ELSEVIER

Discrete Applied Mathematics 82 (1998) 43 - 77

DISCRETE APPLIED MATHEMATICS

The algorithmic use of hypertree structure and maximum neighbourhood orderings

Andreas Brandstädt [a], Victor D. Chepoi [b,1,2], Feodor F. Dragan [b,*,2]

[a] *Universität Rostock, FB Informatik, D 18051 Rostock, Germany*

[b] *Department of Mathematics and Cybern., Moldavian State University, A. Mateevici str. 60, Chişinău 277009, Moldova*

Received 5 July 1996; received in revised form 20 August 1997

Abstract

The use of (generalized) tree structure in graphs is one of the main topics in the field of efficient graph algorithms. The well-known partial k-tree (resp. treewidth) approach belongs to this kind of research and bases on a tree structure of constant-size bounded maximal cliques. Without size bound on the cliques this tree structure of maximal cliques characterizes chordal graphs which are known to be important also in connection with relational database schemes where hypergraphs with tree structure (acyclic hypergraphs) and their elimination orderings (perfect elimination orderings for chordal graphs, Graham-reduction for acyclic hypergraphs) are studied.

We consider here graphs with a tree structure which is dual (in the sense of hypergraphs) to that one of chordal graphs (therefore we call these graphs *dually chordal*). The corresponding vertex elimination orderings of these graphs are the *maximum neighbourhood orderings*. These orderings were studied recently in several papers and some of the algorithmic consequences of such orderings are given.

The aim of this paper is a *systematic treatment of the algorithmic use of maximum neighbourhood orderings*. These orderings are useful especially for dominating-like problems (including Steiner tree) and distance problems. Many problems efficiently solvable for strongly chordal and doubly chordal graphs remain efficiently solvable for dually chordal graphs too.

Our results on dually chordal graphs not only generalize, but also improve and extend the corresponding results on strongly chordal and doubly chordal graphs, since a maximum neighbourhood ordering (if it exists) can be constructed in linear time and we consequently use the underlying structure properties of dually chordal graphs closely connected to hypergraphs. Furthermore, a collection of problems remaining NP-complete on dually chordal graphs is given.
© 1998 Elsevier Science B.V. All rights reserved.

Keywords: Tree structure; Hypertree; Disk hypergraph; Dually chordal graph; Strongly chordal graph; Maximum neighbourhood ordering; Domination; Duality; Location problem; Steiner tree; Linear-time algorithm

* Correspondence address: Universität Rostock, FB Informatik, Albert-Einstein-Str. 21, D-18051 Rostock, Germany. E-mail: dragan@informatik.uni-rostock.de.

[1] Current Address: Laboratoire de Biomathématiques, Université d'Aix Marseille II, 27 Bd Jean Moulin, F-13385 Marseille Cedex 5, France.

[2] Supported by the VW-Stiftung Project No. I 69041.

0166-218X 98/$19.00 © 1998 Elsevier Science B.V. All rights reserved
PII S0166-218X(97)00125-X

1. Introduction

It is well-known that the classical graph problems such as minimum clique covering, minimum coloring, maximum clique and maximum independent set are solvable in polynomial time on perfect graphs, i.e. graphs on which equality between the clique covering number and independence number holds for any induced subgraph. This is mainly due to the result of Lovász [29] that perfect graphs are exactly the line graphs of normal hypergraphs. For normal hypergraphs transversal and matching problems are solved in polynomial time by applying the linear programming algorithms and duality results; see [21]. Although the application of linear programming to perfect graphs leads to polynomial algorithms, in many cases the problems can be solved more efficiently (or even in an optimal way) on special classes of perfect graphs. Chordal graphs represent the most illustrative example to this respect. It is well-known that chordal graphs are characterized as graphs which have a perfect elimination ordering, which may be computed in linear time by using the maximum cardinality search of Tarjan and Yannakakis [39]. Using such an ordering Rose et al. [35] presented linear algorithms for all the four above-mentioned problems. Certainly, this is a particular instance of the fact that chordal graphs are the underlying graphs of α-acyclic hypergraphs, where packing and covering problems are solved in linear time.

Another important class of graph problems is formed by domination-like problems; for references see e.g. [23]. Unfortunately, the perfectness of a given graph is useless in solving such problems. Even for split graphs – a subclass of chordal graphs – the domination problem remains $\mathbb{NP}$-complete. For a long time polynomial algorithms were known only for trees. The first substantial progress was obtained by Farber [17, 18], Chang and Nemhauser [11] and Lubiw [30]. They defined and investigated strongly chordal graphs, a subclass of chordal graphs for which domination problems have efficient solutions [18, 11, 40]. Again, the polynomiality is implicitely based on a special property of hypergraphs: strongly chordal graphs may be characterized as underlying (line) graphs of totally balanced hypergraphs; they are the graphs whose neighbourhood hypergraphs (or equivalently, clique hypergraphs) are totally balanced. Note that the domination problem is nothing else than the transversal problem in the corresponding neighbourhood hypergraph. In addition, total balancedness allows to characterize strongly chordal graphs by a special neighbourhood ordering, called the strong elimination ordering [17]. Note that a strong elimination ordering is a perfect elimination ordering. Unfortunately, to date, there are no linear algorithms to give a strong elimination ordering of a strongly chordal graph $G=(V,E)$: the fastest such algorithm takes $O(|E|\log|V|)$ [33] or $O(|V|^2)$ [36] time. When a strong elimination ordering is given then the domination-like problems are solved in linear time [9, 18, 40, 10, 27].

The most general class of hypergraphs, comprising totally balanced hypergraphs, for which the transversal and matching problems are solvable in linear time is the class of hypertrees (or subtree hypergraphs): A hypergraph H is a *hypertree* if there is a tree T such that the hyperedges of H induce subtrees in T. Graphs whose neighbourhoods form a hypertree were introduced and characterized in [16] (see also [5]).

These graphs are exactly the underlying graphs of hypertrees. Due to the duality between hypertrees and α-acyclic hypergraphs, these graphs are in this sense dual to chordal graphs. That is why we call them *dually chordal graphs* (initially these graphs appeared under the name *HT-graphs* [16]). As well as chordal graphs or strongly chordal graphs, dually chordal graphs have a characteristic ordering of vertices, called the *maximum neighbourhood ordering*. Graphs with maximum neighbourhood ordering were independently introduced and characterized in several papers [2, 16, 37]. Ref. [32] also introduced maximum neighbourhoods but only in connection with chordal graphs (chordal graphs with maximum neighbourhood ordering are called there *doubly chordal graphs*). Using the clique hypergraph properties Moscarini [32] presented a polynomial algorithm for the connected domination problem on doubly chordal graphs. Note that the doubly chordal graphs may be defined as graphs which are simultaneously chordal and dually chordal, i.e. they are the underlying graphs of biacyclic (i.e. acyclic and the dual hypergraph also acyclic) hypergraphs.

Unlike chordal graphs, dually chordal graphs are in general not perfect and not closed under induced subgraphs. Moreover, any given graph may be realized as an induced subgraph of a dually chordal graph by adding a new vertex adjacent to all other vertices. The hereditary dually chordal graphs, i.e. graphs whose induced subgraphs are dually chordal, are exactly the strongly chordal graphs [5].

Maximum neighbourhood orderings are algorithmically useful, especially for domination-like problems and distance problems on dually chordal graphs. Many problems efficiently solvable for strongly chordal and doubly chordal graphs remain polynomial-time solvable for dually chordal graphs [2, 13, 15, 16].

Our results on dually chordal graphs not only generalize, but also improve the corresponding results on strongly chordal and doubly chordal graphs, since a maximum neighbourhood ordering (if it exists) can be constructed in linear time.

In [5] we presented a systematic treatment of dually chordal graphs and of their relationships with hypertrees and chordal graphs. The present paper continues [5]: Here we develop efficient algorithms on dually chordal graphs using the structure theory of [5].

The paper is organized as follows. Section 2 recalls some important notions and properties. Section 3 studies duality results for the r-domination and r-packing problem. Section 4 presents linear time algorithms for finding a maximum neighbourhood ordering of a dually chordal graph and a doubly perfect elimination ordering of a doubly chordal graph. Section 5 deals with the r-domination problem on dually chordal graphs. Section 6 deals with the p-center and q-dispersion problems. Section 7 gives a linear-time algorithm for finding a minimum r-dominating clique and a maximum strict r-packing set of a dually chordal graph. Section 8 gives an algorithm for solving the connected r-domination and Steiner tree problem in linear time on doubly chordal graphs and in quadratic time on dually chordal graphs. Section 9 collects some $\mathbb{NP}$-complete problems on dually chordal graphs – among them the independent domination problem, the dominating cycle problem and the problem of determining the treewidth of a dually chordal graph.

2. Preliminaries

In this section we recall the definitions and some results on dually chordal graphs, which we use in the sequel.

Throughout this paper, let $G=(V,E)$ be a finite simple (i.e. without loops and multiple edges) undirected graph which is always assumed to be connected. Denote by $N(v)$ the *open neighbourhood* $N(v)=\{u: uv\in E\}$ and by $N[v]=N(v)\cup\{v\}$ the *closed neighbourhood*. For $Y\subseteq V$ let $G(Y)$ be the *subgraph induced by* Y. For a graph G with the ordered vertex set $V=(v_1,\ldots,v_n)$ let $G_i=G(\{v_i,v_{i+1},\ldots,v_n\})$ and $N_i[v]$ ($N_i(v)$) be the *closed* (*open*) *neighbourhood* of v in G_i, $i\in\{1,\ldots,n\}$. Let $G-v=G(V\setminus\{v\})$.

A vertex v is *simplicial* if $N[v]$ is a clique. The ordering $(v_1,\ldots,v_n)$ of V is a *perfect elimination ordering* if for all $i\in\{1,\ldots,n\}$ the vertex v_i is simplicial in G_i. The graph G is *chordal* if G has a perfect elimination ordering.

The ordering $(v_1,\ldots,v_n)$ is a *strong elimination ordering* if for all $i\in\{1,\ldots,n\}$ $N_i[v_j]\subseteq N_i[v_k]$ when $v_j,v_k\in N_i[v_i]$ and $j<k$. The graph G is *strongly chordal* if G has a strong elimination ordering [17].

A vertex $u\in N[v]$ is a *maximum neighbour* of v if for all $w\in N[v]$ the inclusion $N[w]\subseteq N[u]$ holds (note that $u=v$ is not excluded). A simplicial vertex which has a maximum neighbour is called *doubly simplicial*. The ordering $(v_1,\ldots,v_n)$ is a *maximum neighbourhood ordering* if for all $i\in\{1,\ldots,n\}$ there is a maximum neighbour $u_i\in N_i[v_i]$; i.e.,

$$\text{for all } w\in N_i[v_i],\ N_i[w]\subseteq N_i[u_i].$$

Furthermore, if each vertex v_i is doubly simplicial in G_i then such an ordering is called *doubly perfect*. The graph G is *dually chordal* [5] (*doubly chordal* [32]) if G has a maximum neighbourhood ordering (resp., a doubly perfect ordering).

There is a close connection between chordal and dually chordal graphs which can be expressed in terms of hypergraphs (for hypergraph notions we follow [3]).

Let $\mathcal{N}(G)=\{N[v]: v\in V\}$ be the (*closed*) *neighbourhood hypergraph* of G and let $\mathcal{C}(G)=\{C: C \text{ is a maximal clique of } G\}$ be the *clique hypergraph* of G. The *distance* $d_G(u,v)$ in G ($d(u,v)$ for short if G is understood) between vertices $u,v\in V$ is the length (i.e. number of edges) of a shortest path connecting u and v. For $v\in D$ and $D\subseteq V$ let $d(v,D)=\min\{d(v,u): u\in D\}$. The *disk* centered at vertex v with radius k is the set of all vertices having distance at most k to v:

$$N^k[v]=\{u\in V: d(v,u)\leqslant k\}.$$

By $\mathcal{D}(G)=\{N^k[v]: v\in V,\ k \text{ a non-negative integer}\}$ we denote the *disk hypergraph* of G. Let also $N_i^k[v]=N^k[v]\cap\{v_i,\ldots,v_n\}$ for $i\in\{1,\ldots,n\}$.

Now let $\mathcal{E}$ be a hypergraph with underlying vertex set V, i.e. $\mathcal{E}$ is a set of subsets of V. The *dual hypergraph* $\mathcal{E}^*$ has $\mathcal{E}$ as its vertex set and for each $v\in V$ a hyperedge $\{e\in\mathcal{E}: v\in e\}$. The *underlying graph* (or *2-section graph*) $I(\mathcal{E})$ of the hypergraph $\mathcal{E}$ has vertex set V and two distinct vertices are adjacent if they are contained in a

common edge of $\mathcal{E}$. The *line graph* $L(\mathcal{E})=(\mathcal{E},E)$ of $\mathcal{E}$ is the intersection graph of $\mathcal{E}$, i.e. $ee'\in E$ iff $e\cap e'\neq\emptyset$. A *partial hypergraph* of hypergraph $\mathcal{E}$ has V as the underlying vertex set and some edges of $\mathcal{E}$.

A hypergraph $\mathcal{E}$ is a *hypertree* (called *arboreal hypergraph* in [3]) if there is a tree T with vertex set V of $\mathcal{E}$ such that every edge $e\in\mathcal{E}$ induces a subtree in T. Equivalently, $\mathcal{E}$ is a hypertree iff the line graph $L(\mathcal{E})$ is chordal and $\mathcal{E}$ has the *Helly property*, i.e. any pairwise intersecting subfamily of edges of $\mathcal{E}$ has a common vertex; see [3]. A hypergraph $\mathcal{E}$ is a *dual hypertree* (*α-acyclic hypergraph*) if there is a tree T with vertex set $\mathcal{E}$ such that for all vertices $v\in V$ $T_v=\{e\in\mathcal{E}: v\in e\}$ induces a subtree of T, i.e. $\mathcal{E}^*$ is a hypertree.

Theorem 2.1 (Dragan et al. [16] and Brandstädt et al. [5]). *Let $G=(V,E)$ be a graph. Then the following conditions are equivalent:*

(i) *G is a dually chordal graph;*
(ii) *$\mathcal{N}(G)$ is a hypertree;*
(iii) *$\mathcal{D}(G)$ is a hypertree;*
(iv) *$\mathcal{C}(G)$ is a hypertree;*
(v) *G is the underlying graph of a hypertree.*

It is well-known [7] that G is chordal iff $\mathcal{C}(G)$ is a dual hypertree, i.e. G is the underlying graph of some dual hypertree. Therefore, the equivalence of parts (i) and (iv) of Theorem 2.1 justifies the name "dually chordal graphs" for graphs with maximum neighbourhood ordering.

Theorem 2.2 (Dragan et al. [16], Moscarini [32] and Brandstädt et al. [5]). *For a graph G the following conditions are equivalent:*

(i) *G is doubly chordal;*
(ii) *G is chordal and dually chordal;*
(iii) *both hypergraphs $\mathcal{C}(G)$ and $(\mathcal{C}(G))^*$ are hypertrees.*

A similar result for strongly chordal graphs and totally balanced hypergraphs was established in [17, 30].

Now we consider the recognition of the graph classes characterized in Theorems 2.1 and 2.2. Since for both graph classes the neighbourhood hypergraph is a hypertree, i.e. the dual hypergraph of an α-acyclic hypergraph the recognition of each class of graphs can be done in time proportional to the size of the corresponding hypergraph. This is a consequence of the linear-time algorithm for testing α-acyclicity of hypergraphs [39]. It is easy to see that the hypergraph $\mathcal{N}(G)$ used in Theorem 2.1 has size proportional to the number of edges of the corresponding graph. In order to recognize doubly chordal graphs first, we apply the maximum cardinality search algorithm of Tarjan and Yannakakis [39] for testing chordality of the graph and then the α-acyclicity testing algorithm of the same paper for the neighbourhood hypergraph.

Corollary 2.3. *It can be tested in linear time whether a graph $G=(V,E)$ is*
1. *dually chordal,*
2. *doubly chordal.*

Since G is always assumed to be connected linear time means $O(|E|)$.

3. r-domination and r-packing: Duality results

In this section we formulate general domination and packing problems and establish some duality results between them.

Let $G=(V,E)$ be a graph and $r: V \to \mathbb{N} \cup \{0\}$ be a non-negative integer-valued function defined on V. A subset $D \subseteq V$ is an *r-dominating set* if for any vertex $v \in V$ there is a vertex $u \in D$ with $d(u,v) \leqslant r(v)$. An *r-packing set* is a subset $P \subseteq V$ such that $d(u,v) > r(u) + r(v)$ for all $u,v \in P$. The *r-domination problem* is to find an r-dominating set with minimum size $\gamma_r(G)$ and the *r-packing problem* is to find an r-packing set with maximum size $\pi_r(G)$. Then $\gamma_r(G)$ and $\pi_r(G)$ are called the *r-domination* and *r-packing numbers* of G.

For a graph G and a function $r: V \to \mathbb{N} \cup \{0\}$ define the partial hypergraph $\mathcal{D}(G,r)$ of the disk hypergraph $\mathcal{D}(G)$ as follows:

$$\mathcal{D}(G,r) = \{N^{r(v)}[v] : v \in V\}.$$

The r-domination and r-packing problems on G may be formulated as the transversal and matching problems on the hypergraph $\mathcal{D}(G,r)$. Recall that a *transversal* of a hypergraph $\mathcal{H}$ is a subset of vertices which meets all edges of $\mathcal{H}$. A *matching* of $\mathcal{H}$ is a subset of pairwise disjoint edges of $\mathcal{H}$. For a hypergraph $\mathcal{H}$, the *transversal problem* is to find a transversal with minimum size $\tau(\mathcal{H})$ and the *matching problem* is to find a matching with maximum size $\nu(\mathcal{H})$. From the definitions we obtain

Lemma 3.1. *Let $G=(V,E)$ be a graph.*
- *D is an r-dominating set of G iff D is a transversal of $\mathcal{D}(G,r)$.*
- *P is an r-packing of G iff P is a matching of $\mathcal{D}(G,r)$.*

Thus, $\tau(\mathcal{D}(G,r)) = \gamma_r(G)$ and $\nu(\mathcal{D}(G,r)) = \pi_r(G)$ hold for any graph G and any function $r: V \to \mathbb{N} \cup \{0\}$.

Recently many variants of the r-domination problem have been extensively studied. The most well-known case is when $r(v) \equiv 1$, i.e. the *1-domination* (or simply *domination*) *problem*. Another particular case is when $r(v) \equiv k$, where k is a positive integer. Then we obtain the *k-domination problem*, or the problem of covering all vertices by a minimum number of disks of radius k. In terms of hypergraphs, the 1-domination (k-domination) problem is nothing else than the transversal problem in the neighbourhood hypergraph $\mathcal{N}(G)$ (k-neighbourhood hypergraph).

The parameters $\gamma_r(G)$ and $\pi_r(G)$ are related by a min-max duality inequality: $\gamma_r(G) \geqslant \pi_r(G)$ for any graph G and any function $r: V \to \mathbb{N} \cup \{0\}$. As was shown in [11] for $r(v) \equiv k$, the equality $\gamma_k(G) = \pi_k(G)$ holds on strongly chordal graphs.

Each partial hypergraph of a hypertree is a hypertree again. Thus, by Theorem 2.1, Lemma 3.1 and the well-known equality $\tau(\mathscr{H}) = \nu(\mathscr{H})$ for hypertrees [3] we are able to derive a more general duality result on dually chordal graphs.

Theorem 3.2 (Dragan et al. [16]). *For any function $r: V \to \mathbb{N} \cup \{0\}$ defined on the vertices of a dually chordal graph G, $\gamma_r(G) = \pi_r(G)$.*

4. Computing maximum neighbourhood orderings

Now we describe a linear-time algorithm for determining maximum neighbourhood orderings of dually chordal graphs. Subsequently special properties of a maximum neighbourhood ordering determined by this algorithm turn out to be of importance. Ref. [13] contains such an algorithm. The algorithm is based on the *maximum cardinality search (MCS) algorithm* of [39] for finding a perfect elimination ordering of chordal graphs. According to MCS the vertices of a graph are numbered from n to 1 in decreasing order. As the next vertex to number, select a vertex adjacent with the largest number of previously numbered vertices, breaking ties arbitrarily [39]. We will prove that the following modification of MCS also gives a perfect elimination ordering of a chordal graph. We call this procedure the *neighbour maximum cardinality search (NMCS)*.

Procedure 4.1. NMCS.
Input: *A chordal graph $G = (V, E)$.*
(0) *initially all $v \in V$ are unnumbered;*
(1) *choose an arbitrary vertex $v \in V$, number v with n, i.e. $v_n = v$;*
repeat
(2) *select a numbered vertex u; select in $N[u]$ an unnumbered vertex v which contains in $N[v]$ a maximum number of numbered vertices;*
(3) *number the vertex v with maximal possible number between 1 and $n-1$ which is still free;*
until *all vertices are numbered.*

Lemma 4.2. *Any ordering of a chordal graph G generated by NMCS is a perfect elimination ordering of G.*

Proof. Let $(v_1, \ldots, v_i, v_{i+1}, \ldots, v_n)$ be an ordering of G generated by NMCS. Assume by induction that the graph G_{i+1} is an induced-path closed subgraph of G. (Recall, that the subgraph H of G is called *induced-path closed* if for any two vertices of H any

induced path connecting them is contained in H.) If G_i is not an induced-path closed subgraph of G then there exists an induced path P outside of G_i whose end-vertices lie in G_i. Necessarily, P connects v_i with some vertex v_j, $j>i$, otherwise we obtain a contradiction with the induction assumption. Let x be a neighbour of v_j which belongs to P. Since v_i is selected by NMCS there exists a vertex $v_k, k>i$, adjacent to v_i such that v_k and v_i have the property:

(*) among all neighbours of v_k outside G_{i+1} the vertex v_i contains the maximum number of neighbours in G_{i+1}.

Pick an arbitrary neighbour v of v_i in G_{i-1}. Since G_{i+1} is an induced-path closed subgraph of G by adding the edge (v, v_i) to the path P we obtain a non-induced path. Since no chordal graph has induced cycles of length greater than three the vertex v is adjacent to all vertices of P. In particular, the vertices v and x are adjacent, i.e. $N_{i-1}(v_i) \subseteq N_{i-1}(x)$. Since x is adjacent to both v_k and v_j while the vertices v_i and v_j are non-adjacent we obtain a contradiction with the choice of v_i. Thus all $G_n, G_{n-1}, \ldots, G_1$ are induced-path closed subgraphs of G. Evidently v_i is simplicial in G_i as a neighbour vertex of G_{i-1}. □

Now we are able to present an algorithm which for dually chordal graphs gives a maximum neighbourhood ordering.

Algorithm 4.3 (*MNO*). *Find a maximum neighbourhood ordering of G.*
Input: *A dually chordal graph* $G=(V,E)$.
Output: *A maximum neighbourhood ordering of G.*
(0) *initially all* $v \in V$ *are unnumbered and unmarked;*
(1) *choose an arbitrary vertex* $v \in V$, *number* v *with* n *i.e.* $v_n = v$ *and let* $mn(v_n) := v$;
repeat
(2) *among all unmarked vertices select a numbered vertex* u *such that* $N[u]$ *contains a maximum number of numbered vertices;*
(3) *number all unnumbered vertices from* $N[u]$ *consecutively with maximal possible numbers between* 1 *and* $n-1$ *which are still free;*
for all of them let $mn(x) := u$;
(4) *mark* u;
until *all vertices are numbered*

The meaning of $mn(x)$ is that of a maximum neighbour of x. Note that the algorithm also yields a maximum neighbour for each vertex, and all vertices of $N[v_n]$ occur consecutively in the ordering on the left of v_n and have v_n as their maximum neighbour. Furthermore for all v_i with $i \leqslant n-1$ $mn(v_i) \neq v_i$ holds.

Subsequently, all maximum neighbourhood orderings are assumed to be generated by the MNO algorithm.

Before proving the correctness of this algorithm we present one modification which allows us to determine a doubly perfect elimination ordering of a doubly chordal graph.

In order to obtain such an ordering we replace step (3) of the MNO algorithm by the slightly modified steps (3a) and (3b) of the procedure NMCS:

repeat

(3a) select in $N[u]$ an unnumbered vertex v which contains in $N[v]$ a maximum number of numbered vertices;

(3b) number the vertex v with the maximal possible number between 1 and $n-1$ which is still free;
put $mn(v) := u$.

until all vertices of $N[u]$ are numbered

We call this algorithm the *doubly maximum neighbourhood ordering (DMNO) algorithm.*

Recall that the *square* G^2 of a graph $G=(V,E)$ has the same vertex set V and two distinct vertices $u,v \in V$ are adjacent in G^2 iff $d(u,v) \leqslant 2$ in G. Denote by $G^2(X)$ the subgraph of G^2 induced by the subset $X \subseteq V$. It is easy to see that for any graph G the graphs G^2 and $L(I(G))$ are isomorphic. Therefore, if G is dually chordal then by Theorem 2.1 the graph G^2 is chordal.

Lemma 4.4. *Let G be a dually chordal graph. Then any numbering of the vertices of G generated by the MNO algorithm can also be generated by a MCS of G^2 and thus this numbering is a maximum neighbourhood ordering of G.*

Proof. Let $(v_1,\ldots,v_n)$ be an ordering obtained by MNO. Since all vertices of $N[v_n]$ occur consecutively at the end of the ordering and form a clique in G^2 the same ordering of $N[v_n]$ can be generated by MCS of G^2. Thus, the assertion is fulfilled for this part of the ordering. Now assume that the end vertices $v_{i+1},\ldots,v_n$ of the ordering can be generated by MCS of G^2 as well and each $v_k, k \geqslant i+1$, has a maximum neighbour in the subgraph G_k. So, we have $N[v_n] \subseteq \{v_{i+1},\ldots,v_n\}$. Let u be the next vertex chosen by MCS of G^2. Suppose that u cannot be replaced by v_i in MCS of G^2. This means that in G^2, the vertex u is adjacent to more numbered vertices than v_i (we call *numbered* only the vertices $v_{i+1},\ldots,v_n$). Necessarily, $v_i \notin N[v_n]$.

Since G^2 is chordal u is a simplicial vertex of $G^2(\{u,v_{i+1},\ldots,v_n\})$. Therefore in G $d(x,y) \leqslant 2$ for any two numbered vertices x,y with $d(u,x) \leqslant 2$ and $d(u,y) \leqslant 2$. Note that u is not adjacent to v_n and, hence, there is at least one vertex whose distance from u is greater than 1. By the Helly property (see Theorem 2.1 (*ii*)) all closed neighbourhoods of numbered vertices of the disk $N^2[u]$ have a common vertex w adjacent to u. We claim that w is numbered. In fact, since u is chosen by MCS

$$|N^2_{i+1}[w]| = |N^2_{i+1}[u]|$$

holds. Since $N_{i+1}[w] = N^2_{i+1}[u]$ also $N_{i+1}[w] = N^2_{i+1}[w]$ holds. This implies that w is adjacent to all numbered vertices, i.e. w and v_n are adjacent or coincide. From $N[v_n] \subseteq \{v_{i+1},\ldots,v_n\}$ we get that w is already numbered.

Now return to vertex v_i chosen by MNO. By this algorithm we have that v_i is adjacent to a numbered vertex v such that $N[v]$ contains the maximum number of numbered vertices. In particular,

$$|N_{i+1}[v]| \geq |N_{i-1}[w]| = |N^2_{i+1}[u]|.$$

Since $N_{i-1}[v] \subseteq N^2_{i+1}[v_i]$ we get a contradiction with the assumption that $|N^2_{i+1}[u]| > |N^2_{i+1}[v_i]|$.

Thus, we obtain that the vertex v_i can be chosen by MCS too, i.e. we can assume w.l.o.g. that $v_i = u$ and

$$N_{i+1}[v] = N_{i-1}[w] = N^2_{i+1}[v_i].$$

Therefore the vertex v_i has a maximum neighbour v. □

Theorem 4.5 (Dragan [13]). (1) *The algorithm MNO finds a maximum neighbourhood ordering of a dually chordal graph G in linear time.*

(2) *The algorithm DMNO finds a doubly perfect elimination ordering of a doubly chordal graph G in linear time.*

Proof. The time bounds of each of these algorithms are linear if a suitable implementation is chosen; for details see [39]. The correctness follows from Lemmas 4.4 and 4.2. □

With a given maximum neighbourhood ordering $(v_1, \ldots, v_n)$ of a dually chordal graph G we can associate a rooted tree T (which we call *MNO-tree*): v_n is the root of T and two vertices v_i and v_j, $i < j$, are adjacent in T if $v_j = mn(v_i)$. Recall that for maximum neighbourhood orderings produced by the MNO algorithm, $v_i \neq mn(v_i)$ if $i \neq n$. Notice also that T is a spanning tree of G. All edges of G which do not belong to T will be called *intermediate edges*.

Maximum neighbourhood orderings of graphs immediately lead to an optimal algorithm for computing the distance matrix for all graphs having such orderings. Let $(v_1, \ldots, v_n)$ be a maximum neighbourhood ordering of a graph G produced by the MNO algorithm. The maximum neighbour $mn(v_i)$ of the vertex v_i in G_i has an important metric property:

> for any vertex v_j, $j > i$, which is non-adjacent to v there exists a shortest path of length $d_G(v_i, v_j)$ in G_i between v_i and v_j which passes through $mn(v_i)$.

To see this assume by way of contradiction that all shortest paths between v_i and v_j in G contain vertices not in G_i. Among these paths, choose a path P whose leftmost vertex u w.r.t. the maximum neighbourhood ordering $(v_1, \ldots, v_n)$ has rightmost position. Let v, w be the neighbours of u in P. Since P is a shortest path the distance of v, w is 2. Now the maximum neighbour $mn(u)$ which according to the MNO algorithm is distinct from u is also adjacent to v and w and on the right of u. Thus, replacing u by $mn(u)$ in P, we obtain a shortest path P' between v_i and v_j whose leftmost vertex is on the right of u – a contradiction.

In particular, we obtain that any graph G_i is a distance-preserving (isometric) subgraph of G, i.e. the distances in G_i are the same as in G.

Let $G=(V,E)$ be a dually chordal graph and let $D(G)=(d(v_i,v_j))_{i,j\in\{1,\ldots,n\}}$ denote the distance matrix of G. By $D_{i+1}(G)$ we denote the submatrix of $D(G)$ which contains the distances between the vertices $v_{i+1},\ldots,v_n$. The next submatrix $D_i(G)$ is obtained from $D_{i+1}(G)$ by adding the ith row and ith column according to the following rule:

for any $k>i$ define

$$d(v_i,v_k)=d(v_k,v_i)=\begin{cases}1 & \text{if } v_i \text{ and } v_k \text{ are adjacent,}\\ d(mn(v_i),v_k)+1 & \text{otherwise.}\end{cases}$$

Evidently, this procedure correctly finds the whole matrix $D(G)$ in optimal time $O(n^2)$. Moreover, the maximum neighbourhood ordering of G for any two query vertices u and v allows to find in time $O(c\cdot d(u,v))$ a shortest path between u and v (c is the necessary time to verify the adjacency of two vertices). Let $num(v)=i$ if $v=v_i$ in the maximum neighbourhood ordering of G.

Procedure 4.6 (sh-path(u,v))
if *u and v are adjacent* **then return** (u,v)
else
  if $num(u) < num(v)$ **then**
    return $(u,$*sh-path*$(mn(u),v))$
  else
    return $($*sh-path*$(u,mn(v)),v))$

A *maximum neighbour path* between any two vertices u,v of a dually chordal graph G is the shortest path between u and v resulting from the procedure sh-path(u,v).

Lemma 4.7. *If $P=(x_1,\ldots,x_k)$ with $u=x_1$ and $v=x_k$ is the maximum neighbour path between u and v then the sequence $x_1,\ldots,x_k$ is unimodal, i.e. $num(x_1)<\cdots<num(x_t)$ and $num(x_t)>\cdots>num(x_k)$ for some t with $1\leqslant t\leqslant k$. Moreover, either P is a path of the MNO-tree T or it consists of two paths $(x_1,\ldots,x_t)$ and $(x_{t+1},\ldots,x_k)$ of T joined by an intermediate edge (x_t,x_{t+1}) of G.*

Proof. We prove the lemma by induction on the distance between u and v. If $uv\in E$ then the assertion is obviously fulfilled. Let $uv\notin E$ and $d(u,v)=k+1$ and w.l.o.g. $num(u)<num(v)$. There is a shortest path joining u and v containing $mn(u)$. Now $d(mn(u),v)=k$ and the induction assumption can be applied which gives a unimodal path between $mn(u)$ and v. By adding the edge $(u,mn(u))$ to this path we obtain the assertion. □

Notice that the MNO-tree T contains all maximum neighbour paths between any vertex and all its predecessors in the rooted tree T.

Summarizing these results we obtain

Theorem 4.8. *Let G be a dually chordal graph. The whole distance matrix $D(G)$ of G can be computed in optimal time $O(n^2)$. It is possible to preprocess G in linear time such that for any two vertices u and v at G, the shortest path between u and v can be found in time $O(c \cdot d(u,v))$, where c is the complexity of testing the adjacency of vertices.*

By preprocessing in this case we understand the application of the MNO algorithm.

5. r-domination problems

The domination (1-domination) problem is $\mathbb{NP}$-complete in general. For special graph classes the situation is sometimes better. For a bibliography on domination we refer to [23]. Unlike for the 1-domination problem, for the more general r-domination problem there are only few cases where a polynomial-time algorithm is known, among them sun-free chordal graphs [9], 3-sun-3-anti-sun-free chordal graphs [12], sun-free bridged graphs and 3-sun-3-anti-sun-free bridged graphs [14] and dually chordal graphs [16].

In [16] we presented $O(n^2)$ time algorithms for r-domination and r-packing problems on dually chordal graphs, using the fact that the disk hypergraphs of dually chordal graphs are hypertrees. Below we present a linear $O(|E|)$ algorithm for the r-domination and r-packing problems on dually chordal graphs, which avoids the construction of the disk hypergraph. The algorithm uses the maximum neighbourhood ordering and is similar to the algorithm of Chang for r-domination on strongly chordal graphs [9]. The algorithm simultaneously finds an r-dominating set D and an r-packing set P such that $|D| = |P|$. This provides an algorithmic proof of duality results between these two problems on dually chordal graphs.

Let $G = (V, E)$ be a dually chordal graph, $r : V \to \mathbb{N} \cup \{0\}$ be a non-negative integer-valued function and $(v_1, \ldots, v_n)$ be the ordering of V generated by the MNO algorithm. The algorithm processes the vertices in the order from v_1 to v_n. In step i the algorithm decides whether the vertex v_i has to be put into the r-dominating set D (where a *step* of the algorithm is understood as one iteration of lines (4)–(10)). If v_i is included in D then a certain personal r-neighbour u_i of v_i i.e. a vertex u_i which is r-dominated only by v_i is included in the r-packing set P. Initially both sets D and P are empty. After processing, vertex v_i is deleted from the graph and an information whether or not v_i was included in D is given to its maximum neighbour $mn(v_i)$ and/or to its other neighbours.

For technical reasons, as in [9] we associate to each vertex v_i not only the domination radius $r(v_i)$ but also a non-negative integer $c(v_i)$. Initially, the values $r(v_i)$, $i \in \{1, \ldots, n\}$ are given by the function $r : V \to \mathbb{N} \cup \{0\}$ of the input and $c(v_i) = \infty$ for all i. Both $r(v_i)$ and $c(v_i)$ keep nonincreasing while the algorithm is executed. At each step $r(v_i)$ becomes the current radius within which the vertex v_i must be r-dominated

in the remaining graph. The meaning of $c(v_i)$ is that in the initial graph G there exists a vertex u from the current set D such that $d(v_i,u)\leqslant c(v_i)$. The radius $r(v_i)>0$ decreases in the case where v_i is the maximum neighbour $mn(v_j)$ of a vertex v_j, $j<i$, that is not properly r-dominated by a vertex of D within distance $r(v_j)$ in step j. In this case, $r(v_j)$ is set to be $r(v_j)-1$. Similarly, in a previous step, $r(v_j)$ is set to be $r(v_k)-1$, $k<j$. Continuing this argument, we find that there is a smallest vertex v_{i^*} that forces ..., k,j,i to decrease their $r(\cdot)$ values, although $r(v_{i^*})$ never changes. We use $fn(v_i)$ (*furthest neighbour* of v_i) to denote this initial vertex v_{i^*} from which $r(v_i)$ decreases. In particular, if $r(v_i)=0$ the furthest neighbour of v_i is v_i itself.

In step i the algorithm processes vertex v_i according to the following rules. When $r(v_i)=0$ then v_i must be in D because no other vertex r-dominates v_i. Moreover, we put $fn(v_i)$ in P and set $c(v_i)=0$. Otherwise, if $r(v_i)>0$ then we distinguish two cases. If $c(v_i)>r(v_i)$ and for any $v\in N_i(v_i)$ $c(v)+1>r(v_i)$ and $r(v)>0$ then we update $r(mn(v_i))$ by

$$r(mn(v_i))=\min\{r(mn(v_i)),r(v_i)-1\}.$$

Finally, if $c(v_i)\leqslant r(v_i)$ or $c(v)+1\leqslant r(v_i)$ for some $v\in N_i[v_i]$ or $r(v)=0$ for some $v\in N_i(v_i)$ then we do nothing. At each step we have to update $c(v)$ for all $v\in N_i(v_i)$:

$$c(v):=\min\{c(v),c(v_i)+1\}.$$

Algorithm 5.1 (*RDP*). *Find a minimum r-dominating set and a maximum r-packing set of a dually chordal graph G.*

Input: *A dually chordal graph $G=(V,E)$ with a maximum neighbourhood ordering $(v_1,\ldots,v_n)$ and with n non-negative integers $r(v_1),\ldots,r(v_n)$*

Output: *A minimum r-dominating set D and a maximum r-packing set P of G*

(1) $D:=\emptyset$; $P:=\emptyset$;
(2) **for all** $v\in V$ **do begin** $c(v):=\infty$; $fn(v):=v$ **end**
(3) **for** $i:=1$ **to** $n-1$ **do**
    begin
(4) **if** $r(v_i)=0$ **then**
        begin
(5) $D:=D\cup\{v_i\}$; $P:=P\cup\{fn(v_i)\}$;
(6) $c(v_i):=0$;
        end
     else
(7) **if** $c(v_i)>r(v_i)$ **and** (for all $v\in N_i(v_i)$ $c(v)+1>r(v_i)$) **and** $r(v)>0$
           then
(8) **if** $r(mn(v_i))\geqslant r(v_i)$ **then**
              begin
(9) $r(mn(v_i)):=r(v_i)-1$;
(10) $fn(mn(v_i)):=fn(v_i)$
              end

(11) **for all** $v \in N_i(v_i)$ **do** $c(v) := \min\{c(v), c(v_i) + 1\}$
end
(12) **if** $r(v_n) < c(v_n)$ **then begin** $D := D \cup \{v_n\}$; $P := P \cup \{fn(v_n)\}$ **end**

Theorem 5.2. *Algorithm RDP is correct and works in linear time.*

Proof. The time bound of the algorithm is obviously linear. Now to the correctness. In order to prove that D is a minimum r-dominating set of G we use the following reformulation of an r-domination problem in terms of $r(v_i)$ and $c(v_i)$:

Find a minimum size set $D \subseteq V$ such that for any vertex $v \in V$ either
(a) $d(u, v) \leqslant r(v)$ for some $u \in D$ (directly dominated) or
(b) $d(v, w) + c(w) \leqslant r(v)$ for some $w \in V$ (indirectly dominated).

Recall that initially $c(v) = \infty$ for all $v \in V$ and so this is the usual r-domination problem. Now we need some auxiliary results; see also [9] for the case of strongly chordal graphs. Throughout the correctness proof of the RDP algorithm we use that any graph G_i is a distance-preserving subgraph of G. By processing v_i the algorithm yields the updates r' of r and c' of c for G_{i-1}.

Lemma 5.3. *If $r(v_i) = 0$ then D is a minimum r-dominating set of G_i if and only iff $D = D' \cup \{v_i\}$ where D' is a minimum r'-dominating set of G_{i+1} with $c'(v) := 1$ for $v \in N_i(v_i)$ and $c'(v) := c(v)$ otherwise, and $r'(v) := r(v)$ for all v.*

Proof. If $r(v_i) = 0$ then v_i is not r-dominated by any other vertex of G_i. So v_i belongs to any r-dominating set of G_i i.e. $v_i \in D$. Let $D' = D \setminus \{v_i\}$. If D' is not r'-dominating in G_{i+1} then there is a vertex v_k, $k \geqslant i+1$ such that $d(v_k, D') > r(v_k)$ and $d(v_k, v) + c(v) > r(v_k)$ for any vertex v. Thus, either v_k is r-dominated by v_i or in G_i $d(v_k, x) + c(x) \leqslant r(v_k)$ holds for some vertex x. In the first case, let $w \in N_i(v_i)$ be a vertex on a shortest path between v_i and v_k. Then $d(v_k, w) + c'(w) = d(v_k, w) + d(w, v_i) = d(v_k, v_i) \leqslant r(v_k)$ – a contradiction. Now to the second case. Either $x \in N_i(v_i)$ or $x = v_i$. If $x \in N_i(v_i)$ then $c'(x) = 1$ and we obtain a similar inequality $d(v_k, x) + c'(x) \leqslant r(v_k)$ (note that $c(x) \neq 0$) in the graph G_{i+1} too. Otherwise, if $d(v_k, v_i) + c(v_i) \leqslant r(v_k)$ then moreover $d(v_k, w) + c'(w) \leqslant r(v_k)$ for any vertex $w \in N_i(v_i)$ on a shortest path between v_i and v_k. Thus, D' is an r'-dominating set of the graph G_{i+1}.

Conversely, for all v_k, $k \geqslant i+1$, $r'(v_k) = r(v_k)$. Therefore, directly dominated vertices in G_{i+1} remain directly dominated in G_i. Let v_k be indirectly dominated by v in G_{i+1}. If $v \notin N_i(v_i)$ then $c'(v) = c(v)$, and thus v_k remains indirectly dominated in G_i as well. If $v \in N_i(v_i)$ then $d(v_k, v) + c'(v) \leqslant r(v_k)$ and $d(v_k, v_i) \leqslant r(v_k)$, i.e. v_k is r-dominated in G_i by v_i. Therefore, if D' is r'-dominating in G_{i+1} then $D' \cup \{v_i\}$ is an r-dominating set of G_i. □

Lemma 5.4. *Assume that $c(v_i) > r(v_i)$, and for all $v \in N_i[v_i]$, $c(v) + 1 > r(v_i)$ and $r(v) > 0$. A subset $D \subseteq (\{v_{i+1}, \ldots, v_n\} \setminus N_i(v_i)) \cup \{mn(v_i)\}$ is a minimum r-dominating*

set of G_i if and only if D is a minimum r'-dominating set of G_{i-1} with $c'(v) := c(v)$ for $v \notin N_i(v_i)$ and $c'(v) := \min\{c(v), c(v_i) + 1\}$ for $v \in N_i(v_i)$ and $r'(v) := r(v)$ for all $v \neq mn(v_i)$ and $r'(mn(v_i)) := \min\{r(v_i) - 1, r(mn(v_i))\}$.

Proof. From our conditions we immediately get that if D' is a minimum r'-dominating set of G_{i+1} then replacing all vertices of $D' \cap N_i(v_i))$ by $mn(v_i)$ we obtain an r'-dominating set D with $|D| \leqslant |D'|$.

Let D be a minimum r-dominating set of G_i such that $D \subseteq (\{v_{i-1}, \ldots, v_n\} \setminus N_i(v_i)) \cup \{mn(v_i)\}$. Evidently all vertices of G_{i-1} which are r-dominated in G_i by vertices of D remain r'-dominated by the same vertices in G_{i+1} too. Further, if $d(v_k, v_i) + c(v_i) \leqslant r(v_k)$ in G_i then $d(v_k, v) + c'(v) \leqslant r'(v_k)$ in G_{i+1} where $v \in N_i(v_i)$ lies on a shortest path between v_i and v_k. So, D is an r'-dominating set in G_{i+1}. The converse is similar. □

Lemma 5.5. *Suppose that $r(v_i) \geqslant 1$. Assume $c(v_i) \leqslant r(v_i)$ or $c(v) + 1 \leqslant r(v_i)$ for some $v \in N_i(v_i)$ or $r(v) = 0$ for some $v \in N_i(v_i)$. A subset $D \subseteq \{v_{i+1}, \ldots, v_n\}$ is a minimum r-dominating set of G_i if and only if D is a minimum r'-dominating set of G_{i+1} with $r'(v) := r(v)$ for all $v \neq v_i$ and $c'(v) := c(v)$ for all $v \notin N_i(v_i)$ and $c'(v) := \min\{c(v), c(v_i) + 1\}$ when $v \in N_i(v_i)$.*

Proof. Evidently, each r-dominating set $D \subseteq \{v_{i+1}, \ldots, v_n\}$ of G_i is r'-dominating in G_{i+1} too. Conversely, suppose that D is an r'-dominating set of G_{i+1}. If a vertex $v \in V$ is r'-dominated by some vertex of D in G_{i+1} then v is directly r-dominated by the same vertex in G_i too. Therefore it is enough to consider only the case when $d(v, w) + c'(w) \leqslant r'(v)$ in G_{i+1} for some vertex $w \in N_i(v_i)$. If $\min\{c(w), c(v_i) + 1\} = c(w)$ then we are done. Otherwise, since $d(v, v_i) \leqslant d(v, w) + 1$ we obtain that $d(v, v_i) + c(v_i) \leqslant d(v, w) + 1 + c(v_i) = d(v, w) + c'(w) \leqslant r(v)$. Hence, D is an r-dominating set of G_i. □

Now we return to the proof of Theorem 5.2. Observe that if $r(v_n) < c(v_n)$ then v_n is not dominated by any vertex of D and thus v_n must be included in D.

Let D be the set determined by the algorithm before step i, and let D' be an arbitrary minimum r-dominating set of the graph G_i with modified functions $r(v)$ and $c(v)$. By Lemmas 5.3–5.5 we obtain that $D \cup D'$ is a minimum r-dominating set of the initial graph G with initial domination radii. Thus, in step n we obtain the required minimum r-dominating set D of G.

Next we concentrate on the proof that the algorithm correctly finds a maximum r-packing set P of G. Hereby we say that a path P is *increasing* if in the given maximum neighbourhood ordering the indices of the vertices of P are increasing i.e. if $P = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ then $i_1 < i_2 < \cdots < i_k$.

Lemma 5.6. $|P| = |D|$.

Proof. From the algorithm it is obvious that $|D| \geqslant |P|$. If $|P| < |D|$ then there are two vertices $u, v \in D$ and a vertex $x \in P$ such that $fn(u) = x = fn(v)$. By the algorithm $fn(u)$ and $fn(v)$ arrive vertices u and v along increasing maximum neighbour paths connecting x, u and x, v resp. Every next vertex of these paths is the maximum neighbour of the previous vertex. Let y be the vertex belonging to both paths which is furthest from x. Note that $y \neq u, v$ since if, for example, $y = u$ then in step $num(u)$ line (5) but not line (10) of the algorithm is applied, and so $fn(v) = x$ is impossible. Now denote by u' and v' the next neighbours of y in these paths. Since both paths are increasing maximum neighbour paths we obtain that $v' = mn(y)$ and $u' = mn(y)$. Hence, by the MNO algorithm $v' = u'$ – a contradiction. □

Since in the proof of the next lemmata, we will consider all steps of the algorithm together for convenience we indicate by $r_i(v)$ ($c_i(v)$, resp.) the value of $r(v)$ ($c(v)$) obtained immediately after step i. Hence for the initial values, $r_0(v) = r(v)$ and $c_0(v) = \infty$ for all $v \in V$ holds.

Consider two vertices $x \in D$ and $u \in P$ such that $u = fn(x)$ and let Q be the maximum neighbour path from u to the root of the MNO-tree T. Evidently this path is increasing. Necessarily, Q contains the vertex x.

Lemma 5.7. *If v is a vertex of Q and $i = num(v)$ then $r_i(mn(v)) = r(u) - d(u, mn(v))$ if $i < num(x)$ and $c_i(mn(v)) \leqslant d(x, mn(v))$ if $i > num(x)$.*

Proof. Let $j = num(w)$, where w is a vertex such that $v = mn(w)$. Obviously, $j \leqslant i - 1$. We proceed by induction on $d(v, u)$ if $i < num(x)$ and by induction on $d(v, x)$ otherwise. Indeed, in the assumption that $r_j(v) = r(u) - d(u, v)$, using the lines (8) and (9) of the algorithm we conclude that $r_i(mn(v)) = r_{i-1}(v) - 1 = r_j(v) - 1$ i.e. $r_i(mn(v)) = r(u) - d(u, v) - 1 = r(u) - d(u, mn(v))$. Otherwise, if $i > num(x)$ then since $c_{num(x)}(x) = 0$, by the induction assumption and line (11) we conclude that $c_i(mn(v)) \leqslant c_{i-1}(v) + 1 \leqslant c_j(v) + 1 \leqslant d(x, v) + 1 = d(x, mn(v))$. □

Lemma 5.8. *P is an r-packing set of G.*

Proof. Assume to the contrary that $d(u, v) \leqslant r(u) + r(v)$ for some vertices $u, v \in P$. By the algorithm, there exist $x, y \in D$ such that $u = fn(x)$ and $v = fn(y)$. Let P' and P'' be maximum neighbour paths between u, x and v, y. Since x and y are predecessors of u and v in the MNO-tree T, both P' and P'' belong to the tree T and are increasing. Note that P' and P'' are two paths along which the values $fn(x)$ and $fn(y)$ are transmitted from u, v to the vertices x and y, respectively. Namely, if $P' = (u = x_0, x_1, \ldots, x_n = x)$ and $P'' = (v = y_0, y_1, \ldots, y_m = y)$ then by line (10) of the algorithm the following chains of equalities are fulfilled:

$$(*)\quad \begin{aligned} fn(x_n) &= \cdots = fn(x_1) = fn(x_0) = u, \\ fn(y_m) &= \cdots = fn(y_1) = fn(y_0) = v. \end{aligned}$$

We assert that P' and P'' are disjoint. Suppose the contrary and let z be a vertex belonging to both paths. Since $fn(z)$ participates in both equalities $(*)$, we deduce that $u=v$, a contradiction.

By Lemma 4.7 the maximum neighbour path between vertices u and v consists of two subpaths P_u and P_v of T and an intermediate edge (u',v'). For the paths P',P_u and P'',P_v we conclude that they should be comparable with respect to inclusion. Because of $d(u,v)\leqslant r(u)+r(v)$ and $d(u,x)=r(u)$ and $d(v,y)=r(v)$, at least one of the inequalities $|P'|>|P_u|$ or $|P''|>|P_v|$ holds. In particular, at least one of the incidences $u'\in P'$ or $v'\in P''$ is satisfied.

First consider the case when $u'\in P'$ and $v'\in P''$. Note that $u'\neq x$ or $v'\neq y$, otherwise $d(v,u)=d(v,y)+1+d(x,u)=r(v)+r(u)-1$. Among adjacent vertices, one from P' and the other from P'', we choose vertices $u^*\in P'$ and $v^*\in P''$ with $u^*\neq x$ and $v^*\neq y$ whose sum $d(u^*,x)+d(v^*,y)$ is minimal. Let $j=num(v^*)>num(u^*)=i$. First suppose that $u^*=x$. Then $v^*\neq y$ and therefore $r_{j-1}(v^*)\neq 0$. By Lemma 5.7 before step i we have $r_{i-1}(u^*)=0$ – thus after this step we obtain $c_i(v^*)=1$. Hence $c_{j-1}(v^*)=1$. But then in step j the first condition in line (7) of the algorithm does not hold, because $c_{j-1}(v^*)=1\leqslant r_{j-1}(v^*)\neq 0$. Therefore the vertex $mn(v^*)\in P''$ does not receive the value $fn(v^*)=v$, contrary to $(*)$. So, let us suppose $u^*\neq x$. Since $num(v^*)>num(u^*)$, v^* is adjacent to the maximum neighbour $mn(u^*)$ of u^*. From the choice of the edge u^*v^* we conclude that $mn(u^*)=x$ and $v^*=y$. First we consider the case $|P''|=1$ i.e. $v=v^*=y$. Then $r_{i-1}(y)=r_0(y)=r(y)=0$ and $fn(y)=v=y$. Since $i<j$ in step i we already have a neighbour $y\in N_i(u^*)$ of u^* with $r_{i-1}(y)=0$. This means that by algorithm x cannot receive the value $fn(u^*)=u$ from u^*. So, let $|P''|\geqslant 2$ and v^+ be the neighbour of y in P''. If $k=num(v^+)<num(u^*)$ then $r_k(y)=0$ and again in step i we already have a neighbour $y\in N_i(u^*)$ of u^* with $r_{i-1}(y)=r_k(y)=0$. Hence $k>i$. Since $x=mn(u^*)$ and $d(u^*,v^+)\leqslant 2$, x and v^+ are adjacent. By the algorithm since $x\in D$ we have $r_i(x)=0$. Now in step k the neighbour $x\in N_k(v^+)$ violates the condition in line (7), contrary to $fn(y)=fn(v^+)=\cdots=v$. This contradiction settles the case when $u'\in P'$ and $v'\in P''$.

Now suppose that $u'\in P'$ and $v'\notin P''$ (the case when $u'\notin P'$ and $v'\in P''$ is similar). Then necessarily $P''\subset P_v$ i.e. the vertex y belongs to the path P_v. Let z be the neighbour of $v'\in P_v$ such that $v'=mn(z)$. Let $d(u',x)=l'$ and $d(v',y)=l''$. Since $d(u,v)\leqslant r(u)+r(v)$ and $d(u,v)=r(u)-l'+1+l''+r(v)$, the inequality $l''<l'$ is necessarily fulfilled. Moreover, since $v'\neq y$ and $2+r(v)\leqslant d(u,v)\leqslant r(u)+r(v)$, we obtain $r(u)\geqslant 2$ and $u'\neq x$. If $i=num(u')<num(z)$ $(<num(v'))$ then $v',z\in N_i[mn(u')]$. Since u' and z are adjacent to both $mn(u')$ and $v'=mn(z)$, by the MNO algorithm we conclude that $v'=mn(u')$. By the RDP algorithm and Lemma 5.7 in step $j=num(v')$ we have $c_{j-1}(v')\leqslant l''$ and $r_{j-1}(v')=l'-1$. Since $l''<l'$ the inequality $c_{j-1}(v')\leqslant r_{j-1}(v')$ holds. Comparing with the condition in line (7) we get $x=v'$. But then $d(u,v)=r(u)+d(x,y)+r(v)>r(u)+r(v)$, which is impossible.

So, assume that $i>num(z)$. If $i<j$ then in step i we have $c_{i-1}(v')\leqslant l''<l'=r_{i-1}(u')$. Therefore $c_{i-1}(v')+1\leqslant r_{i-1}(u')$ and we cannot transmit the value $fn(u')=u$ to the maximum neighbour of u' because the second condition in line (7) is violated. Otherwise,

if $i>j$ then according to Lemma 5.7 $c_{i-1}(u')\leqslant l''+1$. Then $c_{i-1}(u')\leqslant r_{i-1}(u')$ and again applying the second condition of line (7) we obtain a contradiction with (*).

This concludes the proof of the lemma. □

From Lemmas 5.6 and 5.8 and Theorem 3.2 we immediately obtain that the set P computed by the algorithm RDP is a maximum r-packing of G. This completes the proof of the theorem. □

6. p-center and q-dispersion problems

Let $G=(V,E)$ be a graph and $w: V\to\mathbb{R}^+\cup\{0\}$ be a non-negative weight function defined on V. We define the *radius* $r(S)$ of a set $S\subseteq V$ as $\max\{w(u)d(u,S): u\in V\}$.

For a given positive integer $p\leqslant|V|$ we define the *p-radius* of G as

$$r_p=r_p(G)=\min\{r(C): C\subseteq V,\ |C|\leqslant p\}.$$

The *p-center problem* is to find a set of size at most p which realizes the p-radius of G (such a set is called a *p-center* of G).

This is one of the main models in facility location theory; see [24, 38]. For general graphs, the problem of finding p-centers is $\mathbb{NP}$-hard [24]. Moreover, even the ε-approximation variant of this problem remains $\mathbb{NP}$-hard for $\varepsilon<2$ [34], while a polynomial 2-approximation algorithm is given in [34]. Remark also that since the domination problem is a particular instance of the p-center problem [24] the p-center problem is $\mathbb{NP}$-hard in all graph classes where the domination problem is $\mathbb{NP}$-complete.

Polynomial algorithms for the p-center problem are known only in trees [8, 24, 26] and almost-trees [22]. The best known algorithms have time complexity $O(|V|)$ for unweighted trees [19] and $O(|V|\log^2|V|)$ for weighted trees [31].

As already mentioned the domination problem on a graph G is a particular case of the p-center problem. Conversely, the p-center problem can be reduced to solving a logarithmic number of r-domination problems on G: Indeed, the p-radius $r_p=r_p(G)$ is an element of the weighted distance matrix $WD=(w(u)d(u,v))_{u,v\in V}$. To compute r_p we search this matrix for the minimum value which is feasible in the following sense. A value r is *feasible* if there exists a set $C\subseteq V$ of size at most p whose radius $r(C)$ is not greater than r. In order to decide whether a given r is feasible it suffices to solve the r-domination problem on G with $r(u)=\lfloor r/w(u)\rfloor$ if $w(u)\neq 0$ and $r(u)=\infty$ (a sufficiently large number) otherwise, and check if the obtained solution contains not more than p vertices. If so we decrease r by taking the median of the list of elements of WD which are smaller than r. Otherwise, we replace r by the median of elements of WD which are larger than r. The start value of r is the median element of WD. Thus the p-center problem can be solved by computing the weighted distance matrix and searching it by repeatedly using linear-time median finding [4] and solving the corresponding r-domination problem. Since for dually chordal graphs the weighted distance matrix is computed in $O(|V|^2)$ time and the r-domination problem is solved

in $O(|E|)$ time the proposed approach leads to an $O(|V|^2 \log |V|)$ algorithm for the p-center problem in these graphs.

Next we consider the q-dispersion problem [8] which is in some sense dual to the p-center problem. For a given subset $X \subseteq V$ find a q-vertex set $S \subseteq X$ such that its vertices are as far apart as possible, i.e. maximizing $\min\{d(u,v) : u,v \in S, u \neq v\}$. Then S is called the *q-dispersion set* of X and $\max\{\min\{d(u,v) : u,v \in S\} : |S| = q\} = d_q(X)$ is the *q-dispersion* of X. The duality between the p-center and q-dispersion problems in trees was established in different variants in [26, 8]. As we will show a similar result holds for dually chordal graphs too.

Theorem 6.1. *For each dually chordal graph G and each subset $X \subseteq V$ the following equality holds*:

$$\min\{\max\{d(v,C) : v \in X\} : C \subseteq V, |C| = p\}$$
$$= \lfloor (\max\{\min\{d(v,u) : v,u \in S, v \neq u\} : S \subseteq X, |S| = p+1\} + 1)/2 \rfloor.$$

Proof. Let k be an arbitrary nonnegative integer. There exists a $C \subseteq V$, $|C| = p$ such that $\max\{d(v,C) : v \in X\} \leqslant k$ if and only if

$$\min\{|D| : \max\{d(v,D) : v \in X\} \leqslant k, D \subset V\} \leqslant p$$

or, equivalently,

$$\min\{|D| : d(v,D) \leqslant k \text{ for all } v \in X, D \subset V\} \leqslant p.$$

This expression states that $\gamma_r(G) \leqslant p$ for the r-domination problem on graph G with the following radius function r:

$$r(v) = k \quad \text{for all } v \in X \text{ and } r(v) = \infty \text{ otherwise.}$$

By the duality result $\gamma_r(G) = \pi_r(G)$ between r-domination and r-packing (Theorem 3.2) the preceding inequality holds if and only if

$$\max\{|S| : d(u,v) > 2k \quad \text{for all } u,v \in S, S \subset X\} \leqslant p.$$

This expression states that there are at most p vertices such that $d(u,v) > 2k$ for any pair u,v of them. In other words, for every $(p+1)$-vertex set there are at least two vertices u,v such that $d(u,v) \leqslant 2k$. Hence, we obtain that

$$\max\{\min\{d(u,v) : u,v \in S, u \neq v\} : S \subseteq X, |S| = p+1\} \leqslant 2k$$

or, equivalently,

$$\lfloor 1/2(\max\{\min\{d(u,v) : u,v \in S, v \neq u\} : S \subseteq X, |S| = p+1\} + 1) \rfloor \leqslant k. \qquad \square$$

The solution of the q-dispersion problem on a dually chordal graph G may be obtained in the following way. First, solve the p-center problem on G with $p = q - 1$

and $w(u)=1$ if $u\in X$ and $w(u)=0$ if $u\in V\setminus X$. Let r_p be the p-radius of G. By Theorem 6.1 either $d_q(X)=2r_p$ or $d_q(X)=2r_p-1$. In order to compute the exact value of $d_q(X)$ we must solve at most two independent set problems in the subgraph induced by X of the k-th power G^k of G, first for $k=2r_p-1$ and later for $k=2r_p-2$. Recall that the graph G^k has the same vertex set V and two distinct vertices $u,v\in V$ are adjacent in G^k iff $d(u,v)\leqslant k$ in G. If for $k=2r_p-1$ the obtained maximal independent set S of $G^k(X)$ has at least q vertices then we are done: $d_q(X)=2r_p$ and S is a q-dispersion set. Otherwise, the maximal independent set $S\subseteq X$ obtained for $k=2r_p-2$ is the required one and $d_q(X)=2r_p-1$. Observe that in the second case the independent set problem in G^k is equivalent to the r-packing problem on G with $r(u)=r_p-1$ for any $u\in X$ and $r(u)=\infty$ otherwise, that is not true for the first case.

As was shown in [5] all powers of doubly chordal graphs are doubly chordal, thus they are chordal. The independent set problem in chordal graphs is solvable in time linear in the size of the graph [35]. So, the presented algorithm gives an $O(|V|^2\log|V|)$ time bound for computing the q-dispersion set of a doubly chordal graph. Unfortunately, the presented method is not polynomial for dually chordal graphs, because odd powers of dually chordal graphs have no special structure with respect to the independent set problem. Moreover, as we show below the q-dispersion problem for dually chordal graphs is $\mathbb{NP}$-complete.

Let G_0 be an arbitrary graph. Consider the dually chordal graph G, obtained from G_0 by adding a new vertex v adjacent to all vertices of G_0. Evidently, each independent set of G_0 is independent in G and conversely. Let d_q be the q-dispersion of V in G. Then in the graph G, and therefore in the graph G_0 too, there is an independent set with at least q vertices if and only if $d_q=2$. Thus, we reduce the general independent set problem to the q-dispersion problem on dually chordal graphs.

The same construction works for proving that all four classical graph problems (maximum independent set, minimum clique covering, minimum coloring, maximum clique problem) and other problems like e.g. the Hamiltonian circuit problem remain $\mathbb{NP}$-complete for dually chordal graphs. Summarizing the results of this section we obtain

Theorem 6.2. (1) *The p-center problem on a dually chordal graph $G=(V,E)$ can be solved in $O(|V|^2\log|V|)$ time;*

(2) *the q-dispersion problem on dually chordal graphs is $\mathbb{NP}$-complete;*

(3) *the q-dispersion problem on a doubly chordal graph $G=(V,E)$ can be solved in $O(|V|^2\log|V|)$ time.*

Next consider the simplest cases of the q-dispersion and p-center problems when $q=2$ and $p=1$. They are called the *diameter* and the *center problems* in G. Recall some necessary definitions. The *eccentricity* of a vertex $v\in V$ is $e(v)=\max\{d(v,u): u\in V\}$. The *diameter* $d(G)$ of G is the maximum eccentricity, while the *radius* $r(G)$ of G is the minimum eccentricity of vertices of G. A vertex whose eccentricity is equal to $r(G)$ is called a *central vertex*. Vertices $u,v\in V$ form a *diametral pair* of G if $d(u,v)=d(G)$. We present a linear time algorithm for computing a central vertex,

the diameter and a diametral pair of vertices of a dually chordal graph G. Another linear time algorithm for finding a central vertex of G is given in [13].

Let $G=(V,E)$ be a dually chordal graph. According to [13] for any vertex $v \in V$ if $d(v,u)=e(v)$ then $e(u) \geqslant 2r(G)-2$. The value $k=\lfloor (e(u)+1)/2 \rfloor$ is an approximation of the radius of G, more precisely either $k=r(G)$ or $k=r(G)-1$. Next we apply the r-domination algorithm, first with $r(v) \equiv k$ for all $v \in V$ and later with $r(v) \equiv k+1$ for all $v \in V$. If in the first case the obtained minimum r-dominating set D consists of a single vertex x then we are done: x is central and $r(G)=k$. Otherwise, we have $r(G)=k+1$ and the single vertex of the minimum r-dominating set with $r(v) \equiv k+1$ for all $v \in V$ must be central. Thus a central vertex x of a dually chordal graph G can be found in linear time, because the eccentricity of one arbitrary vertex and an r-dominating set are found in linear time.

Next we present a linear time algorithm for computing the diameter $d(G)$ of a dually chordal graph G. This algorithm is a modification of the RDP algorithm for the case $r(v)=r(G)-1$ for all $v \in V$.

Algorithm 6.3 (*Diameter*). *Find the diameter $d(G)$ and a diametral pair of vertices of a dually chordal graph G.*

Input: *A dually chordal graph $G=(V,E)$ with $|V|>2$.*

Output: *The diameter $d(G)$ of G and a diametral pair of vertices*

(1) *find a central vertex v and radius $r=r(G)$ of graph G;*
(2) *using the MNO algorithm find a maximum neighbourhood ordering $(v_1,\ldots,v_n)$ of G with $v_n=v$ and let $M:=\emptyset$;*
(3) **for all** $v \in V$ **do begin** $a(v):=0$; $fn(v):=v$ **end**
(4) **for** $i:=1$ **to** n **do**
(5) **if for all** $v \in N_i[v_i]$ $a(v)<r-1$ **and** $N(v_i) \cap M=\emptyset$ **then**
(6) **if** $a(mn(v_i)) \leqslant a(v_i)$ **then**
begin
(7) $a(mn(v_i)):=a(v_i)+1$;
(8) $fn(mn(v_i)):=fn(v_i)$
end
(9) **else if** $a(v_i)=r-1$ **then** $M:=M \cup \{v_i\}$
(10) **if** (*M has two nonadjacent vertices u and v*)
then $d(fn(u),fn(v))=d(G)=2r(G)$
(11) **else** $d(fn(u),fn(v))=d(G)=2r(G)-1$ *for any pair of vertices $u,v \in M$.*

Theorem 6.4. *The diameter $d(G)$, the radius $r(G)$, a diametral pair and a central vertex of a dually chordal graph G can be found in linear time.*

Proof. It is sufficient to show the correctness of the Algorithm Diameter only. As before we indicate by $a_i(v)$ the value of $a(v)$ obtained immediately after step i of the algorithm, where a step is understood as one iteration of lines (5)–(9). First remark that since v_n is a central vertex of G the MNO-tree T with root v_n has height $r=r(G)$.

Any maximum neighbour path between v_n and an arbitrary vertex $v \in V$ is a decreasing path of T. By the algorithm (lines (7), (8)) every vertex v_i belongs to the maximum neighbour path between $fn(v_i)$ and v_n. Thus, if $a_{i-1}(v_i) = r-1$ then the vertices v_i and v_n must be adjacent if $i < n$. By the algorithm MNO we obtain that all vertices $v_{i+1}, \ldots, v_{n-1}$ are adjacent to the vertex v_n too. Summarizing, we deduce that M is a subset of $N[v_n]$ consisting of vertices v_i with $a_{i-1}(v_i) = r-1$.

We assert that $|M| \geqslant 2$. To show this pick any $v \in M$, $v \neq v_n$. Let $\bar{v}$ be a farthest vertex from v, i.e. $d(v, \bar{v}) \geqslant r(G)$. Then either v_n lies on a shortest path between v and $\bar{v}$ or $d(v_n, \bar{v}) = d(v, \bar{v}) = r$, otherwise v_n is not central. Therefore, either $v_n \in M$ or M contains a neighbour of v_n one step closer to $\bar{v}$.

Next let $\bar{u} = fn(u)$, $\bar{v} = fn(v)$, where u and v are vertices of M selected on lines (10) or (11). Let $P' = (\bar{u}, u_1, \ldots, u)$ and $P'' = (\bar{v}, v_1, \ldots, v)$ be maximum neighbour paths between $\bar{u}$, u and $\bar{v}$, v, respectively. Both these paths are increasing and may be extended to maximum neighbour paths between v_n and $\bar{u}$ and $\bar{v}$. By Lemma 4.7 and since all vertices of $M \setminus \{v_n\}$ are adjacent to v_n in MNO-tree T, the maximum neighbour path between vertices $\bar{u}$ and $\bar{v}$ either coincides with path $P' \cup \{u, v_n, v\} \cup P''$ or consists of a subpath $(\bar{u}, \ldots, u')$ of P', an edge (u', v') and a subpath $(v', \ldots, \bar{v})$ of P''. In the first case u and v are nonadjacent and $d(\bar{u}, \bar{v}) = d(\bar{u}, u) + 2 + d(v, \bar{v}) = r - 1 + 2 + r - 1 = 2r$. Now to the second case. If $u' = u$ and $v' = v$ then we are in conditions of line (11), i.e. the set M induces a complete subgraph. Since $d(x, M) \leqslant r-1$ for all $x \in V$, $d(G) < 2r(G)$. From $d(\bar{u}, \bar{v}) = d(\bar{u}, u') + 1 + d(\bar{v}, v') = 2r(G) - 1$, we obtain that $d(\bar{u}, \bar{v}) = d(G) = 2r(G) - 1$. So, assume that $u' \neq u$ or $v' \neq v$.

Among adjacent vertices $u' \in P'$ and $v' \in P''$ with $u' \neq u$ or $v' \neq v$ we choose adjacent vertices $u^* \in P'$ and $v^* \in P''$ whose sum $d(u^*, v) + d(v^*, v)$ is minimal. Let $j = num(v^*) > num(u^*) = i$. Then evidently $u^* \neq v_n$. If $u^* \neq u$ then $mn(u^*)$ is adjacent to v^*. From the choice of vertices u^* and v^* we get that $mn(u^*) = u$ and $v = v^*$. Then $a_{i-1}(u^*) = r - 2$, otherwise $fn(u) \neq fn(u^*)$. Let v^+ be a vertex of P'' adjacent to v. If $num(v^-) < num(u^*)$ then according to the algorithm the vertex $u = mn(u^*)$ does not receive the value $r - 1$ on step $num(u^*)$ because $a_{i-1}(v^*) = r - 1$, and so $fn(u) \neq fn(u^*)$. Otherwise, if $num(v^+) > num(u^*)$ then u is adjacent to v^+ as maximum neighbour of u^*. In this case the vertex $v = mn(v^+)$ does not receive the value $r - 1$ on step $k = num(v^+)$, because $u \in N_k(v^+)$ and u already has value $r - 1$.

Next assume that $u^* = u$, i.e. $u^* \in N[v_n]$. Since $num(u^*) < num(v^*)$ v_n and v^* are adjacent and $mn(v^*) = v_n$ (by the MNO algorithm). Thus $v_n = v$. Since $u \in N[v^*] \cap M$ according to the algorithm the vertex $v_n = v$ does not get the value $r - 1$ on step $num(v^*)$, a contradiction. □

7. The r-dominating clique problem

An r-dominating set D of a graph G which induces a clique is called an *r-dominating clique* of G. For a given function $r : V \to \mathbb{N} \cup \{0\}$ and a given graph $G = (V, E)$ such a clique does not necessarily exist. As was shown in [6] even for weakly chordal

graphs the problem to decide whether or not a dominating clique ($r(v) \equiv 1$) exists is $\mathbb{NP}$-complete. For chordal graphs and Helly graphs there is a simple criterion for the existence of r-dominating cliques, which leads to polynomial algorithms for finding such a clique if it exists [15]; for the case of chordal graphs and $r(v) \equiv 1$ see [27].

The *r-dominating clique problem* consists in finding the minimum cardinality r-dominating clique of G if such a clique exists. For a bibliography on dominating clique problem we refer to [23]. In [15] a linear-time algorithm for solving the r-dominating clique problem in dually chordal graphs is given. Below we present a modification of this algorithm which allows to solve simultaneously the r-dominating clique problem and its dual problem on dually chordal graphs.

The problem dual to the r-dominating clique problem is: for a given graph G and function $r: V \to \mathbb{N} \cup \{0\}$ find a maximum cardinality vertex set P such that for all $u, v \in P$ $d(u,v) = r(u) + r(v) + 1$ holds. We call such a set P a *strict r-packing set* of G.

Algorithm 7.1 (*RDCSP*). *Find a minimum r-dominating clique and a maximum strict r-packing set of a dually chordal graph G.*

Input: *A dually chordal graph $G = (V,E)$ with a maximum neighbourhood ordering $v_1, \ldots, v_n$ produced by MNO and with non-negative integers $r(v_1), \ldots, r(v_n)$*

Output: *A minimum r-dominating clique D and a maximum strict r-packing set P of G if they exist and answer "NO" otherwise*

(1) $D := \emptyset$; $P := \emptyset$;
(2) **for all** $v \in V$ **do** $fn(v) := v$;
(3) **if for all** $v \in V$ $r(v) > 0$ **then**
(4) **for** $i := 1$ **to** n **do**
begin
(5) **if** $r(mn(v_i)) \geqslant r(v_i)$ **then**
begin
(6) $r(mn(v_i)) := r(v_i) - 1$;
(7) $fn(mn(v_i)) := fn(v_i)$;
end
(8) $G := G - \{v_i\}$;
(9) **if** $r(mn(v_i)) = 0$ **then goto** *outloop*
end
(10) **stop** *with output* $D := P := \{v_n\}$
else
outloop:
begin
(11) *with algorithm MNO find a maximum neighbourhood ordering* $(v'_1, \ldots, v'_p)$ *of the rest graph G with* $r(v'_p) = 0$;
(12) *par* $:= 0$;
(13) **for** $i := 1$ **to** p **do**

(14) **if** $r(v_i')=0$ **then**
begin
(15) $D:=D\cup\{v_i'\}$;
(16) $P:=P\cup\{fn(v_i')\}$;
(17) **if** *par* $=0$ **then** $v:=v_i'$ *and* *par* $:=1$;
end
(18) **else if** (**for all** $x\in N_i[v_i']$ $r(x)>0$) **and** $r(mn(v_i'))\geqslant r(v_i')$ **then**
begin
(19) $r(mn(v_i')):=r(v_i')-1$;
(20) $fn(mn(v_i')):=fn(v_i')$;
end
(21) **if** *D is no clique* **then**
begin
(22) **if** *v and* v_p' *are nonadjacent* **then**
$v':=fn(v)$ *and* $v'':=fn(v_p')$;
(23) **else** $v':=fn(x)$ *and* $v'':=fn(y)$ *where x and y are arbitrary nonadjacent vertices of D*;
(24) *output "there is no r-dominating clique in G, there are two vertices* v' *and* v'' *with* $d(v',v'')>r(v')+r(v'')+1$";
end
else
begin
(25) *D is a minimum r-dominating clique of G*;
(26) *P is a maximum strict r-packing set of G*;
end
end;

Theorem 7.2. *Algorithm RDCSP is correct and works in linear time.*

Proof. The time bound of the algorithm is obviously linear. The correctness proof is based on two claims:

(a) if D is no clique then $d(v',v'')>r(v')+r(v'')+1$ and thus there is no r-dominating clique in G;
(b) if D is a clique then D is an r-dominating clique, P is a strict r-packing set of G and $|D|=|P|$.

According to the algorithm if a vertex v_k is reached such that $v_j=mn(v_k)$ has the current r-value 0, then a maximum neighbourhood ordering $(v_1',\ldots,v_p')$ of the rest graph is computed such that $v_p'=v_j$. Observe that $(v_1,\ldots,v_k,v_1',\ldots,v_p')$ is a maximum neighbourhood ordering of the whole graph G. Denote the obtained ordering by $(v_1,\ldots,v_k,v_{k-1}^{\perp},\ldots,v_n^{\perp})$, where $v_{k+1}^{\perp}=v_1',\ldots,v_n^{\perp}=v_p'$, and let T be its rooted MNO-tree. For any vertex $v\in V$ let $num(v)$ be the index of v in this ordering.

Next we prove assertion (a). In order to present a unified proof we put $x=v_p'$ and $y=v$ in line (22) of the algorithm. Let $P'=(v',\ldots,x)$ and $P''=(v'',\ldots,y)$ be

maximum neighbour paths between v', x and v'', y, respectively. Both these paths belong to T and therefore are increasing. Consider the maximum neighbour path P between v' and v'' and suppose that its length is at most $r(v') + r(v'') + 1$. First assume that both ends of the intermediate edge of P belong to P' and P''. Let (u', u'') be an edge with $u' \in P'$ and $u'' \in P''$ which minimizes $d(u', x) + d(u'', y)$. Since x and y are non-adjacent, $u' \neq x$ or $u'' \neq y$. Without loss of generality assume that $num(u'') < num(u')$. If $u'' \neq y$ then $mn(u'')$ is adjacent to u', in contradiction with the choice of the edge (u', u''). Thus $u'' = y$. First assume that $x \neq v'_p$. Then according to the algorithm the vertex v'_p is adjacent to v. Hence, by the MNO algorithm v'_p is adjacent to all vertices $x \in V$ with $num(x) \geqslant num(v)$. In particular, v'_p is adjacent to both x and y. Moreover, $mn(x) = mn(y) = v'_p$, i.e. v'_p is adjacent to u' and all vertices of the path P' between u' and x. This follows from the fact that the indices of all these vertices are greater than $num(y)$ while $num(y) > k$. But then $mn(u') = v'_p$, in contradiction with our assumption that $v'_p \neq x$. Next assume that $x = v'_p$ and $y = v$ (see lines (17) and (22) of the algorithm). According to the algorithm all vertices of the path P' between v'_p and $fn(v'_p)$ are processed during the first part of the algorithm, i.e. their indices are at most k. This contradicts to the assumption that $num(u') > num(u'') = num(v) > k$. Now, suppose that one end of the intermediate edge of P does not belong to $P' \cup P''$. Then $x = v'_p$ and $y = v$, i.e. the condition of line (22) of the algorithm is fulfilled. Indeed, otherwise v'_p is adjacent to x and y in G and in T. Moreover, since v'_p is the root of the tree T, the maximum neighbour paths from v'_p to v' and v'' pass through x and y, respectively. This is possible only if the ends of the intermediate edge of P coincide with x and v'_p or with y and v'_p. Thus $d(v', v'') = r(v') + r(v'') + 2$. So, let $x = v'_p$ and $y = v$. Pick an edge (u', u'') with $u' \in P'$ and u'' in the path of T between v and v'_p which minimizes $d(u'', v)$ (at least one such edge exists, since the intermediate edge of P is of this kind). Let u be the neighbour of u'' in the subpath of T between u'' and y. By the algorithm all vertices of the path P' between v'_p and $fn(v'_p)$ have indices at most k. In particular, $num(u') \leqslant k$, while $num(u'') > num(u) > k$. This means that the vertices $mn(u')$ and u are adjacent, contrary to the choice of the edge (u', u''). This settles the proof of claim (a).

In order to prove claim (b), we show that the set D constructed by the algorithm is an r-dominating clique of G. Comparing the algorithms RDP and RDCSP we conclude that they work identically until for the first time $r(mn(v_i)) = 0$ occurs (line (9)). This means that for any vertex v both algorithms return identical r-value and one and the same vertex $fn(v)$. In particular we obtain, that if the label "outloop" is never reached by a *goto* command then $\{v_n\}$ is an r-dominating and a strict r-packing set of G. Now compare the actions of both algorithms on the rest graph. Let $v'_1, \ldots, v'_p$ be a maximum neighbourhood ordering such that $r(v'_p) = 0$. Then for any vertex v'_i we have $c(v'_i) = \infty$ in RDP algorithm.

Thus, both algorithms RDP and RDCSP put $r(mn(v'_i)) := r(v'_i) - 1$ and $fn(mn(v'_i)) := fn(v'_i)$ iff $r(x) > 0$ for all $x \in N_i[v'_i]$ and $r(mn(v'_i)) \geqslant r(v'_i)$ until for the first time $r(v'_i) = 0$ occurs in the second part of the algorithm (line (14)). Since both $v = v'_i$ and v'_p have to be in D and D is a clique, v and v'_p are adjacent. Recall that the algorithm MNO

has the property that all neighbours of the last vertex v'_p form an interval directly to the left of v'_p. On these vertices both algorithms simply select neighbours v'_i of vertex v'_p with the property that $r(v'_i)=0$. So, the set D obtained by RDCSP is also obtained by RDP, i.e. D is an r-domination clique. On the other hand, the set P is an r-packing of G and $|P|=|D|$ (see Lemma 5.8). Thus,

$$d(fn(x),fn(y))>r(fn(x))+r(fn(y))$$

for any vertices $x, y \in D$. Since x and y are adjacent and

$$d(x,fn(x))=r(fn(x)), d(y,fn(y))=r(fn(y)),$$

also

$$d(fn(x),fn(y))\leqslant r(fn(x))+r(fn(y))+1.$$

Therefore P is a strict r-packing set. □

The algorithm RDCSP can be used for finding a central clique of a dually chordal graph. The *eccentricity* $e(C)$ of a clique C is the maximum distance from any vertex to C. A *clique center* of G is a clique with minimum eccentricity which is called the *clique radius* of G and is denoted by $cr(G)$. For an arbitrary graph G we have $r(G)\geqslant cr(G)\geqslant r(G)-1$, because the eccentricity of any clique containing a central vertex of G is at most $r(G)$. So, it is sufficient to decide whether G contains an r-dominating clique with $r(v)=r(G)-1$ for all $v\in V$.

Corollary 7.3. *A clique center and the clique radius $cr(G)$ of a dually chordal graph G can be computed in linear time.*

8. Connected r-domination and Steiner tree problems

An r-dominating set D of a graph G which induces a connected subgraph is called *connected r-dominating set* of G. The *connected r-domination problem* consists in finding a minimum cardinality connected r-dominating set of G. For a given graph G and a set $S\subseteq V$ (of *terminal vertices*) a *Steiner tree* is a tree which spans all vertices of S. The *Steiner tree problem* asks for a minimum cardinality Steiner tree. The Steiner tree problem is a particular instance of the connected r-domination problem when $r(v)=0$ for any terminal vertex and $r(v)=\infty$ for all other vertices.

In general, both problems are $\mathbb{NP}$-complete [20]. For more special graph classes the situation is sometimes better (for a bibliography on connected domination cf. [23]). For the connected r-domination problem a polynomial algorithm was known for strongly chordal graphs [9]. In [13] this algorithm was extended to the whole class of dually chordal graphs. Its complexity is linear for doubly chordal graphs and quadratic for dually chordal graphs.

The maximum neighbourhood orderings of dually chordal graphs remain a useful tool for solving the connected r-domination problem too. Let $(v_1,\ldots,v_n)$ be the ordering of vertices of a dually chordal graph $G=(V,E)$ generated by the algorithm MNO. By $r(v_1),\ldots,r(v_n)$ we denote the dominating radii of the corresponding vertices of G. The next algorithm constructs a minimum connected r-dominating set $D\subseteq V$. Initially D contains all vertices v with $r(v)=0$. In step i the algorithm processes the vertex v_i according to the following rules. If $r(v_i)>0$ and $r(v)>0$ for any $v\in N_i[v_i]$ then we update $r(mn(v_i))$ by considering $r(mn(v_i))=\min\{r(mn(v_i)),r(v_i)-1\}$ and including $mn(v_i)$ in D if its radius becomes 0. If $r(v_i)=0$ and $r(v)>0$ for any $v\in N_i(v_i)$ but $r(u)=0$ for some vertex of G_{i-1} then we put $r(mn(v_i))=0$ and include $mn(v_i)$ in D. Otherwise, if $r(v_i)=0$ but $D\cap N_i(v_i)\neq\emptyset$ then we add new edges in G_{i-1} in order to transform the subgraph induced by $D\cap N_i(v_i)$ into a connected subgraph. In such a way we keep the connectedness of all these vertices through the vertex v_i. In this case we also delete all remaining vertices from $N_i(v_i)$, i.e. vertices $v\in N_i(v_i)$ with $r(v)>0$, except the vertex $mn(v_i)$.

Our algorithm works as follows:

Algorithm 8.1 (*CRD*). *Find a minimum connected r-dominating set of a dually chordal graph G.*

Input: *A dually chordal graph G with a maximum neighbourhood ordering* $(v_1,\ldots,v_n)$ *produced by MNO and dominating radii* $r(v_1),\ldots,r(v_n)$

Output: *A minimum connected r-dominating set D of G*

(1) $D:=\emptyset$;
(2) **for all** $v\in V$ **do if** $r(v)=0$ **then** $D:=D\cup\{v\}$;
(3) **if** *D is empty* **then**
(4) **for** $i:=1$ **to** n **do**
begin
(5) $v_j:=mn(v_i)$;
(6) $r(v_i):=\min\{r(v_i)-1,r(v_j)\}$;
(7) $G:=G-v_i$;
(8) **if** $r(v_j)=0$ **then goto** *outloop*
end;
(9) $D:=\{v_n\}$;
(10) **else** {*now* $r(v)=0$ *for some* v *and suppose that G has p vertices*}
(11) *outloop*:
begin
(12) *using MNO algorithm find a maximum neighbourhood ordering* $v_1,\ldots,v_p$ *of the rest graph G with* $r(v_p)=0$;
(13) $D:=D\cup\{v_p\}$;
(14) **for** $i:=1$ **to** p **do** $a(v_i):=0$;
(15) **for** $i:=1$ **to** $p-1$ **do**
(16) **if** $a(v_i)=0$ **then**
begin

```
(17)        v_j := mn(v_i);
(18)        if r(x) > 0 for all x ∈ N_i(v_i) then
(19)          if r(v_i) = 0 then begin r(v_j) := 0; D := D ∪ {v_j} end
(20)          else
                begin
(21)              r(v_j) = min{r(v_i) − 1, r(v_j)};
(22)              if r(v_j) = 0 then D := D ∪ {v_j}
                end
(23)        else if r(v_i) = 0 then
              begin
(24)            F := ∅;
(25)            for all x ∈ N_i(v_i) do
(26)              if r(x) = 0 then F := F ∪ {x}
(27)              else if x ≠ v_j then a(x) := 1;
(28)            add some new edges (in case of need) so that the subgraph
                induced by set F becomes connected;
(29)            update sets N_i(x) for all x ∈ N_i(v_i) ∩ F;
              end
          end
      end;
```

The running time of this algorithm is $O(|E|+|E'|+t)$, where E' is the set of added edges and t is the total cost of lines (28) and (29).

Theorem 8.2. *Algorithm CRD is correct and works in time* $O(|V|^2)$. *For doubly chordal graphs the running time of this algorithm is linear.*

The proof of this theorem requires some auxiliary results. Let v be a vertex of a dually chordal graph G which has a maximum neighbour $u \neq v$.

Lemma 8.3. *In a graph G with at least two vertices there exists a minimum connected r-dominating set D in which $v \notin D$ if and only if $r(v) \neq 0$.*

As we already saw a similar result holds for the general r-domination problem; see the proof of Theorem 5.2. This is true in our case too since if $r(v) \neq 0$ and $v \in D$ we can replace v by its maximum neighbour, obtaining a new connected r-dominating set of the same cardinality.

Lemma 8.4. *Let $r(v) > 0$. A subset $D \subseteq V \setminus \{v\}$ is a minimum connected r-dominating set of G if D is a minimum connected r'-dominating set of $G' = G - v$ with $r'(x) = r(x)$ when $x \neq u$, and $r'(u) = r(u)$ when $r(w) = 0$ for some $w \in N(v)$ and $r'(u) = \min\{r(u), r(v) - 1\}$ otherwise.*

Proof. Any connected r'-dominating set of the graph G' is an r-dominating set of G too. Let $D \subseteq V\backslash\{v\}$ be a minimum connected r-dominating set of G. Evidently, D induces a connected subgraph of G' too. Since G' preserves all distances between vertices of $V\backslash\{v\}$ a vertex x is r'-dominated in G' by the same vertex of D as in G if only $r'(x) = r(x)$. So, suppose that $r'(u) = r(v) - 1 < r(u)$. Consider the vertex $z \in D$ which r-dominates the vertex v in G. If $d(z,v) \geqslant 2$ then $d(u,z) = d(v,z) - 1 \leqslant r'(u)$, i.e. u is r'-dominated by z. Finally, if $z \in N(v)$ then since $r(z) > 0$ we can replace in D the vertex z by u. The obtained set D is an r-dominating and an r'-dominating set. □

Lemma 8.5. *Let $r(v) = 0$ and $r(x) = 0$ for some vertex $x \in V\backslash\{v\}$, and $r(w) > 0$ for every $w \in N(v)$. D is a minimum connected r-dominating set of G if $D = D' \cup \{v\}$ where D' is a minimum connected r'-dominating set of $G' = G - v$ with $r'(y) = r(y)$ when $y \neq u$ and $r'(u) = 0$.*

Proof. Again, if D' is a connected r-dominating set of G' then $D = D' \cup \{v\}$ is a connected r-dominating set of G too. Let D be a minimum connected r-dominating set of G. Necessarily $v \in D$. By the connectedness of D some neighbour of v, say z must be in D. Since $r(z) \neq 0$ then $D\backslash\{z\} \cup \{u\}$ is r-dominating and connected in G. Therefore, $D'\backslash\{z\} \cup \{u\}$ is a connected r'-dominating set of G'. □

Lemma 8.6. *Let $r(v) = 0$ and $r(x) = 0$ for some vertex $x \in N(v)$. A set D is a minimum connected r-dominating set of G if $D = D' \cup \{v\}$ where D' is a minimum connected r-dominating set of a graph G' obtained from $G - v$ by deleting all vertices $w \in N(v)$ with $r(w) > 0$ and $w \neq u$, and by adding new edges (if necessary) so that the subgraph induced by the set $F = \{x \in N(v) : r(x) = 0\}$ becomes connected.*

Proof. Evidently, if D is a connected r-dominating set of G then $D\backslash\{v\}$ is a connected r-dominating set of G'. Conversely, let D' be a connected r-dominating set of G'. All vertices $w \in N(v)$ of G with $r(w) > 0$ are dominated by v. Remark that G' preserves distances between all pairs of vertices of G, except vertices of $N(v)$. Therefore, if some vertex $x \notin N(v)$ is r-dominated in G' by a vertex $z \in D'$ then z r-dominates x in G too. Thus, $D' \cup \{v\}$ is a connected r-dominating set of G. □

Lemma 8.7. *Let $v = v_1, v_2, \ldots, v_n$ be an ordering of vertices of a dually chordal graph G obtained by the MNO algorithm and let G' be a graph defined in Lemma 8.6. Then G' is a dually chordal graph and the ordering $v_{i_1}, \ldots, v_{i_p}$ of G' induced by the ordering $v_1, \ldots, v_n$ is a maximum neighbourhood ordering with the same maximum neighbours of vertices.*

Proof. Assume to the contrary that we delete a vertex $y \in N(v)$ which is a maximum neighbour of some vertex x of G. Then x must be adjacent to the maximum neighbour u of v. According to the MNO algorithm vertices u and y necessarily coincide. This is

in contradiction with our assumption because we do not delete the maximum neighbour of v.

As we already mentioned in the proof of Lemma 8.6. by adding a new edge we can only change the distance between the end-vertices of this edge from 2 to 1. Therefore, for any vertex v_{i_c} of G' if $N^2_{i_c}[v_{i_c}] = N^1_{i_c}[mn(v_{i_c})]$ holds in G then this equality remains true in the graph G' too. □

Proof of Theorem 8.2. The correctness of the algorithm CRD follows by induction on the number of vertices from Lemmas 8.3–8.7. Remark that if G is doubly chordal and the maximum neighbourhood ordering is obtained by the DMNO algorithm then $E' = \emptyset$ and $t = 0$. Thus in this case the complexity of the algorithm is linear. Note also that in the general case all added edges E' are edges from the square G^2 of a graph G.

Consider an arbitrary edge $(x, y) \in E'$. Let (x, y) be added to G in step i. According to the algorithm we have that $r(mn(v_i)) \neq 0$ in this step. otherwise the set F is already connected and we do not need to add new edges. This means that both vertices x and y are adjacent to $mn(v_i)$ in the initial graph G. Therefore $d(x, y) = 2$ in G and we are done. □

Next we consider the problem of finding a minimum cardinality Steiner tree in dually chordal graphs. As we already mentioned this problem is a particular instance of the connected r-domination problem, thus we can apply the CRD algorithm to solve it. As we will show below, in this case the CRD algorithm is nothing else than one of the implementations of the following "greedy method".

Let S be a set of terminal vertices and let $C_1, C_2, \ldots, C_p$ be the connected components of the graph $G(S)$ induced by S. For each component C_i by v_i^+ we denote the vertex with maximum number in the maximum neighbourhood ordering of G. We call such a vertex v_i^- the *maximal vertex* of the connected component C_i. In each step we choose the vertex v^- such that $num(v^+) = \min\{num(v_i^-), \ldots, num(v_p^+)\}$. We add the maximum neighbour $mn(v^+)$ of v^+ to the set S and update the connected components of S and their maximal vertices. We repeat this step until the number of connected components is at most two.

In order to prove the correctness of this method it is enough to show that the first vertex added to S by the method will be chosen as a first vertex added to the set D according to Lemmas 8.3, 8.5 and 8.6. To prove this we proceed by induction on the number of vertices. Consider the leftmost vertex v in the maximum neighbourhood ordering of G. If $v \notin S$ or $v \in S$ and $N(v) \cap S \neq \emptyset$ then by Lemmas 8.3 and 8.6 we will not add new vertices to the set D. In this case we do not add by the last method a new vertex to the set S. Moreover, if $N(v) \cap S \neq \emptyset$ then adding new edges (see Lemma 8.6) we do not change the set of connected components and their maximal vertices. By the induction assumption the maximum neighbour of vertex v^+ selected by the greedy method is also included in D.

Finally, assume that $v \in S$ but $N(v) \cap S = \emptyset$. In this case v is the maximal vertex of its connected component. Moreover, we immediately get that $v = v^+$, i.e. $mn(v)$ is the

first vertex which is included in S. On the other hand, by Lemmas 8.3 and 8.5, the vertex $mn(v)$ is added to D.

Unfortunately, we are not able to implement this method in linear time. Standard data structures and UNION-FIND techniques give an $O(|E| + |\overline{S}| \log |S|)$ bound for time complexity, where S is the set of terminal vertices and $|\overline{S}|$ is the size of the obtained Steiner tree. The time needed to look for and update minimum v^+ is at most $O(|\overline{S}| \log |S|)$. To merge the connected components when a Steiner point (vertex $mn(v^-)$) is added, one can use "rename smaller sets" (see e.g. [1]) to get an $O(|\overline{S}| \log |S|)$ bound in total for this case.

9. NP-complete problems

As we already mentioned all four classic graph problems are NP-complete on dually chordal graphs. On the other hand, the main goal of this paper is to consider the domination-like problems. The r-domination problem, the connected r-domination problem and the r-dominating clique problem are efficiently solvable on dually chordal graphs. Below we show that the independent domination problem and the dominating cycle problem are NP-complete on these graphs.

Recall that the problems of finding a minimum dominating set D of G with the additional requirement to induce an independent set or to have a Hamiltonian cycle in $G(D)$ are known as the *independent domination problem* and the *dominating cycle problem*. It is known that both these problems are NP-complete [20].

Theorem 9.1. *The independent domination problem and the dominating cycle problem are NP-complete on dually chordal graphs.*

Proof. (1) Let $G=(V,E)$ be an arbitrary graph. Construct a new graph $H=(V',E')$ by adding to G a new vertex v adjacent to all vertices of G, a new vertex w adjacent to v and $|V|+1$ new vertices pendant to w (see Fig. 1). Evidently, H is dually chordal. We claim that H has an independent dominating set of size $k \leqslant |V| + 1$ if and only if G has an independent dominating set of size $k-1$.

Let D be an independent dominating set of H of size $k \leqslant |V| + 1$. Then necessarily $D \subseteq V \cup \{w\}$. Hence $D \setminus \{w\}$ is an independent dominating set of G. Conversely, if $D \subseteq V$ is an independent dominating set of G of size $k-1$ then $D \cup \{w\}$ is an independent dominating set of H.

(2) Let $G=(V,E)$ be an arbitrary graph. Construct a new graph $H=(V',E')$ by adding pendant vertices to all vertices of G and a new vertex v adjacent to all vertices of G (see Fig. 2). Again, H is a dually chordal graph. We claim that H has a dominating cycle if and only if G has a Hamiltonian path.

Let H have a dominating cycle C. Evidently, $C \subseteq V \cup \{v\}$. If $v \in C$ then $C \setminus \{v\}$ is a Hamiltonian path of G. Otherwise, C is a Hamiltonian cycle of G. Conversely, let G have a Hamiltonian path P. Then $P \cup \{v\}$ is a dominating cycle of H. □

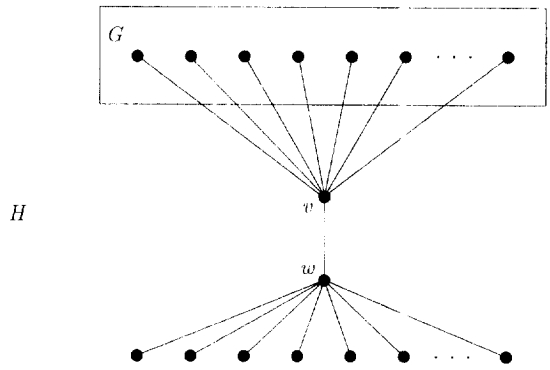

Fig. 1.

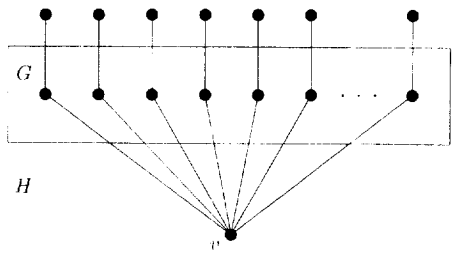

Fig. 2.

Another type of domination problems, namely the *total domination problem* [23] consists in finding a minimum dominating set which induces a subgraph without isolated vertices. This problem has a polynomial time solution for strongly chordal graphs [9]. For dually chordal graphs this problem is still open. Our conjecture is that it can be solved in polynomial time too.

One of the approaches to solve $\mathbb{NP}$-complete problems on graphs is to apply efficient dynamic programming algorithms to graphs with bounded treewidth [25]. A *triangulation* of a graph G is a graph H with the same vertex set as G, such that G is a subgraph of H and H is chordal. The *treewidth* of a graph G is the minimum value k for which there exists a triangulation H of G whose maximal clique has size $k+1$; see [25] for equivalent definitions and properties. Unfortunately, this approach is useless for dually chordal graphs, since although the treewidth of these graphs is bounded by the maximal vertex degree, its exact computation is an $\mathbb{NP}$-complete problem. To prove this, we use our standard construction: for an arbitrary graph $G=(V,E)$ add a new vertex v adjacent to all vertices of G. The obtained graph H is dually chordal and the treewidths of graphs G and H coincide. Thus we have

Theorem 9.2. *The problem of computing the treewidth of a dually chordal graph is* $\mathbb{NP}$*-complete.*

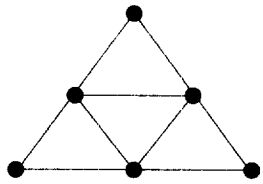

Fig. 3. A 3-sun.

In the present paper we use the maximum neighbourhood ordering to develop optimal algorithms for domination-like problems. The existence of such an ordering as we see from Theorem 2.1 is a consequence of the fact that the disk hypergraph of a dually chordal graph is a hypertree, i.e. the disk family of a dually chordal graph fulfills the Helly property and its intersection graph is chordal. As the next result shows, assuming only the Helly property for disks does not lead to polynomial algorithms.

Theorem 9.3. *The following problems are* $\mathbb{NP}$*-complete on graphs whose disk hypergraph has the Helly property: the domination problem, the connected domination problem, the dominating clique problem and the Steiner tree problem.*

Proof. Let $G=(V,E)$ be a graph without triangles and let H be the *split graph* of G : H has $V \cup E$ as vertex set. V induces a clique in H, E is independent in H, and a pair (v,e), where $v \in V$ and $e \in E$, is an edge of H iff $v \in e$. We reduce the $\mathbb{NP}$-complete vertex-edge covering problem on triangle-free graphs [20] to the domination problem on their split graphs.

Since G is triangle-free, H has no 3-suns as induced subgraphs (for the shape of a 3-sun see Fig. 3).

It is easy to see by induction that disks of split graphs without 3-suns have the Helly property. Note that G has a vertex-edge covering set of size k if and only if H has a dominating set D (necessarily contained in V) of the same size. Since $D \subseteq V$ the set D induces a complete subgraph. Thus, D is simultaneously a dominating set, a connected dominating set and a dominating clique of H. Moreover, D is the set of Steiner vertices from the Steiner tree which spans the set of terminal vertices E of the graph $H=(V \cup E, E')$. □

Acknowledgements

The authors are grateful to Dr. G. Chang, Dr. G. Frederickson and Dr. A. Tamir for acquaintance with their recent papers on related topics. We wish to acknowledge the anonymous referees for suggestions leading to improvements in the presentation of the results.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1976.

[2] H. Behrendt, A. Brandstädt, Domination and the use of maximum neighbourhoods, Technical Report SM-DU-204, University of Duisburg, 1992

[3] C. Berge, Hypergraphs, North-Holland, Amsterdam, 1989.

[4] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection, J. Comput. System. Sci. 7 (1973) 448–461.

[5] A. Brandstädt, F.F. Dragan, V.D. Chepoi, V.I. Voloshin, Dually chordal graphs, in: J. van Leeumen (Ed.), Proceedings 19th International Workshop "Graph-Theoretic Concepts in Computer Science", Utrecht, The Netherlands, 1993, Lecture Notes in Computer Science, vol. 790, Springer, Berlin, 1994, pp. 237–251; SIAM J. Discrete Math., to appear.

[6] A. Brandstädt, D. Kratsch, Domination problems on permutation and other graphs, Theoret. Comput. Sci. 54 (1987) 181–198.

[7] P. Buneman, A characterization of rigid circuit graphs, Discrete Math. 9 (1974) 205–212.

[8] R. Chandrasekaran, A. Doughety, Location on tree networks: p-center and q-dispersion problems, Math. Oper. Res. 6 (1981) 50–57.

[9] G.J. Chang, Labeling algorithms for domination problems in sun-free chordal graphs, Discrete Appl. Math. 22 (1988/89) 21–34.

[10] G.J. Chang, M. Farber, Z. Tuza, Algorithmic aspects of neighbourhood numbers, SIAM J. Discrete Math. 6 (1993) 24–29.

[11] G.J. Chang, G.L. Nemhauser, The k-domination and k-stability problems on sun-free chordal graphs, SIAM J. Algebraic Discrete Methods 5 (1984) 332–345.

[12] F.F. Dragan, Dominating and packing in triangulated graphs, Meth. Discrete Anal. (Novosibirsk) 51 (1991) 17–36 (in Russian).

[13] F.F. Dragan, HT-graphs: centers, connected r-domination and Steiner trees, Comput. Sci. J. Moldova 1(2) (1993) 64–83.

[14] F.F. Dragan, Domination in Helly graphs without quadrangles, Cybernet. System Anal. (Kiev) 6 (1993) 47–57 (in Russian).

[15] F.F. Dragan, A. Brandstädt, r-dominating cliques in Helly graphs and chordal graphs, Proceedings of the 11th STACS, Caen, France, Lecture Notes in Computer Science, vol. 775, Springer, Berlin, 1994, pp. 735–746; Discrete Math. 162 (1996) 93–108.

[16] F.F. Dragan, C.F. Prisacaru, V.D. Chepoi, Location problems in graphs and the Helly property, Discrete Math. Moscow 4 (1992) 67–73 (in Russian), (the full version appeared as preprint: F.F. Dragan, C.F. Prisacaru, V.D. Chepoi, r-domination and p-center problems on graphs: special solution methods and graphs for which this method is usable, Kishinev State University, preprint MoldNIINTI, N. 948-M88, 1987 (in Russian)).

[17] M. Farber, Characterizations of strongly chordal graphs, Discrete Math. 43 (1983) 173–189.

[18] M. Farber, Domination, independent domination and duality in strongly chordal graphs, Discrete Appl. Math. 7 (1984) 115–130.

[19] G.N. Frederickson, Parametric search and locating supply centers in trees, Proceedings of the Workshop on Algorithms and Data Structures (WADS'91), Lecture Notes in Computer Science, vol. 519, Springer, Berlin, 1991, pp. 299–319.

[20] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of $\mathbb{NP}$-Completeness, W.H. Freeman, San Fransisco, 1979.

[21] M. Grötschel, L. Lovász, A. Schrijver, Polynomial algorithms for perfect graphs, Ann. Discrete Math. 21 (1984) 325–356.

[22] Y. Gurevich, L. Stockmeyer, U. Vishkin, Solving $\mathbb{NP}$-hard problems on graphs that are almost trees and an application to facility location problems, J. ACM 31 (1984) 459–473.

[23] S.C. Hedetniemi, R. Laskar (Eds.), Topics on Domination, Ann. Discrete Math. 48 (1991).

[24] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems, I: the p-centers, SIAM J. Appl. Math. 37(3) (1979) 513–538.

[25] T. Kloks, Treewidth-Computations and Approximations, Lecture Notes in Computer Science, vol. 842, Springer, Berlin, 1994.

[26] A.W.J. Kolen, Duality in tree location theory. Cah. Cent. Etud. Rech. Oper. 25 (1983) 201–215.
[27] D. Kratsch, P. Damaschke, A. Lubiw, Dominating cliques in chordal graphs, Discrete Math. 128 (1994) 269–275.
[28] J. Lehel, A characterization of totally balanced hypergraphs, Discrete Math. 57 (1985) 59–65.
[29] L. Lovász, Normal hypergraphs and the perfect graph conjecture, Discrete Math. 2 (1972) 253–267.
[30] A. Lubiw, Doubly lexical orderings of matrices, SIAM J. Comput. 16 (1987) 854–879.
[31] N. Megiddo, A. Tamir, E. Zemel, R. Chandrasekaran, An $O(n \log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems, SIAM J. Comput. 10 (1981) 328–337.
[32] M. Moscarini, Doubly chordal graphs, Steiner trees and connected domination, Networks 23 (1993) 59–69.
[33] R. Paige, R.E. Tarjan, Three partition refinement algorithms, SIAM J. Comput. 16 (1987) 973–989.
[34] J. Plesnik, A heuristic for the p-center problem in graphs, Discrete Math. 17 (1987) 263–268.
[35] D.J. Rose, R.E. Tarjan, G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Comput. 5 (1976) 266–283.
[36] J.P. Spinrad, Doubly lexical ordering of dense 0–1-matrices, Inform. Process. Lett. 45 (1993) 229–235.
[37] J.L. Szwarcfiter, C.F. Bornstein, Clique graphs of chordal and path graphs, SIAM J. Discrete Math. 7 (1994) 331–336.
[38] A. Tamir, A class of balanced matrices arising from location problems, SIAM J. Alg. Discrete Meth. 4 (1983) 363–370.
[39] R.E. Tarjan, M. Yannakakis, Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13(3) (1984) 566–579.
[40] K. White, M. Farber, W. Pulleyblank, Steiner trees, connected domination and strongly chordal graphs, Networks 15 (1985) 109–124.