An Output Sensitive Algorithm for Maximal Clique Enumeration in Sparse Graphs

George Manoussakis

LRI-CNRS, Université Paris Sud, Université Paris Saclay, France
georgе@lri.fr

Abstract

The *degeneracy* of a graph G is the smallest integer k such that every subgraph of G contains a vertex of degree at most k. Given an n-order k-degenerate graph G, we present an algorithm for enumerating all its maximal cliques. Assuming that α is the number of maximal cliques of G, our algorithm has setup time $\mathcal{O}(n(k^2 + s(k+1)))$ and enumeration time $\alpha\mathcal{O}((k+1)f(k+1))$ where $s(k+1)$ (resp. $f(k+1)$) is the preprocessing time (resp. enumeration time) for maximal clique enumeration in a general $(k+1)$-order graph. This is the first output sensitive algorithm whose enumeration time depends only on the degeneracy of the graph.

1998 ACM Subject Classification G.2.2 Graph Theory, Graph Algorithms

Keywords and phrases Enumeration algorithms, maximal cliques, k-degenerate graphs

Digital Object Identifier 10.4230/LIPIcs.IPEC.2017.27

1 Introduction

Degeneracy, introduced by Lick *et al.* [12], is a common measure of the sparseness of a graph and is closely related to other sparsity measures such as arboricity and thickness. Degenerate graphs often appear in practice. For instance, the World Wide Web graph, citation networks, and collaboration graphs have low arboricity, and therefore have low degeneracy [18]. Furthermore, planar graphs have degeneracy at most five [12] and the Barabàsi-Albert model of preferential attachment [9], frequently used as a model for social networks, produces graphs with bounded degeneracy.

Cliques are complete subgraphs of a graph. The problem of listing all maximal cliques in general and k-degenerate graphs has been extensively studied. We can essentially distinguish between two families of algorithms. On one side, *worst-case output size* algorithms have been proposed. Their complexities match the maximal number of maximal cliques one can find in the considered graphs. For instance, Tomita *et al.* [16] propose an algorithm enumerating all maximal cliques of a general n-order graph in time $\mathcal{O}(3^{n/3})$. This is worst-case output size optimal in general graphs as for instance the Moon-Moser graphs have $\Theta(3^{n/3})$ cliques [3, 15]. Thus, even printing the cliques of these graphs would require at least $\Omega(3^{n/3})$ time. Similarly, for k-degenerate graphs, Eppstein *et al.* [8] prove a $\mathcal{O}((n-k)3^{k/3})$ bound on the maximal number of maximal cliques and then show an algorithm running in time $\mathcal{O}(k(n-k)3^{k/3})$. The two algorithms described above rely on ideas of the Bron-Kerbosch algorithm [2]. These results are summarized in the first three rows of Table 1. This table is largely inspired by the one provided by Conte *et al.* [7].

Another family is the one of *polynomial delay output sensitive* algorithms. Their time complexities can be divided into a preprocessing phase followed by an enumeration phase. During the enumeration phase, maximal cliques of the graph are outputted with polynomial delay: the wait between the output of two maximal cliques is bounded by some polynomial

Table 1 Bounds for maximal clique enumeration where $q-1 \leq k \leq \Delta \leq n-1$. $^+$: these are polynomial time delay algorithms. Their delay is equal to their enumeration time divided by the number of maximal cliques α. The space bounds do not include the space needed to store the graph.

Algorithm	Setup	Enumeration	Space
Bron-Kerbosch [2]	$\mathcal{O}(m)$	unbounded	$\mathcal{O}(n+q\Delta)$
Tomita *et al.* [16]	$\mathcal{O}(m)$	$\mathcal{O}(3^{n/3})$	$\mathcal{O}(n+q\Delta)$
Eppstein *et al.* [8]	$\mathcal{O}(m)$	$\mathcal{O}(k(n-k)3^{k/3})$	$\mathcal{O}(n+k\Delta)$
Johnson *et al.* [11]$^+$	$\mathcal{O}(mn)$	$\alpha\mathcal{O}(mn)$	$\mathcal{O}(\alpha n)$
Tsukiyama *et al.* [17]$^+$	$\mathcal{O}(n^2)$	$\alpha\mathcal{O}((n^2-m)n)$	$\mathcal{O}(n^2)$
Chiba *et al.* [5]$^+$	$\mathcal{O}(m)$	$\alpha\mathcal{O}(mk)$	$\mathcal{O}(m)$
Makino *et al.* [13]$^+$	$\mathcal{O}(mn)$	$\alpha\mathcal{O}(\Delta^4)$	$\mathcal{O}(m)$
Chang *et al.* [4]$^+$	$\mathcal{O}(m)$	$\alpha\mathcal{O}(\Delta h^3)$	$\mathcal{O}(m)$
Makino *et al.* [13]$^+$	$\mathcal{O}(n^2)$	$\alpha\mathcal{O}(n^{2.37})$	$\mathcal{O}(n^2)$
Comin *et al.* [6]$^+$	$\mathcal{O}(n^{5.37})$	$\alpha\mathcal{O}(n^{2.09})$	$\mathcal{O}(n^{4.27})$
Conte *et al.* [7]$^+$	$\mathcal{O}(m\log^{\mathcal{O}(1)}(m+n))$	$\alpha\mathcal{O}(qd(\Delta+qd)\log^{\mathcal{O}(1)}(m+n))$	$\mathcal{O}(q)$
Conte *et al.* [7]$^+$	$\mathcal{O}(m\log^{\mathcal{O}(1)}(m+n))$	$\alpha\mathcal{O}(\min\{mk, qk\Delta\}\log^{\mathcal{O}(1)}(m+n))$	$\mathcal{O}(k)$

Δ = max degree k = degeneracy q = maximum clique size
α = number of maximal cliques
h = smallest integer such that $|\{v \in V : |N(v)| \geq h\}|$, where $k \leq h \leq \Delta$.

in the parameters of the graph. For example, the algorithm of Johnson *et al.* [11] has setup time $\mathcal{O}(mn)$ and polynomial time delay $\mathcal{O}(mn)$. Thus, after the setup phase, this algorithm requires $\alpha\mathcal{O}(mn)$ time, α being the number of maximal cliques, to output all the maximal cliques of the graph. It is *output sensitive* since the enumeration time depends on the number of maximal cliques of the graph. All the algorithms that fall into this category are listed in the last nine rows of Table 1. For these specific algorithms, the time delay is equal to the enumeration time divided by α, the number of maximal cliques.

Our contribution. Given a k-degenerate graph, we present an output sensitive algorithm with setup time $\mathcal{O}(n(k^2+s(k+1)))$ and enumeration time $\alpha\mathcal{O}((k+1)f(k+1))$, where $s(k+1)$ (resp. $f(k+1)$) is the preprocessing time (resp. enumeration time) for maximal clique enumeration in a general $(k+1)$-order graph. For example, using the algorithm of Makino *et al.* [13] which has setup time $\mathcal{O}((k+1)^2)$ and enumeration time $\mathcal{O}((k+1)^{2.37})$ for a $(k+1)$-order graph, our algorithm has setup time $\mathcal{O}(n(k^2+(k+1)^2))$ and enumeration time $\alpha\mathcal{O}((k+1)(k+1)^{2.37})$. Since in a $(k+1)$-order graph all the graph parameters (number of edges and vertices, the maximum degree, the clique size, etc.) are bounded by some function of k, our algorithm will always have enumeration time depending only on the degeneracy of the graph, whatever output sensitive algorithm of Table 1 we use. This is the first such algorithm.

On the downside, we were not able to prove that our algorithm has polynomial time delay. It also requires that the maximum cliques be stored. Thus, since the maximal cliques of a k-degenerate graph are of size at most $k+1$, our algorithm needs $\mathcal{O}((k+1)\alpha)$ space, besides the space needed to store the graph (in our case, the graph can be stored using adjacency lists). Further improvements are discussed in the conclusion.

The organization of the document is as follows. In Section 2 we introduce some notations and definitions. In Section 3 we prove basic results. These results are used in Section 4 to prove the correctness and time complexity of Algorithm 1, which is the main contribution of the paper.

2 Definitions

2.1 Graph terminologies

We consider graphs of the form $G = (V, E)$ which are simple, undirected, connected, with n vertices and m edges. We assume that they are stored in memory using adjacency lists. If $X \subset V$, the subgraph of G induced by X is denoted by $G[X]$. 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 vertex adjacent to x in G. *The closed neighborhood* of x is defined as $N[x] = N(x) \cup x$. Given an ordering $v_1, ..., v_n$ of the vertices of G, V_i is the set of vertices following v_i including itself in this ordering, that is, the set $\{v_i, v_{i+1}, ..., v_n\}$. By G_i we denote the induced subgraph $G[N[v_i] \cap V_i]$. A graph is *k-degenerate* if there is an ordering $v_1, ..., v_n$ of its vertices such that for all i, $1 \leq i \leq n$, $|N(v_i) \cap V_i| \leq k$. The degeneracy ordering can be computed in $\mathcal{O}(m)$ time [1]. Given a graph G we will denote by σ_G its degeneracy ordering and if $x \in V(G)$ then $\sigma_G(x)$ will be the ranking of x in σ_G.

2.2 Word terminologies

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.

3 Basic results

When not specified, we always assume that, given a k-degenerate graph G, we have its degeneracy ordering, denoted by σ_G. When referring to an ordering of the vertices, we always refer to σ_G. The family of subgraphs $G_i, i \in [n]$ described in Section 2.1 will always be constructed following the degeneracy ordering of G. Thus, these graphs have at most $k + 1$ vertices, since in a degeneracy ordering $v_1, ..., v_n$ of the vertices of G, the inequality $|N[v_i] \cap V_i| \leq k + 1$ holds.

We want to show in this section that, roughly, given some k-degenerate graph G, it is enough to compute all the maximal cliques of the induced subgraphs $G_i, i \in [n]$ to get all the maximal cliques of G. We first start by proving that the induced subgraphs $G_i, i \in [n]$ can be easily computed. To prove that we first introduce a special adjacency structure, in the following definition.

▶ **Definition 1.** Let $G = (V, E)$ be a k-degenerate graph. Assume that G is given by the adjacency lists for each vertex. The degenerate adjacency list of a vertex $x \in V$ is its adjacency list in which every vertex that has lower ranking that x in σ_G has been deleted.

▶ **Lemma 2.** *The degenerate adjacency lists of a n-order k-degenerate graph G can be computed in time $\mathcal{O}(m)$.*

Proof. Compute the degeneracy ordering σ_G of G. As described before this can be done in $\mathcal{O}(m)$ time. Assume that we have the adjacency lists of G. Let $x \in V$ and let d_x be its degree. In time $\mathcal{O}(d_x)$ remove all vertices from its adjacency lists that have lower ranking in σ_G. Repeat the procedure for all the vertices of the graph. This is done in total time $\mathcal{O}(m)$. ◀

▶ **Lemma 3.** *Given a k-degenerate graph G, there is an algorithm constructing the induced subgraphs $G_i, i \in [n]$ in time $\mathcal{O}(nk^2)$ and $\mathcal{O}(m)$ space.*

Proof. Compute the degenerate adjacency lists of G. This is done in time $\mathcal{O}(m)$ by Lemma 2. Observe first that the vertex set of graph $G_i, i \in [n]$ corresponds to i-th vertex of σ_G plus all the vertices of its degenerate adjacency list. Thus it only remains to show how to compute the adjacency lists of each of these graphs. We proceed as follows. For every vertex $x \in V(G_i)$, go through its degenerate adjacency list and remove vertices which are not in the vertex set $V(G_i)$. Observe that this can be done in $\mathcal{O}(k)$ by coloring the vertices of $V(G_i)$ blue and removing non blue vertices from the degenerate adjacency list of x. This procedure takes time $\mathcal{O}(k^2)$ for each graph $G_i, i \in [n]$, thus in total, we need time $\mathcal{O}(nk^2 + m) = \mathcal{O}(nk^2)$, as $m = \mathcal{O}(nk)$. ◀

Now that we have seen how the induced subgraphs $G_i, i \in [n]$ can be constructed, we want to characterize their maximal cliques with respect to the maximal cliques of the graph. We show in the next two lemmas that the maximal cliques of graphs $G_i, i \in [n]$ which are not maximal in G can be easily described.

▶ **Lemma 4.** *Let G be a k-degenerate graph, σ_G its degeneracy ordering, and let K be a maximal clique of an induced subgraph $G_i, i \in [n]$. Clique K is not a maximal clique of G if and only if there exists a maximal clique C of G which is an induced subgraph of a G_j with $j < i$ and such that K is a strict induced subgraph of C.*

Proof. Let σ_G be the degeneracy ordering of G. Assume that K is a maximal clique of an induced graph G_i for $i = 1, ..., n-k$ but is not a maximal clique of G. Observe that $v_i \in V(K)$ since, by definition, v_i is connected to all the vertices of $V(G_i) \backslash v_i$. Since K_i is a clique which is not maximal, then there exists a set A of vertices such that $A \cap V(K) = \emptyset$ and the graph induced on $V(K) \cup A$ is a maximal clique of G. Let v_j be the vertex of A with lower ranking in σ_G. We have that $\sigma_G(v_j) < \sigma_G(v_i)$ since v_j is connected to v_i but does not appear in $V(G_i)$. (It does not appear otherwise $A \cap V(K) \neq \emptyset$). Let C be the maximal clique induced on $V(K) \cup A$. Clique C is an induced subgraph of G_j with $j < i$. Observe that K does not have v_j in its vertex set. Therefore K is a strict induced subgraph of C.

Conversely, assume that K is a maximal clique of G_i and C a maximal clique of $G_j, j < i$ such that K is an induced subgraph of C. Since K is a strict induced subgraph of a maximal clique of G then K cannot be a maximal clique of G. ◀

▶ **Corollary 5.** *Let G be a k-degenerate graph and let K be a maximal clique of an induced subgraph $G_i, i \in [n]$ such that K is not maximal in G. Let C be a maximal clique 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 cliques K and C which have been ordered following σ_G. Then $W(K)$ is a proper suffix of $W(C)$.*

Proof. Observe first that by Lemma 4, clique C is well defined. Since K is a strict subgraph of C then $V(K) \subset V(C)$. Recall that by definition, graph $G_i = G[N[v_i] \cap V_i]$ where v_i is the i-th vertex of the degeneracy ordering. Observe that since v_i is the vertex of $V(K)$

with smallest ranking in σ_G then v_i appears first in $W(K)$. We also have that $v_i \in V(C)$. Assume now by contradiction that $W(K)$ is not a proper suffix of $W(C)$. This implies that there exists at least a vertex $x \in V(C) \backslash V(K)$ that appears after vertex v_i in $W(C)$. If that was not the case then $W(K)$ would have been a proper suffix of $W(C)$. This implies that vertex x appears after vertex v_i in σ_G. Observe now that x is connected to all the vertices of K since $x \in V(C)$ and $V(K) \subset V(C)$. Thus $G[V(K) \cup \{x\}]$ is a maximal clique of G_i, which is a contradiction by maximality of K. ◀

To conclude the section, we prove some additional results regarding the maximal cliques of graphs $G_i, i \in [n]$, in the next three lemmas.

▶ **Lemma 6.** *Let G be a k-degenerate graph. Every clique which is maximal in some subgraph $G_i, i \in [n]$ is not maximal in any subgraph G_j with $j \neq i$.*

Proof. Let K be a maximal clique of some subgraph $G_i, i \in [n]$. Assume by contradiction that there exists a $j \in [n]$ with $j \neq i$ such that K is maximal in G_j. Assume first that $i < j$. Since vertex v_i is connected to all the vertices of graph G_i then necessarily $v_i \in V(K)$ or K is not maximal in G_i. Since we assumed $i < j$ then $v_i \notin V(G_j)$. This implies that K cannot be a subgraph of G_j, which gives a contradiction in that case. Thus assume now that $j < i$. The proof is similar. Vertex v_j which is connected to all the vertices of G_j does not belong to graph G_i. Since K is maximal in G_i and since $v_j \notin V(K)$ then K cannot be maximal in G_j. ◀

▶ **Lemma 7.** *Let G be a k-degenerate graph, σ_G its degeneracy ordering. Every maximal clique of G is a subgraph of exactly one graph $G_i, i \in [n]$.*

Proof. let K be some maximal clique of G. We first prove that K is a subgraph of at least a subgraph $G_i, i \in [n]$. Let $x \in V(K)$ be the vertex of K which has minimum ranking in σ_G. Observe now that clique K is subgraph of graph $G_{\sigma_G(x)}$. The fact that clique K is a subgraph of at most a graph $G_i, i \in [n]$ is a consequence of Lemma 6. ◀

▶ **Lemma 8.** *Let G be a k-degenerate graph. Let G_i, $i \in [n]$ be the family of induced subgraphs as defined in Section 2.1 and constructed in Lemma 3. Let α denote the number of maximal cliques of G and α_i the number of maximal cliques of graph G_i. We have that $\sum_{j=1}^{n} \alpha_j \leq \alpha(k+1)$.*

Proof. Let max_i denotes the number of maximal cliques of $G_i, i \in [n]$ which are maximal in G and $Nmax_i$ the number of maximal cliques of $G_i, i \in [n]$ which are not maximal in G. We have that $\alpha_i = max_i + Nmax_i$. By Lemma 7, every maximal clique of G is a subgraph of exactly one graph $G_i, i \in [n]$. This implies that $\sum_{j=1}^{n} max_j = \alpha$. Let X be the set of cliques which are maximal in some graph $G_i, i \in [n]$ but not maximal in G and let $x \in X$. By Lemma 5, the word obtained from the vertices of x which have been ordered following σ_G is a proper suffix of the word obtained from ordering the vertices, following σ_G, of some maximal clique of G. This implies that X is of size at most $k\alpha$ since a maximum clique of a k-degenerate has at most $k+1$ vertices and that a word with $k+1$ letters has at most k proper suffixes. To conclude the proof, Lemma 6 implies that clique x is maximal in an unique graph $G_i, i \in [n]$ which implies that $\sum_{j=1}^{n} Nmax_j \leq k\alpha$. Thus in total $\sum_{j=1}^{n} \alpha_j \leq \alpha + \alpha k = \alpha(k+1)$. ◀

Algorithm 1:

Data: A graph G.
Result: All the maximal cliques of G.

```
1  Compute k the degeneracy of G and σ_G.
2  Construct the graphs G_i, i ∈ [n].
3  Initialize T an empty generalized suffix tree.
4  for j = 1 to n do
5      Compute all maximal cliques of graph G_j.
6      for every maximal clique K of graph G_j do
7          Order the vertices of K following σ_G
8          Search for K in T.
9          if there is a match then
10             Reject it.
11         else
12             Insert the proper suffixes of K in T.
13             Output K.
```

4 Algorithm for maximal clique enumeration

Before we describe the algorithm, we introduce suffix trees. We need a data structure to store the proper suffixes of all maximal cliques. Given a word of size n, we can construct a suffix tree containing all its suffixes in space and time $\mathcal{O}(n)$, see [14, 18, 19]. 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 an online fashion, in space and time $\mathcal{O}(\sum_{i=1}^{r} |x_i|)$, see [10, chapter 6] and [18] for instance.

The outline of the algorithm is the following. We start by computing the induced subgraphs $G_i, i \in [n]$. Then we consider each such subgraph, starting from G_1 up to G_n. We find all its maximal cliques and try to find them in a generalized suffix tree. If there is a match, the clique is rejected, otherwise it is outputted and its proper suffixes are inserted into the generalized suffix tree. The procedure is described in Algorithm 1. Its correctness is proved in Theorem 9 and its time complexity in Theorem 10.

▶ **Theorem 9.** *Given a k-degenerate graph G, Algorithm 1 outputs exactly all its maximal cliques, without duplication.*

Proof. By Lemma 7, every maximal clique of the graph is a subgraph of exactly one graph $G_i, i \in [n]$. Thus, every maximal clique K of the graph is considered exactly once in Line 6 of the algorithm. If K is matched in the generalized suffix tree at Line 7 then the vertices of K ordered following σ_G form a proper suffix of some clique of the graph. This contradicts the fact that K is maximal in G. Thus, every maximal clique is outputted exactly once. Moreover, all the proper suffixes of all the maximal cliques are stored in the generalized tree. By Corollary 5 the word obtained from a maximal clique in some graph $G_i, i \in [n]$ which is not maximal in G form a proper suffix of the world obtained from some maximal clique of G. Thus, all such cliques will be rejected in Line 9 of Algorithm 1. In conclusion, we proved that only the maximum cliques of G are outputted, without duplication. ◀

▶ **Theorem 10.** *Given a k-degenerate graph G, Algorithm 1 has setup time $\mathcal{O}(n(k^2+s(k+1)))$ and enumeration time $\alpha\mathcal{O}((k+1)f(k+1))$ where α is the number of maximal cliques of G and $s(k+1)$ (resp. $f(k+1)$) is the preprocessing time (resp. enumeration time) of maximal clique enumeration in a general $(k+1)$-order graph.*

Proof. Computing the degeneracy of G in Line 1 is done in $\mathcal{O}(m)$ time. Constructing the graphs $G_i, i \in [n]$ in Line 2 is done in $\mathcal{O}(nk^2)$, by Lemma 3. To compute all the maximal cliques of every graph $G_i, i \in [n]$, we can use any output sensitive algorithm of Table 1. The chosen algorithm has preprocessing time $s(|V(G_i)|) = \mathcal{O}(s(k+1))$ and enumeration time $f(|V(G_i)|) = \mathcal{O}(f(k+1))$ for each graph $G_i, i \in [n]$ since these graphs have at most $k+1$ vertices. We first preprocess every such graph G_i in total time $\mathcal{O}(n * s(k+1))$. Thus the preprocessing phase takes time $\mathcal{O}(nk^2 + m + n * s(k+1)) = \mathcal{O}(n(k^2 + s(k+1)))$. Then we enumerate all the maximal cliques of the graphs $G_i, i \in [n]$ in total time $(\sum_{j=1}^{n} \alpha_j) * \mathcal{O}(f(k+1))$ where α_j is the number of maximal cliques of graph G_j. By Lemma 8, $\sum_{j=1}^{n} \alpha_j \leq (k+1)\alpha$. Thus enumerating all the maximal cliques of the graphs $G_i, i \in [n]$ takes total time $\alpha\mathcal{O}((k+1)f(k+1))$. Searching and inserting the generated cliques in the suffix tree takes total time $\alpha\mathcal{O}((k+1)^2)$. In conclusion, Algorithm 1 has preprocessing time $\mathcal{O}(n(k^2+s(k+1)))$ and enumeration time $\alpha\mathcal{O}((k+1)f(k+1))$, as claimed. ◀

5 Conclusion

We presented the first output sensitive algorithm for maximal clique enumeration whose enumeration time depends only on the degeneracy of the graph. We were not able to prove that it has polynomial time delay. Our intuition is that in its current state, our algorithm has time delay $\mathcal{O}(k\alpha)$. Thus, we first ask whether this is true or not and if yes, if there is a way to modify our approach as to get a polynomial time delay. The second question that we ask is whether or not we can improve the space complexity. In its current state, our algorithm requires that the maximal cliques be stored. Can we modify our approach as to avoid that?

References

1. V. Batagelj and M. Zaversnik. An $\mathcal{O}(m)$ algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003. URL: `http://arxiv.org/abs/cs.DS/0310049`.
2. C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973. `doi:10.1145/362342.362367`.
3. F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564–568, 2008.
4. L. Chang, J. X. Yu, and L. Qin. Fast maximal cliques enumeration in sparse graphs. *Algorithmica*, 66(1):173–186, 2013.
5. N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.
6. C. Comin and R. Rizzi. An improved upper bound on maximal clique listing via rectangular fast matrix multiplication. *arXiv preprint arXiv:1506.01082*, 2015.
7. A. Conte, R. Grossi, A. Marino, and L. Versari. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 148, pages 1–148, 2016.
8. David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18, 2013. `doi:10.1145/2543629`.

9 M. Farach. Optimal suffix tree construction with large alphabets. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pages 137–, Washington, DC, USA, 1997. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=795663.796326.

10 D. Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, 1997.

11 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988. doi:10.1016/0020-0190(88)90065-8.

12 D. R. Lick and A. T. White. d-degenerate graphs. *Canad. J. Math.*, 22:1082–1096, 1970. URL: http://www.smc.math.ca/cjm/v22/p1082.

13 K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Scandinavian Workshop on Algorithm Theory*, pages 260–272. Springer, 2004.

14 Edward M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, 1976. doi:10.1145/321941.321946.

15 J. W Moon and L. Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

16 Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1):28–42, 2006. doi:10.1016/j.tcs.2006.06.015.

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

18 Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995. doi:10.1007/BF01206331.

19 Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973*, pages 1–11. IEEE Computer Society, 1973. doi:10.1109/SWAT.1973.13.