Efficient enumeration of maximal induced bicliques

Danny Hermelin, George Manoussakis

▶ **To cite this version:**

Danny Hermelin, George Manoussakis. Efficient enumeration of maximal induced bicliques. Discrete Applied Mathematics, 2021, 303, pp.253-261. ⟨10.1016/j.dam.2020.04.034⟩. ⟨hal-04520508⟩

HAL Id: hal-04520508

https://hal.science/hal-04520508v1

Submitted on 22 Jul 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC BY-NC 4.0 - Attribution - Non-commercial use - International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S0166218X20302365
Manuscript_e948d455d3f0d23be0801999b97c66b8

Efficient Enumeration of Maximal Induced Bicliques

Danny Hermelin, George Manoussakis[1]

Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

Given a graph G of order n, we consider the problem of enumerating all its maximal induced bicliques. We first propose an algorithm running in time $\mathcal{O}(n3^{n/3})$. As the maximum number of maximal induced bicliques of a graph with n vertices is $\Theta(3^{n/3})$, the algorithm is worst-case output size optimal. Then, we prove new bounds on the maximum number of maximal induced bicliques of graphs with respect to their maximum degree Δ and degeneracy k, and propose a near-optimal algorithm with enumeration time $\mathcal{O}(nk(\Delta + k)3^{\frac{\Delta+k}{3}})$. Then, we provide output sensitive algorithms for this problem with enumeration time depending only on the maximum degree of the input graph. Since we need to store the bicliques in these algorithms, the space complexity may be exponential. Thus, we show how to modify them so they only require polynomial space, but with a slight time complexity increase.

1. Introduction

Bicliques have been studied extensively and in many different contexts, as they are well-known and fundamental graph structures. Various applications such as automata and language theory, partial orders, artificial intelligence, and biology are discussed in [1]. These structures also find applications in community detection problems in real world graphs [2], as well as in data mining questions [3]. In chemistry, enumeration of bicliques can be useful for the analysis of the structure of organic compounds [4]. In biology, bicliques are often used for the representation of relationships of heterogeneous data types and their enumeration is an important task [5].

[1]Current address: Université de Versailles Saint-Quentin-En-Yvelines, UFR des sciences, 45 avenue des États-Unis, 78035 Versailles cedex.

© 2020 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

A biclique is a bipartite graph $B = X \cup Y$ such that every vertex of X is adjacent to every vertex of Y. If sets X and Y are non-empty then B is proper, otherwise B is an independent set (or the empty graph). When the requirement that X, Y are independent sets is dropped, B is a non-induced biclique (it is induced otherwise). A biclique is maximal if it is not properly contained in any other biclique. In this paper, we study the question of enumerating all maximal proper induced bicliques of some input graph.

There are a few algorithms for the enumeration of maximal non-induced bicliques. Eppstein [6] proposes an algorithm running in time $\mathcal{O}(a^3 2^{2a} n)$ where a is the arboricity of the input graph (the arboricity is a graph parameter within a constant factor of the degeneracy). Eppstein also proves that the maximum number of maximal non-induced bicliques is lower bounded by $\Omega(2^a n/a)$ and upper bounded by $\mathcal{O}(2^{2a} n)$. Alexe *et al.* [7] propose algorithms whose complexity depends on the size α of the output (that is, the number of maximal non-induced bicliques of the graph). They give *polynomial time delay* algorithms.

An algorithm is polynomial time delay if, between the output of two bicliques, at most some polynomial time (in the input) has elapsed. The complexity of such algorithms is often given as the complexity of the time delay. For example, one algorithm of Alexe *et al.* [7] has time delay $\mathcal{O}(n^3)$ with n the order of the graph. Thus, their algorithm has total enumeration time $\mathcal{O}(\alpha n^3)$, where α is the number of maximal non-induced bicliques of the input graph. We refer to algorithms which have complexity polynomial in the parameters of the graph and the output size, but are not polynomial time delay as *output sensitive* algorithms. Alexe *et al.* [7] also show that enumerating all maximal non-induced bicliques in a graph of order n can be reduced to the problem of enumerating all maximal cliques of a special graph of order $2n$. Other polynomial time delays algorithms have been proposed by Dias *et al.* [8] for this problem.

Results also exist for the enumeration of maximal non-induced bicliques when the input graph is restricted. For instance, Makino and Uno [9] propose a polynomial time delay algorithm with delay $\mathcal{O}(\Delta^2)$ (with Δ the maximum degree of the graph) when the input graph is bipartite. Damaschke [10] also provides polynomial time delay algorithms for bipartite graphs with special degree distributions.

In this paper, we are interested in the enumeration of maximal induced bicliques. Moreover we restrict our study to bicliques which are proper, that is, that have at least an edge. Non-proper bicliques are independent

sets, and we feel that their enumeration is a different problem. Therefore, in the remainder of the paper we consider the problem of enumerating all maximal induced proper bicliques. The lower bound of Prisner [11] $\Omega(3^{n/3})$ combined with upper bound of Gaspers *et al.* [12] $\mathcal{O}(3^{n/3})$ shows that the maximum number of maximal induced proper bicliques is $\Theta(3^{n/3})$, in a graph of order n. Most relevant to the enumeration of such bicliques is the paper of Dias *et al.* [13]. In their work, Dias *et al.* propose a polynomial time delay algorithm enumerating all maximal proper induced biclique in lexicographic order, with time delay $\mathcal{O}(n^3)$ and requiring space to store all such bicliques. Gély *et al.* [14] provide algorithms for the problem with time delay $\mathcal{O}(nm)$ for a general graph or $\mathcal{O}(n^2)$ if the input graph is bipartite, using polynomial space. It is also possible to apply the more general results presented by Cohen *et al.* [15] to our problem. This would yield and algorithm with complexity at least quadratic in the number of vertices. We could also apply the proximity search method of Conte and Uno [16] to this problem and get an algorithm with time delay $\mathcal{O}(n\Delta^3)$. Another approach is using Takata's idea [17, Remark in Sect.6] which yields a $\mathcal{O}(nT)$ delay algorithm with n the number of processes and T the maximum delay of the processes.

Dias *et al.* [13] also prove that, similarly to ideas of Alexe *et al.*, the enumeration of all maximal induced bicliques (proper and non-proper) in a graph G of order n can be reduced to the problem of maximal clique enumeration in some graph of order $2n$ built from copies of the complement of G. However, as observed in the same paper, the number of non-proper bicliques can be exponential in the number of maximal proper bicliques. Consider for example the graph composed of n disjoint edges. It has n maximal proper bicliques but 2^n maximal independent sets. Thus, it seems relevant to design algorithms enumerating specifically all maximal proper induced bicliques.

Our contributions. In this paper, we assume the standard model in which the algorithm must output the actual solutions. We first prove algorithms that are worst-case output-size optimal or near-optimal. An algorithm is worst-case output size optimal if its complexity matches the maximum number of maximal proper induced bicliques of the graph. As discussed above, this number is $\Theta(3^{n/3})$ and these bicliques can be of size n. We provide an algorithm running in optimal worst-case time $\mathcal{O}(n3^{n/3})$. We then prove upper and lower bounds on their maximum number with respect to the maximum degree Δ and the degeneracy k of the graph. We prove a $\Omega(\frac{n}{\Delta+k}3^{\frac{\Delta+k}{3}})$ lower

bound, a $\mathcal{O}(nk3^{\frac{\Delta+k}{3}})$ upper bound and show that maximal proper induced bicliques can be of size $\Delta + k$. Then, we provide an algorithm enumerating them in time $\mathcal{O}(nk(\Delta + k)3^{\frac{\Delta+k}{3}})$. This algorithm is near-optimal as it is at most a factor $k(\Delta + k)$ away from the optimal complexity.

In a third part, we prove an output sensitive algorithm whose enumeration time depends only on the maximum degree Δ (and degeneracy k, but we recall that degeneracy is upper bounded by the maximum degree) of the graph. To the best of our knowledge, this is the first such result. Its complexity is as follows. The algorithm has setup time $\mathcal{O}(kn^2+m\Delta+s(\Delta+k)kn)$ and enumeration time $\alpha\mathcal{O}(r(\Delta + k)k(\Delta + k))$, where $r(\Delta + k)$ (resp. $s(\Delta + k)$) is the time delay (resp. setup) time for maximal clique enumeration in a graph of order $\Delta + k$. For example, we can use the algorithm of Makino and Uno [9]. Their algorithm is polynomial time delay with setup time $\mathcal{O}(n^2)$ and time delay $\mathcal{O}(n^{2.37})$ for a graph of order n. Thus, using their algorithm, our algorithm has setup time $\mathcal{O}(kn^2 + m\Delta + (\Delta + k)^2kn)$ and enumeration time $\alpha\mathcal{O}((\Delta + k)^{2.37}k(\Delta + k)) = \alpha\mathcal{O}(k(\Delta + k)^{3.37})$. Note that our algorithm is not polynomial time delay but only output sensitive. We were not able to bound the delay between the enumeration of two bicliques. It also requires to store the maximal bicliques of the graph. As described in the time complexity, the algorithm depends on our ability to enumerate maximal cliques in a graph of order $\mathcal{O}(\Delta + k)$. Fortunately for us, the maximal clique enumeration problem is a very fundamental question in computer science and there exist many algorithms tackling it. Giving an in-depth description of these algorithms would be slightly beyond the scope of this paper. The ones relevant to our work are the output sensitive and polynomial time delay algorithms, such as [18, 19, 20, 21, 9, 22, 23]. The reader can refer to [20] for a recent overview of the state of the art on this question. Two other important results that we use for our algorithms are related to the maximum number of maximal cliques in a graph of order n. Moon and Moser [24], proved that this number is $\Theta(3^{n/3})$. Then, Tomita *et al.* [25] provided a worst-case output size optimal algorithm running in time $\mathcal{O}(n3^{n/3})$. The results are used in the algorithms of Section 4. In subsection 4.3 we show how to improve the space complexities of the previous algorithms. Since they need to store the solutions in memory, they may need exponential space. Thus, we modify them so they only need polynomial space but at the price of a slightly larger time complexity.

The paper is organized as follows. In Section 2, we give some initial

definitions and prove simple lemmas. In Section 3, we provide preliminary results that we use in Section 4 for the proofs of the main results of the paper.

2. Definitions

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 ordering $\sigma = v_1, ..., v_n$ of the vertices of G, set V_i consists of the vertices following v_i including itself in this ordering, that is, the set $\{v_i, v_{i+1}, ..., v_n\}$. In the ordering σ, the *rank* of v_i, denoted by $\sigma(v_i)$, is its position in the ordering (i in that case). By $[n]$, we denote the set of integers $\{1, 2, ..., n\}$. The distance between two vertices u, 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 (we consider in the paper that $N_i^1(v_i)$ and $N_i(v_i)$ are equal). We now define a family of induced subgraphs of G, with respect to some ordering of its vertices. This construction is inspired by the one provided by Gaspers *et al.* [12].

Definition 1. *Let $G = (V, E)$ be a graph and let $v_1, \ldots, v_n$ be an ordering of its vertices. Graph $G_i, i \in [n]$, is the graph with vertex set $\{v_i\} \cup N_i(v_i) \cup N_i^2(v_i)$ and edges:*

- *$xy \in G_i$, if $x \in N_i(v_i)$, $y \in N_i(v_i)$ and $xy \in E$*
- *$xy \in G_i$, if $x \in N_i^2(v_i)$, $y \in N_i^2(v_i)$ and $xy \in E$*
- *$xy \in G_i$, if $x \in N_i(v_i)$, $y \in N_i^2(v_i)$ and $xy \notin E$.*

A graph G_i consists of the neighbors at distance one and two of v_i with higher rank in the ordering. Its edges are the same as for the graph G except for edges between vertices at distance one and two, which are complemented. For technical reasons, we also add vertex v_i, but note that it is not connected to any vertex in G_i. We state this in the following lemma which we use later.

Lemma 1. *Every maximal independent set of a graph $G_i, i \in [n]$, includes vertex v_i.*

What we will show is that computing all the maximal induced proper bicliques of the graph can be reduced to computing all maximal proper independent set of graphs $G_i, i \in [n]$ (or equivalently, all maximal cliques of their complement). This property can be seen through the following lemma. We say that an independent set I in a graph $G_i, i \in [n]$, is *proper* if it has at least one vertex in $N_i(v_i)$.

Lemma 2. *The vertices of an independent set in a graph $G_i, i \in [n]$, induce a biclique in G.*

Proof. Let I be an independent set in some graph $G_i, i \in [n]$. If $I \cap N_i(v_i) = \emptyset$, then $G[I]$ is an independent set and the lemma holds. Therefore assume now that I is proper. Let $x \in (I \cap N_i(v_i))$ and $y \in I \cap (N_i^2(v_i) \cup \{v_i\})$. By Lemma 1, vertex y exists. Since I is an independent set, then there is no edge between x and y in G_i. This in turn implies that there is an edge between these two vertices in G. This yields the proof. □

We also study this enumeration problem with respect to some parameters of the input graph, namely its maximum degree and degeneracy. The degree of a vertex is its number of neighbors. The maximum degree, denoted Δ, is the maximum degree among the vertices of the graph. The degeneracy of a graph can be defined in a few different ways. The one relevant to this paper is the following. A graph has degeneracy k, or is *k-degenerate*, if there exists an ordering $v_1, ..., v_n$ of its vertices such that for all $i \in [n]$, we have $|N(v_i) \cap V_i| \leq k$ [26]. The degeneracy ordering can be computed in time $\mathcal{O}(m)$ [27]. The idea is essentially to iteratively remove vertices of minimum degree until the graph is empty. The order in which the vertices are removed yields the degeneracy ordering.

In our results we will frequently consider vertex subsets of a graph as strings (consisting of a single unique letter per vertex). We introduce some of the vocabulary that we use in the paper. Let Σ be an alphabet, that is, a non-empty finite set of symbols. Let a string s be any finite sequence of symbols from Σ; s will be a substring of a string t if there exists strings u and v such that $t = usv$. If u or v is not empty then s is a proper substring of t. It will be a *suffix* of t if there exists a string u such that $t = us$. If u is not empty, s is called a *proper suffix* of t. We also use a data structure to store all the proper suffixes of a given string. It is called *suffix tree* in the literature. Given a word of size n, we can construct a suffix tree containing all its suffixes in

space and time $\mathcal{O}(n)$, see [28, 29, 30]. For a set of words $X = \{x_1, x_2, ..., x_r\}$, it is possible to construct a *generalized suffix tree* containing all the suffixes of the words in X in space and time $\mathcal{O}(\sum_{i=1}^{r} |x_i|)$, see [31, chapter 6], and [29] for instance. Once a generalized suffix tree has been constructed, adding a new word x can be done in $\mathcal{O}(|x|)$ time.

3. Preliminary results

In this section we give basic results that we use in the analysis of the different variations of our main algorithm which is studied in Section 4. We first show in Corollary 1 that the maximal induced proper bicliques of a graph G can be related to the maximal proper independent sets of graphs $G_i, i \in [n]$. In the next lemma, we show that the vertices of maximal induced proper bicliques correspond to a maximal independent set in at least one graph G_i, for some $i \in [n]$.

Lemma 3. *Let G be a graph and σ an ordering of its vertices. Let B be a maximal proper induced biclique of G. The vertices $V(B)$ of biclique B form a maximal proper independent set in graph G_v where v is the vertex of B with lowest ranking in σ.*

Proof. Assume that $B = X \cup Y$. Assume without loss of generality that $v \in X$. We have that $V(Y) \subseteq N_{\sigma(v)}(v)$ and $V(X)\backslash\{v\} \subseteq N^2_{\sigma(v)}(v)$. Therefore, $V(B) \subseteq V(G_{\sigma(v)})$ and $V(B)$ is an independent set in $G_{\sigma(v)}$, by Definition 1. It is maximal, or otherwise B is not a maximal biclique, which yields a contradiction. □

In the following lemma and corollary, we prove that the vertices of maximal proper induced bicliques form a maximal independent set in at most one graph $G_i, i \in [n]$. This result, together with the previous lemma yield the proof of Corollary 1.

Lemma 4. *Let G be a graph and $v_1, \ldots, v_n$ an ordering of its vertices. Let B be a maximal independent set of $G_i, i \in [n]$. Then B is not a maximal independent set of any graph $G_j, j \neq i$.*

Proof. Assume that $V(B)$ is a maximal independent set of some graph $G_i, i \in [n]$. Assume by contradiction that there exists $j \in [n]$ with $j \neq i$ such that $V(B)$ is a maximal independent set of G_j. Assume first that $i < j$. By

Lemma 1, vertex v_i belongs to all maximal independent sets of $G_i, i \in [n]$. Therefore, $v_i \in V(B)$. Since we assumed $i < j$, then $v_i \notin V(G_j)$. This implies that $V(B)$ cannot be a maximal independent set of G_j, which yields the contradiction. Assume now that $j < i$. By definition, $V(B)$ is a maximal independent set in G_i. Since $v_j \notin V(G_i)$, then $v_j \notin V(B)$. Therefore, $V(B)$ cannot be maximal in G_j, since by Lemma 3, all its maximal independent sets include vertex v_j. □

Corollary 1. *Let G be a graph and $v_1, \ldots, v_n$ an ordering of its vertices. Let B be a maximal proper induced biclique of G. The vertices $V(B)$ of biclique B form a maximal proper independent set exactly in graph G_v, where v is the vertex of B with lowest ranking in σ.*

In the remainder of the section, we characterize maximal proper independent sets in graphs $G_i, i \in [n]$, which correspond to proper induced bicliques in G which are not maximal.

Lemma 5. *Let G be a graph and $\sigma = v_1, \ldots, v_n$ an ordering of its vertices. Let B be a maximal proper independent set of a graph $G_i, i \in [n]$. Then, $G[B]$ is a proper biclique. Moreover, $G[B]$ is not maximal in G if and only if there exists a proper maximal independent set C of a graph G_j, with $j < i$, and such that $B \subset C$.*

Proof. Assume first that B is a maximal proper independent set of a graph $G_i, i \in [n]$. By Lemma 2, $G[B]$ is a biclique. Since B is proper in G_i, then $G[B]$ is proper as well. Assume now that it is not maximal in G. This implies that there exists a non-empty set A of vertices such that $A \cap B = \emptyset$ and $G[C]$, with $C = B \cup A$ a maximal biclique of G. Therefore, we have that $B \subset V(C)$. Let $v \in A$. By definition $v \notin B$. Thus, $v \notin G_i$ or B is not maximal in G_i. If $\sigma(v) > i$, then v should be included in either $N_i(v_i)$ or $N_i^2(v_i)$, which yields a contradiction. Therefore, assume that $\sigma(v) < i$ (it cannot be equal since A is non-empty and $A \cap B = \emptyset$). Assuming without loss of generality that v is the vertex of A with lowest ranking in σ, we have by Corollary 1 that $V(C)$ is a maximal independent set of graph G_v.

For the other direction, assume that B is a maximal proper independent set of G_i, that C is a maximal proper independent set of G_j with $j < i$, and that $B \subset C$. By Lemma 2, B and C induce bicliques in G. Thus, $G[B]$ is not maximal in G. □

Corollary 2. *Let G be a graph and σ an ordering of its vertices. Let B be a maximal proper independent set of a graph $G_i, i \in [n]$. Then B corresponds to a non-maximal proper biclique of G if and only if:*

- *There exists C, a maximal proper independent set of a graph $G_j, j < i$ such that B is a proper subgraph of C, and*
- *if $W(B)$ and $W(C)$ are the words obtained from the vertices of B and C which have been ordered following σ, then $W(B)$ is a proper suffix of $W(C)$.*

Proof. Similarly as in the proof of Lemma 5, we have that biclique $G[B]$ is well defined. Assume that it is non-maximal. The first point holds by Lemma 5. Since B is a strict subgraph of C, then $V(B) \subset V(C)$. By Lemma 3, since v_i is the vertex of $V(B)$ with smallest ranking in σ_G then it appears first in $W(B)$. We also have that $v_i \in V(C)$. Assume now by contradiction that $W(B)$ is not a proper suffix of $W(C)$. This implies that there exists at least a vertex $x \in V(C) \backslash V(B)$ that appears after vertex v_i in $W(C)$. Thus, vertex x appears after vertex v_i in σ_G. Then, x should be included in either $N_i(v_i)$ or $N_i^2(v_i)$ and this contradicts the maximality of B. The other direction is straightforward. □

4. Algorithms for maximal biclique enumeration

In this section, we give the main results of the paper. In subsection 4.1, we describe various exponential algorithms and show that some of them are optimal or near optimal with respect to the worst-case output size. In subsection 5, we describe an output sensitive algorithm, with enumeration time depending only on the maximum degree of the graph. Results of both subsections are based on different analyses of the complexity of Algorithm 1, described in the next subsection.

4.1. Exponential algorithms

Algorithm 1 is based on the exploitation of some special independent sets of the graphs $G_i, i \in [n]$ introduced in Definition 1 of Section 2. Thus, we first show how to compute these graphs, in the next lemma.

Lemma 6. *Graphs $G_i, i \in [n]$, can be constructed in time $\mathcal{O}(n^3)$ with an adjacency matrix or time $\mathcal{O}(m\Delta)$ with an adjacency list.*

Proof. With an adjacency matrix, for each vertex v_i, we can determine $N_i(v_i)$ in $\mathcal{O}(n)$ time and find the vertices of $N_i^2(v_i)$ in $\mathcal{O}(n^2)$ time. Therefore, we need $\mathcal{O}(n^3)$ time for all graphs $G_i, i \in [n]$. With an adjacency list, we can determine $N_i(v_i)$ in $\mathcal{O}(d_{v_i})$ time. To determine $N_i^2(v_i)$, we need at most $\mathcal{O}(d(v_i)\Delta)$ time. Overall, we need $\mathcal{O}(m\Delta)$ time for all graphs $G_i, i \in [n]$. □

We now present the pseudo-code for Algorithm 1. We show in Theorem 1 that it outputs correctly all maximal induced proper bicliques of the input graph. Then, we give its time complexity in Theorem 2.

Algorithm 1:

Data: A graph G.
Result: All its maximal induced bicliques.

```
1  Consider any ordering σ of the vertices of G.
2  Construct the graphs G_i, i ∈ [n].
3  Initialize T an empty generalized suffix tree.
4  for i = 1 to n do
5      Compute all maximal independent sets of graph G_i.
6      for every maximal independent set I of graph G_i do
7          if I is proper then
8              Order the vertices of I following σ.
9              Search for I in T.
10             if there is a match then
11                 Reject it.
12             else
13                 Insert the proper suffixes of I in T.
14                 Output I.
```

Theorem 1. *Given a graph G, Algorithm 1 outputs exactly all its maximal bicliques, without duplication.*

Proof. By Corollary 1, the vertices of every maximal proper biclique B of the graph correspond to a maximal proper independent set I in exactly one graph $G_i, i \in [n]$. Thus, the set of vertices $V(B) = I$ of B is considered exactly once in the **for** loop starting at Line 4 of the algorithm. If I is

matched in the generalized suffix tree at Line 10 then by Corollary 2, I corresponds to a non maximal biclique in G and this is a contradiction. Thus, Algorithm 1 correctly outputs all maximal proper bicliques, without duplication.

We prove now that it does not output incorrect bicliques. First, by Lemma 2, all maximal independent sets of graphs $G_i, i \in [n]$ correspond to bicliques in G. If some maximal proper independent set I of a graph G_i corresponds to a non-maximal biclique in G, then the following holds, by Corollary 2:

- there exists a maximal proper independent set I' corresponding to a maximal proper biclique of G such that I' belongs to a graph G_j with $j < i$.
- The vertices of I ordered following the ordering form a proper suffix of the word formed by the vertices of I' ordered following the ordering. This implies that I will be rejected in the **if** loop in Line 10.

Therefore, Algorithm 1 outputs exactly all maximal proper bicliques, without duplication, which gives the correction of the algorithm. □

Theorem 2. *Given a graph G of order n, Algorithm 1 runs in time $\mathcal{O}(n3^{n/3})$.*

Proof. Computing the graphs $G_i, i \in [n]$ in Line 2 is done in $\mathcal{O}(n^3)$, by Lemma 6. Computing all the maximal independent sets in Line 5 is done in time $(n-i)3^{(n-i)/3}$ for each G_i, using the algorithm of Tomita *et al.* [25] (which enumerates all maximal cliques in time $\mathcal{O}(n3^{n/3})$ for a graph of order n), and since a graph G_i is of order $\mathcal{O}(n-i)$. Note that the algorithm of Tomita *et al.* enumerates maximal cliques. Thus, we run it on the complement of the graphs $G_i, i \in [n]$. Checking if an independent set is proper can be done in time $\mathcal{O}(n-i)$ by remembering whether a vertex belongs to $N_i(v_i)$ or $N_i^2(v_i)$ when computing graphs $G_i, i \in [n]$. Line 8 can be done in time $\mathcal{O}(n-i)$ by renaming (one time) the vertices of G_i in time $\mathcal{O}(n)$ and inserting the vertices of I in their correct position in an array of size $\mathcal{O}(n-i)$. Searching and inserting I in the suffix tree is done in time $\mathcal{O}(n-i)$ as well. Therefore, overall, the algorithm runs in time $\mathcal{O}(n^3+n^2+\sum_{i=1}^{n}(n-i)3^{n-i/3}) = \mathcal{O}(n3^{n/3})$, as claimed. □

In the next theorem, we prove that with some slight modifications, the complexity of Algorithm 1 can be parameterized by the maximum degree

and degeneracy of the input graph. The main modifications consist first in considering a degeneracy ordering in the vertices instead of just any ordering in Line 1 of the algorithm. Then we provide a new way of computing all maximal proper independent sets of graphs $G_i, i \in [n]$, which will yield the claimed complexity.

Theorem 3. *Given a graph G with maximum degree Δ and degeneracy k, Algorithm 1 can be modified to have setup time $\mathcal{O}(kn^2+m\Delta)$ and enumeration time $\mathcal{O}(nk(\Delta+k)3^{\frac{\Delta+k}{3}})$.*

Proof. We modify the algorithm as follows. First, in Line 1, we consider a degeneracy ordering of the graph. Such an ordering can be computed in time $\mathcal{O}(m)$, as discussed in Section 2. To compute all the proper maximal independent sets in graphs $G_i, i \in [n]$, we proceed slightly differently than in Theorem 2. For each graph $G_i, i \in [n]$ we compute $\mathcal{O}(k)$ graphs G_{ik} of size $\mathcal{O}(k+\Delta)$ as follows. For each vertex $x \in N_i(v_i)$ (since $|N_i(v_i)| \leq k$, there are at most k such graphs), we construct the graph $G_i[\{v_i\} \cup N_i(v_i) \cup (N_i(x) \cap N_i^2(v_i))]$. This graph consists of the neighbors $N_i(v_i)$ of v_i in the ordering and of its neighbors at distance 2 in the ordering, which are in the neighborhood of x. These graphs can be constructed in time $\mathcal{O}(k(k+\Delta))$ for each $G_i, i \in [n]$. Observe that each of these k graphs is of size $\mathcal{O}(\Delta+k)$ since $|N_i(v_i)| \leq k$ and $|(N_i(x) \cap N_i^2(v_i))| \leq \Delta - 1$.

We show that the set of maximal proper independent sets of $G_i, i \in [n]$ is equal to the set of maximal proper independent sets of graphs $G_{ik}, i \in [n]$. We call this property 1. To achieve that, we prove now that an independent set is proper maximal in G_i if and only if it is proper maximal in one of these k graphs.

For the first direction, consider a proper maximal independent set I in G_i. Since I is proper, there exists a vertex x such that $x \in I \cap N_i(v_i)$. Observe now that $I \cap N_i^2(v_i) \subseteq N_i(x) \cap N_i^2(v_i)$ holds since $G[I]$ is a biclique by Lemma 2. This yields the proof for the first direction. For the other direction, It is enough to prove the maximality of I since if it is proper in one graph, it is also proper in the other. Thus, assume by contradiction that an independent set I is proper maximal in one of these k graphs (call it G_i') but proper and not maximal in G_i. Therefore, there exists a non empty set A of vertices such that $A \subseteq V(G_i) \backslash V(G_i')$ and $\{I \cup A\}$ forms a proper maximal independent set in G_i. Recall that graph G_i' is the graph $G_i[\{v_i\} \cup N_i(v_i) \cup (N_i(x) \cap N_i^2(v_i))]$, for some vertex x of $N_i(v_i)$. Let $a \in A$.

First, observe that $a \notin N_i(v_i)$ or I not maximal in G'_i. Thus, necessarily $a \in N_i^2(v_i)$. Since $I \cup \{a\}$ forms an independent set in G_i then there is no edge between vertex x and vertex a, which in turn implies, by definition of G_i, that $a \in N_i(x)$. Observe that this implies that I is not maximal in G'_i and yields a contradiction.

The second property (that we call property 2) that we prove is that a maximal independent set I of a graph G_{ik} is always proper (that is has at least one vertex in $N_i(v_i)$). To prove that, recall that each of these graphs consists of all the vertices of $N_i(v_i)$ and of the neighbors of one vertex $x \in N_i(v_i)$. This implies that x belongs to all maximal independent sets of G_{ik}, since there are no edges between x and $N_i(x) \cap N_i^2(v_i)$ in G_i.

Therefore, by the previous two properties that we proved above, to find all the maximal proper independent sets of G_i, it is enough to compute all maximal independent sets of graphs G_{ik}. Since there are k such graphs and that they are of size at most $\mathcal{O}(\Delta + k)$ then they have at most $\mathcal{O}(k3^{\frac{\Delta+k}{3}})$ independent sets (by the Moon and Mooser bound [24]). We can generate all these graphs in time $\mathcal{O}(k(\Delta + k)3^{\frac{\Delta+k}{3}})$ using the algorithm of Tomita *et al.* [25]. This gives the time needed to execute Line 5 of Algorithm 1. Checking if an independent set is proper can be done $\mathcal{O}(\Delta + k)$ using the same argument as in Theorem 2. To order the vertices of an independent set I following the degeneracy ordering in Line 8 we proceed as follows. We first rename the vertices of each graph G_{ik} (one time). This takes time $\mathcal{O}(n)$ for each such graph and $\mathcal{O}(n^2k)$, overall (since there are $\mathcal{O}(nk)$ such graphs). Then, we insert the vertices of each independent set I in their correct position in an array of size $\mathcal{O}(\Delta + k)$, with same time complexity. Searching and inserting I in the suffix tree is done in time $\mathcal{O}(\Delta + k)$ as well. Overall, this modified version of Algorithm 1 has preprocessing time $\mathcal{O}(kn^2 + m\Delta)$ and enumeration time $\mathcal{O}(nk(\Delta + k)3^{\frac{\Delta+k}{3}})$, as claimed.

We prove now the correction of this modified version of Algorithm 1. Essentially, the proof is the same as for Theorem 1. All maximal proper independent sets of graphs $G_i, i \in [n]$, will be computed in this modified version, as proved in the second paragraph. Thus, we have that only maximal proper bicliques will be outputted by Algorithm 1. However, duplicates maximal proper independent sets can also be computed (two duplicates may appear in two different graphs G_{ik} and $G_{ik'}$ for some $i \in [n]$). Let I be such a duplicate proper maximal independent set. Consider the first time it is generated by the algorithm. If it is matched in the suffix tree, then it will be

rejected and all its duplicates as well. If it is not matched, it will be inserted in the suffix tree, and all its duplicates will be rejected (since they will be in turn matched in the tree). Thus, the correctness holds. □

Corollary 3. *A graph G of order n with maximum degree Δ and degeneracy k has at most $\mathcal{O}(nk3^{\frac{\Delta+k}{3}})$ maximal induced proper bicliques.*

Proof. The algorithm described in Theorem 3 generates at most $\mathcal{O}(nk3^{\frac{\Delta+k}{3}})$ candidate vertex sets corresponding to the maximal induced proper bicliques of the input graph. The claim thus follows from the correctness of our algorithm. □

To conclude the subsection, we show that the algorithm described in Theorem 3 has almost optimal complexity, up to a small function of the maximum degree and degeneracy of the graph.

Theorem 4. *For any tuple of integers k, Δ, n such that $k \leq \Delta \leq n$, we can construct a graph G of order n with maximum degree Δ and degeneracy k such that G has $\Omega(\frac{n}{\Delta+k}3^{\frac{\Delta+k}{3}})$ maximal induced proper bicliques.*

Proof. Assume that $\Delta \equiv 0 \pmod 3$ and $k \equiv 0 \pmod 3$. Consider graph B consisting of the two following sets of vertices. Let X be a set of $\Delta/3$ disjoint triangles and let set Y consists of $k/3$ disjoint triangles. Connect all the vertices of X to all the vertices of Y. The maximum degree of B is Δ. It has degeneracy k. To prove that, observe that every vertex has at least degree k which implies that the graph cannot have degeneracy $k-1$. We can easily find an ordering of the vertices such that every vertex has at most k neighbors with higher rank in the ordering. To do that, simply put first in the ordering the vertices of X (in any order) followed by the vertices of Y (in any order). This proves that B has degeneracy k. The number of maximal induced proper bicliques of this graph is $3^{\Delta/3}3^{k/3} = 3^{\frac{\Delta+k}{3}}$. Consider graph G consisting of $\frac{n}{\Delta+k}$ disjoint copies of B. It is of order n, has maximum degree Δ, degeneracy k and has the claimed number of maximal induced proper bicliques.

For the other possible combinations of values of $\Delta \pmod 3$ and $k \pmod 3$, the proofs are very similar. For example, when $\Delta \equiv 1 \pmod 3$, it is sufficient to replace one triangle in X by a square. □

Lemma 7. *Let G be a graph of order n, degeneracy k and maximum degree Δ. Let σ be a degeneracy ordering of G. A maximal induced biclique of G is of size at most $\Delta + k$.*

Proof. Let $X = Y \cup Z$ be a maximal proper induced biclique of G. Let $y \in Y$ be the vertex of X with smallest rank in σ. By definition of a degeneracy ordering, we have that $|N_{\sigma(x)}(x)| \leq k$, which implies that $|V(Z)| \leq k$. Let $z \in Z$, we have that $Y \subseteq N(z)$, which implies that $|Y| \leq \Delta$. □

To conclude the section, note that the lower bound proved in Theorem 4 implies that any optimal algorithm requires at least time $\Omega(n3^{\frac{\Delta+k}{3}})$ in the worst-case to output all maximal induced proper bicliques, since they are of size $\mathcal{O}(\Delta+k)$, as proved in Lemma 7. By Corollary 3, any optimal algorithm requires at most time $\mathcal{O}(nk(\Delta + k)3^{\frac{\Delta+k}{3}})$. Thus, our algorithm described in Theorem 3 is at most a factor $\mathcal{O}(k(\Delta+k))$ away from the optimal complexity.

4.2. Output sensitive algorithms

In this section we start by proving some additional results concerning the number of maximal proper independent sets in graphs $G_i, i \in [n]$, which correspond to non-maximal proper bicliques in graph G. This is done in Lemma 8 and Corollary 4. Using these new results, we prove that Algorithm 1 can be slightly modified in order to achieve a complexity which is output sensitive. This result is stated in Theorem 5.

Lemma 8. *Let G be a graph of order n and σ an ordering of its vertices. Let α denote the number of maximal proper induced bicliques of G and α_i the number of maximal proper independent sets of graph $G_i, i \in [n]$. We have that $\sum_{j=1}^{n} \alpha_j \leq \alpha n$.*

Proof. Let max_i denotes the number of maximal proper independent sets of $G_i, i \in [n]$ which correspond to maximal proper bicliques of G, and $Nmax_i$ the number of maximal proper independent sets of $G_i, i \in [n]$, which correspond to non-maximal proper bicliques of G. We have that $\alpha_i = max_i + Nmax_i$. By Corollary 1, the vertices of every maximal proper biclique of G correspond to a maximal proper independent set of exactly one graph $G_i, i \in [n]$. This implies that $\sum_{i=1}^{n} max_i = \alpha$.

Let X be the multiset of independent sets which are maximal and proper in graphs $G_i, i \in [n]$ but do not correspond to maximal and proper induced

bicliques in G. Let $x \in X$. By Corollary 2, the word obtained from the vertices of x which have been ordered following σ is a proper suffix of the word obtained from ordering the vertices, following σ, of some maximal proper induced biclique of G. This implies that X has at most $(n-1)\alpha$ different elements since a maximum biclique of G has at most n vertices and that a word with n letters has at most $n-1$ proper suffixes. Moreover, Lemma 4 implies that x is maximal in an unique graph $G_i, i \in [n]$, which implies that $\sum_{j=1}^{n} Nmax_j \leq (n-1)\alpha$. Thus, in total $\sum_{j=1}^{n} \alpha_j \leq \alpha + (n-1)\alpha = \alpha n$. □

Corollary 4. *Let G be a graph of order n, degeneracy k, maximum degree Δ and let σ be a degeneracy ordering of its vertices. Let α denote the number of maximal proper induced bicliques of G and α_i the number of maximal proper independent sets of graph $G_i, i \in [n]$. We have that $\sum_{j=1}^{n} \alpha_j \leq \alpha(\Delta + k)$.*

Proof. The proof is exactly the same as for Lemma 8, except that in this case, by Lemma 7, a maximal proper induced biclique of G is of size at most $\Delta + k$. □

We are now ready to describe the output sensitive algorithm for maximal biclique enumeration in the next theorem.

Theorem 5. *Given a graph G with maximum degree Δ and degeneracy k, Algorithm 1 can be modified to have setup time $\mathcal{O}(kn^2+m\Delta+s(\Delta+k)kn)$ and enumeration time $\alpha\mathcal{O}(r(\Delta+k)k(\Delta+k))$, where $r(\Delta+k)$ (resp. $s(\Delta+k)$) is the time delay (resp. setup time) for maximal clique enumeration in a graph of order $\Delta + k$.*

Proof. The proof is very similar to the one of Theorem 3. We recall its main steps. In Line 1 of Algorithm 1, we consider a degeneracy ordering of the graph. To compute all the proper maximal independent sets in graphs $G_i, i \in [n]$, we first compute $\mathcal{O}(k)$ graphs G_{ik} of size $\mathcal{O}(k + \Delta)$ as follows. For each vertex $x \in N_i(v_i)$, we construct the graph $G_{ix} = G_i[\{v_i\} \cup N_i(v_i) \cup (N_i(x) \cap N_i^2(v_i))]$. This graph consists of the neighbors $N_i(v_i)$ of v_i in the ordering and of its neighbors at distance 2 in the ordering which are in the neighborhood of x. These graphs can be constructed in time $\mathcal{O}(nk)$ for each $G_i, i \in [n]$ and they are of size $\mathcal{O}(\Delta + k)$. In the proof of Theorem 3, we proved that the set of maximal proper independent sets of $G_i, i \in [n]$, is equal to set of maximal independent sets of graphs $G_{ik}, i \in [n]$.

The main difference with the algorithm described in Theorem 3 lies in the way we compute all maximal independent sets of these graphs $G_{ik}, i \in [n]$. In Theorem 3 we used the worst-case output size optimal algorithm of Tomita *et al.* [25]. Here we use any polynomial time delay or output sensitive algorithm for maximal clique enumeration (such as [18, 19, 20, 21, 9, 22, 23]). For a given graph G_i, there are k such graphs G_{ik} and they are of size at most $\mathcal{O}(\Delta + k)$. To get their maximal independent sets, we compute the maximal cliques of their complement graph. Let α_i be the number of maximal proper independent sets of graph G_i and let α_{ik} be the number of maximal independent sets of graph G_{ik}. We proved that $\sum_{x \in N_i(v_i)} \alpha_{ix} \leq k\alpha_i$ (this holds because of the two properties in the proof of Theorem 3 and because there can be at most k duplicates of some maximal independent set in the graphs G_{ik}). Let $\mathcal{O}(r(\Delta + k))$ be the time delay for maximal clique enumeration for a graph of order $\Delta + k$. Computing all the maximal independent sets of a graph G_{ik} can be done in time $\alpha_{ik}\mathcal{O}(r(\Delta + k))$, since these graphs (and therefore their independent sets) are of order at most $\Delta+k$. Thus, computing all the maximal proper independent sets of G_i can be done in $\sum_{x \in N_i(v_i)} (\alpha_{ix}\mathcal{O}(r(\Delta + k))) \leq \alpha_i\mathcal{O}(r(\Delta + k)k)$. Thus, to compute all the maximal proper independent sets of graph G takes $\sum_{i=1}^{n} \alpha_i\mathcal{O}(r(\Delta + k)k) \leq \alpha\mathcal{O}(r(\Delta + k)k(\Delta + k))$, by Corollary 4.

For the last part of Algorithm 1, that is, to sort the vertices of computed independent sets and search and insert them in the suffix tree, we proceed as in the proof of Theorem 3. This takes time $\mathcal{O}(\Delta + k)$ for each computed independent set. Overall, this last step requires $\mathcal{O}(\sum_{i=1}^{n} \sum_{x \in N_i(v_i)} \alpha_{ix}(\Delta + k)) \leq \alpha\mathcal{O}(k(\Delta + k)^2)$.

We give now the complexity of the setup phase of this modified version of Algorithm 1. The first modification is to compute a degeneracy ordering. This takes time $\mathcal{O}(m)$, as discussed earlier. Then, we compute n graphs $G_i, i \in [n]$. This takes time $\mathcal{O}(m\Delta)$ by Lemma 6. Then, we need to compute, for each of these graphs, k graphs G_{ik}. This takes overall time $\mathcal{O}(n^2k)$ as seen in the proof of Theorem 3. Last, to compute all the independent sets of these graphs using a polynomial time delay or output sensitive algorithm of [18, 19, 20, 21, 9, 22, 23], we need to preprocess them. Since there are overall $\mathcal{O}(kn)$ such graphs, and that they are of order at most $\Delta + k$, this

takes time $\mathcal{O}(s(\Delta+k)kn)$, where $s(\Delta+k)$ is the preprocessing time of the chosen algorithm for a graph of order $\Delta+k$.

The proof of correction of this modified version of Algorithm 1 is similar to the one provided for the algorithm described in Theorem 3. □

4.3. Improving the space complexity

In this section, we show how to improve the space complexities of the previous algorithms. Since we need to store the solutions in the suffix trees, then their space complexity is exponential if the number of solutions is exponential. We can achieve polynomial space by getting rid of the suffix trees altogether, and checking the maximality and uniqueness of the generated induced bicliques with other methods. We describe this process for each of the three algorithms.

Theorem 6 improves the space complexity of the algorithm with time complexity $\mathcal{O}(n3^{n/3})$ whose correctness and complexity are proved in Theorems 1 and 2, respectively. However, as a result, the time complexity becomes a bit larger, and we lose the worst-case output size optimality.

Theorem 6. *Algorithm 1 can be modified to use polynomial space but requires $\mathcal{O}(n\Delta^3)$ additional time.*

Proof We remove in Algorithm 1 the suffix tree, initialized in Line 2. For each maximal independent set I of graph G_i found in Line 5, we do the following. By Lemma 1, set I includes vertex v_i. Therefore, if there exists a vertex $x \in (N(v_i) \cup N^2(v_i))\backslash(N_i(v_i) \cup N_i^2(v_i))$ such that $I \cup \{x\}$ forms a biclique in G then $G[I]$ is not a maximal biclique of the graph. There are at most $\mathcal{O}(\Delta^2)$ such candidate vertices and checking each of them takes time $\mathcal{O}(n\Delta)$ with adjacency lists. □

Theorem 7. *The algorithm described in Theorem 3 can be modified to use polynomial space but requires $\mathcal{O}(k\Delta^3)$ additional time. Similary, the algorithm described in Theorem 5 can be modified to use polynomial space but requires additional $\mathcal{O}(k\Delta^3)$ enumeration time.*

Proof Recall that the algorithms described in Theorem 3 and Theorem 5 generate $\mathcal{O}(k)$ graphs G_{ik} for each vertex $x \in N_i(v_i)$. Assume that these graphs are considered for each x in increasing index. Let $G_{ik'}$ be such a graph, x the corresponding vertex in $N_i(v_i)$, and let I be a proper maximal

independent set of this graph, considered by the algorithm. To check if some duplicate biclique has already been outputted, we proceed as follows. Let x' be the lowest rank vertex of $I \cap N_i(v_i)$. If $x' = x$ then we check if I is maximal, similarly as in the proof of Theorem 6 and output it accordingly. Otherwise we do not output it because I has already been processed when graph $G_{ik''}$ corresponding to vertex x' was considered by the algorithm. The complexity analysis is similar to the one in Theorem 6, except that, since the considered bicliques are of size $\mathcal{O}(\Delta + k)$, we need $\mathcal{O}(k\Delta^3)$ additional time.

□

5. Acknowledgements

This research was supported by Grant No. 2016049 from the United States-Israel Binational Science Foundation (BSF).

References

[1] J. Amilhastre, M. C. Vilarem, P. Janssen, Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs, Discrete Applied Mathematics 86 (2) (1998) 125 – 144. doi:https://doi.org/10.1016/S0166-218X(98)00039-0.

[2] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, Trawling the web for emerging cyber-communities, Computer networks 31 (11-16) (1999) 1481–1493.

[3] M. J. Zaki, Scalable algorithms for association mining, IEEE transactions on knowledge and data engineering 12 (3) (2000) 372–390.

[4] S. Koichi, M. Arisaka, H. Koshino, A. Aoki, S. Iwata, T. Uno, H. Satoh, Chemical structure elucidation from 13c nmr chemical shifts: Efficient data processing using bipartite matching and maximal clique algorithms, Journal of chemical information and modeling 54 (4) (2014) 1027–1035.

[5] Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, M. A. Langston, On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types, BMC bioinformatics 15 (1) (2014) 110.

[6] D. Eppstein, Arboricity and bipartite subgraph listing algorithms, Information Processing Letters 51 (4) (1994) 207 – 211. doi:https://doi.org/10.1016/0020-0190(94)90121-X.
URL http://www.sciencedirect.com/science/article/pii/002001909490121X

[7] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, B. Simeone, Consensus algorithms for the generation of all maximal bicliques, Discrete Applied Mathematics 145 (1) (2004) 11–21.

[8] V. M. Dias, C. M. de Figueiredo, J. L. Szwarcfiter, On the generation of bicliques of a graph, Discrete Applied Mathematics 155 (14) (2007) 1826 – 1832, 3rd Cologne/ Twente Workshop on Graphs and Combinatorial Optimization. doi:https://doi.org/10.1016/j.dam.2007.03.017.
URL http://www.sciencedirect.com/science/article/pii/S0166218X07000649

[9] K. Makino, T. Uno, New algorithms for enumerating all maximal cliques, in: Scandinavian Workshop on Algorithm Theory, Springer, 2004, pp. 260–272.

[10] P. Damaschke, Enumerating maximal bicliques in bipartite graphs with favorable degree sequences, Information Processing Letters 114 (6) (2014) 317 – 321. doi:https://doi.org/10.1016/j.ipl.2014.02.001.
URL http://www.sciencedirect.com/science/article/pii/S0020019014000192

[11] E. Prisner, Bicliques in graphs i: Bounds on their number, Combinatorica 20 (1) (2000) 109–117. doi:10.1007/s004930070035.
URL https://doi.org/10.1007/s004930070035

[12] S. Gaspers, D. Kratsch, M. Liedloff, On independent sets and bicliques in graphs, in: International Workshop on Graph-Theoretic Concepts in Computer Science, Springer, 2008, pp. 171–182.

[13] V. M. F. Dias, C. M. de Figueiredo, J. L. Szwarcfiter, Generating bicliques of a graph in lexicographic order, Theoretical Computer Science 337 (1) (2005) 240 – 248. doi:https://doi.org/10.1016/j.tcs.2005.01.014.
URL http://www.sciencedirect.com/science/article/pii/S030439750500037X

[14] A. Gély, L. Nourine, B. Sadi, Enumeration aspects of maximal cliques and bicliques, Discrete Applied Mathematics 157 (7) (2009) 1447 – 1459. doi:https://doi.org/10.1016/j.dam.2008.10.010.
URL http://www.sciencedirect.com/science/article/pii/S0166218X08004563

[15] S. Cohen, B. Kimelfeld, Y. Sagiv, Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties, Journal of Computer and System Sciences 74 (7) (2008) 1147 – 1159. doi:https://doi.org/10.1016/j.jcss.2008.04.003.
URL http://www.sciencedirect.com/science/article/pii/S0022000008000512

[16] A. Conte, T. Uno, New polynomial delay bounds for maximal subgraph enumeration by proximity search, in: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019, pp. 1179–1190.

[17] K. Takata, Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph, Discrete Applied Mathematics 158 (15) (2010) 1660–1667.

[18] L. Chang, J. X. Yu, L. Qin, Fast maximal cliques enumeration in sparse graphs, Algorithmica 66 (1) (2013) 173–186.

[19] C. Comin, R. Rizzi, An improved upper bound on maximal clique listing via rectangular fast matrix multiplication, Algorithmica (Jan 2018). doi:10.1007/s00453-017-0402-5.
URL https://doi.org/10.1007/s00453-017-0402-5

[20] A. Conte, R. Grossi, A. Marino, L. Versari, Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques, in: 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), Vol. 148, 2016, pp. 1–148.

[21] D. S. Johnson, M. Yannakakis, C. H. Papadimitriou, On generating all maximal independent sets, Information Processing Letters 27 (3) (1988) 119 – 123. doi:http://dx.doi.org/10.1016/0020-0190(88)90065-8.

[22] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa, A new algorithm for generating all the maximal independent sets, SIAM Journal on Computing 6 (3) (1977) 505–517.

[23] G. Manoussakis, An output sensitive algorithm for maximal clique enumeration in sparse graphs, in: 12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria, 2017, pp. 27:1–27:8. doi:10.4230/LIPIcs.IPEC.2017.27.
URL https://doi.org/10.4230/LIPIcs.IPEC.2017.27

[24] J. W. Moon, L. Moser, On cliques in graphs, Israel journal of Mathematics 3 (1) (1965) 23–28.

[25] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363 (1) (2006) 28–42.

[26] D. R. Lick, A. T. White, d-degenerate graphs, Canad. J. Math. 22 (1970) 1082–1096.
URL http://www.smc.math.ca/cjm/v22/p1082

[27] V. Batagelj, M. Zaversnik, An $\mathcal{O}(m)$ algorithm for cores decomposition of networks, CoRR cs.DS/0310049 (2003).

[28] E. M. McCreight, A space-economical suffix tree construction algorithm, J. ACM 23 (2) (1976) 262–272. doi:10.1145/321941.321946.
URL http://doi.acm.org/10.1145/321941.321946

[29] E. Ukkonen, On-line construction of suffix trees, Algorithmica 14 (3) (1995) 249–260. doi:10.1007/BF01206331.
URL http://dx.doi.org/10.1007/BF01206331

[30] P. Weiner, Linear pattern matching algorithms, in: Switching and Automata Theory, 1973. SWAT '08. IEEE Conference Record of 14th Annual Symposium on, 1973, pp. 1–11. doi:10.1109/SWAT.1973.13.

[31] D. Gusfield, Algorithms on strings, trees and sequences: computer science and computational biology, Cambridge University Press, 1997.