Efficient Enumeration of k-Plexes and k-Defective Cliques

Mohamed Jiddou
LI-PARAD, Université de Versailles Saint-Quentin-En-Yvelines, Paris-Saclay, France

George Manoussakis
LI-PARAD, Université de Versailles Saint-Quentin-En-Yvelines, Paris-Saclay, France

Abstract

We investigate the enumeration of dense subgraphs under two well-known relaxations of cliques: k-plexes and k-defective cliques. Our main contribution is a family of algorithms with improved worst-case and output-sensitive complexities, driven by a decomposition technique based on graph degeneracy. We first propose a worst-case output-size near-optimal algorithm to enumerate all maximal k-plexes of size at least $2k-1$, achieving a total time complexity of $\mathcal{O}(n(dk)^3 2^d \Delta^k)$, where d is the degeneracy and Δ the maximum degree of the input graph. We then refine this result to obtain a fixed-parameter tractable output-sensitive algorithm with complexity $\mathcal{O}(\alpha f(k) p(d\Delta))$, where α is the number of solutions, $f(k)$ is an arbitrary function of k, and p is a polynomial. We then extend this framework to the enumeration of k-defective cliques and also show a linear-time $O(n)$ algorithm for the enumeration of 2-plexes for graphs with bounded degeneracy. To the best of our knowledge, these complexities are competitive with or better than the current state of the art.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, enumeration algorithms, maximal cliques enumeration

Digital Object Identifier 10.4230/LIPIcs.IPEC.2025.22

1 Introduction

Finding dense substructures in graphs is a fundamental problem in data mining [9, 19], social network analysis [16, 23], bio-informatics [15, 14], and database theory, where uncovering dense groups of entities reveals critical insights into the organization of complex systems. In data mining, dense patterns often signify meaningful associations or frequent co-occurrences; in social networks, they correspond to densely connected communities; in biological networks, they may represent protein complexes or functional modules; and in database systems, they inform efficient indexing and query optimization over graph-structured data.

Among the most well-known dense patterns is the clique, a subset of vertices in which every pair is mutually adjacent. Cliques represent idealized notions of cohesion but are rarely found in real-world networks, which are typically large, sparse, and noisy. The requirement of full connectivity makes cliques overly restrictive for practical applications. As a result, various relaxed clique models have been proposed to capture dense – but not necessarily complete – subgraphs that better reflect real-world connectivity.

Among these, the k-plex and k-defective clique models have been widely used [2, 19, 10, 20, 24]. A k-plex allows each vertex to be non-adjacent to up to k other vertices in the subgraph, while a k-defective clique requires the induced subgraph to miss at most k edges. These definitions offer a balance between density and noise tolerance, enabling the discovery of dense structures even in imperfect data. These models have demonstrated practical utility in numerous domains: detecting non-clique-like but cohesive communities in social networks [20, 23], identifying biologically relevant modules in protein-protein interaction networks [22], and mining patterns in incomplete or uncertain data sources [8].

The versatility of relaxed clique models makes them an important tool in modern graph analysis, where the goal is often to find structures that are both dense and resilient to noise. Consequently, the development of efficient algorithms to enumerate such subgraphs has become a central research problem.

2 State of the art

In this paper, we focus on the enumeration of two families of graph-theoretic structures: k-defective cliques and k-plexes. Both serve as models of relaxed cliques. Although the enumeration of k-plexes has been extensively studied, less attention has been paid to k-defective cliques. In this section, we review existing techniques and results concerning the enumeration of maximal k-plexes and k-defective cliques. In both the related work and throughout this paper, we encounter two common classes of output-sensitive algorithms for enumeration problems: *Polynomial Total Time* and *Polynomial Delay*. A problem is said to be solvable in polynomial total time if the total time required to enumerate all solutions is bounded by a polynomial in the size of the input and the number of solutions. This notion is typically used when the user is interested in generating the full output. However, Polynomial Delay ensures that the time between any two consecutive outputs is bounded by a polynomial in the input size. This is particularly valuable in practice, as it allows the user to process or visualize each solution incrementally without waiting for the enumeration to complete.

To our knowledge, there exist only two enumeration algorithms specifically designed for maximal s-defective cliques, which are given in [10]. The first is a polynomial delay algorithm, meaning that the time elapsed between two consecutive outputs is bounded by a polynomial function. Specifically, the delay is $\mathcal{O}(n^2 N_m^{2k+1}/4^k)$, where n is the number of vertices in the graph, N_m is the size of the largest k-defective clique. The second algorithm, which follows a more brute force approach and whose complexity is not output sensitive, uses a branch-and-bound approach with an optimized pivoting technique. Its time complexity is $\mathcal{O}(n\alpha_s^n)$, where α_s is a real constant strictly less than 2. An optimization based on the degeneracy ordering of the graph reduces this complexity to $\mathcal{O}(n^{k+2}\alpha_k^\delta)$, where δ is the degeneracy of the graph, a parameter that is typically small in real-world networks.

For the problem of enumerating all maximal k-plexes, several algorithms have been proposed. Some of these are adaptations of the Bron–Kerbosch algorithm [3], which was originally designed to enumerate maximal cliques. Wu and Pei [20] extended the Bron–Kerbosch algorithm to handle maximal k-plexes by introducing a set of pruning rules to eliminate unnecessary branches in the search space. The time complexity of their algorithm is $\mathcal{O}(2^n)$. Zhou et al. [24] introduced a novel branching heuristic that improves the running time to $\mathcal{O}(\alpha_k^n)$, where α_k is a constant depending on k and is strictly less than 2. In a related approach, Conte et al. [1] proposed a decomposition-based Bron–Kerbosch algorithm to enumerate k-plexes with diameter at most 2. Although no explicit time complexity is provided, a brief analysis suggests that the algorithm runs in $\mathcal{O}^*(2^{d\Delta})$, where d denotes the degeneracy and Δ the maximum degree of the graph.

Other approaches have also been proposed. In Wang et al. [19], the authors present an algorithm to enumerate all k-plexes with a time complexity of $\mathcal{O}(n^{2k} + n(d\Delta)^{k+1}\alpha_k^d)$. In Dai et al. [11], they describe an algorithm to enumerate all maximal k-plexes of size at least q, with a time complexity of $\mathcal{O}(nr_1^k r_2 \alpha_k^d)$, where $r_1 = \min\left\{\frac{d\Delta}{q^{-2k+2}}, n\right\}$ and $r_2 = \min\left\{\frac{d\Delta^2}{q^{-2k+2}}, n\right\}$. It is also possible to apply the more general framework of Cohen et al. [7], which provides algorithms to enumerate maximal induced subgraphs satisfying hereditary properties. In fact, in a subsequent paper of Berlowitz et al. [2], they applied this framework to the specific case

of k-plexes, resulting in a fixed-parameter tractable polynomial-time delay algorithm with time complexity $\mathcal{O}(f(k)p(n))$ with $p(n)$ a polynomial function of n and $f(k)$ an arbitrary function of k.

Closely related to our problems, the computation of the maximum k-defective clique has also been studied. It is known to be NP-hard [21], and several non-trivial algorithms have been developed to address it [4, 6, 5, 13]. These methods achieve exponential time complexities with base constants strictly less than 2, often derived from the roots of characteristic polynomials. Some offer tighter theoretical bounds, while others emphasize practical performance through preprocessing and reduction techniques. Certain approaches also exploit structural properties of the graph, such as degeneracy, to improve practical runtime. However, even optimized methods become impractical for large values of k, particularly on real-world datasets [13].

3 Our contributions

In this paper, we investigate the enumeration of dense subgraphs under the models of k-plexes and k-defective cliques. More specifically, we focus on connected k-plexes and k-defective cliques of diameter at most 2, as they better capture the notion of a community. To achieve this, Lemma 4 ensures that it suffices to enumerate k-plexes of size at least $2k-1$ and k-defective cliques of size at least $k+2$.

Our main goal is to develop algorithms to enumerate all maximal k-plexes of size at least $2k-1$. We then leverage this approach to enumerate maximal k-defective cliques by exploiting the structural relationship between the two concepts.

We begin by presenting general combinatorial results in Section 4.2. Then, in Section 4.3, we focus on the enumeration of maximal k-plexes. As a starting point, we build on the decomposition introduced in [9]. We first prove that every maximal k-plex of size at least $2k-1$ is contained within a small subgraph of size $\mathcal{O}(d\Delta)$. Using this structural insight, we design an enumeration algorithm that explores each subgraph G_i in time $\mathcal{O}((dk)^3 2^d \Delta^k)$, ensuring that all maximal k-plexes of size at least $2k-1$ are reported without duplication. A subsequent filtering step based on suffix trees guarantees a total enumeration time bounded by $\mathcal{O}(n(dk)^3 2^d \Delta^k)$. This result is formalized in Theorem 16. We also establish an upper bound on the number of such k-plexes, namely $\mathcal{O}(nk2^d\Delta^{k-1})$, and demonstrate the near-optimality of this bound by constructing a family of graphs in which the number of maximal k-plexes of size at least $2k-1$ is $\Omega(n3^{d/3}\Delta^{k-1})$. It follows that our algorithm achieves near-optimal runtime in the worst case.

In Berlow et al. [2], the authors introduced an algorithm for the enumeration of maximal k-plexes of size at least $2k-1$, with fixed-parameter tractable polynomial delay $\mathcal{O}(f(k)p(n))$, where $f(k)$ is an arbitrary function of k, and $p(n)$ is a polynomial in the number of vertices n. The considered parameter is k. We improve upon this result by applying the enumeration to the subgraphs G_i, leveraging the fact that each G_i contains at most $\mathcal{O}(d\Delta)$ vertices. Combined with our upper bound on the total number of solutions across all subgraphs G_i in Lemma 12, this enables the design of a polynomial total algorithm with runtime $\mathcal{O}(\alpha f(k)p(d\Delta))$, where α is the total number of output solutions. To the best of our knowledge, this is the first algorithm with total runtime polynomial both in the output size and in the degeneracy and maximum degree of the input graph.

By applying the same technique to the polynomial delay algorithm of Dai et al. [10] for the enumeration of maximal k-defective cliques of size at least $k+2$, mentioned earlier, we also derive a second output sensitive algorithm with polynomial total runtime. These results are presented in Theorems 17 and 19, respectively.

Finally, we propose an algorithm that lists all maximal 2-plexes in time $\mathcal{O}(n)$ for graphs of bounded degeneracy. This result is presented in Theorem 23. An overview of the existing results, as well as our claimed complexities, can be found in Table 1.

Table 1 Existing algorithms for the problems considered in this paper. Here d is the degeneracy, Δ the maximum degree, n the graph order, N_m the size of the largest k-defective clique, α_k, α_s are constants < 2, and α the output size.

Algorithm	**Output**	**Complexity**
Wu and Pei [20]	Maximal k-plexes	$O(2^n)$
Zhou et al. [24]	Maximal k-plexes	$O(\alpha_k^n)$
Conte et al. [1]	Maximal k-plexes with diameter ≤ 2	$O^*(2^{d\Delta})$
Wang et al. [19]	Maximal k-plexes	$O(n^{2k} + n(d\Delta)^{k+1}\alpha_k^d)$
Dai et al. [11]	Maximal k-plexes of size $\geq q$	$O(nr_1^k r_2 \alpha_k^d)$, where $r_1 = \min\left\{\frac{d\Delta}{q-2k+2}, n\right\}$, $r_2 = \min\left\{\frac{d\Delta^2}{q-2k+2}, n\right\}$
Dai et al. [10]	Maximal k-defective cliques	$O(n^2 N_m^{2k+1}/4^k)$ (*polynomial-time delay*)
Dai et al. [10] (Pivoting)	Maximal k-defective cliques	$O(n\alpha_s^n)$
Dai et al. [10] (Degeneracy-optimized)	Maximal k-defective cliques	$O(n^{k+2}\alpha_k^d)$
Berlowitz et al. [2]	Maximal k-plexes	$O(f(k)\,p(n))$, where $p(n)$ is polynomial in n and $f(k)$ is any function of k (*fixed-parameter tractable polynomial delay*)
This paper	Maximal k-plexes of size $\geq 2k-1$	$O(n(dk)^3 2^d \Delta^k)$
This paper	Maximal k-defective cliques of size $\geq k+2$	$O(\alpha(d\Delta)^2 \frac{(d+k)^{2k+2}}{4^k})$ (*polynomial total time*)
This paper	Maximal 2-plexes	$O(n)$ (when $d = O(1)$)
This paper	Maximal k-plexes of size $\geq 2k-1$	$O(\alpha f(k) p(d\Delta))$, where $p(d\Delta)$ is polynomial in $d\Delta$ and $f(k)$ is any function of k (*fixed-parameter tractable polynomial total time*)

4 Results

4.1 Notations

We consider graphs of the form $G = (V, E)$, which are simple, undirected, with n vertices and m edges. If $X \subseteq V$, the subgraph of G induced by X is denoted by $G[X]$. If $X \subset V$, then X is a proper subgraph of G. When not clear from the context, the vertex set of G will be denoted by $V(G)$. The set $N(x)$ is called the open neighborhood of the vertex x and consists of the vertices adjacent to x in G.

Given an arbitrary ordering $\sigma = v_1, \ldots, v_n$ of the vertices of G, the set V_i consists of the vertices following v_i in this ordering, including v_i itself, that is, $\{v_i, v_{i+1}, \ldots, v_n\}$. In the ordering σ, the rank of v_i, denoted by $\sigma(v_i)$, is its position in the ordering (here, i). By $[n]$, we denote the set of integers $\{1, 2, \ldots, n\}$. The distance between two vertices u and v is the length of the shortest path from u to v. Let $N_i^k(v) = V_i \cap N^k(v)$, where $N^k(v)$ is the set of vertices at distance k from v (in this paper, we consider $N_i^1(v_i) = N(v_i)$).

4.2 Preliminaries

In this section, we present a set of preliminary definitions and combinatorial results that constitute the theoretical foundation of our algorithms. We begin by introducing a decomposition framework based on subgraphs G_i, which plays a central role throughout the paper. We then formally define the two central notions of subgraph density considered in this work: k-plexes and k-defective cliques. These definitions enable a rigorous formulation of the enumeration problems that we aim to solve.

The section proceeds with a series of lemmas that establish the essential properties of these structures. In particular, Lemma 5 and Lemma 6 guarantee that any maximal k-plex of size at least $2k-1$ in a graph G is entirely contained in a unique subgraph G_i of the decomposition, Lemma 11 provides a procedure to verify whether a k-plex is maximal in its corresponding subgraph G_i in time $\mathcal{O}((d+k)\Delta)$, making the approach efficient in practice. These results are central in justifying the correctness and non-redundancy of our enumeration strategy. Further, Lemmas 9 and 10 provide tight bounds on the size and number of such k-plexes in terms of the graph's degeneracy d and maximum degree Δ. In addition, Lemma 12 establishes an upper bound on the total number of such k-plexes in all subgraphs G_i, showing that their sum is proportional to the number of maximal k-plexes in G. This result is crucial in the design and analysis of our output-sensitive algorithms.

▶ **Definition 1.** *Let $G = (V, E)$ be a graph, and let $v_1, \ldots, v_n$ be an ordering of its vertices. For each $i \in [n]$, we define the graph G_i as the subgraph of G induced by the vertex set $V(G_i) = \{v_i\} \cup N_i(v_i) \cup N_i^2(v_i)$, that is, the vertex v_i itself, its neighbors in V_i, and the vertices at distance two from v_i within V_i.*

▶ **Definition 2.** *Let $G = (V, E)$ be an undirected graph and let k be a non-negative integer. A subset $S \subseteq V$ is called a k-plex if, in the induced subgraph $G[S]$, each vertex has at most k non-neighbors (equivalently, at least $|S| - k$ neighbors) within S.*

A k-plex S is said to be maximal in G if there does not exist a strict superset $S' \supset S$ such that S' is also a k-plex in G; that is, S is inclusion-maximal among the k-plexes of G.

▶ **Definition 3.** *Let $G = (V, E)$ be an undirected graph and let k be a non-negative integer. A subset $C \subseteq V$ is called a k-defective clique if the induced subgraph $G[C]$ is missing at most k edges to be a complete clique. A k-defective clique C is said to be maximal in G if there does not exist a strict superset $C' \supset C$ such that C' is also a k-defective clique in G; that is, C is inclusion-maximal among the k-defective cliques of G.*

▶ **Lemma 4** (Two-hop property [18])**.** *If S is a k-plex in G with $|S| \geq 2k-1$, then the subgraph $G[S]$ is connected and has diameter at most two.*

▶ **Lemma 5.** *Let G be a graph, and let $\sigma = (v_1, \ldots, v_n)$ be a degeneracy ordering of its vertices. Let $C \subseteq V(G)$ be a maximal k-plex such that the size of C is at least $2k-1$. Then, there exists a unique index $i \in [n]$ such that $C \subseteq V(G_i)$ and $v_i \in C$.*

Proof. Let C be a maximal k-plex in G with $|C| \geq 2k-1$, and let $i \in [n]$ be the smallest index such that $v_i \in C$. By Lemma 4, the subgraph $G[C]$ has diameter at most two and contains v_i, which implies that every vertex $u \in C \setminus \{v_i\}$ is at distance at most two from v_i. Therefore, each such vertex u is either a neighbor of v_i or is at distance two from v_i, so by the definition of G_i, it follows that $C \subseteq V(G_i)$.

To show uniqueness, suppose for contradiction that there exists an index $j \neq i$ such that $C \subseteq V(G_j)$ and $v_j \in C$. If $j < i$, then $v_j \in C$, which contradicts the minimality of i. If $j > i$, then since $C \subseteq V(G_j)$, we must have $v_i \in V(G_j)$. By definition, $V(G_j) =$

$\{v_j\} \cup N(v_j) \cup N_2(v_j)$, so $v_i \in N(v_j) \cup N_2(v_j)$. However, this is impossible, since all vertices in $N(v_j) \cup N_2(v_j)$ have indices strictly greater than j, whereas $i < j$. Therefore, such an index j cannot exist, proving the uniqueness. ◀

▶ **Lemma 6.** *Let G be a graph, and let $\sigma = (v_1, \ldots, v_n)$ be a degeneracy ordering of its vertices. Let $C \subseteq V(G)$ be a maximal k-plex in subgraph G_i, for some $i \in [n]$, such that the size of C is at least $2k-1$. Then C is not maximal in G if and only if there exists a maximal k-plex $C' \subseteq V(G)$ of size at least $2k-1$, maximal in G, such that $C \subsetneq C' \subseteq V(G_j)$ for some $j < i$.*

Proof. Suppose that C is a maximal k-plex of size at least $2k-1$ in G_i, for some $i \in [n]$, but that C is not maximal in G. Then there exists a set $S \subseteq V(G) \setminus C$ such that the subgraph $G[C \cup S]$ is a k-plex of size at least $2k-1$, and is maximal in G. Let $v_j \in S$ be the vertex with the smallest rank according to the ordering σ.

If $j > i$, then since $G[C \cup S]$ has a diameter of at most 2, the vertex v_j must be at distance at most 2 from every vertex in C. By the definition of G_i, this means $v_j \in V(G_i)$, so $C \cup \{v_j\} \subseteq V(G_i)$ forms a k-plex of size at least $2k-1$ in G_i, contradicting the maximality of C in G_i. Therefore, it must be that $j < i$. Now set $C' = C \cup S$, which by hypothesis is a k-plex of size at least $2k-1$, maximal in G. Since $v_j \in C' \setminus C$, we have $C \subsetneq C'$, and $C' \subseteq V(G_j)$ with $j < i$. This completes the first implication.

Conversely, suppose that there exists a maximal k-plex C' of size at least $2k-1$ in G, included in a subgraph G_j for some $j < i$, such that $C \subsetneq C'$. Then, by definition, C cannot be maximal in G, which completes the proof. ◀

▶ **Corollary 7.** *Let G be a d-degenerate graph and let K be a maximal k-plex of size at least $2k-1$ of an induced subgraph G_i, $i \in [n]$, such that K is not maximal in G. Let C be a maximal k-plex of size at least $2k-1$ of G, which is a subgraph of some graph G_j, $j < i$, and such that K is a subgraph of C. Let $W(K)$ and $W(C)$ be the words obtained from the vertices of the k-plexes K and C ordered according to σ_G. Then $W(K)$ is a proper suffix of $W(C)$.*

Proof. First observe that, by Lemma 6, the k-plex C is well defined and $K \subsetneq C$. Let v_i be the vertex of K with the smallest index in σ_G; thus, v_i appears first in $W(K)$. Since $K \subsetneq C$, we also have $v_i \in C$.

Assume for contradiction that $W(K)$ is not a proper suffix of $W(C)$. This means that there exists at least one vertex $x \in C \setminus K$ which appears after v_i in $W(C)$. Otherwise, $W(K)$ would indeed be a proper suffix of $W(C)$. Let v_j denote the vertex of C with the smallest index in σ_G; according to Lemma 6, we have $j < i$. Since any subset of a k-plex is itself a k-plex, it follows that $K \cup \{x\}$ is a k-plex in G_j. Moreover, as K has size at least $2k-1$, so does $K \cup \{x\}$. By Lemma 4, $K \cup \{x\}$ must have diameter at most 2, implying $x \in N(v_i) \cup N_i^2(v_i)$. Thus, $K \cup \{x\}$ is a k-plex of size at least $2k-1$ in G_i, which contradicts the maximality of K in G_i. ◀

▶ **Lemma 8.** *Let G be a graph of degeneracy d. Then the size of any maximal k-plex in G of size at least $2k-1$ is at most $d+k$.*

Proof. Let C be a maximal k-plex in G of size at least $2k-1$, and let σ be a degeneracy ordering of G with degeneracy d. Let v_i be the vertex of C with the smallest rank in σ_G. By Lemma 5, $C \subseteq V(G_i)$. Therefore, C can contain at most $k-1$ vertices in $N_i^2(v_i)$; otherwise, the number of v_i's non-neighbors would exceed the allowed $k-1$, contradicting the definition of a k-plex. Moreover, by the definition of degeneracy, v_i has at most d neighbors in V_i, so at most d vertices in $N_i(v_i)$ can belong to C. Since $N_i(v_i)$ and $N_i^2(v_i)$ are disjoint, we obtain the following inequality: $|C| \leq d + k$. ◀

▶ **Lemma 9.** *Let G be a graph, and d its degeneracy. Then the number of maximal k-defective cliques of size at least $k+2$ is $\mathcal{O}(nk2^d\Delta^{k-1})$.*

Proof. Let G be a d-degenerate graph and σ a degeneracy ordering of its vertices. Denote by $\mathcal{P}([d])$ the set of all subsets of $[1,d]$, and by $\mathcal{P}_{k-1}$ the set of all subsets of $N_i^2(v_i)$ of size at most $k-1$. Let α_i denote the number of maximal k-plex of size at least $2k-1$ in the graph G_i, for $i \in [n]$, containing v_i, and let α be the number of maximal k-plex of size at least $2k-1$ in the graph G. According to Lemma 5, for every maximal k-plex of size at least $2k-1$ in G, there exists $i \in [n]$ such that $C \subseteq V(G_i)$ and $v_i \in C$. Furthermore, by Lemma 6, there may exist a maximal k-plex of size at least $2k-1$ in G_i containing v_i (for some $i \in [n]$), which are not maximal in G. It follows that $\alpha \leq \sum_{i=1}^{n} \alpha_i$.

Let $i \in [n]$. By definition of σ, the vertex v_i has at most d neighbors of higher rank (i.e., greater index) in σ; in other words, the cardinality of $N(v_i)$ is at most d. Each vertex in $N(v_i)$ is adjacent to at most Δ vertices in $N_{2i}(v_i)$, so $|N_{2i}(v_i)| \leq d\Delta$. Moreover, by arguments analogous to those in the proof of Lemma 8, any maximal k-plex of size at least $2k-1$ in G_i containing v_i contains at most d vertices from $N(v_i)$ and at most $k-1$ vertices from $N_{2i}(v_i)$. Thus, for k-plex C, there exist two sets $S_1 \subseteq \mathcal{P}([d])$ and $S_2 \subseteq \mathcal{P}_k$ such that $C = \{v_i\} \cup S_1 \cup S_2$. Therefore, we obtain the following inequality: $\alpha_i \leq |\mathcal{P}([d]) \times \mathcal{P}_{k-1}|$, with $|\mathcal{P}_{k-1}| = \sum_{r=0}^{k-1} \binom{d\Delta}{r}$ and $|\mathcal{P}([d])| = 2^d$. As $\sum_{r=0}^{k-1} \binom{d\Delta}{r} \in \mathcal{O}(k\Delta^{k-1})$, it follows that $|\mathcal{P}([d]) \times \mathcal{P}_{k-1}| \in \mathcal{O}(k2^d\Delta^{k-1})$, and thus $\alpha_i \in \mathcal{O}(k2^d\Delta^{k-1})$. Consequently, $\alpha \leq \sum_{i=1}^{n} \alpha_i$ which is bounded by $\mathcal{O}(nk2^d\Delta^{k-1})$. ◀

▶ **Lemma 10.** *For any tuple (k,d,Δ,n) such that $d \leq k \leq \Delta \leq n$, we can construct a graph of order n with maximum degree Δ and degeneracy d such that G has $\Omega\left((n-d-\Delta)3^{d/3}\Delta^{k-1}\right)$ maximal k-plexes of size at least $2k-1$.*

Proof. Let $G = (V,E)$ be a simple, undirected, d-degenerate graph with maximum degree Δ, and let $\sigma = (v_1, v_2, \ldots, v_n)$ be a degeneracy ordering. Set $S_1 = \{v_i : 1 \leq i \leq n-d-\Delta\}$, $S_2 = V \setminus S_1$, and consider a partition $S_2 = S_2^1 \cup S_2^2$ such that $|S_2^1| = d$, $G[S_2^2]$ is a clique, and S_1 is an independent set.

For every $i \in \{1, \ldots, n-d-\Delta\}$, let G_i be the subgraph induced by the vertex set $\{v_i\} \cup S_2$, where v_i is adjacent to all vertices in S_2^1, and v_i is not adjacent to any vertex in S_2^2. Let α_i denote the number of maximal k-plexes of size at least $2k-1$ in G_i, and let α denote the number of such k-plexes in G. For $i \in \{1, \ldots, n-d-\Delta\}$, Consider $C = \{v_i\} \cup C_1 \cup C_2$, where C_1 is a maximal clique of S_2^1 and $C_2 \subseteq S_2^2$ is a subset of cardinality $k-1$. We show that C is a maximal k-plex of size at least $2k-1$ in G_i. First, $C \subseteq V(G_i)$, and since $v_i \notin C_2$, we have $|C| = d + k \geq k+2$ by Lemma 8. Moreover, since S_2^2 induces a clique and v_i is not adjacent to any vertex in S_2^2, the subgraph $G_i[C]$ missing exactly $k-1$ edges, namely those between v_i and each vertex of C_2. Thus, C is indeed a k-plex in G_i.

Assume for contradiction that C is not maximal in G; then there exists $u \in V \setminus C$ such that $C \cup \{u\}$ is a k-defective clique. Since $|S_2^1| = d$, then v_i and u are not adjacent; hence, v_i has exactly k non-neighbors in $G[C \cup \{u\}]$, contradicting the definition of a k-plex. C is therefore maximal both in G and in G_i.

For every choice of a pair (C_1, C_2), we obtain a distinct maximal k-plex. There are exactly $3^{d/3}$ maximal cliques in S_2^1 and $\binom{|S_2^2|}{k-1} = \binom{\Delta}{k-1}$ subsets of size k in S_2^2 (since $|S_2^2| = \Delta$). Thus, we obtain $3^{d/3}\binom{\Delta}{k-1}$ maximal k-plexes of the form C. Therefore, for each $i \in \{1, \ldots, n-d-\Delta\}$, we have $\alpha_i \geq 3^{d/3}\binom{\Delta}{k-1}$. Moreover, since S_1 is an independent set, every maximal k-plex of size at least $2k-1$ in G_i is also maximal in G, for each such i.

Finally, we obtain: $\alpha = \sum_{i=1}^{n-d-\Delta} \alpha_i \geq (n-d-\Delta)3^{d/3}\binom{\Delta}{k-1}$ Now, since $\binom{\Delta}{k-1} \in \Omega(\Delta^{k-1})$, it follows that $\alpha \in \Omega((n-d-\Delta)3^{d/3}\Delta^{k-1})$. ◀

▶ **Lemma 11.** *Let G be a d-degenerate graph with maximum degree Δ, σ a degeneracy ordering of its vertices, and G_i the subgraph defined as in Definition 1 for some $i \in [n]$. If $C \subseteq V(G_i)$ is a k-plex of size at least $2k-1$, then one can verify whether C is maximal in G_i in time $\mathcal{O}((d+k)d\Delta)$.*

Proof. Let $C \subseteq V(G_i)$ be a k-plex of size at least $2k-1$. To check whether C is maximal in G_i, it suffices, by Definition 2, to verify whether there exists a vertex $u \in V(G_i) \setminus C$ such that $C \cup \{u\}$ is still a k-plex in G_i. The cardinality of $V(G_i)$ is bounded by $1 + d + d\Delta$ (i.e., $O(d\Delta)$), since G_i contains at most d direct neighbors of v_i and $d\Delta$ vertices at distance two. Therefore, the size of $V(G_i) \setminus C$ is also $O(d\Delta)$. For each $u \in V(G_i) \setminus C$, one needs to check whether $|C \cup \{u\}| - \deg_{G[C\cup\{u\}]}(v) \leq k-1$, for each $v \in C \cup \{u\}$, which can be done in $O(d+k)$ time, since $|C| \leq d+k$ by Lemma 4. If this condition holds for every $v \in C \cup \{u\}$), then C is not maximal in G_i. Therefore, the maximality of a k-plex can be verified in $O((d+k)d\Delta)$ time. ◀

▶ **Lemma 12.** *Let G be a d-degenerate graph, and let G_i, $i \in [n]$, be the family of induced subgraphs defined in Definition 1. Let α denote the number of maximal k-plexes of size at least $2k-1$ in G, and let α_i denote the number of such k-plexes in G_i. We have that $\sum_{i=1}^{n} \alpha_i \leq (d+k)\alpha$.*

Proof. Let m_i denote the number of maximal k-plexes of size at least $2k-1$ in G_i, $i \in [n]$ that are also maximal in G, and let Nm_i be the number of such k-plexes in G_i that are not maximal in G. Thus, $\alpha_i = m_i + Nm_i$.

Let C be a maximal k-plex of size at least $2k-1$ in G. By Lemma 8, the number of vertices in C is at most $d+k$. Consequently, $W(C)$, the word formed by the vertices of C ordered according to σ_G, can have at most $d+k-1$ proper suffixes.

Moreover, by Corollary 7, every maximal k-plex K of size at least $2k-1$ in some subgraph G_i, $i \in [n]$, that is not maximal in G, satisfies $W(K)$ being a proper suffix of $W(C)$ for some maximal k-plex C in G. Therefore, $\sum_{i=1}^{n} Nm_i \leq (d+k-1)\alpha$, and it follows that $\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n}(m_i + Nm_i) \leq \sum_{i=1}^{n} m_i + \sum_{i=1}^{n} Nm_i \leq \alpha + (d+k-1)\alpha = (d+k)\alpha$. ◀

4.3 Algorithms for the enumeration of k-plexes

We now describe our enumeration framework for computing all maximal k-plexes of size at least $2k-1$ in a graph G. This section presents two algorithms: a base algorithm that enumerates all such k-plexes in each subgraph G_i, and a refined version that guarantees uniqueness of output using a generalized suffix tree. The first algorithm operates locally in each G_i and its enumeration time depends solely on the degeneracy of G, ensuring efficient exploration of the search space. The second algorithm builds upon the first by incorporating a filtering mechanism based on suffix tree rejection to avoid duplicate outputs. Both algorithms are described in detail and formally analyzed. The correctness and non-redundancy of the enumerated solutions are established in Theorems 13 and 14, while their complexities are proved in Theorems 15 and 16. Finally, in Theorem 17, we derive an output-sensitive result by applying the polynomial delay algorithm from [12] to the subgraphs G_i defined in Definition 1. By exploiting the fact that each G_i has size $\mathcal{O}(d\Delta)$, and combining this with the bound established in Lemma 12 on the total number of maximal k-plexes across all G_i's, we design an output-sensitive algorithm with a total running time polynomial in $d\Delta$, and linear in the number of solutions.

Algorithm 1 The procedure to find the local maximal k-plexes.

Data: A graph G_i, integer k
Result: All maximal k-plexes of size at least $2k-1$ in G_i containing v_i
1 Initialize $C \leftarrow \emptyset$;
2 Let $N_1 \leftarrow N(v_i)$;
3 Let $N_2 \leftarrow N_i^2(v_i)$;
4 **for** *every subset* $S_1 \subseteq N_1$ **do**
5 **for** *every subset* $S_2 \subseteq N_2$ *such that* $|S_2| \leq k-1$ **do**
6 Let $C \leftarrow \{v_i\} \cup S_1 \cup S_2$;
7 **if** $|C| \geq 2k-1$ *and* $G_i[C]$ *is a* k*-plex* **then**
8 **if** *no proper superset of* C *in* G_i *satisfies the same properties* **then**
9 Add C to the output;
10 **return** All cliques C found;

Algorithm 2 The algorithm to find all maximal k-plexes.

Data: A graph G, integer k
Result: All maximal k-plexes of size at least $2k-1$ in G
1 Compute d the degeneracy of G and σ_G;
2 Initialize T an empty generalized suffix tree;
3 **for** $i = 1$ *to* n **do**
4 Algorithm 1(G_i, k)
5 **for** *every maximal* k*-plex* C *of graph* G_i **do**
6 Order the vertices of C following σ_G;
7 Search for C in T;
8 **if** *there is a match* **then**
9 Reject it;
10 **else**
11 Insert the proper suffixes of C in T;
12 Output C;

4.3.1 Proof of correctness

▶ **Theorem 13.** *Given a subgraph G_i as defined in Definition 1, Algorithm 1 outputs exactly all maximal k-plexes of size at least $2k-1$, containing v_i, without duplication.*

Proof. Assume for contradiction that there exists a k-plex C of size at least $2k-1$, containing v_i and maximal in G_i, which is not reported by Algorithm 1. Let $C_1 = C \cap N_1$ and $C_2 = C \cap N_2$, so that $C = \{v_i\} \cup C_1 \cup C_2$. Observe that since $C_1 \subseteq N_1$, the subset $S_1 = C_1$ is examined at line 4 of the algorithm. Moreover, as v_i has no neighbors in C_2, it follows that $|C_2| \leq k-1$. Thus, the subset $S_2 = C_2$ is included in the enumeration at line 5. Consequently, the set C is constructed at line 6, and by assumption, both the size and k-plex constraints at line 7, as well as the maximality criterion at line 8, are satisfied. Hence, Algorithm 1 must output C, a contradiction. It follows that every k-plex of size at least $2k-1$, containing v_i and maximal in G_i, is indeed generated by the algorithm.

We now show that Algorithm 1 does not produce any duplicates. Suppose, to the contrary, that a k-plex C is generated more than once, i.e., there exist distinct pairs $(S_1, S_2) \neq (S_1', S_2')$ such that $C = \{v_i\} \cup S_1 \cup S_2 = \{v_i\} \cup S_1' \cup S_2'$. By construction, $S_1, S_1' \subseteq N_1$, $S_2, S_2' \subseteq N_2$, with $N_1 \cap N_2 = \varnothing$. Thus, the decomposition $C = \{v_i\} \cup S_1 \cup S_2$ is unique, as the sets S_1 and S_2 are disjoint and correspond to non-overlapping neighborhoods. Therefore, if $(S_1, S_2) \neq (S_1', S_2')$, the resulting sets are necessarily distinct, contradicting our assumption. This establishes that each clique is output at most once by the algorithm. ◀

▶ **Theorem 14.** *Given a subgraph G, and an integer k, Algorithm 2 outputs exactly all maximal k-plexes of size at least $2k-1$, without duplication.*

Proof. Assume for contradiction, that there exists a maximal k-plex C of size at least $2k-1$ in G that is not output by Algorithm 2. By Lemma 5, there exists a unique index $i \in [n]$ such that C is maximal in G_i and $C \subseteq V(G_i)$. According to Theorem 13, Algorithm 1, which is called at line 4 of Algorithm 2, enumerates all maximal k-plexes in G_i without duplication. Thus, C must necessarily be produced at line 5. It follows that C can only be rejected by the suffix-tree rejection test (lines 7–9) if it is detected as a proper suffix of some already produced k-plex C' that is maximal in G, with $C \subsetneq C'$, as ensured by Corollary 7. However, the inclusion $C \subsetneq C'$ contradicts the maximality of C in G. Therefore, C must indeed be output by Algorithm 2. We conclude that Algorithm 2 outputs all maximal k-plexes of size at least $2k-1$ in G.

We now show that there is no duplication. Assume for contradiction, that this is not the case; that is, there exists a maximal k-plex C of size at least $2k-1$ in G that is output at least twice by Algorithm 2. Since Algorithm 1 enumerates, without duplication, all maximal k-plexes of each subgraph G_i, $i \in [n]$, at line 4, the only possibility is that C is enumerated in two different subgraphs G_i and G_j with $i \neq j$. However, this contradicts Lemma 5, which guarantees that a maximal k-plex of size at least $2k-1$ in G is included in exactly one such subgraph G_i.

Hence, by contradiction, Algorithm 2 outputs exactly all maximal k-plexes of size at least $2k-1$ in G, without duplication. ◀

4.3.2 Proofs of complexity

This subsection establishes the computational complexity of our enumeration algorithms for maximal k-plexes of size at least $2k-1$. We analyze both the local enumeration within individual subgraphs G_i and then the enumeration across the entire graph G. These bounds exploit the structural properties of degeneracy orderings and leverage the combinatorial limits established in Lemma 9, combined with efficient suffix tree operations to ensure uniqueness of the output.

▶ **Theorem 15.** *Given a graph G_i, $i \in [n]$, and an integer k, the time complexity of Algorithm 1 is in $\mathcal{O}((dk)^3 2^d \Delta^k)$.*

Proof. Let G be a d-degenerate graph, and let σ be a degeneracy ordering of its vertices. For each vertex v_i, the algorithm operates on the subgraph G_i, as defined in Definition 1. The `for` loop at line 4 enumerates all subsets $S_1 \subseteq N_1$, yielding 2^d iterations. The loop at line 5 iterates over all $S_2 \subseteq N_2$ with $|S_2| \leq k-1$, corresponding to exactly $\sum_{r=0}^{k-1} \binom{d\Delta}{r}$ iterations, which is $\mathcal{O}(k\Delta^{k-1})$. The checks at line 7 are performed in $\mathcal{O}(d+k)$ time, and the maximality verification at line 8 requires $\mathcal{O}(d(d+k)\Delta)$ time according to Lemma 11. Therefore, the body of the inner loop runs in $\mathcal{O}(d(d+k)^2\Delta$ time. In total, the algorithm has a complexity of $\mathcal{O}(dk(d+k)^2\Delta 2^d \Delta^{k-1}) = \mathcal{O}(dk(d+k)^2 2^d \Delta^k)$, that is, $\mathcal{O}((dk)^3 2^d \Delta^k)$. ◀

▶ **Theorem 16.** *Given a graph G with degeneracy d and maximum degree Δ, and an integer k, the time complexity of Algorithm 2 is in $\mathcal{O}(n(dk)^3 2^d \Delta^k)$.*

Proof. Let G be a d-degenerate graph and let σ be a degeneracy ordering of its vertices. The computation of the degeneracy and the ordering σ at line 1 can be performed in $\mathcal{O}(m)$ time, where m is the number of edges. For lines 3–4, we consider each subgraph G_i for $i \in [n]$, so this step is executed for n subgraphs in total. Enumerating all maximal k-plexes of size at least $2k-1$ in each G_i takes $\mathcal{O}((dk)^3 2^d \Delta^k)$ time by Theorem 15. According to Lemma 9, each subgraph G_i contains $\mathcal{O}(2^d \Delta^k)$ maximal k-plexes of size at least $2k-1$. Thus, the `for` loop at line 5 runs $\mathcal{O}(2^d \Delta^k)$ times. Regarding lines 5–12: Ordering the vertices of C according to σ can be done in $\mathcal{O}(|C|) \subseteq \mathcal{O}(d+k)$ time, by Lemma 8. Suffix tree operations (search and insertion) for words of length at most $d+k$ can be performed in $\mathcal{O}(d+k)$ time (see [17]). Thus, the body of the inner loop at line 5 has complexity $\mathcal{O}((d+k)2^d \Delta^k)$, and so the outer loop at line 3 has total complexity $\mathcal{O}\big(n((dk)^3 2^d \Delta^k + n(d+k)2^d \Delta^k)\big) = \mathcal{O}(n(dk)^3 2^d \Delta^k)$. Since all other steps require at most $\mathcal{O}(m)$ operations, it follows that the overall complexity of the algorithm is $\mathcal{O}(n(dk)^3 2^d \Delta^k)$. ◀

▶ **Theorem 17.** *Given a graph G of degeneracy d and maximum degree Δ, and an integer k, one can enumerate all maximal k-plexes of size at least $2k-1$, without duplication, in time $\mathcal{O}\left(\alpha\, f(k)\, p(d\Delta)\right)$, where α denotes the number of maximal k-plexes of size at least $2k-1$ in G, $p(d\Delta)$ is a polynomial function of $d\Delta$, and $f(k)$ is an arbitrary function of k. The algorithm is output-sensitive, with total running time proportional to the number of solutions.*

Proof. Let α denote the number of maximal k-plexes of size at least $2k-1$ in G, and let α_i denote the number of such k-plexes in the subgraph G_i, for $i \in [n]$. In [2], the authors applied this framework to the specific case of k-plexes, resulting in a fixed-parameter tractable algorithm with polynomial delay and time complexity $\mathcal{O}(f(k)\, p(n))$, where n is the order of the graph, $p(n)$ is a polynomial function of n, and $f(k)$ is an arbitrary function of k. This algorithm can be adapted to our decomposition into subgraphs G_i, which yields a significant improvement in complexity: $\mathcal{O}\big(\alpha\, f(k)\, p(d\Delta)\big)$. More concretely, the approach consists of applying this algorithm to each subgraph G_i, for $i \in [n]$. For each $i \in [n]$, the order of G_i is $\Theta(d\Delta)$, where Δ is the maximum degree of G. Consequently, the enumeration in each G_i can be performed with a delay between two outputs in $\mathcal{O}\big(f(k)\, p(d\Delta)\big)$. Since there are α_i such cliques in G_i, for $i \in [n]$, the enumeration in G_i takes $\mathcal{O}\left(\alpha_i f(k)\, p(d\Delta)\right.$ time.

Moreover, by applying the same arguments used in the proof of Lemma 12, it can be shown that $\sum_{i=1}^{n} \alpha_i \leq (d+k-1)\alpha$. It follows that the enumeration over all subgraphs G_i can be performed in total time $\mathcal{O}\left(f(k)\, p(d\Delta) \sum_{i=1}^{n} \alpha_i\right) \in \mathcal{O}\left(\alpha f(k)\, p(d\Delta)\right)$. To ensure that only cliques which are maximal in G are retained, a similar filtering procedure to that of Algorithm 2 can be used. This step requires $\mathcal{O}\big((d+k)\alpha\big)$ time, as established in the proof of Theorem 16. Therefore, by Lemma 5, Lemma 6, and Corollary 7, this approach yields an exhaustive, duplication-free enumeration of all maximal k-defective cliques of size at least $k+2$ in G, with total time complexity $\mathcal{O}\left(\alpha f(k)\, p(d\Delta)\right)$. ◀

4.4 Application to the enumeration of k-defective cliques

In this section, we show how to effectively extend our analytical framework to the enumeration of k-defective cliques. We leverage the fact that a k-defective clique is, by definition, a $(k+1)$-plex, which allows us to reuse the algorithms previously developed for the enumeration of k-plexes. By inserting an additional verification step into the enumeration algorithm, we

ensure that only those $(k+1)$-plexes satisfying the k-defectiveness condition are retained. This refinement enables the duplication-free generation of all maximal k-defective cliques of size at least $k+2$ in a given graph, as established in Theorem 16.

We then introduce an alternative approach based on an output-sensitive algorithm that fully exploits the degeneracy and the maximum degree of the graph to improve the computational complexity. Theorem 19 formalizes this optimization by providing a bound on the total running time in terms of the number of solutions. To the best of our knowledge, these are the first output-sensitive bounds for this problem that are expressed solely in terms of the parameters k, Δ, d, and the number of maximal k-defective cliques. Since the proofs of Theorems 18 and 19 are similar to the proofs presented in Section 4.3.

▶ **Theorem 18.** *Given a graph G of degeneracy d and maximum degree Δ, and an integer k, one can enumerate all maximal k-defective cliques of size at least $k+2$, without duplication, in time $\mathcal{O}\left(n(dk)^5 2^d \Delta^{k+1}\right)$.*

Proof. Since every k-defective clique is a $(k+1)$-plex, the enumeration algorithm for maximal $(k+1)$-plexes can be leveraged to enumerate all maximal k-defective cliques. Specifically, Algorithm 2 is applied to enumerate all maximal $(k+1)$-plexes of size at least $2k+1$. Immediately after Line 11 of Algorithm 2, an additional verification step is incorporated to test whether the current $(k+1)$-plex is also a k-defective clique. This verification consists in counting the number of missing edges in the subgraph induced by the current $(k+1)$-plex. It can be performed in $\mathcal{O}((d+k)^2)$ time. According to Theorem 16, Algorithm 2 enumerates all maximal $(k+1)$-plexes of size at least $2k+1$ in $\mathcal{O}\left(n(dk)^3 2^d \Delta^{k+1}\right)$ time. Since the additional verification for the k-defective clique condition is performed once per candidate and requires at most $\mathcal{O}((d+k)^2)$ time, the total time per candidate becomes $\mathcal{O}\left((dk)^3(d+k)^2 2^d \Delta^{k+1}\right) = \mathcal{O}\left((dk)^5 2^d \Delta^{k+1}\right)$. Finally, by Lemma 9, Lemma 10, and Corollary 7, the suffix-tree-based filtering guarantees that only the k-defective cliques which are maximal in G are retained, and that no duplicates are produced. Hence, this approach yields an exhaustive, duplication-free enumeration of all maximal k-defective cliques of size at least $k+2$ in G, with overall time complexity $\mathcal{O}\left((dk)^5 2^d \Delta^{k+1}\right)$. ◀

▶ **Theorem 19.** *Given a graph G of degeneracy d and maximum degree Δ, and an integer k, one can enumerate all maximal k-defective cliques of size at least $k+2$, without duplication, in time $\mathcal{O}\left(n\alpha(d\Delta)^2 \frac{(d+k)^{2k+2}}{4^k}\right)$, where α denotes the number of maximal k-defective cliques of size at least $k+2$ in G. The algorithm is output-sensitive, with total running time proportional to the number of solutions.*

Proof. Let α denote the number of maximal k-defective cliques of size at least $K+2$ of G, and let α_i denote the number of maximal k-defective cliques of size at least $k+2$ in the subgraph G_i, $i \in [n]$. In [10], Algorithm 3 enumerates all maximal k-defective cliques of size at least $k+2$ in a graph G, with a delay of $\mathcal{O}\big(n^2 N_m^{2k+1}/4^k\big)$ between two outputs, where n is the order of the graph and N_m denotes the size of the largest maximal k-defective clique in G. This algorithm can be adapted to our decomposition into subgraphs G_i, which allows for a significant complexity improvement to $\mathcal{O}\big(n(d\Delta)^2(d+k)^{2k+2}/4^k\big)$, where d is the degeneracy of the input graph and n its order. More concretely, the approach consists in applying Algorithm 3 to each subgraph G_i, $i \in [n]$. For every $i \in [n]$, the order of G_i is $\Theta(d\Delta)$, where Δ is the maximum degree of G. Moreover, Lemma 4 ensures that the size of the largest maximal k-defective clique of size at least $k+2$ in G_i is at most $d+k+1$. Thus, the enumeration in each G_i can be performed with a polynomial delay of $\mathcal{O}\big((d\Delta)^2(d+k)^{2k+1}/4^k\big)$. Consequently, the enumeration in each G_i can be performed

with a polynomial delay between two outputs in $\mathcal{O}\left((d\Delta)^2 \frac{(d+k)^{2k+1}}{4^k}\right)$. Since there are α_i such cliques in G_i, for $i \in [n]$, the enumeration in G_i takes $\mathcal{O}\left(\alpha_i (d\Delta)^2 \frac{(d+k)^{2k+1}}{4^k}\right)$ time. Moreover, by applying exactly the same arguments used in the proof of Lemma 12, it can be shown that $\sum_{i=1}^{n} \alpha_i \leq (d+k-1)\alpha$. It follows that the enumeration over all subgraphs G_i can be performed in total time $\mathcal{O}\left((d\Delta)^2 \frac{(d+k)^{2k+1}}{4^k} \sum_{i=1}^{n} \alpha_i\right) \in \mathcal{O}\left(\alpha (d\Delta)^2 \frac{(d+k)^{2k+2}}{4^k}\right)$. To ensure that only cliques which are maximal in G are retained, one can use a similar filtering procedure as in Algorithm 2, which requires $\mathcal{O}\big(\alpha(d+k)\big)$ time, as established in the proof of Theorem 16. Therefore, by Lemma 5, Lemma 6, and Corollary 7, this approach yields an exhaustive, duplication-free enumeration of all maximal k-defective cliques of size at least $k+2$ in G with$\mathcal{O}\big(\alpha (d\Delta)^2 \frac{(d+k)^{2k+3}}{4^k}\big)$ time. ◀

4.5 Application to the enumeration of 2-plexes

In this section, we focus on the enumeration of maximal 2-plexes. We build upon some of the combinatorial results presented in the previous sections and apply a variation of a result by Eppstein [12] concerning the enumeration of non-induced maximal bicliques. This allows us to derive a linear-time algorithm, with complexity $\mathcal{O}(n)$, to enumerate maximal 2-plexes in graphs of constant degeneracy.

Recall that a non-induced biclique is a bipartite subgraph $A \times B$ such that every vertex in A is adjacent to every vertex in B, without the requirement that A and B form independent sets. Such a biclique is said to be maximal if it is not properly contained in any other biclique of the same form. Moreover, given an undirected graph G, a d-bounded orientation of G is any orientation of the edges such that each vertex has out-degree at most d. In this section, the term biclique always refer to non-induced bicliques.

▶ **Lemma 20.** *Let G be an undirected d-degenerate graph, and let C be a 2-plex of size at least 3 and maximal in G. Then there exists a maximal non-induced biclique $A \times B$ in G such that $C \subseteq A \cup B$.*

Proof. Let G be an undirected d-degenerate graph, and let C be a maximal 2-plex of size at least 3 in G. By Lemma 5, there exists $i \in [n]$ such that $C \subseteq V(G_i)$ and $v_i \in C$. Let us define $A = C \cap N(v_i)$, and $B = C \cap (N_i^2(v_i) \cup \{v_i\})$, so that $C = A \cup B$. By construction, $A \subseteq N(v_i)$, and thus $|A| \leq d$. Since C is a 2-plex, C can contain at most one vertex in $N_i^2(v_i)$, ensuring that v_i has at most one non-neighbor in C. Hence, $|B| \leq 2$. Furthermore, by the definition of $N(v_i)$, the vertex v_i is adjacent to every vertex in A, and every $u \in B$ is adjacent to every vertex in A as well; otherwise, u would have more than one non-neighbor in C, contradicting the fact that C is a 2-plex. Consequently, $A \times B$ forms a biclique. If $A \times B$ is maximal, the lemma holds, otherwise it is included in a maximal non-induced biclique containing $A \times B$. ◀

▶ **Lemma 21** (Folklore)**.** *If a graph G has degeneracy d, it has a d-bounded orientation.*

▶ **Lemma 22** (Eppstein [12])**.** *Given a graph G with a d-bounded orientation, and a collection of sets A_i, where each A_i is a 2-plex of size at most d in G, it is possible to compute the corresponding non-induced bicliques $A_i \times B_i$ in total time $\mathcal{O}(d2^d n + d^2 r)$, where r denotes the total size of all sets A_i.*

Proof. We rely on a result established in [12], which shows that, given a graph equipped with a d-bounded orientation and a collection of sets $A_1, \ldots, A_m$, where each A_i is a subset of vertices, all non-induced bicliques $A_i \times B_i$ can be listed in total time $\mathcal{O}(d^2 n + d^2 r)$,

where r denotes the total size of sets A_i. In our case, we consider as candidates the sets A_i defined as 2-plexes of size at most d in the subgraphs G_j, $j \in [n]$, according to Definition 1. More specifically, we restrict to sets $A_i \subseteq N(v_j)$, since $N(v_j)$ contains at most d vertices due to the degeneracy of the graph. Therefore, each G_j contains at most 2^d such subsets. Summing over all n vertices v_j, the total number of sets A_i is thus bounded by $n2^d$. All these sets can therefore be listed in time $\mathcal{O}(n2^d)$. Then, by applying the result from [12] to this collection, all non-induced bicliques $A_i \times B_i$ can be listed in total time $\mathcal{O}(d^2n + d^2n2^d + n2^d) = \mathcal{O}(d^2n + d^2 2^d n)$, since $r = n2^d$. ◀

The proof of the following theorem is similar to that given by Eppstein [12] for their linear-time algorithm to enumerate maximal non-induced bicliques, with a few additional simple steps.

▶ **Theorem 23.** *If an undirected graph G has degeneracy $d = \mathcal{O}(1)$, then all maximal 2-plexes of G can be listed in $\mathcal{O}(n)$ time.*

Proof. We rely on Theorem 1 from [12], which establishes that for any graph G with arboricity $a = \mathcal{O}(1)$, all maximal non-induced bicliques of G can be listed without duplication in total time $\mathcal{O}(n)$, where n denotes the order of the graph. In their proof, the arboricity is used to ensure the existence of a $2a$-bounded acyclic orientation of the graph. However, in our context, Lemma 21 guarantees that any d-degenerate graph admits a d-bounded orientation. Moreover, Lemma 3 from [12] ensures that such a d-bounded orientation implies the existence of a $2d$-bounded acyclic orientation, which can be computed in linear time. It follows that a graph with degeneracy $d = \mathcal{O}(1)$ admits a $2d$-bounded acyclic orientation, thus allowing the application of Theorem 1 from [12] within our framework. To make use of this result, we adapt the technical lemmas from [12] by replacing their Lemmas 2 and 4 with our Lemmas 21 and 22, respectively. Indeed, in the proof of Lemma 21, we define a collection of sets A_i as the 2-plexes of size at most d in the subgraphs G_j, which enables the efficient listing of non-induced bicliques of the form $A_i \times B_i$. Then, Lemma 20 ensures that every maximal 2-plex in G is included in some maximal biclique of G.

However, this is not sufficient to recover exactly all maximal 2-plexes. Recall that in Lemma 20, it is shown that a maximal 2-plex is composed of a vertex v_i (the vertex with the smallest rank in the 2-plex, with respect to a degeneracy ordering), a set A_i of its neighbors with higher rank at distance one, and a single vertex with the highest rank at distance two. Therefore, given a maximal non-induced biclique $A_i \times B_i$, to find all 2-plexes included in that biclique, it suffices to identify the vertices in B_i that are not neighbors of the vertex v_i. This can be done in time $\mathcal{O}(n \times d)$. According to Theorem 1 of Eppstein [12], the computation of the sets B_i requires $\mathcal{O}(a^3 2^{2a} n)$ time, where a is the arboricity of the graph. Recalling that $d = 2a$, the total time complexity becomes $\mathcal{O}(a^3 2^{2a} n + d2^{2a} n) = \mathcal{O}((a^3 + 2a)2^{2a} n)$, which yields the claimed result. ◀

References

1 A.Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. Versari. D2k: Scalable community detection in massive networks via small-diameter k-plexes. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1272–1281. ACM, 2018. doi:10.1145/3219819.3220093.

2 D. Berlowitz, S. Cohen, and B. Kimelfeld. Efficient enumeration of maximal k-plexes. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 431–444. ACM, 2015. doi:10.1145/2723372.2746484.

3 C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973. doi:10.1145/362342.362367.

4 Y. Cai and M. Xiao. A new exact algorithm for the maximum k-defective clique problem, 2023. arXiv:2309.02635.

5 Y. Cai and M. Xiao. A simple yet effective branching algorithm for the maximum k-defective clique problem, 2024. arXiv:2407.16588.

6 X. Chen, Y. Zhou, J.-K. Hao, and M. Xiao. Computing maximum k-defective cliques in massive graphs. *Computers & Operations Research*, 127:105131, 2021. doi:10.1016/J.COR.2020.105131.

7 S. Cohen, B. Kimelfeld, and Y. Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *Journal of Computer and System Sciences*, 74(7):1147–1159, 2008. doi:10.1016/j.jcss.2007.12.006.

8 A. Conte, P. Foggia, and M. Vento. Exploring community structures using relaxed cliques. *European Journal of Operational Research*, 226(1):103–113, 2013. doi:10.1016/j.ejor.2012.10.031.

9 A. Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. Versari. D2k: Scalable community detection in massive networks via small-diameter k-plexes. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1272–1281. ACM, 2018. doi:10.1145/3219819.3220101.

10 Q. Dai, R.-H. Li, M. Liao, and G. Wang. Maximal defective clique enumeration. *Proceedings of the ACM on Management of Data*, 1(1):Article 77, 2023. doi:10.1145/3588931.

11 Q. Dai, R.-H. Li, and G. Wang. Output-sensitive enumeration of maximal k-plexes of minimum size in massive graphs, 2024. arXiv:2402.13008.

12 D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters*, 51(4):207–211, 1994. doi:10.1016/0020-0190(94)90121-X.

13 T. Gao, Z. Xu, R. Li, and M. Yin. An exact algorithm with new upper bounds for the maximum k-defective clique problem in massive sparse graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 10174–10183, 2022.

14 M. Grbić, A. Kartelj, S. Janković, D. Matić, and V. Filipović. Variable neighborhood search for partitioning sparse biological networks into the maximum edge-weighted k-plexes. *arXiv preprint arXiv:1807.01160*, 2018. arXiv:1807.01160.

15 X. Li, H. Chen, and H. Yan. The identification of protein complexes from weighted ppi networks using a novel core-attachment method. *BMC Bioinformatics*, 11(1):1–13, 2010. doi:10.1186/1471-2105-11-595.

16 Y. Liu, L. Zhou, and Q. Liu. Community detection based on k-plexes in complex networks. *Physica A: Statistical Mechanics and its Applications*, 503:999–1009, 2018. doi:10.1016/j.physa.2018.02.159.

17 G. Manoussakis. A new decomposition technique for maximal clique enumeration for sparse graphs. *Theoretical Computer Science*, 770:25–33, 2019. doi:10.1016/j.tcs.2018.10.014.

18 C. Verma and A. K. Tripathi. Enumeration of k-defective cliques, 2024. arXiv:2407.16588.

19 Z. Wang, Y. Zhou, M. Xiao, and B. Khoussainov. Listing maximal k-plexes in large real-world graphs. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 1517–1527, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3485447.3512198.

20 B. Wu and X. Pei. A parallel algorithm for enumerating all the maximal k-plexes. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 476–483. Springer, 2007. doi:10.1007/978-3-540-71701-0_46.

21 M. Yannakakis. Node- and edge-deletion np-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–264. ACM, 1978. doi:10.1145/800133.804355.

22 H. Yu, X. Zhao, and H. Huang. Discovering protein complexes based on k-plexes in protein interaction networks. *PLOS ONE*, 9(7):e101541, 2014. doi:10.1371/journal.pone.0101541.

23 J. Zhou, X. Chen, J. Huang, and Q. Fang. Efficient community detection based on k-plexes in social networks. *Information Sciences*, 512:1172–1192, 2020. doi:10.1016/j.ins.2019.10.057.

24 Y. Zhou, J. Xu, Z. Guo, M. Xiao, and Y. Jin. Enumerating maximal k-plexes with worst-case time guarantee. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2442–2449, 2020.