Efficient Enumeration of Maximal k-Degenerate Subgraphs in a Chordal Graph

Alessio Conte[1(✉)], Mamadou Moustapha Kanté[2], Yota Otachi[3], Takeaki Uno[4], and Kunihiro Wasa[4]

[1] Università di Pisa, Pisa, Italy
conte@di.unipi.it
[2] Université Clermont Auvergne, LIMOS, CNRS, Aubière, France
mamadou.kante@uca.fr
[3] Kumamoto University, Kumamoto, Japan
otachi@cs.kumamoto-u.ac.jp
[4] National Institute of Informatics, Tokyo, Japan
{uno,wasa}@nii.ac.jp

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. The problem generalizes that of enumerating maximal independent sets and maximal induced forests, which correspond to respectively 0-degenerate and 1-degenerate subgraphs.

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 [12], dense subgraphs [13] or complete subgraphs [5,9]. A more comprehensive list can be found in [20].

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, [21] addresses the enumeration of induced trees in k-degenerate graphs.

The *degeneracy* of a graph is the smallest value 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 not a $k+1$-core, and is a widely used sparsity measure [5,9,15,19,21]. Several studies tend to take into account the degeneracy of graphs, as it tends to be very small

M.M. Kanté is supported by French Agency for Research under the GraphEN project (ANR-15-CE-0009)

Y. Cao and J. Chen (Eds.): COCOON 2017, LNCS 10392, pp. 150–161, 2017.
DOI: 10.1007/978-3-319-62389-4_13

in real-world networks [19], many important graph classes in structural graph theory are degenerate [15]. 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.* [16] 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 [2,14] for other recent studies on degeneracy.

In this paper we address the enumeration of maximal k-degenerate induced subgraphs, and provide an efficient polynomial delay algorithm for chordal input graphs. An enumeration algorithm is of polynomial delay if the maximum computation time between two 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 [7,20]).

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 [3,8,11,17,18]. 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) [17,18]; the graphs that allow a *clique tree* [3] (see Sect. 2.1); the intersection graphs of subtree families in trees [11]. 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 sets of vertices S and X, decide whether there is a maximal k-degenerate graph 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. This problem, however, can be shown to be NP-Complete for generic graphs (the proof is omitted for space reasons), and even on chordal graphs its complexity is not clear. We thus need some additional techniques: for our algorithm we do

not consider all possible sets (S, X) for the Extension problem, but only some 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 containing $S \cup S'$ and avoiding $X \cup X'$.

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 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 a linear ordering of its vertices. 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 any 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 $N_G(x)$.

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 $x \in V(G) \setminus C$. Depending on the context, C may refer to its set of vertices, the subgraph induced by C or the corresponding node in the clique tree. 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 $\mathcal{Q}(G, v)$ the set of maximal cliques containing x.

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. In what follows, we omit the subscript G and fix a graph $G = (V(G), E(G))$ if it is clear from the context.

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 *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 $\mathcal{QT}(G)$ [11]. Moreover, for every chordal graph G, one can compute a clique tree in linear time (see for instance [10]). In the rest of the paper all clique trees are considered rooted.

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 [5,9,15,19,21]. 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-degenrate 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 neighbour is a clique [17]. Therefore, in any chordal graph we can find a vertex of degree at most $\omega(G)-1$. □

3 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 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 efficient enumeration algorithm, we have to limit such redundant testing as much as possible. To achieve this we focus on the rooted clique tree $\mathcal{QT}(G)$ and introduce the concepts of *greedy filling* and *partial solution*. In what follows we let G be a fixed chordal graph.

3.1 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 be the set of vertices in $C \setminus P$. 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 1. *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 $v \in X$, $A(v, X) = \sum_{C \in \mathcal{Q}(G,v)} A(C, X)$, 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 S would cause S to not be k-degenerate anymore, we say that C is ***saturated*** *in* S if $|C \cap S| = k + 1$.

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

Lemma 2. *Let $G = (V, E)$ be a chordal graph and $M \subseteq V$ be a k-degenerate subgraph of G, with $X = V \setminus M$. M is maximal if and only if $A(x, X) \geq 1$ for each $x \in X$.*

Proof. Assume $A(x, X) \geq 1$ for each $x \in X$ and there exists a k-degenerate subgraph $M' \supset M$, with $v \in M' \setminus M$. 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 $M = V \setminus X$, we have $|C \setminus X| = |C \cap M| \geq k + 1$. As $M \cup \{v\} \subseteq M'$ we have $|C \cap M'| \geq k + 2$, 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 $|(M \cup \{x\}) \cap C| \leq k + 1$, since $|C \setminus X| = |C \cap M| < k + 1$. Thus the largest clique in $M \cup \{x\}$ has size at most $k + 1$, and as $M \cup \{x\}$ 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:

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. $\forall x \in X, A(x, X) \geq 1$,
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 $M \supseteq S$ with $M \cap X = \emptyset$. 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 3. *For any partial solution (S, X), the* greedy filling *yields a maximal k-degenerate subgraph M of G such that $S \subseteq M$ and $M \cap X = \emptyset$.*

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

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 $A(x, X_M) \geq 1$ for each $x \in X_M$, making M a maximal k-degenerate subgraph by Lemma 2. 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. □

3.2 Binary Partition Method

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

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 3). 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 4 in Sect. 3.3) 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 1 of Definition 1 still holds, and so (S, X) is still a partial solution. It is important

Algorithm 1. $k\texttt{MIG}$: 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) ← the 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)
```

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 2 that S is a maximal k-degenerate induced subgraph of G.

3.3 Correctness

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

Theorem 2. *Let G be a chordal graph and k be a non-negative integer. Then kMIG(G, k) outputs all and only maximal k-degenerate induced subgraphs of G without duplicates.*

As mentioned in the description, kMIG(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 2 we know that when $V = S \cup X$ then S is a maximal k-degenerate induced subgraph, thus kMIG(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 base condition assume that (S, X) is a partial solution such that $S \subseteq M, X \cap M = \emptyset$; 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 in $Q_i \setminus (S \cup X)$. If $v \in M$, 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, $v \notin M$, 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 $V \setminus M$, i.e. $(S, X \cup \{v\})$. In both cases the base condition is still true, thus by induction kMIG(G, k) will find M. In order to prove Theorem 2, 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 4. *Let (S, X) be any partial solution of G, Q be a clique such that its ancestor cliques are fully contained in $S \cup X$, and $v \notin S \cup X$ be 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 3.

Otherwise, 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 2. □

Thus Theorem 2 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 Complexity Analysis

In this section we analyze the cost of our algorithm, and prove that it can enumerate all maximal k-degenerate 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 [10]). 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 5. *In a chordal graph G, the number of 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 4. We recall that $\omega(G)$ denotes the maximum size of a clique in G.

Lemma 6. *Line 13 can be executed in time $O(\omega(G) \cdot |N(v)|)$.*

Proof. By Lemma 4 it is sufficient to check, for every vertex in $x \in N[v]$, whether there must be 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 $|Q' \setminus (X \cup \{v\})| \geq k + 1$, there exists a clique Q' such that $|Q' \setminus X| \geq k + 1 > |Q' \setminus (X \cup \{v\})| = k$, 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 5. 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 3. *$k\texttt{MIG}(G, k)$ runs with delay $O(m \cdot \omega(G))$.*

Proof. First, we need to compute $\mathcal{QT}(G)$, which takes $O(n+m)$ time [10]. 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 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 6), let us explain how to initialise them (we already explain in Lemma 6 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 $\mathcal{QT}(G)$: initialising $B(Q)$ 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 initialising the tables A and B take thus $O(n+m) = O(m)$ time (see Remark 1).

Let $v_1, \ldots v_t$ 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 6. 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.

5 Conclusion

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. It would be interesting for future work to investigate the feasibility of an output-polynomial algorithm for general graphs. It is worth noticing that the enumeration of maximal independent sets in graphs is a special case as X is an independent set in G if and only if $G[X]$ is 0-degenerate.

References

1. Alon, N., Kahn, J., Seymour, P.D.: Large induced degenerate subgraphs. Graph. Combin. **3**(1), 203–211 (1987)
2. Bauer, R., Krug, M., Wagner, D.: Enumerating and generating labeled k-degenerate graphs. In: 2010 Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics, pp. 90–98. SIAM, Philadelphia (2010)
3. Blair, J.R., Peyton, B.: An introduction to chordal graphs and clique trees. In: George, A., Gilbert, J.R., Liu, J.W.H. (eds.) Graph Theory and Sparse Matrix Computation, pp. 1–29. Springer, New York (1993)
4. Chandran, L.S.: A linear time algorithm for enumerating all the minimum and minimal separators of a chordal graph. In: Wang, J. (ed.) COCOON 2001. LNCS, vol. 2108, pp. 308–317. Springer, Heidelberg (2001). doi:10.1007/3-540-44679-6_34
5. Conte, A., Grossi, R., Marino, A., Versari, L.: Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In: ICALP (2016)
6. Diestel, R.: Graph Theory (Graduate Texts in Mathematics). Springer, New York (2005)
7. Eiter, T., Makino, K., Gottlob, G.: Computational aspects of monotone dualization: a brief survey. Discrete Appl. Math. **156**(11), 2035–2049 (2008)
8. Enright, J., Kondrak, G.: The application of chordal graphs to inferring phylogenetic trees of languages. In: Fifth International Joint Conference on Natural Language Processing, IJCNLP, pp. 545–552 (2011)
9. Eppstein, D., Löffler, M., Strash, D.: Listing all maximal cliques in sparse graphs in near-optimal time. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010. LNCS, vol. 6506, pp. 403–414. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17517-6_36
10. Galinier, P., Habib, M., Paul, C.: Chordal graphs and their clique graphs. In: Nagl, M. (ed.) WG 1995. LNCS, vol. 1017, pp. 358–371. Springer, Heidelberg (1995). doi:10.1007/3-540-60618-1_88
11. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. J. Comb. Theor. Ser. B **16**(1), 47–56 (1974)
12. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings IEEE International Conference on Data Mining, pp. 313–320. IEEE (2001)
13. Lee, V.E., Ruan, N., Jin, R., Aggarwal, C.: A survey of algorithms for dense subgraph discovery. In: Aggarwal, C.C., Wang, H. (eds.) Managing and Mining Graph Data, pp. 303–336. Springer, Heidelberg (2010)
14. Lukot'ka, R., Mazák, J., Zhu, X.: Maximum 4-degenerate subgraph of a planar graph. Electron. J. Comb. **22**(1), P1–11 (2015)

15. Nešetřil, J., Ossona de Mendez, P.: Sparsity: Graphs, Structures, and Algorithms. Algorithms and Combinatorics, vol. 28. Springer, Heidelberg (2012)
16. Pilipczuk, M., Pilipczuk, M.: Finding a maximum induced degenerate subgraph faster than 2^n. In: Thilikos, D.M., Woeginger, G.J. (eds.) IPEC 2012. LNCS, vol. 7535, pp. 3–12. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33293-7_3
17. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. **5**(2), 266–283 (1976)
18. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput. **13**(3), 566–579 (1984)
19. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the facebook social graph. arXiv preprint arXiv: 1111.4503 (2011)
20. Wasa, K.: Enumeration of enumeration algorithms. arXiv preprint arXiv:1605.05102 (2016)
21. Wasa, K., Arimura, H., Uno, T.: Efficient enumeration of induced subtrees in a K-degenerate graph. In: Ahn, H.-K., Shin, C.-S. (eds.) ISAAC 2014. LNCS, vol. 8889, pp. 94–102. Springer, Cham (2014). doi:10.1007/978-3-319-13075-0_8