{1-2c})}).$$

□

6 Conclusions

In this paper we presented several exponential time algorithms to solve the Minimum Dominating Set problem on graph classes for which this problem remains NP-hard. All these algorithms are faster than the best known algorithm to solve MDS on general graphs. We would like to mention that any faster algorithm for the Minimum Set Cover problem, i.e. of running time $O(\alpha^{|\mathcal{U}|+|\mathcal{S}|})$ with $\alpha < 1.2303$, could immediately be used to speed up all our algorithms.

Besides classes of sparse graphs (see e.g. [7]) two more classes are of great interest in this respect: split and bipartite graphs. For split graphs, combining ideas of [10] and [8] one easily obtains an $O(1.2303^n)$ algorithm. Unfortunately, despite our efforts we could not construct an exponential time algorithm to solve MDS on bipartite graphs beating the best known algorithm for general graphs.

References

[1] Alber, J., H. L. Bodlaender, H. Fernau, T. Kloks and R. Niedermeier, Fixed parameter algorithms for dominating set and related problems on planar graphs, *Algorithmica* **33**, (2002), pp. 461–493. 3, 10

[2] Bertossi, A. A., Dominating sets for split and bipartite graphs., *Inform. Process. Lett.* **19**, (1984), pp. 37–40. 6

[3] Blair, J. R. S. and B. W. Peyton, An introduction to chordal graphs and clique trees, *Graph theory and sparse matrix computation*, IMA Vol. Math. Appl., vol. 56, Springer, 1993, pp. 1–29. 3

[4] Bodlaender, H. L. and D. M. Thilikos, Graphs with branchwidth at most three, *J. Algorithms* **32**, (1999), pp. 167–194. 1

[5] Booth, K. S. and J. H. Johnson, Dominating sets in chordal graphs, *SIAM J. Comput.* **11**, (1982), pp. 191–199. 1

[6] Brandstädt, A., V. Le, and J. P. Spinrad, *Graph classes: A survey*, SIAM Monogr. Discrete Math. Appl., Philadelphia, 1999. 2

[7] Fomin, F.V., and K. Høie, Pathwidth of cubic graphs and exact algorithms, Technical Report 298, Department of Informatics, University of Bergen, Norway, 2005. 1, 6

[8] Fomin, F.V., F. Grandoni, D. Kratsch, Measure and conquer: Domination - A case study, *Proceedings of ICALP 2005*, *LNCS* **3380**, (2005), pp. 192–203. 1, 3, 6

[9] Fomin, F.V., F. Grandoni, D. Kratsch, Some new techniques in design and analysis of exact (exponential) algorithms, *Bull. EATCS*, **87**, (2005), pp. 47–77. 1, 4

[10] Fomin, F.V., D. Kratsch, and G. J. Woeginger, Exact (exponential) algorithms for the dominating set problem, *Proceedings of WG 2004*, *LNCS* **3353**, (2004), pp. 245–256. 1, 6

[11] Golumbic, M. C., *Algorithmic graph theory and perfect graphs*, Academic Press, New York, 1980. 2, 3

[12] Grandoni, F., A note on the complexity of minimum dominating set, *J. Discrete Algorithms*, to appear. 1

[13] Keil, J. M., The complexity of domination problems in circle graphs, *Discrete Appl. Math.* **42**, (1993), pp. 51–63. 1

[14] Kloks, T., *Treewidth. Computations and approximation*, LNCS **842**, Springer-Verlag, Berlin, 1994. 4

[15] Kloks, T., Treewidth of Circle Graphs, *Internat. J. Found. Comput. Sci.* **7**, (1996) pp. 111–120. 4, 11

[16] Randerath, B., and I. Schiermeyer, Exact algorithms for Minimum Dominating Set, Technical Report zaik-469, Zentrum fur Angewandte Informatik, Köln, Germany, 2004. 1

[17] Robertson, N. and P. D. Seymour, Graph Minors. II. Algorithmic Aspects of Tree-Width, *J. Algorithms* **7**, (1986), pp. 309–322. 2

[18] Schiermeyer, I., Problems remaining NP-complete for sparse or dense graphs, *Discuss. Math. Graph Theory* **15**, (1995), pp. 33–41. 5

[19] Woeginger, G.J., Exact algorithms for NP-hard problems: A survey, *Combinatorial Optimization - Eureka, You Shrink!*, *LNCS* **2570**, (2003), pp. 185–207. 1

Appendix

An easy way to show that an NP-complete graph problem remains NP-complete for c-dense graphs, for any c with $0 < c < 1/2$, is to construct the graph G' by adding a sufficiently large complete graph as new component to the original graph G such that G' is c-dense. This simple reduction can be used to show that various NP-complete graph problems remain NP-complete for c-dense graphs. To name a few problems: INDEPENDENT SET (since $\alpha(G') = \alpha(G) + 1$), DOMINATING SET (since $\gamma(G') = \gamma(G) + 1$), PARTITION INTO CLIQUES (since $\kappa(G') = \kappa(G) + 1$), VERTEX COVER, FEEDBACK VERTEX SET and MINIMUM FILL-IN.

Theorem 21. *For any constant c with $0 < c < 1/2$, the problem to decide, whether a c-dense graph has a dominating set of size at most k, is NP-complete.*

Proof. Let c be any constant with $0 < c < 1/2$. Clearly DOMINATING SET, and thus also DOMINATING SET for c-dense graphs is in NP.

It is show in [2] that the problem of determining if a split graph has a dominating set of size k is NP-complete. We shall provide a polynomial many-one reduction from DOMINATING SET for split graphs to DOMINATING SET for c-dense graphs. Let k be an integer and $G = (V, E)$ a split graph where I and C form a partition of the vertices of G such that I is an independent set and C is a clique.

First we construct a c-dense graph $G_D = (V_D, E_D)$ with $E_D \geq c \cdot |V_D|^2$. The graph $G_D = (V_D, E_D)$ is obtained from the graph $G = (C \cup I, E)$ by adding a clique C' of size $\lceil (1+4c|I \cup C| + \sqrt{1 + 8c|I \cup C|(1 + |I \cup C|)})/(2-4c) \rceil$ to G and adding all edges with one end point in C and the other in C'. Note that this guarantees that G_D is a split graph with a partition of V_D into an independent set I and a clique $C \cup C'$. Furthermore the number of edges of G_D is greater than $c(|I \cup C| + |C'|)^2$, and hence G_D is a c-dense graph.

Now we show that G has a dominating set of size a most k if and only if G_D has a dominating set of size at most k.

First, assume that G_D has a dominating set D with $|D| \leq k$. Since $N_{G_D}[x'] \subseteq N_{G_D}[x]$ for all $x' \in C'$ and all $x \in C$ we may replace each vertex of C' belonging to D by a vertex of C. In this way we obtain a dominating set $D' \subseteq I \cup C$ of G_D such that $|D'| \leq k$. Consequently D' is also a dominating set of G.

Conversely, assume that D is a dominating set of G of size at most k. If D contains at least one vertex of C then D is also a dominating set of G_D since each vertex of C' is adjacent to all vertices of C. Otherwise, D contains no vertex of C and thus each vertex in C has at least one neighbor in $D \cap I$. In this second case we replace any vertex $s \in D \cap I$ by a neighbor $t \in C$ and obtain $D' = (D \setminus \{s\}) \cup \{t\}$. Then D' is a dominating set of G since $N_{G_D}[s] \subseteq N_{G_D}[t]$. Furthermore, since D' contains a vertex of C it is also a dominating set of G_D. Hence G_D has in each case a dominating set of size at most k.

Thus we obtain that the problem of deciding whether a c-dense graph has a dominating set of size at most k is NP-complete. □