Efficient Enumeration of Maximal k-Degenerate Induced Subgraphs of a Chordal Graph[☆]

Alessio Conte[a], Mamadou Moustapha Kanté[b,1], Yota Otachi[c], Takeaki Uno[a], Kunihiro Wasa[a]

[a] *National Institute of Informatics, Tokyo, Japan*
[b] *Université Clermont Auvergne, LIMOS, CNRS, Aubière, France*
[c] *Kumamoto University, Kumamoto, Japan*

Abstract

In this paper we consider the problem of listing the maximal k-degenerate induced subgraphs of a chordal graph, and propose an output-sensitive algorithm using delay $O(m \cdot \omega(G))$ for any n-vertex chordal graph with m edges, where $\omega(G) \leq n$ is the maximum size of a clique in G. Degeneracy is a well known sparsity measure, and k-degenerate subgraphs are a notion of sparse subgraphs, which generalizes other problems such as independent sets (0-degenerate subgraphs) and forests (1-degenerate subgraphs).

Many efficient enumeration algorithms are designed by solving the so-called *Extension problem*, which asks whether there exists a maximal solution containing a given set of nodes, but no node from a forbidden set. We show that solving this problem is NP-complete for maximal k-degenerate induced subgraphs, motivating the need for additional techniques.

Keywords: Graph enumeration; Graph algorithms; Polynomial delay; Degeneracy; Chordal graphs; Clique trees

1. Introduction

One of the fundamental problems in network analysis is finding subgraphs with some desired properties. A great body of literature has been devoted to develop efficient algorithms for many different types of subgraphs, such as frequent subgraphs [14], dense subgraphs [15] or complete subgraphs [5, 9]. A more comprehensive list can be found in [22].

Dense subgraphs are object of extensive research, especially due to their close relationship to community detection; however, one may be interested in finding *sparse* graphs as many networks are sparse even if locally dense. For instance, the paper [23] addresses the enumeration of induced trees in k-degenerate graphs.

The *degeneracy* of a graph is the smallest integer k for which every subgraph of the graph has a vertex of degree *at most* k. A graph is said to be k-degenerate if its degeneracy is k or less. Degeneracy is also referred to as the coloring number or k-core number, as a k-degenerate graph may contain a k-core but

[☆]This work was partially supported by JST CREST, Grant Number JPMJCR1401, Japan.
[*]Alessio Conte (Corresponding author)
Email addresses: conte@nii.ac.jp (Alessio Conte), mamadou.kante@uca.fr (Mamadou Moustapha Kanté), otachi@cs.kumamoto-u.ac.jp (Yota Otachi), uno@nii.ac.jp (Takeaki Uno), wasa@nii.ac.jp (Kunihiro Wasa)
[1]M.M. Kanté is supported by French Agency for Research under the GraphEN project (ANR-15-CE40-0009).

not a k+1-core, and is a widely used sparsity measure [5, 9, 17, 21, 23]. Several studies tend to take into account the degeneracy of graphs, as it tends to be very small in real-world networks [21], many important graph classes in structural graph theory are degenerate [17]. Furthermore, it is straightforward to see that k-degenrate subgraphs generalize well known structures, as 0-degenerate subgraphs correspond to independent sets, while 1-degenerate subgraphs correspond to induced forests.

Alon *et al.* [1] investigated the size of the largest k-degenerate induced subgraph in a graph, giving tight lower bounds in relation to the degree sequence of the graph. Whilst Pilipczuk *et al.* [18] showed that a maximum k-degenerate induced subgraph can be found in randomized time $O((2-\epsilon_k)^n n^{O(1)})$, for some $\epsilon_k > 0$ depending only on k, and moreover showed that there are at most $(2-\epsilon_k)^n$ such subgraphs. See [2, 16] for other recent studies on degeneracy.

In this paper we address the enumeration of all maximal k-degenerate induced subgraphs of a given graph, and provide an efficient polynomial delay algorithm for input chordal graphs. An enumeration algorithm is of polynomial delay if the maximum computation time between two consecutive outputs is bounded by a polynomial in the size of the input. Enumeration algorithms are of high importance in several areas such as data-mining, biology, artificial intelligence, or databases (see for instance [7, 22]).

Chordal graphs (also known as triangulated graphs) have been a topic of intensive study in computer science due to the applications in phylogenetic networks and also many NP-complete problems become tractable when the inputs are chordal graphs [3, 8, 13, 19, 20]. A graph is chordal if and only if every cycle of length 4 or more has a chord, i.e. an edge joining two non-consecutive vertices. Chordal graphs have been equivalently characterized in different ways: They are the graphs that allow a *perfect elimination ordering*, that is an elimination ordering in which every eliminated vertex is *simplicial* (its neighbors form a clique) [19, 20]; the graphs that allow a *clique tree* [3] (see Section 2.1); the intersection graphs of subtree families in trees [13]. In our case, we will consider the characterization by clique-trees. It is well-known that n-vertex chordal graphs have at most n maximal cliques. A clique-tree of a chordal graph G is a tree T whose nodes are in bijection with the set of maximal cliques, and such that for each vertex x the set of maximal cliques containing x form a subtree of T.

Our algorithm is based on the well-known *Extension Problem* (also known as *backtracking* or *flashlight* or *binary partition*) and uses the clique-tree. The enumeration can be reduced to the following question: Given two subsets of vertices S and X, decide whether there is a maximal k-degenerate induced subgraph which contains S and does not intersect X. Indeed, if we can answer this question in polynomial time, the algorithm can be summarized as follows: Start from the empty set, and in each iteration with given sets (S, X) pick a vertex v and partition the problem into those containing v (a call to the iteration $(S \cup \{v\}, X)$) or those not containing v (a call to the iteration $(S, X \cup \{v\})$, both calls depending on the answer given by the Extension problem. The delay of such algorithms is usually $O(n \cdot poly(n))$ with $poly(n)$ being the time to decide the Extension problem. As we will show in Section 3, however, this problem is NP-complete for generic graphs, and even for *split graphs*, which are a subset of chordal graphs. This motivates the investigation of

the problem, which is not solvable by trivial application of known techniques, and highlights the need for a deeper understanding of the problems' structure. On one hand, we want to define suitable constraint so that the Extension problem can be solved efficiently. On the other hand, if the constraints are too strong (say, requiring the excluded set X to be empty), the result will not be powerful enough to yield an exact algorithm. For our algorithm, thus, we do not consider all possible sets (S, X) for the Extension problem, but a sufficient amount of special cases driven by the clique-tree. Our special case of the Extension problem is the following (we consider the clique-tree T to be rooted):

Input. A node C of T, a partition (S, X) of the set of vertices in all the cliques preceding C in a pre-order traversal of T and a partition (S', X') of $C \setminus (S \cup X)$.

Output. Decide whether there is a maximal solution M such that $S \cup S' \subseteq V(M)$ and $(X \cup X') \cap V(M) = \emptyset$.

We propose a notion of *greedy solution* and show that this special case of the Extension problem is a Yes-instance if and only if a greedy solution exists; we also propose an $O(m)$-time algorithm to compute the greedy solution.

2. Preliminaries

An algorithm is said to be *output-polynomial* if the running time is bounded by a polynomial in the input and the output sizes. The delay is the maximum computation time between two consecutive outputs, pre-processing, and post-processing. If the delay is polynomial in the input size, the algorithm is called *polynomial delay*.

For two sets A and B we denote by $A \setminus B$ the set $\{x \in A \mid x \notin B\}$. Our graph terminology is standard, we refer to the book [6]. In this paper, we assume that graphs are simple, finite, loopless, and each graph is given with an arbitrary linear ordering of its vertices. Whenever we will say that a vertex is "smaller" or "bigger" than another, we mean that it occurs respectively earlier or later than the other in this ordering. We can further assume graphs to be connected as the solutions of a non-connected graph are obtained by combining those of its connected components. We use n and m to denote respectively the numbers of vertices and edges in the input graph. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. The subgraph of G induced by $X \subseteq V(G)$, denoted by $G[X]$, is the graph $(X, (X \times X) \cap E(G))$, and we write $G \setminus X$ to denote $G[V(G) \setminus X]$. For a vertex x of G we denote by $N_G(x)$ the set of neighbors of x, i.e., the set $\{y \in V(G) \mid xy \in E(G)\}$, and we let $N_G[x]$, the *closed neighborhood of* x, be $N_G(x) \cup \{x\}$; the degree of a vertex x, $d_G(x)$, is defined as the size of $N_G(x)$. In what follows, we usually omit the subscript G if it is clear from the context.

A tree is an acyclic connected graph. A *clique* of a graph G is a subset C of G that induces a complete graph, and a *maximal clique* is a clique C of G such that $C \cup \{x\}$ is not a clique for all $x \in V(G) \setminus C$. We denote by $\mathcal{Q}(G)$ the set of maximal cliques of G, and by $\omega(G)$ the maximum number of vertices in a clique in $\mathcal{Q}(G)$. For a vertex x, we denote by $\mathcal{Q}(G, x)$ the set of maximal cliques containing x.

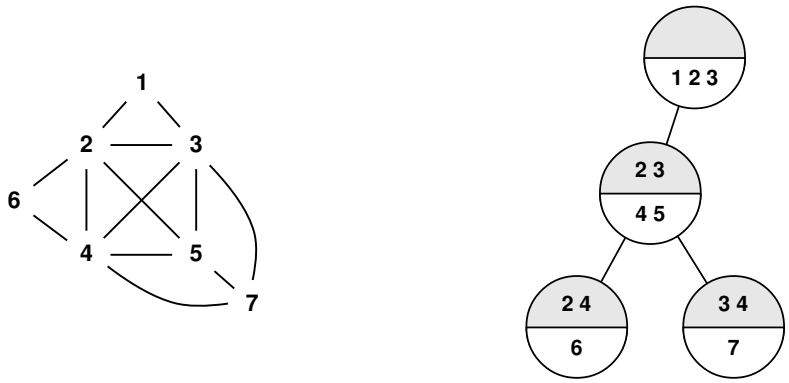

Figure 1: A chordal graph (left) and its clique tree (right) rooted in the clique $\{1, 2, 3\}$. Each node of the clique tree corresponds to a maximal clique of the graph, with its private vertices (i.e., those not in common with the parent clique) in the bottom half.

For a *rooted tree* T and two nodes u and v of T, we call v an *ancestor* of u, and u a *descendant* of v, if v is on the unique path from the root to u; u and v are *incomparable* if v is neither an ancestor or descendant of u.

2.1. Chordal Graphs and Clique Trees

A graph G is a *chordal graph* if it does not contain an induced cycle of length more than three. It is well-known that a chordal graph G has at most n maximal cliques, and they can be enumerated in linear time [4]. With every chordal graph G, one can associate a tree that we denote by $\mathcal{QT}(G)$, called a *clique tree*, whose nodes are the maximal cliques of G and such that for every vertex $x \in V(G)$ the set $\mathcal{Q}(G, x)$ is a subtree of $\mathcal{QT}(G)$ [13]. Moreover, for every chordal graph G, one can compute a clique tree in linear time (see for instance [11]). In the rest of the paper all clique trees are considered rooted. Furthermore, for any clique in the clique tree, we call *private* vertices the vertices of the clique which are *not* in common with its parent in the tree. A visual example is given in Figure 1.

In what follows, for a maximal clique C of a chordal graph G, C depending on the context, may refer to its set of vertices, the subgraph induced by C, or the corresponding node in the clique tree.

2.2. k-Degenerate Graphs

A graph G is a *k-degenerate graph* if for any induced subgraph H in G, H has a vertex whose degree is at most k. The degeneracy of a graph is the minimum value k for which the graph is k-degenerate, and is a well known sparsity measure [5, 9, 17, 21, 23]. We consider the following question.

Problem 1. *Given a chordal graph G and a positive integer k, enumerate all maximal k-degenerate induced subgraphs in G, with polynomial delay.*

Note that a complete graph K_n is an $(n-1)$-degenerate graph, as all its vertices have degree $n-1$. Therefore, for any clique C of a graph G, any k-degenerate induced subgraph of G may have no more than $k+1$ vertices belonging to C. Chordal graphs have the following property.

Theorem 1. *The degeneracy of a chordal graph is exactly* $\omega(G) - 1$.

Proof. Since the degeneracy is a hereditary property (i.e., any subgraph of a k-degenerate graph is k-degenerate), and the complete graph K_n has degeneracy $n-1$, $\omega(G)-1$ is a lower bound for the degeneracy of any graph. The fact that $\omega(G) - 1$ is an upper bound on chordal graphs relies on the fact that every chordal graph has at least a vertex whose neighbor is a clique [19]. Therefore, in any chordal graph (and all its subgraphs) we can find a vertex of degree at most $\omega(G) - 1$. □

We also remark that, in a given graph G, there is an essential difference between k-degenerate *induced* subgraphs (defined by subsets of the vertices), and k-degenerate *edge* subgraphs (defined by subsets of the edges): For example the 0-degenerate induced subgraphs of G are its independent sets, while any graph has a single 0-degenerate edge subgraph corresponding to an empty set of edges.[2] In this paper we focus on induced subgraphs, and leave open the problem of enumerating maximal k-degenerate edge subgraphs.

3. Hardness of the Extension Problem

In this section we consider the *Extension problem* for maximal k-degenerate induced subgraphs, which corresponds to answering the following question:

Problem 2 (Extension problem for maximal k-degenerate induced subgraphs)**.**
Input. *A graph G, and two sets of vertices $S \subseteq V(G)$ and $X \subseteq V(G)$ such that $S \cap X = \emptyset$ and $G[S]$ is k-degenerate.*
Output. ***Yes****, if there is a maximal k-degenerate induced subgraph M such that $S \subseteq V(M)$ and $X \cap V(M) = \emptyset$.* ***No****, otherwise.*

Assuming that $G[S]$ is k-degenerate is without loss of generality: Indeed if $G[S]$ is not k-degenerate then the answer to the problem is always negative, as any subgraph containing S may not have smaller degeneracy than $G[S]$.

For brevity we reuse the notation in Problem 2 in the remainder of the section, so as not to similarly redefine G, S, X and M multiple times. Furthermore, we use simply "Extension problem" as a shorthand for "Extension problem for maximal k-degenerate induced subgraphs".

As we mentioned in Section 1, if we could answer this question in polynomial time, say, $p(|V(G)|)$, then we could use this as a black box to create a binary partition algorithm with polynomial delay, as it is straightforward to see that its delay would be bounded by $O(|V(G)| \cdot p(|V(G)|))$.

In the following, however, we show that this problem is NP-complete, even when G is a *split graph*. A graph G is a split graph if its vertices can be partitioned into a clique and an independent set. It has long been known that this is equivalent to saying that both G and its complement $\bar{G}$ (i.e., the graph on the same vertices as G that has an edge iff the edge is not in G) are chordal [10].

[2] Indeed, any edge induces a subgraph which has degeneracy at least 1.

As split graphs are chordal, this hardness result also applies to chordal graphs, and, of course, to generic graphs.

3.1. Hardness Proof

In the following we show that, for arbitrary subsets S and X of vertices, the Extension problem is NP-complete even on split graphs.

In order to do so, we will use a polynomial-time reduction from an instance of the *Hitting Set* problem, whose definition we recall below, to that of solving the Extension problem on a split graph G.

Problem 3 (Hitting Set Problem)**.**
Input. *A set of elements U, called* universe*, an integer h, and a family $\mathcal{W}$ of subsets of U whose union equals U.*
Output. ***Yes****, if there is a subset $T \subseteq U$ of size at most h, such that for each $W \in \mathcal{W}$, $|W \cap T| \geq 1$.* ***No****, otherwise.*

The Hitting Set problem is a well known NP-complete problem, and is equivalent to *set cover*, one of Karp's 21 NP-complete problems. The problem is also reported in [12], which shows that it is still NP-complete even if $\mathcal{W}$ is constrained so that every $W \in \mathcal{W}$ has at most two elements.

Theorem 2. *Deciding whether the Extension problem for maximal k-degenerate induced subgraphs (Problem 2) has a positive answer is* NP*-complete, even if the considered graph G is a split graph.*

Proof. Firstly, the problem is in NP: The degeneracy of a graph can be found in linear time (see, *e.g.*, [5]), and as the degeneracy is hereditary, it means that if a k-degenerate subgraph M of G is not maximal, then there exists a node $x \in V(G) \setminus V(M)$ such that $G[V(M) \cup \{x\}]$ is k-degenerate. Thus verifying a positive instance of the problem takes polynomial time.

We now prove the problem's hardness by reducing an arbitrary instance of the *Hitting Set* problem to the Extension problem on a split graph. In the following we consider an instance $(U, \mathcal{W}, h)$ of the Hitting Set problem, referring to the notation in Problem 3. We assume $|U| > h$ without loss of generality, as otherwise the problem is trivial (U itself would be a solution).

Building the instance. Let us build a suitable instance of the Extension problem (G, S, X) for the given Hitting Set problem. Let V be a set of vertices, in bijection with $\mathcal{W}$ so that $W_v \in \mathcal{W}$ is the set corresponding to $v \in V$; let U be a set of vertices disjoint from V, corresponding to the universe of the given Hitting Set problem, and let v_x be a vertex disjoint from $U \cup V$. Finally, let G be a graph with vertex set $V \cup U \cup \{v_x\}$ whose edges are as follows: $U \cup \{v_x\}$ is a clique in G, V is an independent set in G, and all the other edges are of the form uv, for each $u \in U$ and $v \in V$ such that $u \notin W_v$. G is clearly a split graph, whose vertex set can be partitioned into the clique U and the independent set $I = V \cup \{v_x\}$, and its size is polynomial in $|U| + |\mathcal{W}|$, the size of the input Hitting Set problem. Let $S = V$, $X = \{v_x\}$ and $k = h - 1$. We claim that

the Extension problem with instance (G, S, X) has positive answer iff the Hitting Set problem has positive answer.

Equivalence. Let M be a maximal $(h-1)$-degenerate induced subgraph of G, with $S = V \subseteq V(M)$ and $X \cap V(M) = \emptyset$. Recalling Proposition 1, since $v_x \in I \cap X$, then $|V(M) \cap U| = h$, *i.e.*, M must contain exactly h vertices from U. Moreover, for each $v \in V = I \cap S$, $|N(v) \cap (V(M) \cap U)| < h$. Since, for $u \in U$ and $v \in V$, $uv \in E(G)$ only when $u \notin W_v$, we can conclude that $|(V(M) \cap U) \setminus N(v)| = |(V(M) \cap U) \cap W_v| \geq 1$, for all $W_v \in \mathcal{W}$, and thus the answer to the Hitting Set problem is yes, with $T = V(M) \cap U$. Conversely, let T be a solution to the Hitting Set problem. We will show that there exists a set T' such that $M = G[V \cup T']$ is a maximal $(h-1)$-degenerate induced subgraph of G.

We define T' as a set such that $T \subseteq T' \subset U$, and $|T| = h$. T' may be equal to T (*i.e.*, when $|T| = h$), and always exists since $|U| > h$ by assumption.

Since each set W_v intersects T, it also intersects T'. By the definition of the edge set of G, there is a vertex in T' (corresponding to a vertex in $W_v \cap T'$) that is not adjacent to v in G. As $|T'| = h$ it implies that $|N(v) \cap T'| < h$, meaning that v may not participate in a clique of size $h+1$ in $M = G[V \cup T']$. This means that M may not have a clique of size larger than h, so it is $(h-1)$-degenerate by Theorem 1 since it is chordal.

Furthermore, it is easy to see that M is maximal: Since U is a clique, for each $u \in (U \setminus T') \cup \{v_x\}$ we have $|N(u) \cap T'| = h$, *i.e.*, $\{u\} \cup T'$ is a clique of size $h+1$, thus $G[V(M) \cup \{u\}]$ is not $(h-1)$-degenerate. The statement follows. □

As mentioned above, this hardness result extends to chordal graphs, as split graphs are chordal, and of course to general graphs, motivating our interest in finding more refined techniques for the associated enumeration problem. Furthermore, this is a somewhat surprising result if compared to our proposed algorithm k`MIG`, as the algorithm itself exploits a restricted form of the Extension problem, which can be solved in polynomial time, but is still powerful enough to allow the algorithm to find all solutions. Let's first show in the next subsection a somewhat characterization of maximal k-degenerate graphs in split graphs, which yields a restricted form of the Extension problem that can be solved in polynomial time. This restricted form of the Extension problem allows a polynomial delay algorithm on split graphs, which is a special case of k`MIG` on split graphs.

3.2. Polynomiality of restricted cases

The following considering the case $X = \emptyset$ is folklore and informs that computing a solution is easy.

Lemma 1. *For any graph G, if $X = \emptyset$ then the answer to the Extension problem is always* yes.

Proof. As X is empty, M may not contain vertices of X, thus the question is simply whether there exists a maximal k-degenerate induced subgraph M which contains S. As mentioned in the proof of Theorem 1, any subgraph of a k-degenerate graph is itself k-degenerate. This means that M may be found in a greedy

fashion by initially setting $M = G[S]$, and adding arbitrary vertices to it for which M is still k-degenerate. When no such vertex can be found, then M is a maximal k-degenerate induced subgraph which contains all vertices in S but none of those in X (which is empty). □

More in general, the above lemma can be applied to the Extension problem of not just maximal k-degenerate subgraphs, but to that of maximal subgraphs for all hereditary properties.

We now restrict our attention to split graphs, in which we can prove the polynomiality of a less restricted case. Firstly, Proposition 1 shows some necessary condition for the Extension problem to have a positive answer on split graphs, which will be useful in the following.

Proposition 1. *For any split graph G, whose vertices are partitioned into a clique C and an independent set I, a maximal k-degenerate induced subgraph M of G satisfies $|N(x) \cap V(M)| = k+1$ for all $x \in I \setminus V(M)$.*

Proof. Assume there is a vertex $x \in I \setminus V(M)$ with $|N(x) \cap V(M)| < k+1$. Then $M' = G[V(M) \cup \{x\}]$ is k-degenerate: Indeed, the largest clique in M has size at most $k+1$, since x has degree at most k in M', and M' may not contain cliques larger than $k+1$ not involving x since $M = M' \setminus \{x\}$ is k-degenerate. Thus M' is k-degenerate by Theorem 1, which contradicts the maximality of M. □

We now show a second, less strict, case in which the Extension problem is still solvable in polynomial time.

Proposition 2. *Let G be a split graph, with vertices partitioned in a clique C and an independent set I. If $I \cap S = \emptyset$ and $I \cap X \neq \emptyset$, then the answer to the Extension problem can be found in polynomial time. Furthermore, if $S \cup X \subset C$, and $|C \setminus X| \geq k+1$ then the answer to the Extension problem is* yes *iff $|S| \leq k+1$.*

Proof. Firstly, recall that k-degenerate subgraphs may contain cliques with up to $k+1$ vertices, that induced subgraphs of chordal graphs are chordal, and by Theorem 1 that chordal graphs have degeneracy $\omega(G) - 1$. Thus a subgraph of G is k-degenerate if and only if its largest clique is of size at most $k+1$. Note that every vertex in I is simplicial (*i.e.*, its neighbors form a clique) as its neighbors are a subset of the clique C.

Case 1: $I \cap S = \emptyset$ and $I \cap X \neq \emptyset$. First notice that, if a k-degenerate induced subgraph M not intersecting X is maximal, then $|N(x) \cap V(M)| = k+1$ for any $x \in I \setminus V(M)$ by Proposition 1. Thus, we can conclude that "$|N(x) \cap V(M)| = k+1$ for each $x \in I \cap X$" is a *necessary* condition if $I \cap X$ is not empty, independently from S. We now show that this is also sufficient and a suitable M can be found in polynomial time, if any exists.

Let $C' = \bigcap_{x \in I \cap X} N(x)$, and note that $C' \subseteq C$. Recall also that a maximal k-degenerate induced subgraph of G can contain at most $k+1$ vertices from C' since C is a clique. Moreover, if M is a maximal k-degenerate induced subgraph of G, then $V(M) \cap C \subseteq C'$, otherwise there is a vertex $x \in I \cap X$ such that $|N(x) \cap (V(M) \cap C)| \leq k$, *i.e.*, $G[V(M) \cup \{x\}]$ is also k-degenerate, contradicting the maximality of M. Therefore, if $|C' \setminus X| < k+1$ or $(C \setminus C') \cap S \neq \emptyset$, then the answer to the Extension problem is *no*. Assuming $|C' \setminus X| \geq k+1$ and $(C \setminus C') \cap S = \emptyset$, let C_M be any arbitrary subset of $C' \setminus X$ of size $k+1$, and I_M be

the set of all vertices v in $I \setminus X$ such that $|N(v) \cap C_M| < k+1$, and let $M = G[C_M \cup I_M]$. Clearly, all the vertices w in $I \setminus I_M$ may not be added to M as it would form a clique of size $k+2$ with C_M. Hence, M is a maximal k-degenerate induced subgraph of G, which means the answer to the Extension problem is *yes*. Since constructing C', C_M and I_M can all be done in polynomial time, we are done.

Case 2: $S \cup X \subset C$, and $|C \setminus X| \geq k+1$. As $S \subseteq C$ is a clique, if $|S| > k+1$ then $G[S]$ is not k-degenerate and the answer to the Extension problem is clearly *no*. Otherwise, let C_M be a set of vertices such that $S \subseteq C_M \subseteq C \setminus X$ and $|C_M| = k+1$ (we know that a suitable C_M always exists from the assumption that $|C \setminus X| \geq k+1$). Furthermore, let I_M be the set of all vertices $v \in I$ such that $|N(v) \cap C_M| < k+1$. We claim that $M := G[C_M \cup I_M]$ is a maximal k-degenerate induced subgraph of G. Indeed, as $|C_M| = k+1$, no vertex from C can be added to M, otherwise M would have a clique of size $k+2$. Also, any vertex w in $I \setminus I_M$ is such that $C_M \subseteq N(w)$, *i.e.*, $G[C_M \cup \{w\}]$ is a clique of size $k+2$. Thus M is a maximal k-degenerate induced subgraph of G and the answer to the Extension problem is *yes*. □

Let G be a split graph with its vertex set partitioned into a clique C and an independent set I. By Propositions 1 and 2, one easily checks that if M is a maximal k-degenerate induced subgraph of G, then $k \leq |M \cap C| \leq k+1$, and

- if $|M \cap C| = k+1$, then $I \setminus M = \{x \in I \mid M \cap C \subseteq N_G(x)\}$,
- if $|M \cap C| = k$, then $I \subset M$, and for all $x \in C \setminus M$, there is $v \in I$ such that $C \cup \{x\} \subseteq N_G(x)$.

We can therefore in G restrict the Extension problem to instances (S, X) with $S \cup X$ being a partition of C. By Proposition 2 this restricted Extension problem can be solved in polynomial time, and yields a polynomial delay algorithm as explained in Section 1. Our algorithm k`MIG` for chordal graphs is an extension of this algorithm, but requires a more refined characterization of maximal k-generate induced subgraphs.

4. Enumeration algorithm

This section describes our algorithm for enumerating all maximal k-degenerate induced subgraphs of a given chordal graph $G = (V, E)$. In the following, we sometimes refer to maximal k-degenerate induced subgraphs as *solutions*, and we denote them by their vertex set as for cliques, to ease the reading.

Our proposed algorithm is based on the binary partition method. The outline of our algorithm is as follows. We start with an empty induced subgraph S. Then we pick a vertex v from G and add v to S. If $S + v$ is a maximal k-degenerate induced subgraph, then we output $S + v$, otherwise we choose another vertex and add it to $S + v$. After that we backtrack and add v to an excluded set X, to generate all solutions that contain S and not v. By recursively applying the above operation to G we can enumerate all solutions. However, certain pairs (S, X) may not generate a solution, as there may be no maximal k-degenerate induced subgraph containing S but no vertex in X (*e.g.*, if $S = \emptyset, X = V$). If we test all the possibilities that will not lead to a solution, the cost of this process is not output sensitive, *i.e.*, not bounded by a polynomial in

the number of solutions. To develop an efficient enumeration algorithm, we have to limit such redundant testing as much as possible by answering instances of the Extension problem. But, this later problem is NP-complete. To overcome this difficulty, we focus on the rooted clique tree $\mathcal{QT}(G)$ to restrict the instances of the Extension problem to tractable cases. For doing so, we introduce the concepts of *greedy filling* and *partial solution*. In what follows we let G be a fixed chordal graph.

4.1. Greedy filling strategy

Let R be a fixed maximal clique, called the *root* of $\mathcal{QT}(G)$, and let us root $\mathcal{QT}(G)$ at R. For a maximal clique C of G, whose parent in $\mathcal{QT}(G)$ is the clique P, we call ***private** vertices of* C, denoted by $Pv(C)$, the set of vertices in $C \setminus P$.[3] Because all cliques in $\mathcal{QT}(G)$ are different and inclusion-maximal, and by the properties of the clique tree, one can deduce the following.

Lemma 2. *Given a clique tree $\mathcal{QT}(G)$, every clique in $\mathcal{QT}(G)$ contains at least one private vertex, and every vertex v is private in exactly one clique in $\mathcal{QT}(G)$.*

Let C be a maximal clique of G. For $X \subseteq V(G)$, let $A(C, X) = 1$ if $|C \setminus X| \geq k+1$, and $A(C, X) = 0$ otherwise. For any vertex $v \in X$, let $A(v, X) = \sum_{C \in \mathcal{Q}(G,v)} A(C, X)$, *i.e.*, the number of maximal cliques containing v for which $|C \setminus X| \geq k+1$. As adding more than $k+1$ vertices from the same clique to any solution M would cause M to not be k-degenerate anymore, we say that C is ***saturated** in* M if $|C \cap V(M)| = k+1$.

The function A allows us to check the maximality of a k-degenerate induced subgraph, thanks to the following lemma.

Lemma 3. *Let $G = (V, E)$ be a chordal graph and M be a k-degenerate induced subgraph of G, with $X = V \setminus V(M)$. Then, M is maximal if and only if $A(x, X) \geq 1$ for each $x \in X$.*

Proof. Assume that $A(x, X) \geq 1$ for each $x \in X$, and there exists a k-degenerate induced subgraph M' with $V(M') \supset V(M)$; let $v \in V(M') \setminus V(M)$. As $v \in X$ we have $A(v, X) \geq 1$, thus there exists a clique C containing v s.t. $|C \setminus X| \geq k+1$. As $V(M) = V \setminus X$, we have $|C \setminus X| = |C \cap V(M)| \geq k+1$. As $V(M) \cup \{v\} \subseteq V(M')$ we have $|C \cap V(M')| \geq k+2$, thus M' contains a complete subgraph with $k+2$ vertices and is not k-degenerate, which contradicts the hypothesis.

On the other hand, if for a vertex $x \in X$ we have $A(x, X) = 0$, then for any clique C containing x we have $|(V(M) \cup \{x\}) \cap C| \leq k+1$, since $|C \setminus X| = |C \cap V(M)| < k+1$. Thus the largest clique in $G[V(M) \cup \{x\}]$ has size at most $k+1$, and as $V(M) \cup \{x\}$ is a chordal graph (it is an induced subgraph of G) it is k-degenerate by Theorem 1. Thus M is not maximal, which contradicts the hypothesis. □

We now define the notion of *partial solution* as a pair of disjoint vertex subsets (S, X), where S contains vertices (to include) in the k-degenerate induced subgraph, and X is a set of vertices that must be excluded from the solution, with some additional properties:

[3] As the root R of the clique tree has no parent, we consider all its vertices private in R.

Definition 1 (Partial solution)**.** *A pair (S, X) of subsets of $V(G)$ with $S \cap X = \emptyset$ is a* partial solution *if*

1. *$|S \cap C| \leq k + 1$ for any maximal clique C,*
2. *$A(x, X) \geq 1$ for each $x \in X$, and*
3. *for each maximal clique C, if $Pv(C) \cap (S \cup X) \neq \emptyset$, then $C' \subseteq S \cup X$ for all ancestors C' of C.*

Given a pair (S, X) of disjoint subsets of $V(G)$, it is not trivial to decide whether there exists a solution M, with $V(M) \supseteq S$ and $V(M) \cap X = \emptyset$ (see Section 3.1). However, as we will later demonstrate, this is always true if (S, X) is a partial solution. Next, we introduce the strategy that will be used by our algorithm to guarantee the existence of solutions. Let $\pi : \{1, \ldots, |\mathcal{Q}(G)|\} \to \mathcal{Q}(G)$ be a fixed linear ordering of $\mathcal{Q}(G)$ obtained from a pre-order traversal of $\mathcal{QT}(G)$, and let us call $\pi^{-1}(C)$ the *rank* of $C \in \mathcal{Q}(G)$. We use the rank of the cliques to define the order in which they are considered by the following procedure.

Definition 2 (Greedy filling)**.** *The* greedy filling *of a partial solution (S, X) consists in the following. Let C be the maximal clique with the smallest* rank *for which $C \setminus (S \cup X) \neq \emptyset$. Add vertices one by one from C to S until C is saturated for S or $C \setminus (S \cup X) = \emptyset$. Then add the remaining vertices in $C \setminus (S \cup X)$ to X, if any, and repeat the process until no such clique C exists.*

Finally, we can now show that a partial solution can always be extended into a maximal one by means of a greedy filling.

Lemma 4. *For any partial solution (S, X), the* greedy filling *yields a maximal k-degenerate induced subgraph M of G such that $S \subseteq V(M)$ and $V(M) \cap X = \emptyset$.*

Proof. Let M be the greedy filling of (S, X). By definition, $S \subseteq V(M)$ and $X \cap V(M) = \emptyset$.

We prove the statement by showing that at all times during the greedy filling (S, X) maintains the property of being a partial solution (see Definition 1), so in the end we have $A(x, V \setminus V(M)) \geq 1$ for each $x \in V \setminus V(M)$, making M a maximal k-degenerate induced subgraph by Lemma 3. Let Q be the maximal clique of the smallest *rank* for which $Q \setminus (S \cup X) \neq \emptyset$. Let (S', X') be the new pair constructed from Q by the greedy filling, and let (S_Q, X_Q) be the partition of $Q \setminus (S \cup X)$ such that $S' = S \cup S_Q$ and $X' = X \cup X_Q$. First notice that for all the ancestors Q' of Q we have $Q' \setminus (S \cup X) = \emptyset$ as their rank is smaller than the one of Q.

By definition of greedy filling, $|Q \cap S'| = |(Q \cap S) \cup S_Q| \leq k + 1$. If $X_Q = \emptyset$, then $X' = X$ and $A(x, X') = A(x, X) \geq 1$ for each $x \in X'$. Otherwise, by definition of greedy filling, Q is saturated in S' ($|Q \cap S'| = k+1$). Hence, $A(Q, X') = 1$, and for each $x \in Q$ $A(x, X') \geq 1$, while for each $x \in X' \setminus Q = X \setminus Q$ $A(x, X') = A(x, X) \geq 1$. Thus, (S', X') is a partial solution, which completes the proof. □

4.2. Binary partition method

We are now ready to describe our algorithm $k\texttt{MIG}(G, k)$, whose pseudo-code is given in Algorithm 1.

Algorithm 1: kMIG: Enumerating all maximal k-degenerate induced subgraphs in a chordal graph $G = (V, E)$

```
1  Procedure kMIG(G, k)
2      Compute 𝒬𝒯(G) of G;
3      R ← the root clique of 𝒬𝒯(G);
4      π : {1, ... |𝒬(G)|} → 𝒬(G) be a pre-order traversal of 𝒬𝒯(G);
5      Call SubkMIG(G, R, ∅, ∅, k);
6  Procedure SubkMIG(G, Q, S, X, k)
7      if V = S ∪ X then
8          Output S;
9      if Q \ (S ∪ X) ≠ ∅ then
10         v ← the smallest vertex in Q \ (S ∪ X);
11         if |Q ∩ S| < k + 1 then
12             SubkMIG(G, Q, S ∪ {v}, X, k);
13         if there exists a solution S* s.t. S ⊆ S* ∧ S* ∩ (X ∪ {v}) = ∅ then
14             SubkMIG(G, Q, S, X ∪ {v}, k);
15     else
16         Q' ← π(π^{-1}(Q) + 1);
17         SubkMIG(G, Q', S, X, k);
```

The principle is to start from the partial solution $S = \emptyset, X = \emptyset$, where S represent the vertices that will be in the solution, and X the vertices that are excluded from the solution, and proceed with binary partition: In each recursive call we consider a vertex $v \in Q$, initially from the clique Q with the smallest rank, *i.e.*, the root of $\mathcal{QT}(G)$; we will first add v to S and find all the solutions containing $S \cup \{v\}$ and nothing in X; then add v to X and find all the solutions containing S and nothing in $X \cup \{v\}$, if any exists. At any step, we keep the invariant that (S, X) is a partial solution: If we add v to S (Line 12), this is equivalent to performing a step of the greedy filling, thus we know that $(S \cup \{v\}, X)$ is still a partial solution (see proof of Lemma 4). When, on the other hand, we try to add v to X (Line 14), we only explore this road if there exists a solution that contains all the vertices in S and no vertex in $X \cup \{v\}$. Thanks to (S, X) being a partial solution we will be able to discover this efficiently, and we will demonstrate (Lemma 5 in Section 4.3) that this is true if and only if $(S, X \cup \{v\})$ is still a partial solution. Only once $Q \setminus (S \cup X)$ is empty, we then proceed to the clique Q' next in the ranking (Lines 16-17). This guarantees that Q is always the clique of smallest rank such that $Q \setminus (S \cup X) \neq \emptyset$, thus Condition 3 of Definition 1 still holds, and so (S, X) is still a partial solution. It is important to remark that, as all ancestors of Q are fully contained in $S \cup X$, and $v \notin S \cup X$, then v is always a *private vertex* of Q, not contained in the ancestors of Q.

Finally, if $S \cup X = V$ we can output S as a solution: By keeping the invariant that (S, X) is a partial solution, we know by Lemma 3 that S is a maximal k-degenerate induced subgraph of G.

4.3. Correctness

In this section we show the following theorem, that is the correctness of our algorithm.

Theorem 3. *Let G be a chordal graph and k be a non-negative integer. Then $k\texttt{MIG}(G,k)$ outputs all and only maximal k-degenerate induced subgraphs of G without duplicates.*

As mentioned in the description, $k\texttt{MIG}(G,k)$ uses binary partition, thus every recursive call has either a single child (Line 17) which will simply extend the current solution, or will produce two recursive calls (Lines 12 and 14) that will lead to different solutions, as the first one considers only solutions for which $v \in S$, and the second only solutions for which $v \notin S$ (if any). Thus the same solution cannot be found more than once.

Furthermore, as we keep the invariant that (S, X) is a partial solution, by Lemma 3 we know that when $V = S \cup X$ then S is a maximal k-degenerate induced subgraph, thus $k\texttt{MIG}(G,k)$ outputs only solutions.

Finally, any solution, *i.e.*, maximal k-degenerate induced subgraph M is found by the algorithm, and we can prove this by induction: Consider the set of cliques $Q_1, Q_2, \ldots$ in $\mathcal{QT}(G)$, ordered by ranking. As a base condition, assume that (S, X) is a partial solution such that $S \subseteq V(M), X \cap V(M) = \emptyset$; this is always true in the beginning, when $(S = \emptyset, X = \emptyset)$. Let Q_i be the clique that we are considering, *i.e.*, the one of smallest rank such that $Q_i \setminus (S \cup X) \neq \emptyset$, and v be the smallest vertex[4] in $Q_i \setminus (S \cup X)$. If $v \in V(M)$, then the recursive call in Line 12 will consider a partial solution which has one more vertex in common with M, *i.e.*, $(S \cup \{v\}, X)$. Otherwise, $v \notin V(M)$, that is, there exists a solution S^* such that $S \subseteq S^*$ and $S^* \cap (X \cup \{v\}) = \emptyset$, thus the recursive call in Line 14 is executed; this recursive call will consider a partial solution that has one more vertex in common with $V \setminus V(M)$, *i.e.*, $(S, X \cup \{v\})$. In both cases the base condition is still true, thus by induction $k\texttt{MIG}(G,k)$ will find M. In order to prove Theorem 3, it only remains to show how to decide whether, given (S, X), there is a solution containing S but nothing in $X \cup \{v\}$, *i.e.*, how to compute Line 13. This is shown in the following lemma.

Lemma 5. *Let (S, X) be any partial solution of G, Q a clique such that its ancestor cliques are fully contained in $S \cup X$, and $v \notin S \cup X$ a private vertex of Q. Then, there exists a solution S^* such that $S \subseteq S^*$ and $S^* \cap (X \cup \{v\}) = \emptyset$, if and only if $A(x, X \cup \{v\}) \geq 1$ for each vertex $x \in N[v] \cap (X \cup \{v\})$.*

Proof. Let $X' = X \cup \{v\}$. If for each vertex $x \in N[v] \cap X'$, $A(x, X') \geq 1$, then (S, X') still satisfies all the properties in Definition 1, as $A(w, X)$ is unchanged for any vertex $w \in X \setminus N(v)$. Thus (S, X') is a partial solution, and a solution S^* is given by Lemma 4.

Suppose that there is a vertex $x \in X'$ such that $A(x, X') = 0$, *i.e.*, there is no clique Q containing x such that $|Q \setminus X'| \geq k+1$. As $X' \subseteq V \setminus S^*$ for any solution S^* disjoint from X', there is no clique Q containing x such that $|Q \setminus (V \setminus S^*)| \geq k+1$, thus $A(x, V \setminus S^*) = 0$, and there is no maximal solution S^* by Lemma 3. □

Thus Theorem 3 is true, and $k\texttt{MIG}(G,k)$ finds all and only maximal k-degenerate induced subgraphs of the chordal graph G exactly once.

[4] According to the arbitrary linear ordering associated to G.

5. Complexity Analysis

In this section we analyze the cost of our algorithm, and prove that it can enumerate all maximal k-degenerate induced subgraphs of G in $O(m \cdot \omega(G))$ time per solution. First, we recall some important properties of cliques in chordal graphs.

Remark 1 (From [3] and [11])**.** *Let G be a connected chordal graph with $n > 1$ vertices and m edges. Then the number of maximal cliques in G is at most $n - 1$, and the sum of their sizes is $\sum_{C \in \mathcal{Q}(G)} |C| = O(m)$.*

And regarding the cliques in G containing a specific node, we can state the following.

Lemma 6. *In a chordal graph G, the number of maximal cliques containing a vertex v is at most $|N(v)|$.*

Proof. Consider $G[N[v]]$, the subgraph of G induced by vertices of $N[v]$. $G[N[v]]$ is chordal as it is an induced subgraph of a chordal graph, it has $|N[v]|$ vertices, and at most $|N[v]| - 1 = |N(v)|$ maximal cliques, which exactly correspond to the maximal cliques in G containing v. □

Now, consider the cost of executing Line 13, which dominates the cost of each iteration of the algorithm. We show in the next lemma that it can be done efficiently by exploiting Lemma 5. We recall that $\omega(G)$ denotes the maximum size of a clique in G.

Lemma 7. *Line 13 can be executed in time $O(\omega(G) \cdot |N(v)|)$.*

Proof. By Lemma 5 it is sufficient to check, for every vertex $x \in N[v]$, whether there is a clique Q' containing x such that $|Q' \setminus (X \cup \{v\})| \geq k+1$. As (S, X) is a partial solution, if a vertex x is not contained in any clique such that $|Q' \setminus (X \cup \{v\})| \geq k+1$, then there exists a clique Q' such that $|Q' \setminus X| \geq k+1 > |Q' \setminus (X \cup \{v\})| = k$, and thus x is contained in one of the cliques containing v.

Assume we have a table that keeps track of the value $B(Q) = |Q \setminus (X \cup \{v\})|$ for every clique Q, and one that keeps the value $A(x) = |\{Q \mid x \in Q \text{ and } B(Q) \geq k+1\}|$. When adding v to X, we can update the B table by decrementing $B(Q)$ by 1 for every clique containing v. The number of such cliques in a chordal graph is at most $|N(v)|$ by Lemma 6. Every time the value of $B(Q)$ is decremented to less than $k+1$, we can update the A table by decrementing $A(x)$ by 1 for each vertex x in Q. During this process, the check fails if and only if $A(x)$ is decremented to 0 for any x. The time required is $|Q| \leq \omega(G)$ for each considered clique, for a total cost of $O(\omega(G) \cdot |N(v)|)$. □

Finally, we are ready to prove the complexity bound for $k\texttt{MIG}(G, k)$.

Theorem 4. *$k\texttt{MIG}(G, k)$ runs with delay $O(m \cdot \omega(G))$.*

Proof. First, we need to compute $\mathcal{QT}(G)$, which takes $O(n+m)$ time [11]. Note that $O(m+n) = O(m)$ as G is connected. Computing a pre-order traversal of $\mathcal{QT}(G)$ takes $O(n)$ time as $\mathcal{QT}(G)$ has at most n nodes.

In each recursive call we add a vertex either in S or in X or consider a next maximal clique. Hence, the depth of the tree of recursive calls is bounded by $2n$. To bound the delay between two solutions M and M',

it is enough to bound the sum of the cost of all recursive calls in the path from the recursive call outputting M to the one that outputs M'. For clarity, let us use the term *recursive node* to refer a node in the tree of the recursive calls. Note that the recursive nodes that output a solution are exactly the leaves of this tree, thus the path between M and M' is bounded by the sum of the cost of a root-to-leaf and a leaf-to-root path.

As to execute Line 13 we use tables A and B (see Lemma 7), let us explain how to initialise them (we already explain in Lemma 7 how to update them). For each vertex x, we set $A(x) = |\{Q \in \mathcal{Q}(G, x) \mid |Q| \geq k+1\}|$, and set $B(Q) = |Q|$ for each $Q \in \mathcal{Q}(G)$. In order to set these values we can simply iterate over all maximal cliques in $\mathcal{QT}(G)$: Initializing $B(Q)$ takes $O(1)$ time, and if $|Q| \geq k+1$ we increment $A(x)$ by 1 for each $x \in Q$, which takes $O(|Q|)$ time. The total running time for initializing the tables A and B take thus $O(n+m) = O(m)$ time (see Remark 1).

Let $v_1, \dots v_t$ be the recursive nodes in the path from the root to the node that outputs M'. First, $t \leq 2n$ as in each step either we add v to S or to X or we take another Q. The delay now is the sum of the cost of each v_i. Lines 9-14 can be done in time $O(|N(x)| \cdot \omega(G))$ by Lemma 7. The cost for Lines 16-17 is $O(1)$. By summing, we have the upper bound $\sum_{Q \in \mathcal{Q}(G)} O(1) + \sum_{x \in V(G)} O(|N(x)| \cdot \omega(G)) = O(m \cdot \omega(G))$. The $O(m)$ preprocessing cost is negligible as there always exists at least one solution. □

Note that this holds for any value of k: Indeed, by Theorem 1 we know that chordal graphs are $\omega(G)-1$-degenerate, thus for any $k \geq \omega(G)$, the problem is trivial as the only maximal solution is G itself.

6. Conclusions

We presented the first output-polynomial algorithm for enumerating maximal k-degenerate induced subgraphs in a chordal graph. The algorithm runs in $O(m \cdot \omega(G))$ time per solution for any given k. While the enumeration problem on general graphs seems challenging, partially due to the proven hardness of the Extension problem, special cases seem to be solvable: For example, listing maximal 0-degenerate induced subgraphs corresponds to listing maximal independent sets, for which output-polynomial algorithms are known. It would thus be interesting for future work to investigate the feasibility of an output-polynomial algorithm in a more general setting. Another interesting problem which remains open is the enumeration of maximal k-degenerate *edge* subgraphs, which is in essence a different problem.

References

[1] Alon, N., Kahn, J., Seymour, P. D., 1987. Large induced degenerate subgraphs. Graphs and Combinatorics 3 (1), 203–211.

[2] Bauer, R., Krug, M., Wagner, D., 2010. Enumerating and generating labeled k-degenerate graphs. In: 2010 Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics. SIAM, Philadelphia, PA, pp. 90–98.

[3] Blair, J. R., Peyton, B., 1993. An introduction to chordal graphs and clique trees. In: Graph theory and sparse matrix computation. Springer, pp. 1–29.

[4] Chandran, L. S., 2001. A linear time algorithm for enumerating all the minimum and minimal separators of a chordal graph. In: Wang, J. (Ed.), Proceedings of the 7th Annual International Computing and Combinatorics Conference, COCOON. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 308–317.

[5] Conte, A., Grossi, R., Marino, A., Versari, L., 2016. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy. pp. 148:1–148:15.

[6] Diestel, R., 2005. Graph Theory (Graduate Texts in Mathematics). Springer.

[7] Eiter, T., Makino, K., Gottlob, G., 2008. Computational aspects of monotone dualization: a brief survey. Discrete Applied Mathematics 156 (11), 2035–2049.

[8] Enright, J., Kondrak, G., 2011. The application of chordal graphs to inferring phylogenetic trees of languages. In: Fifth International Joint Conference on Natural Language Processing, IJCNLP. pp. 545–552.

[9] Eppstein, D., Löffler, M., Strash, D., 2013. Listing all maximal cliques in large sparse real-world graphs. ACM Journal of Experimental Algorithmics 18.

[10] Foldes, S., Hammer, P. L., 1977. Split graphs having dilworth number two. Canadian Journal of Mathematics 29 (3), 666–672.

[11] Galinier, P., Habib, M., Paul, C., 1995. Chordal graphs and their clique graphs. In: Nagl, M. (Ed.), Graph-Theoretic Concepts in Computer Science: 21st International Workshop, WG '95 Aachen, Germany, June 20–22, 1995 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 358–371.

[12] Garey, M. R., Johnson, D. S., 1990. Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA.

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

[14] Kuramochi, M., Karypis, G., 2001. Frequent subgraph discovery. In: Proceedings of the 2001 IEEE International Conference on Data Mining. ICDM '01. IEEE Computer Society, Washington, DC, USA, pp. 313–320.

[15] Lee, V. E., Ruan, N., Jin, R., Aggarwal, C., 2010. A survey of algorithms for dense subgraph discovery. In: Managing and Mining Graph Data. Springer, pp. 303–336.

[16] Lukoťka, R., Mazák, J., Zhu, X., 2015. Maximum 4-degenerate subgraph of a planar graph. The Electronic Journal of Combinatorics 22 (1), P1–11.

[17] Nešetřil, J., Ossona de Mendez, P., 2012. Sparsity. Vol. 28 of Algorithms and Combinatorics. Springer, Heidelberg, graphs, structures, and algorithms.

[18] Pilipczuk, M., Pilipczuk, M., 2012. Finding a maximum induced degenerate subgraph faster than 2^n. In: International Symposium on Parameterized and Exact Computation. Springer, pp. 3–12.

[19] Rose, D. J., Tarjan, R. E., Lueker, G. S., 1976. Algorithmic aspects of vertex elimination on graphs. SIAM Journal on computing 5 (2), 266–283.

[20] Tarjan, R. E., Yannakakis, M., 1984. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal on computing 13 (3), 566–579.

[21] Ugander, J., Karrer, B., Backstrom, L., Marlow, C., 2011. The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503.

[22] Wasa, K., 2016. Enumeration of enumeration algorithms. arXiv preprint arXiv:1605.05102.

[23] Wasa, K., Arimura, H., Uno, T., 2014. Efficient enumeration of induced subtrees in a k-degenerate graph. In: International Symposium on Algorithms and Computation, ISAAC. Springer, pp. 94–102.