Diskrete Mathematik und Optimierung

Helena Bergold, Winfried Hochstättler, Uwe Mayer:

The Neighborhood Polynomial of Chordal Graphs

Technical Report feu-dmo
Contact: helena.bergold@fernuni-hagen.de, winfried.hochstaettler@fernuni-hagen.de, privat@uwemayer.info

FernUniversität in Hagen
Fakultät für Mathematik und Informatik
Lehrgebiet für Diskrete Mathematik und Optimierung
D – 58084 Hagen

2000 Mathematics Subject Classification:
Keywords: neighborhood polynomial, domination polynomial, chordal graph, comparability graph, leafage, anchor width

The Neighborhood Polynomial of Chordal Graphs

Helena Bergold, Winfried Hochstättler, Uwe Mayer
FernUniversität in Hagen, Germany

Abstract

The neighborhood polynomial of a graph G is the generating function of subsets of vertices in G that have a common neighbor. In this paper we study the neighborhood polynomial and the complexity of its computation for chordal graphs. We will show that it is NP-hard to compute the neighborhood polynomial on general chordal graphs. Furthermore we will introduce a parameter for chordal graphs called anchor width and an algorithm to compute the neighborhood polynomial which runs in polynomial time if the anchor width is polynomially bounded. Finally we will show that we can bound the anchor width for chordal comparability graphs and chordal graphs with bounded leafage. The leafage of a chordal graphs is the minimum number of leaves in the host tree of a subtree representation. In particular, interval graphs have leafage at most 2. This shows that the anchor width of interval graphs is at most quadratic.

1 Introduction

In this paper we study the neighborhood polynomial of graphs. Throughout the paper, all graphs are simple, finite and undirected. For a graph $G = (V, E)$, the *neighborhood* of a vertex $v \in V$ is the set of all adjacent vertices, denoted by $N_G(v) = \{u \in V \mid (u, v) \in E\}$. The *neighborhood complex* of a graph G was first introduced by Lovász [Lov78] and consists of all subsets of vertices $W \subseteq V$ which have a common neighbor, i.e.

$$\mathcal{N}_G = \{U \subseteq V \mid \exists v \in V : U \subseteq N_G(v)\}.$$

This set-system is clearly hereditary, hence it is a simplicial complex.
To count the number of sets with cardinality k in $\mathcal{N}_G$, we define the *neighborhood polynomial* $N_G(x)$, which is the generating function of the neighborhood complex $\mathcal{N}_G$, i.e.,

$$N_G(x) = \sum_{U \in \mathcal{N}_G} x^{|U|}.$$

Since we only consider finite graphs, the sum is finite and $N_G(x)$ is a polynomial such as all other generating functions considered in this paper. The constant term of $N_G(x)$ for every graph G is always 1, since the empty set is in the neighborhood complex and the only set of cardinality 0. Furthermore, it is easy to determine the linear coefficient which is the number of non-isolated vertices.

In this paper, we investigate the neighborhood polynomial of some graph classes. In particular, we look at chordal graphs and subclasses like interval graphs, split graphs and chordal comparability graphs. In Section 4, we discuss the complexity of computing the neighborhood polynomial for each of these classes,. In order to do this, we introduce the *anchor width* of a graph and develop an algorithm for computing the neighborhood polynomial in Section 3. We will see that the anchor width is the essential parameter. If for a graph class the anchor width is polynomially bounded in the number of vertices, our algorithm is efficient. In Section 4.3 we investigate the leafage $l(G)$ of a chordal graph G on n vertices and show that the anchor width is at most $n^{l(G)}$. For interval graphs, which are the graphs with leafage at most two, we give a family with quadratic anchor width.

2 Preliminaries

The neighborhood polynomial was introduced by Brown and Nowakowski [BN08] who also investigated the effect on the neighborhood polynomial of some elementary graph operations. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ on disjoint vertex sets, the *union* $G_1 \cup G_2$ of the graphs is the graph consisting of both graphs without adding edges, i.e. the graph on the vertex set $V_1 \cup V_2$ with edge set $E_1 \cup E_2$. The *join* $G_1 + G_2$ of the two graphs is the graph on the vertex set $V_1 \cup V_2$ consisting of both graphs together with all possible edges between vertices in V_1 and vertices in V_2, i.e. $E = E_1 \cup E_2 \cup \{v_1v_2 \mid v_1 \in V_1, v_2 \in V_2\}$.

Proposition 1 ([BN08])**.** *Let G_1 and G_2 be two graphs on disjoint vertex sets. Then the neighborhood polynomial of the disjoint union $G_1 \cup G_2$ is*

$$N_{G_1 \cup G_2}(x) = N_{G_1}(x) + N_{G_2}(x).$$

Proposition 2 ([BN08])**.** *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ be two graphs on disjoint vertex sets. Then the neighborhood polynomial of the join $G_1 + G_2$ is*

$$N_{G_1+G_2}(x) = (1 + x)^{|V_2|} N_{G_1}(x) + (1 + x)^{|V_1|} N_{G_2}(x) - N_{G_1}(x)N_{G_2}(x).$$

This two graph operations, disjoint union and join, are used to define cographs. *Cographs* are exactly the graphs which do not contain an induced P_4. They can be constructed recursively. Starting with a single vertex as a cograph, the disjoint union and the join of two cographs are cographs. For this and other well-known graph theoretic facts, we refer to [Gol80]. The neighborhood polynomial of a single vertex graph is $N(K_1, x) = 1$ and the two operations disjoint union and join, given by the two formulas in Proposition 1 and Proposition 2 are computable in linear time. Note that $(1 + x)^n$ can be computed in linear time using the binomial theorem. Furthermore Corneil et al. [CPS85] present a linear time algorithm to recognize cographs and construct their cotree. Hence it is possible to compute the neighborhood polynomial of a cograph in quadratic time.

Another graph operation is attaching a vertex v to a subset of vertices of a graph G. This operation was studied by Alipour and Tittmann [AT18], who gave an explicit formula for a neighborhood polynomial after attaching a vertex to a subset of vertices. More formally for a graph $G = (V, E)$, a subset $U \subseteq V$

of vertices and an additional vertex $v \notin V$, we denote by $G_{U \triangleright v}$ the graph with vertex set $V \cup \{v\}$ and edge set $E \cup \{uv \mid u \in U\}$. In the following we use the notation

$$N_G^{\cap}(W) = \bigcap_{w \in W} N_G(w) \quad \text{and}$$
$$N_G^{\cup}(W) = \bigcup_{w \in W} N_G(w)$$

for $W \subseteq V$.

Proposition 3 ([AT18]). *Let $G = (V, E)$ be a graph, $U \subseteq V$ and $v \notin V$. Then the neighborhood polynomial of $G_{U \triangleright v}$ is*

$$N_{G_{U \triangleright v}}(x) = N_G(x) + \sum_{W \subseteq U} \phi_W + \sum_{W \subseteq U} (-1)^{|W|+1} x(1+x)^{|N_G^{\cap}(W)|},$$

where

$$\phi_W = \begin{cases} x^{|W|}, & \text{if } N_G^{\cap}(W) = \emptyset; \\ 0, & \text{otherwise.} \end{cases}$$

Using this formula, Alipour and Tittmann [AT18] showed that for a fixed integer k, computing the neighborhood polynomial of k-degenerate graphs is possible in polynomial time. A *k-degenerate graph* is a graph where every subgraph has a vertex v with $\deg(v) \leq k$. Using the degeneracy, we can pick one vertex of degree $\leq k$ after another and update the neighborhood polynomial by the formula of Proposition 3 in order to get a polynomial runtime. As a corollary it follows that there is a polynomial-time algorithm to compute the neighborhood polynomial for planar and k-regular graphs [AT18]. The update formula of Alipour and Tittman (see Proposition 3) was the starting point of our investigations for chordal graphs.

A graph G is said to be *chordal* if there is no induced cycle of length ≥ 4. Equivalently a graph is chordal if and only if it has a perfect elimination order. A *perfect elimination order* is an ordering of the vertices $v_1, \ldots, v_n$ such that for all i the neighborhood of v_i in $G[\{v_i, \ldots v_n\}]$ is a clique. Here for any subset $U \subseteq V$ the graph $G[U]$ is the of U induced subgraph of G. A vertex v_i, where the neighborhood is a clique is called *simplicial*. It is well-known that every chordal graph has at least two simplicial vertices, which gives us the perfect elimination order (cf. [Gol80]).

In order to study the neighborhood polynomial of chordal graphs and subclasses, we make use of the perfect elimination order in reverse order. We will adapt the formula of Alipour and Tittmann (Proposition 3) to our use. To construct a chordal graph, it is enough to attach one vertex after another to a clique C of a graph G.

To get some complexity results of computing neighborhood polynomial, the connection to the domination polynomial is useful. To define the domination polynomial, we introduce dominating sets. A *dominating set* of a graph $G = (V, E)$ is a set of vertices $D \subseteq V$ such that

$$D \ \cup N_G^{\cup}(D) = V.$$

The family of all dominating sets of a graph G is denoted by $\mathcal{D}_G$ and the *domination polynomial* $D_G(x)$ is the generating function of $\mathcal{D}_G$, i.e.

$$D_G(x) = \sum_{U \in \mathcal{D}_G} x^{|U|}.$$

The following relation between domination polynomials and neighborhood polynomials holds, for a proof see for example [HT18] or [Day17].

Proposition 4. *For any graph $G = (V, E)$ and its complement graph $\overline{G}$ it holds:*

$$D_{\overline{G}}(x) + N_G(x) = (1 + x)^{|V|}.$$

With other words this proposition says that every vertex set has either a common neighbor in the graph or is a dominating set in the complement graph. The connection of these two polynomials can be used to determine the complexity of computing the neighborhood polynomial. In particular, the neighborhood polynomial is computable in polynomial time if and only if the domination polynomial of the complement graph is computable in polynomial time. Furthermore the contributions to the well-known graph problem DOMSET, the problem of finding a minimal dominating set in a graph, imply some complexity results for the neighborhood polynomial.

Corollary 5. *Let $\mathcal{G}$ be a class of graphs and $\overline{\mathcal{G}}$ the class of the complement graphs of $\mathcal{G}$. If DOMSET is NP-hard on $\overline{\mathcal{G}}$, then computing the neighborhood polynomial on $\mathcal{G}$ is NP-hard as well.*

Proof. DOMSET can be reduced to computing the domination polynomial as we can easily find the smallest k such that x^k has a nonzero coefficient in $D_{\overline{G}}(x)$. With Proposition 4 the corollary holds. □

DOMSET is NP-hard on many graph classes such as chordal graphs [BJ82]. Afterwards Bertossi [Ber84] showed that it is NP-hard on bipartite graphs and split graphs. *Split graphs* are the graphs where the vertex set can be partitioned into a clique and an independent set. Since split graphs are exactly the graphs which are chordal and co-chordal (i.e. the complement graph is chordal) [Gol80], DOMSET is also NP-hard on co-chordal graphs. This together with Corollary 5 shows the NP-hardness of computing the neighborhood polynomial in split graphs (cf. [Day17]) and hence in chordal graphs.

3 Algorithm for Chordal Graphs

In this section, we derive an algorithm to compute the neighborhood polynomial of chordal graphs, which runs in polynomial time if the so called anchor width is polynomially bounded in the number of vertices. In chordal graphs the anchor width can be exponential in the number of vertices (see Section 4.1), but there are some subclasses such as chordal comparability graphs (Section 4.2) and interval graphs or more generally chordal graphs with bounded leafage (Section 4.3) which have polynomially bounded anchor width. Our algorithm relies on the perfect elimination order of chordal graphs and comes from the vertex-attachment formula of Alipour and Tittmann, see Proposition 3. First, we adapt this formula to the special case that the set, to which we attach a vertex is a clique. To study the new arising neighborhood sets we introduce anchor sets, which are subsets of a clique appearing as a common neighborhood. The maximal number of anchor sets of a clique, which we denote as anchor width, is the essential parameter in this algorithms in order to get a polynomial runtime for our algorithm.

Let C be a clique in a graph $G = (V, E)$. We define the set of neighbors of the clique C, not including the clique itself as the *periphery* of C, denoted by

$$P_G(C) = N_G^{\cup}(C) \setminus C.$$

A subset $M \subseteq P_G(C)$ of the periphery is called *periphery set*. Note that the empty set is also a periphery set. We call a non-empty subset A of C *anchor set*, if it is the common neighborhood of some periphery set M. See Figure 1 for an illustration. For any $M \subseteq P_G(C)$ we define the *corresponding anchor set* as

$$A_G(M, C) = N_G^{\cap}(M) \cap C$$

if the intersection is non-empty.

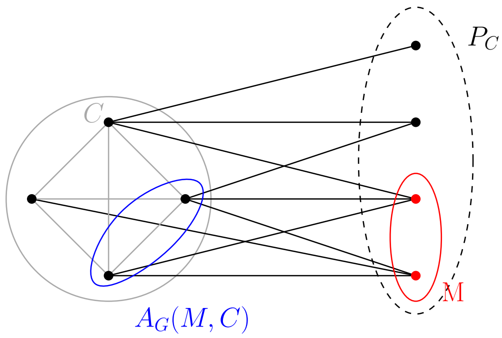

Figure 1: An illustration of the introduced sets, the periphery P_C and an periphery set M with corresponding anchor set $A_G(M, C)$ of a clique C.

Note that several periphery sets M and M' can correspond to the same anchor set, i.e. $A_G(M, C) = A_G(M', C)$. For an anchor set A, the periphery sets $M \subseteq P_G(C)$, whose common neighborhood in C is A, build the family

$$\mathcal{P}_G(A, C) = \{M \subseteq P_G(C) \mid A_G(M, C) = A\}.$$

The generating function

$$P_G(A, C, x) = \sum_{M \in \mathcal{P}_G(A,C)} x^{|M|}$$

of $\mathcal{P}_G(A, C)$ is called *periphery polynomial*. Note that $\mathcal{P}_G(A, C) = \emptyset$ and $P_G(A, C, x) = 0$ if A is not an anchor set of C. Furthermore we define the family of all anchor sets of a clique C as

$$\mathcal{A}_G(C) = \{A \subseteq C \mid A \neq \emptyset \text{ and } \exists M \subseteq P_G(C) : A = A_G(M, C)\}.$$

Note that for every clique, it holds $C \in \mathcal{A}_G(C)$, since it is the anchor set of the empty periphery set. The *anchor width* of a graph G on n vertices is the smallest number k such that $|\mathcal{A}_G(C)| \le k$ for all cliques C in G or in other words the maximal number of anchor sets appearing in a clique C of G.

For a maximal clique C_{max} and a clique C contained in C_{max} the following relations hold.

Lemma 6. *Let C be a clique and C_{max} a maximal clique containing C in a graph $G = (V, E)$. Then the following conditions hold:*

(a) $C_{max} \backslash C \subseteq P_G(C) \subseteq P_G(C_{max}) \cup (C_{max} \backslash C)$

(b) $\mathcal{A}_G(C) = \{A \cap C \mid A \cap C \neq \emptyset$ and $A \in \mathcal{A}_G(C_{max})\}$

(c) For any $A \in \mathcal{A}_G(C)$ the periphery polynomial is

$$P_G(A, C, x) = (1+x)^{|C_{max} \backslash C|} \sum_{\substack{A' \in \mathcal{A}_G(C_{max}) \\ A' \cap C = A}} P_G(A', C_{max}, x)$$

Proof. (a) Since every vertex in the clique $C_{max} \backslash C$ is adjacent to C, the first inclusion holds. Furthermore, every element which is adjacent to one of the elements in C is either an element of $C_{max} \backslash C$ or it is adjacent to an element of C_{max}.
To show (b), we check which subsets of the clique C can appear as an anchor set. Let $M' \subseteq P_G(C)$ be a periphery set such that $A_G(M, C)$ is a non-empty anchor set. Using (a) we distinguish three cases.
If $M' \subseteq C_{max} \backslash C$, the anchor set of M' is C itself. If $M' \subseteq P_G(C_{max})$ there exists an anchor set $A = A_G(M, C_{max})$. Since M' only consists of elements of the periphery of C, the intersection of A with C provides the anchor set $A_G(M, C)$, which is non-empty. In the final case, M' consist of elements of $C_{max} \backslash C$ and $P_G(C_{max})$. We only need to consider $M = M' \cap P_G(C_{max})$, since the elements of $C_{max} \backslash C$ just lead to another intersection with C. We continue as in the second case.
On the other hand, a set $A' = A \cap C \neq \emptyset$ for $A \in \mathcal{A}_G(C_{max})$ is always an anchor set. For A there exists a periphery set $M \subseteq P_G(C_{max})$ which has A as common neighborhood $A_G(M, C_{max})$ in C_{max}. If we take all elements of M which are in the periphery of C, the common intersection of those elements inside C is exactly A'. Hence A' is an anchor set.
(c) The periphery polynomial $P_G(A, C, x)$ counts the different periphery sets with respective to the size, where A is the corresponding anchor set. As we have seen in (b), an anchor set A is given by $A = A' \cap C$ for an anchor set $A' \in \mathcal{A}_G(C_{max})$. Since there are different possibilities to choose A', we sum over all corresponding periphery polynomials which are counted in $P_G(A', C, x)$. Furthermore all elements in $C_{max} \backslash C$ are in the periphery of C (see (a)) with neighborhood C. Hence we can add elements of $C_{max} \backslash C$ to any periphery set M' with anchor set $A_G(M', C_{max}) = A'$ and still have as anchor set $A_G(M, C) = A$. For the polynomial as generating function, we multiply $P_G(A', C, x)$ by $(1+x)^{|C_{max} \backslash C|}$.
For different anchor sets A' and A'' with $A'' \cap C = A = A' \cap C$, the corresponding periphery sets are pairwise different, since the common neighborhood inside

C_{max} is different. So in order to get all periphery sets corresponding to A, we need to add the polynomials. □

Due to Lemma 6, it is sufficient to provide the information about anchor sets and periphery polynomials for all maximal cliques. With this information we are able to compute the necessary information for all other cliques which are not maximal. Furthermore the anchor width only depends on the size of the anchor family of the maximal cliques.

For any set $U \subseteq V$ of vertices we define the *local neighborhood* of U as the family consisting of all vertex sets of G which have a common neighbor in U, i.e.

$$\mathcal{N}_G(U) = \{W \subseteq V \mid \exists v \in U : \ W \subseteq N_G(v)\}.$$

In particular $\mathcal{N}_G(V) = \mathcal{N}_G$. For a clique C, we can partition the local neighborhood $\mathcal{N}_G(C)$ by the following lemma into the sets

$$\mathcal{N}_G(A, C) = \{N \in \mathcal{N}_G(C) \mid N \cap P_G(C) \in \mathcal{P}_G(A, C)\}$$

for all anchor sets A of C.

Lemma 7. *For any clique C of the graph G, it holds*

$$\mathcal{N}_G(C) = \dot{\bigcup_{A \in \mathcal{A}_G(C)}} \mathcal{N}_G(A, C).$$

Proof. For every $N \in \mathcal{N}_G(C)$, there is an element as common intersection in C. Hence the common neighborhood of $N \cap P_G(C)$ inside C is non-empty. This common neighborhood is an anchor set A. Since these anchor sets differ for different families $\mathcal{N}_G(A, C)$, the union is disjoint. □

Lemma 7 is useful since we only have to determine the generating functions of $\mathcal{N}_G(A, C)$ for every $A \in \mathcal{A}_G(C)$. Adding these generating functions, we maintain the generating function of the local neighborhood $\mathcal{N}_G(C)$. In the next lemma, we derive a formula to compute the generating function of $\mathcal{N}_G(A, C)$ for every $A \in \mathcal{A}_G(C)$.

Lemma 8. *For a given anchor set $A \in \mathcal{A}_G(C)$ of a clique C, the generating function of $\mathcal{N}_G(A, C)$ is*

$$N_G(A, C, x) = P_G(A, C, x) \left((1 + x)^{|C|} - x^{|A|} (1 + x)^{|C| - |A|} \right).$$

Proof. We count the number of sets with respect to the cardinality in $\mathcal{N}_G(A, C)$. Every $M \in \mathcal{P}_G(A, C)$ is in $\mathcal{N}_G(A, C)$. Furthermore we can extend every M to a set N such that we have a common neighbor in C. Since we look at all $M \in \mathcal{P}_G(A, C)$, it is enough to look at extensions $N = M \cup X$, where X is a subset of C. In order to keep N in the local neighborhood $\mathcal{N}_G(C)$, we need a common neighbor in C. Since the common neighborhood of M inside C is the anchor set A, the common neighborhood of N must contain an element of A. Hence X cannot be the whole anchor set A. In particular, the possibilities to extend M are the elements of the family

$$\mathcal{X} = \{X \mid \exists a \in A : X \subseteq C \backslash \{a\}\}.$$

All sets in $\mathcal{X}$ consist of a disjoint union of a proper subset of A and a subset of $C\backslash A$. This leads to a generating function

$$\begin{aligned}&\left((1+x)^{|A|}-x^{|A|}\right)(1+x)^{|C|-|A|}\\&=(1+x)^{|C|}-x^{|A|}(1+x)^{|C|-|A|}\end{aligned}$$

of $\mathcal{X}$. The generating function of $\mathcal{P}_G(A,C)$, which counts the different possibilities of M is counted by $P_G(A,C,x)$. □

This leads us to the update formula similar to Proposition 3 adapted to attaching a vertex to a clique.

Corollary 9. *Let $G=(V,E)$ be a graph and C a clique in the graph. The neighborhood polynomial of $G_{C\triangleright v}$ with vertex set $V\cup\{v\}$ can be computed as follows:*

$$\begin{aligned}N_{G_{C\triangleright v}}(x)&=N_G(x)+\phi_G(C)\\&\quad+x\sum_{A\in\mathcal{A}_G(C)}P_G(A,C,x)\left((1+x)^{|C|}-x^{|A|}(1+x)^{|C|-|A|}\right),\end{aligned}$$

where

$$\phi_G(C)=\begin{cases}x^{|C|}, & \text{if } C \text{ is a maximal clique in } G;\\ 0, & \text{otherwise}.\end{cases}$$

Proof. Let $X\in\mathcal{N}_{G_{C\triangleright v}}$ be a neighborhood set in the graph $G_{C\triangleright v}$. We consider the following three cases:

- If $X\subseteq V$ but $X\nsubseteq N_G(v)$, then $X\in\mathcal{N}_G$ is in the neighborhood complex of G. These sets are counted in the first summand $N_G(x)$.
- Now let $X\subseteq V$ and $X\subseteq N_G(v)$. If X is a proper subset of C, it already has a common neighbor in G, hence it is already counted in the first summand. Similarly this holds if $X=C$ and C is not a maximal clique in G, i.e. C has a common neighbor in G. Thus the only case where a new neighborhood arises is if C is a maximal clique in G. In $G_{C\triangleright v}$ the common neighbor of C is v. This is counted in $\phi_G(C)$.
- Let us now consider the case $v\in X$, i.e. $X\nsubseteq V$. Since v is connected to all elements in C, we need to count all subsets $Y\subseteq V$ which have a common neighbor in C. This is equivalent to count the number of elements in $\mathcal{N}_G(C)$. Combining Lemma 7 and Lemma 8, we obtain

  $$\sum_{A\in\mathcal{A}_G(C)}P_G(A,C,x)\left((1+x)^{|C|}-x^{|A|}(1+x)^{|C|-|A|}\right)$$

  as a generating function of those subsets $Y\subseteq V$. In X there is just one additional element v. Hence we have to multiply the polynomial with x.

Since the above cases are disjoint, this leads to the formula of the neighborhood polynomial as stated. □

With this formula, we are able to compute the neighborhood polynomial after attaching a vertex v to a clique C in a graph. In order to compute the neighborhood polynomial of a chordal graph G, we need the perfect elimination order $v_1, \ldots, v_n$. If the chordal graph is connected, we add the vertices in reverse order, starting with v_n and then adding v_i to the corresponding clique in $G[v_{i+1}, \ldots, v_n]$. For attaching one vertex, the neighborhood polynomial can be computed with Corollary 9. If the chordal graph is not connected we use the same procedure explained above for every connected component and compute the neighborhood polynomial by adding the polynomials of the connected components as in Proposition 1.
In order to compute the formula, we need the anchor family of C and the corresponding periphery polynomials $P_G(A, C, x)$ for every $A \in \mathcal{A}_G(C)$. As we have seen in Lemma 6 it is enough to store these informations for the maximal cliques and compute them in every step for the required clique C.

In the next paragraph, we see how we update the anchor families and periphery polynomials for the maximal cliques after attaching a vertex in order to have the correct ones in the next step. The graph with attached vertex v is denoted by $G^+ = G_{C \triangleright v}$. We get a new maximal clique $C^+ = C \cup \{v\}$ which we have to add to the list of maximal cliques in the graph. The periphery of C^+ in G^+ is $P_{G^+}(C^+) = P_G(C)$ and the family of anchor sets $\mathcal{A}_{G^+}(C^+) = \mathcal{A}_G(C)$ with the same periphery polynomials as in G, i.e. $P_{G^+}(A, C, x) = P_G(A, C, x)$ for all $A \in \mathcal{A}_{G^+}(C^+)$.
The maximal cliques in G which have no intersection with C, do not change in G^+. Let C_{max} be a maximal clique in G with $C_{max} \cap C \neq \emptyset$. If $C = C_{max}$, we are done since this is not a maximal clique in G^+. So we assume $C \neq C_{max}$. The periphery of C_{max} in G^+ consists of the periphery of C_{max} in G together with the new element v, i.e.

$$P_{G^+}(C_{max}) = P_G(C) \cup \{v\}.$$

In the next step, we identify the anchor sets of C_{max} in G^+. Any anchor set of C_{max} in G remains an anchor set in G^+. Since the new vertex v is attached to the subset $C_{max} \cap C$ of the considered clique C_{max}, this subset $C_{max} \cap C \neq \emptyset$ is a new anchor set in G^+, if it was not already an anchor set in G. Furthermore all subsets of C_{max}, which occur as non-empty intersection of $C_{max} \cap C$ with an anchor set in $\mathcal{A}_G(C_{max})$ build an anchor set in G^+. Since the whole clique C_{max} is an anchor set in G, the intersection $C_{max} \cap C$ occurs as intersection $C_{max} \cap (C_{max} \cap C)$ with an anchor set of G. This shows

$$\begin{aligned}&\mathcal{A}_{G^+}(C_{max}) = \\ &\mathcal{A}_G(C_{max}) \cup \{A \cap (C_{max} \cap C) \mid A \cap (C_{max} \cap C) \neq \emptyset \text{ and } A \in \mathcal{A}_G(C_{max})\}.\end{aligned}$$

Now we determine the periphery polynomial $P_{G^+}(C_{max})$ for every anchor set $A \in \mathcal{A}_{G^+}(C_{max})$. Since $C \neq C_{max}$, the intersection $C_{max} \cap C$ is a proper subset of C_{max}. We consider the following three cases

- If A is a proper subset of $C \cap C_{max}$, all corresponding periphery sets in G are a corresponding periphery set in G^+ and we can add v to corresponding periphery set M in G, since the intersection with the neighborhood

$N_{G^+}(v) = C$ does not change the anchor set. In this case the periphery polynomial is

$$P_{G^+}(A, C_{max}, x) = (1+x)P_G(A, C_{max}, x).$$

- If $A = C \cap C_{max}$, the periphery sets in G with corresponding anchor set A which are counted in $P_G(A, C_{max}, x)$ still have the same anchor set in G^+. Furthermore v is a new periphery set with anchor set $A = C \cap C_{max}$ and all periphery sets which have a superset A' of A as corresponding anchor set, form together with v a periphery set with anchor set A. Hence the updated periphery polynomial is

$$P_{G^+}(A, C_{max}, x) = P_G(A, C_{max}, x) + x \left(1 + \sum_{A' \supseteq A, A' \in \mathcal{A}_G(C_{max})} P_G(A', C_{max}, x) \right). \quad (1)$$

- Otherwise A is not a subset of $C \cap C_{max}$, hence v is not in a periphery set with anchor set M. So the periphery polynomial stays the same, i.e.

$$P_{G^+}(A, C_{max}, x) = P_G(A, C_{max}, x).$$

This concludes the algorithm to compute the neighborhood polynomial of chordal graphs.
Recall that the anchor width of a graph G is the smallest k such that $|\mathcal{A}_G(C)| \le k$ for all cliques of C. The algorithm explained above leads to a polynomial time algorithm if the anchor width is polynomially bounded.

Proposition 10. *Let G be a chordal graph with n vertices with anchor width at most k. Computing the neighborhood polynomial takes at most $\mathcal{O}(n^3k + n^2k^2)$ time.*

Proof. Using lexicographic breadth-first search, we get a perfect elimination order of the chordal graph G in linear time (cf. [Gol80]). After attaching one vertex, there is one new maximal clique, containing v. We add this maximal clique to the list of all maximal cliques in the graph, possibly removing the neighborhood of v if it was in the list. This shows that we have at most n maximal cliques. Furthermore we have an ordering of the vertices given by the perfect elimination order which is the reverse order of attaching the vertices to the graph. Hence in order to compute an intersection or test a subset relation, we only need to compare the elements in the clique in the attached order. Both is possible in linear time.
To update the neighborhood polynomial after vertex attachment to a clique C, we first need to find a maximal clique, containing C and compute its anchor family and its periphery polynomials. In order to find this maximal clique we test for every maximal clique (at most n) whether C is a subset. This needs at most $\mathcal{O}(n^2)$ time. Computing the anchor family takes $\mathcal{O}(nk)$ and for one anchor set the periphery polynomial takes $\mathcal{O}(nk)$ and since we have at most k anchor sets in a clique, we need $\mathcal{O}(nk^2)$ for this step. Now using the update formula to compute the neighborhood polynomial is possible in $\mathcal{O}(k)$.

Now in the next step, we have to update the anchor families and periphery polynomials for all maximal cliques with non-empty intersection with C. We have at most n maximal cliques, where we need to update this information. Updating the anchor family for one maximal clique takes $\mathcal{O}(nk)$ time, see (1). Hence for all maximal cliques this takes at most $\mathcal{O}(n^2k)$. For one anchor set, updating the periphery polynomials takes at most $\mathcal{O}(kn)$ (see case (2)). In the other cases this is possible in linear time. And since case (2) only appears for one anchor set, this leads to a runtime of at most $\mathcal{O}(kn^2)$ for all maximal cliques. Since we add in total n vertices, one after another, this leads to a total runtime of $\mathcal{O}(n^3k + n^2k^2)$. □

4 Complexity of the Anchor Width

In this section, we will discuss some subclasses of chordal graphs with bounded anchor width.

4.1 Anchor Width of Split Graphs

The anchor width of split graphs is not polynomially bounded. Since split graphs are chordal, the algorithm introduced in Section 3 might take super-polynomial time.

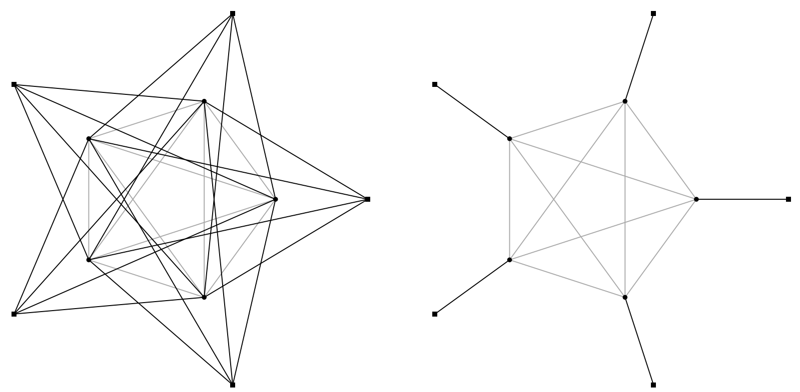

Figure 2: left: The split graph S_5 with anchor width 31 and right: The complement graph of S_5.

Proposition 11. *The anchor width of split graphs can grow exponentially.*

Proof. In order to prove this proposition, we construct an infinite family of split graphs S_m on $n = 2m$ vertices such that the anchor width is $2^{\frac{n}{2}}$.
We start with a clique $C = \{c_1, \dots, c_m\}$ of size m and attach vertices $p_1, \dots, p_m$ such that every p_i $(1 \leq i \leq m)$ is adjacent to c_j for all $j \neq i$, see Figure 2. This constructed graph is a split graph since C is a clique and $\{p_1, \dots p_m\}$ build an independent set. All vertices p_i are therefore in the periphery P_C of C, hence $|P_C| = m = \frac{n}{2}$. All non-empty subsets of C are an anchor set, hence the anchor width of this graph is $2^m - 1 = 2^{\frac{n}{2}} - 1$. □

4.2 Anchor Width of Chordal Comparability Graphs

In this section we introduce chordal comparability graphs and show that the anchor degree of this graph class is bounded. A graph $G = (V, E)$ is a *comparability graph* if there is a poset $(V, \prec)$ such that two vertices $u, v \in V$ are adjacent in G if and only if $u \prec v$.

Proposition 12. *The anchor width of a chordal comparability graph with n vertices is at most $2n$.*

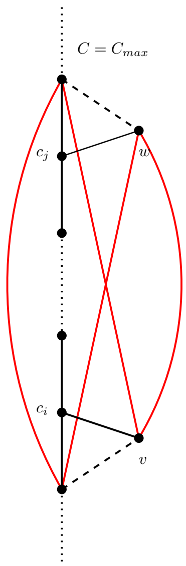

Figure 3: Hasse diagram of a poset corresponding to a comparability graph with induced C_4 which gives a contradiction in the proof of Proposition 12

.

Proof. Let G be a chordal comparability graph. Consider a maximal clique $C_{max} = \{c_1, \dots c_m\}$ in G. Since G is a comparability graph, it comes from a poset $(V, \prec)$. A clique in the graph corresponds to a chain in the poset. Hence the maximal clique C_{max} of size m corresponds to a maximal chain $c_1 \prec \cdots \prec c_m$ of length m in the poset. So whenever there is a vertex $v \notin C_{max}$, which is connected to $c_i \in C_{max}$ such that $v \prec c_i$ then v is connected to c_k for all $k \geq i$. Let i be the minimal element of the clique such that $v \prec c_i$. Since the clique is maximal, $i > 1$ and v is not comparable to c_{i-1}. Similar we get for every vertex $w \notin C_{max}$ which is connected to a vertex c_j of the clique with $w \succ c_j$ that w is connected to all elements c_k of the clique with $k \leq j$. Let c_j be the maximal element of the clique connected to w, then $j < m$ and c_{j+1} is not comparable to w. Hence an anchor set in C_{max} is a chain of the form $c_i \prec c_{i+1} \prec \dots \prec c_{j-1} \prec c_j$. Assume there is an anchor set with $1 < i < j < m$ and vertices v and w such that $v \prec c_i$ and $w \succ c_j$. Then v and w are connected by an edge since $w \succ c_j \succ c_i \succ v$ holds. And since w and c_{j+1} do not share an edge and analogously v and c_{i-1}, we get an induced cycle of length 4 which is not possible since the graph is chordal. In Figure 3 the poset is illustrated by its Hasse diagram and gives an illustration of the contradiction. This shows that all anchor sets of C_{max} are of the form $c_1 \prec \ldots \prec c_{j-1} \prec c_j$ for $j \leq m$ or $c_i \prec c_{i+1} \prec \ldots \prec c_m$ for $i \geq 1$. We have at most $2m - 1 \leq 2n$ possibilities for those sets. □

4.3 Leafage of Chordal Graphs

For a family of sets, the *intersection graph* is the graph consisting of one vertex for every set in the family. Two vertices are adjacent if and only if the corresponding sets have a non-empty intersection. The intersection graph of a family of intervals is an *interval graph* which is chordal. Equivalently an interval graph is an intersection graph of subtrees of a path. Chordal graphs are exactly the graphs which are an intersection graph of a family of subtrees of a *host tree* [Gav74]. We call a representation of a chordal graph by a family of subtrees a *subtree configuration*. Lin et al. [LMW98] introduced a parameter of a chordal graph, which measures how close a chordal graph is to an interval graph. The *leafage* $l(G)$ of a chordal graph G is defined as the minimal number of leaves of the host tree among all subtree configurations. We call a subtree representation *optimal* if it has the minimal number of leaves in the host tree. The interval graphs are exactly the chordal graphs with leafage at most 2. The split graphs S_m constructed in Section 4.1 have leafage m and the host tree is a star. In the following, we describe a subtree of a tree by the vertices, which induce the subtree.

We study the connection between the anchor width and the leafage of a chordal graph, and show that the anchor width of a chordal graph G is bounded from above by $n^{l(G)}$, where n is the number of vertices. In particular the anchor width of interval graphs is at most n^2.

Every pairwise intersecting family of subtrees has the *Helly property*, i.e. the intersection of all subtrees is non-empty [Gol80]. Hence there is at least one common vertex v_C in the host tree for every clique C of the chordal graph G.

Proposition 13. *For a chordal graph G with leafage $l = l(G)$ and n vertices, the anchor width is at most n^l.*

Proof. Let $C = C_{max}$ be a maximal clique in the graph G. We consider an optimal subtree representation of G in a host tree T. Let v_C be a vertex in the host tree which corresponds to the clique C. We may assume there is only one vertex in the host tree which is in the intersection of all subtrees corresponding to the clique C. If there is a non-trivial subtree in the common intersection, we can contract its edges without increasing the number of leaves. Since C is a maximal clique, there is no other subtree in the subtree representation containing v_C. From v_C there is a unique path in the host tree to all l leaves which we denote by $P_1, \ldots, P_l$.

For a periphery set $M \subseteq P_C$, the corresponding anchor set consists of those elements of the clique which have a non-empty intersection with all elements of M. For every $w \in M$, there is a tree T_w representing w in the subtree configuration. For every path P_i, we define a vertex v_i representing M on P_i as follows:

$$v_i \in \operatorname*{arg\,min}_{v \in T_{w_i} \cap P_i} dist(v, v_c),$$

where w_i is an element from the periphery such that

$$w_i \in \arg\max_{w \in M} \min_{v \in T_w \cap P_i} dist(v, v_c).$$

So for every $w \in M$ such that $T_w \cap P_i \neq \emptyset$, we choose the closest vertex v_w to v_C on the path P_i of the corresponding tree T_w. Among those vertices $\{v_w\}_w$, the vertex v_i is the vertex with maximal distance to v_C. If there is no subtree T_w of the periphery which has a non-empty intersection with the path P_i, we set $v_i = v_C$. Note that the v_i's are not necessarily distinct.
Now the anchor set $A = A_G(M, C)$ consists exactly of all subtrees of the clique, which contain all $v_1 \dots, v_l$ and v_C. If there is no such subtree corresponding to an element of the clique, there is no corresponding anchor set to M in C. The anchor set A is fully determined by the vertices v_i.
Gavril [Gav74] and Shibata [Shi88] showed that there are subtree representations such that every vertex of the host tree belongs to a maximal clique of the chordal graph. Hence we may assume that there are at most n vertices in the host tree. This implies that there are at most n choices for every v_i. In total we have at most n choices for every i, which are n^l choices for the tuple $(v_1, \dots v_l)$ and hence at most n^l different anchor sets. This shows the upper bound for the anchor width. □

Since interval graphs are the graphs with leafage at most 2, it follows:

Corollary 14. *The anchor width of interval graphs is at most n^2.*

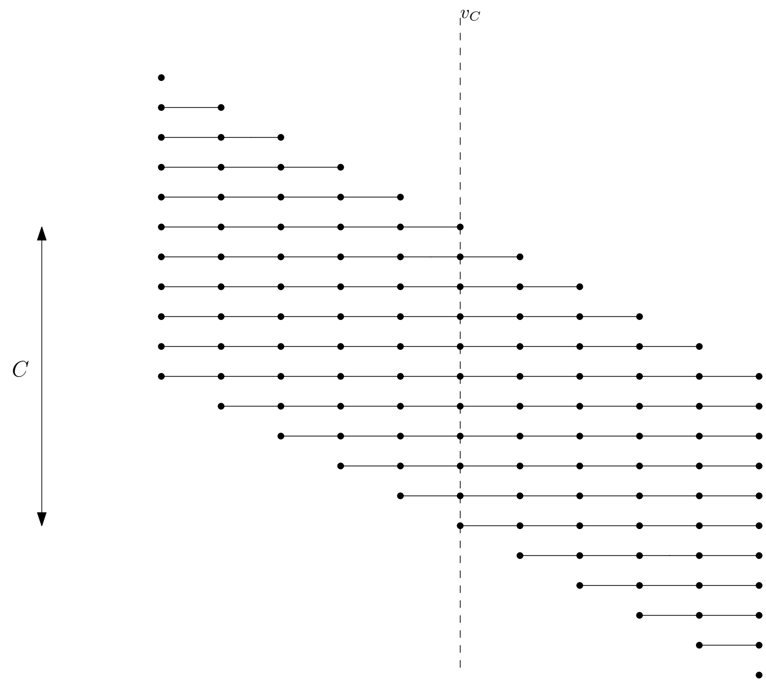

Figure 4: Construction of an interval graph with 21 vertices and a maximal clique of size 11 and 25 anchor sets.

Furthermore we can show that there is an infinite family of interval graphs on $n = 4m+1$ vertices with a clique of size $2m+1$ which has at least $m^2 = \left(\frac{(n-1)}{4}\right)^2$ different anchor sets. For the construction (see Figure 4), we take the path P_{2m+1} on $2m + 1$ vertices $v_{-m,\dots,v_0,\dots,v_m}$. The subtrees corresponding to the clique C are the $2m + 1$ paths on the vertices

$$\{v_{-m}, \dots, v_i\} \text{ for } i = 0, \dots, m \quad \text{and}$$
$$\{v_i, \dots, v_m\} \text{ for } i = -m, \dots, 0.$$

The common intersection v_C of the clique is the vertex v_0. Furthermore we define the following subpaths, which are in the periphery of C:

$$\{v_{-m}, \dots, v_i\} \text{ for } i = -m, \dots, -1 \quad \text{and}$$
$$\{v_i, \dots v_m\} \text{ for } i = 1, \dots m.$$

For an illustration of the construction see Figure 4. For any choice of $i \in \{-m \dots, -1\}$ and $j \in \{1, \dots, m\}$, we consider the two paths:

$$\{v_{-m}, \dots, v_i\} \quad \text{and} \quad \{v_j, \dots v_m\}$$

of the periphery. The anchor set corresponding to this two-element periphery set consists of all paths in the host tree corresponding to a clique vertex which contain v_i and v_j. For every choice of i and j these anchor sets differ. Hence there are at least m^2 anchor sets.

Acknowledgments

The authors thank Kolja Knauer and Manfred Scheucher for helpful discussions.

References

[AT18] M. Alipour and P. Tittmann. Graph operations and neighborhood polynomials. arXiv:1807.03971, 2018.

[Ber84] A. A. Bertossi. Dominating sets for split and bipartite graphs. *Information Processing Letters*, 19(1):37–40, 1984.

[BJ82] K. S. Booth and J. H. Johnson. Dominating sets in chordal graphs. *SIAM Journal on Computing*, 11(1):191–199, 1982.

[BN08] J. I. Brown and R. J. Nowakowski. The neighbourhood polynomial of a graph. *Australasian Journal of Combinatorics*, 42:55–68, 2008.

[CPS85] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[Day17] D. Day. *On the neighbourhood polynomial.* Master thesis, Dalhousie University, 2017.

[Gav74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[Gol80] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. 1980.

[HT18] I. Heinrich and P. Tittmann. Neighborhood and domination polynomials of graphs. *Graphs and Combinatorics*, 34:1203–1216, 2018.

[LMW98] I.-J. Lin, T. A. McKee, and D. B. West. The leafage of a chordal graph. *Discussiones Mathematicae Graph Theory*, 18(1):23–48, 1998.

[Lov78] L. Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

[Shi88] Y. Shibata. On the tree representation of chordal graphs. *Journal of Graph Theory*, 12(3):421–428, 1988.