Another algorithm listing all maximal cliques in sparse graphs in optimal time

George Manoussakis,\textsuperscript{1,*}

\textsuperscript{1}LRI, University Paris XI, France
*gomanous@gmail.com

Abstract

A graph is $k$-degenerate if there exists an ordering of its vertices $v_1, \ldots, v_n$ such that for all $i$, $|N(v_i) \cap V_i| \leq k$ where $N(v_i)$ is the open neighbourhood of $v_i$ and $V_i = \{v_i, \ldots, v_n\}$. In \cite{EppsteinLOfierStrash2013}, it was proved that the largest possible number of maximal cliques in an $n$-vertex $k$-degenerate graph is $(n-k)3^{k/3}$. Then the authors prove an algorithm listing all the maximal cliques of these graphs in optimal time $O(k(n-k)3^{k/3})$. In this paper we prove another optimal algorithm.

1 Introduction

Degeneracy, introduced by Lick et al. \cite{Licketal1970} is a common and robust measure of the sparseness of a graph. Many real world graphs are sparse and have low degeneracy \cite{Licketal1970}. A graph is $k$-degenerate if every induced subgraph has a vertex of degree at most $k$. Equivalently, as proved by Lick et al. \cite{Licketal1970}, a $k$-degenerate graph admits an ordering of its vertices $v_1, \ldots, v_n$ such that vertex $v_i$ has at most $k$ neighbours after it in the ordering. For instance trees and forests are $1$-degenerate graphs. Planar graphs are $5$-degenerate. See Figure \ref{fig:deg_graphs}.

![Figure 1: Examples of $k$-degenerate graphs for (a) $k = 2$ and (b) $k = 3$. Labels correspond to the degeneracy ordering.](image)

\cite{Licketal1970}
A clique of a graph $G$ is a complete induced subgraph where each pair of vertices is connected. A clique is maximal if it can not be extended by including one more vertex. A clique is maximum if it is of largest possible size in the graph. Cliques have been studied extensively since they are widely used in bioinformatics and social networks, among other domains.

Finding a maximum clique of a graph is one of Karp’s NP-complete problems [11]. There are results on sparse graphs concerning this problem and its variants. For instance, Buchanan et al. [3] prove an algorithm to find the maximum clique of an $n$-vertex $k$-degenerate graph in $O(nm + n2^{k/4})$, later improved to $O(1.2127^k(n - k + 1))$ [12].

To list all the maximal cliques of a general graph, the Bron-Kerbosch algorithm [2], a simple backtracking procedure works well in practice. One of its variants has been shown optimal [14], in the sense that it runs in time $O(3^{n/3})$, which is proportional to the maximum possible number of maximal cliques (excluding time to print the output). Concerning $k$-degenerate graphs, Eppstein et al. [7] show that they may have at most $O((n - k)3^{k/3})$ maximal cliques. In the same paper they prove a fixed-parameter tractable algorithm (with parameter the degeneracy) reporting all the maximal cliques in time $O(nk3^{k/3})$. It is nearly-optimal as defined previously. Later, the same authors showed how to modify it to attain the optimal complexity $O(k(n - k)3^{k/3})$ in [8]. The idea of these algorithms is, roughly, to modify the Bron-Kerbosch algorithm by considering the vertices following the degeneracy ordering and then show how this improves the overall complexity.

We prove another fixed-parameter tractable algorithm parametrized by the degeneracy running in optimal time $O(k(n - k)3^{k/3})$. The main idea is to compute a family of specials induced subgraphs. We apply the optimal variant of the Bron-Kerbosch algorithm to list their maximal cliques and store these cliques in some way. With further work, we output exactly all the maximal cliques of the graph.

## 2 Notations

### 2.1 Graph terminologies

Throughout the paper, 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 neighbourhood of the vertex $x$. The closed neighbourhood of $x$ is defined as $N[x] = N(x) \cup x$. Given an ordering $v_1, \ldots, v_n$ of the vertices of $G$, $V_i$ is the set of vertices following $v_i$ including itself in this ordering, that is, the set $\{v_i, \ldots, v_n\}$. Let $G_i$ denote the induced subgraph $G[N(v_i) \cap V_i]$. $G_i^*$ denotes the induced subgraph $G[N(v_i) \cap V_i]$ and $G_{n-k+1}^*$ is the graph induced on $V_{n-k+1}$. A graph is $k$-degenerate if there is an ordering $v_1, \ldots, v_n$ of its vertices such that for all $i$, $1 \leq i \leq n$, $|N(v_i) \cap V_i| \leq k$. 
2.2 Word terminologies

Let $\Sigma$ be an alphabet, that is, a non-empty finite set of symbols. Let a string $s$ be any finite sequence of symbols from $\Sigma$; $s$ 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 Vertex ordering properties

We give results concerning maximal cliques related to orderings of vertices in general and $k$-degenerate graphs. We start by showing how to build a special family of induced subgraphs. Lemma 1 and Lemma 2 have been proved by Manoussakis [12] but the proofs are given for the sake of completeness.

Lemma 1. [12] Given a $k$-degenerate graph $G$, there is an algorithm constructing the induced subgraphs $G_i$ for $i = 1,\ldots, (n-k)$ and graph $G_{n-k+1}^*$ in time $O((n-k+1)k^3)$, using $O(m)$ memory space.

Proof. Assume that $G$ is represented by its adjacency lists, using therefore $O(m)$ memory space. Degeneracy, along with a degeneracy ordering, can be computed by greedily removing a vertex with smallest degree (and its edges) from the graph until it is empty. The degeneracy ordering is the order in which vertices are removed from the graph and this algorithm can be implemented in $O(m)$ time [11].

Using this degeneracy ordering we construct below the vertex sets of the graphs $G_i$ for $i = 1,\ldots, (n-k)$ and of the graph $G^*_{n-k+1}$ as follows. Assume that initially all the vertices of $G$ are coloured blue. Consider iteratively, one by one, the first $n-k$ vertices $v_1, v_2, \ldots, v_{n-k}$ of the ordering. At Step $i$, we start by colouring vertex $v_i$ in red. Then, we scan its neighbourhood (using an adjacency list), we skip its red neighbours and put its blue neighbours in $V(G_i)$. This is because if one of its neighbour is red, it means that it appears before it in the ordering and thus should not be put in $V(G_i)$. At the end of the $(n-k)$ first iterations put the remaining $k$ vertices in the vertex set $V(G^*_{n-k+1})$. This construction can be done in $O(m)$ time since each iteration takes time proportional to the degree of the vertex we are considering in the order.

Now we construct the edge sets of the graphs $(G_i)$ for $i = 1,\ldots, n-k$ and of the graph $G^*_{n-k+1}$ as follows. For the vertex sets $V(G_i)$ for $i = 1,\ldots, (n-k)$ and for $V(G^*_{n-k+1})$ we start by sorting their vertices following the degeneracy ordering in time $O(klog(k))$ for each such set. This takes total time $O((n-k+1)klog(k))$. This will give us, for each vertex $v_1,\ldots, v_n$, a sorted array $D_i = d_1,\ldots, d_k$ containing its neighbours coming later in the degeneracy ordering.

This takes space $O(nk) = O(m)$ space since every such array is at most of size $k$. Using this structure, we now show how to build the edge sets. Assume that we want to build the edge set of the graph $G^*_i$. Since $v_i$ is connected to all the vertices of $G^*_i$, we go through $D_i$ and add all the corresponding edges to the adjacency list of $G^*_i$ (we add $v_i$ to the adjacency lists of every vertex in $D_i$ and add every vertex of $D_i$ in the adjacency list of $v_i$). Then,
for each element $d_j$ for $j = 1, \ldots, k$ of $D_i$, we check for every element $d'_j$ of $D_i$ with $j' > j$ if it appears in $D_d_j$. If it is the case, we add the corresponding edge. This is done in $O(k^3)$ for all the elements of $D_i$. Therefore, to build all the graphs $(G_i)$ for $i = 1, \ldots, n - k$ and of graph $G^*_{n-k+1}$ we need, overall, $O((n - k + 1)k^3 + m) = O((n - k + 1)k^3 + nk) = O((n - k + 1)k^3)$, and $O(m)$ space, as claimed.

☐

In the next Lemma, we show how the maximal cliques are related to the special family of induced subgraphs we built previously.

**Lemma 2.** Let $G$ be a graph and let $v_1, \ldots, v_n$ be any ordering of its vertices. Every maximal clique of $G$ belongs to exactly one induced subgraph $G^+_i$.

**Proof.** Let $K$ be a maximal clique of $G$. Consider the vertex $v_i$ of $K$ that appears first in the ordering. It is clear that $K$ belongs to $G^+_i$. Now we claim that $K$ can not belong to another subgraph $G^+_j$ for $j \neq i$. To show that, we assume by contradiction that $K$ appears in two induced subgraphs, say $G^+_i$ and $G^+_j$, with $i < j$. By the maximality property of $K$ we observe that necessarily $v_i, v_j \in K$. Therefore we have a contradiction since $v_i \notin V(G^+_j)$.

☐

In the next Lemmas, we characterize maximal cliques of the induced subgraphs we built in Lemma that are not maximal in the graph.

**Lemma 3.** Let $G$ be a graph and let $K$ be a maximal clique of an induced subgraph $G^+_i$ or of $G^*_{n-k+1}$, $K$ is not a maximal clique of $G$ if and only if there is a maximal clique $C$ of $G$ which is an induced subgraph of a $G^+_j$ with $j < i$ (or $j < n - k + 1$) and such that $K$ is a strict induced subgraph of $C$.

**Proof.** Assume that we have a degeneracy ordering $v_1, \ldots, v_n$ of the graph $G$. Consider the first case when $K$ is a maximal clique of an induced graph $G^+_i$ for $i = 1, \ldots, n - k$ but which 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 $G_i$. Since $K$ 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 $A$ that appears first in the degeneracy ordering of $G$. Let $i, j$ be respectively the positions of $v_i, v_j$ in the degeneracy ordering, We have that $j < i$ since $v_j$ is connected to $v_i$ but does not appear in $V(G^+_j)$. (It does not appear otherwise $A \cap V(K) \neq \emptyset$). Let $C$ be the maximal clique induced on $V(K) \cup A$. $C$ 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 $C$.

Consider now the second case where $K$ is maximal clique of graph $G^*_{n-k+1}$ but is not a maximal clique of $G$. Let $B$ be the set of vertices such that $G[V(K) \cup B]$ is a maximal clique of the graph. Notice that no vertex $x$ of $B$ can be in $V(G^*_{n-k+1})$ otherwise $G[V(K) \cup \{x\}]$ is maximal clique of $G^*_{n-k+1}$ which contradicts the maximality of $K$ in $G^*_{n-k+1}$. The proof for this case now goes on as for the first case.

Conversely, assume that $K$ is a maximal clique of $G^+_i$ and $C$ a maximal clique of $G^+_j$ such that $K$ is an induced subgraph of $C$. Since $j < i$, $K$ is a strict induced subgraph of a maximal clique of the graph. Therefore $K$ can not be a maximal clique of the graph. The same holds if $K$ is a maximal clique of $G^*_{n-k+1}$.
Lemma 4. Let \( G \) be a graph and let \( K \) be a maximal clique of an induced subgraph \( G_i^+ \) or of \( G_{n-k+1}^* \). Assume that the vertices of the maximal cliques of graphs \( G_i \) for \( i = 1, \ldots, (n-k) \) and of graph \( G_{n-k+1}^* \) are ordered following some ordering \( v_1, \ldots, v_n \) of the vertices of graph \( G \). \( K \) is not a maximal clique of \( G \) if and only if there is a maximal clique \( C \) of \( G \) which is an induced subgraph of a \( G_j^+ \) with \( j < i \) (or \( j < n-k+1 \)) and such that \( V(K) \) is a proper suffix of \( V(C) \).

Proof. Assume that \( K \) is a maximal clique of a graph \( G_i^+ \) for some \( i \) but is not a maximal clique of \( G \). By Lemma 3 there is 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 \( C \). Let \( A = \{V(C) \setminus V(K)\} \). Observe that \( A \neq \emptyset \). If a vertex \( x \) of \( A \) appears after any vertex of \( K \) in the degeneracy ordering, \( x \) must appear in \( K \) (or \( K \) would not be maximal) which is a contradiction by definition of \( A \). Therefore \( V(K) \) is a proper substring of \( V(C) \) such that all the letters of \( V(K) \) are after the letters of \( A \) which proves that \( V(K) \) is a proper suffix of \( V(C) \). The proof is the same if \( K \) is an induced subgraph of \( G_{n-k+1}^* \).

Conversely, the proof remains the same as for Lemma 3. Since \( V(K) \) is a proper suffix of the vertex set of a maximal clique of the graph, \( K \) can not be a maximal clique of \( G \). 

4 Algorithm listing all the maximal cliques

Before we describe the algorithm, we introduce suffix trees. We need a data structure to store the maximal cliques and their suffixes. Given a word of size \( n \), we can construct a suffix tree containing all its suffixes in space and time \( O(n) \), see [16, 13, 15], for instance. This holds if the alphabet is either of constant size or if it is integer [9], that is for a word of size \( n \), the alphabet is the integers in interval \([1, \ldots, n]\). For a set of words \( X = \{x_1, x_2, \ldots, x_n\} \), 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 \( O(\sum_{i=1}^{n} |x_i|) \), see [10] for instance. This holds, to the best of our knowledge, if the alphabet is either of constant size or integer. If the words we consider are of size \( k \) over an alphabet of size \( n \) with \( k < n \), we can obtain the same time complexities but using more space, see [10] for instance.

We start by proving a non-optimal algorithm. We then modify it to obtain the claimed complexity. The outline of the algorithm is as follows:

INPUT: A \( k \)-degenerate graph \( G \) represented by adjacency lists.
OUTPUT: The maximal cliques of \( G \).

1. Construct the graphs \( G_i \) for \( i = 1, \ldots, (n-k) \) and the graph \( G_{n-k+1}^* \);
2. For each graph, in increasing index order do begin
   2.1. Compute all its maximal cliques using the variant of the Bron-Kerbosch algorithm.
   2.2. For each clique do begin
2.2.1. Sort its vertices following the degeneracy ordering to obtain a word;
2.2.2. Try to match it in a global generalized suffix tree:
   2.2.2.1. If there is a match, reject it;
   2.2.2.2. If there is no match, accept and insert it;
end
end;

Theorem 5. Given a $k$-degenerate graph $G$, there is an algorithm listing all its maximal cliques in time $O(k(n-k+1)3^{k/3})$.

Proof. We apply a variant of the classical Bron-Kerbosch algorithm [14] to graphs $G_i$ for $i = 1, \ldots, (n-k)$ and graph $G^*_{n-k+1}$. Using a pivot strategy minimizing the number of recursive calls, it reports all maximal cliques of a $n$-vertex graph in time $O(n3^{n/3})$. By definition, $v_i$ is connected to all the vertices of $G_i$. Therefore we add to all the reported maximal cliques of $G_i$ vertex $v_i$. Because graphs $G^*_i$ for $i = (n-k+1), \ldots, n$ are induced subgraphs of $G^*_{n-k+1}$ and by Lemma 2 this procedure will list at least all the maximal cliques of $G$. This can be done in $O(k(n-k+1)3^{k/3})$ time. But this algorithm can also list cliques that are maximal in some induced subgraph $G^*_i$ or $G^*_{n-k+1}$ but not maximal in $G$. To tackle this problem, we proceed as follows. We construct iteratively a generalized suffix tree containing all the suffixes of the reported maximal cliques.

We start with graph $G^*_1$. By definition $v_1$ appears in all the maximal cliques of $G_1$. By Lemma 3 all the maximal cliques of $G^*_1$ are maximal cliques of $G$. Since $v_1$ does not appear in any clique of some graph $G^{*+}_i$ with $i > 1$, we only store the suffixes of the maximal cliques of $G_1$. We need to consider a given ordering of the vertices that we keep through the algorithm to check if some clique is in the generalized suffix tree. Therefore we sort the vertices of $G_1$ in time $O(k\log(k))$ following the degeneracy ordering. Then we attribute to every vertex an integer between 1 and $k$ which is its rank in the sorting. This is possible since $|V(G_1)| \leq k$. Now, every time the Bron-Kerbosch algorithm lists a clique in $G_1$, we sort its vertices following the degeneracy ordering in time $O(k\log(k))$ using the ranks of its vertices and the Counting Sort algorithm, see [5] for instance. Notice that we can do this for all the maximal cliques of $G_1$ in time $O(k3^{k/3})$.

Consider induced subgraph $G_2$. We start by sorting its vertices in time $O(k\log(k))$. Then we attribute its rank to every vertex. Since $v_2$ will appear in every maximal clique of $G^*_2$, we look in the generalized suffix tree if $v_2$ appears in first position of some suffix in the tree. This can be done in time $O(1)$. If it is not the case all the maximal cliques of $G^*_2$ do not appear in the suffix tree (because all these cliques start with $v_2$). In this case, we know by Lemma 4 that all the maximal cliques of $G^*_2$ are maximal cliques of the graph. If $v_2$ appears in the generalized suffix tree, we save its position in the tree. We compute the cliques of $G_2$, for each clique we order its vertices following the degeneracy ordering in time $O(k)$ (using the ranks of its vertices). Now we use the generalized suffix tree to check if the reported maximal cliques are maximal in $G$. We begin in the tree at the position of $v_2$, since all the cliques start with $v_2$. If the clique $K$ we are considering in $G_2$ is a suffix in the subtree starting at $v_2$, by Lemma 4 we reject it. Notice that this can be done in $O(k)$ time.
Theorem 6. Given a k-degenerate graph G, there is an algorithm listing all its maximal cliques in time \( O(k(n - k)3^{k/3}) \).

Proof. We modify the algorithm of Theorem 3 to get the claimed complexity. For the last two graphs \( G_{n-k}^* \) and \( G_{n-k+1}^* \) we proceed differently. We check if all the vertices of \( G_{n-k+1}^* \) are in the neighbourhood of \( v_{n-k} \). This can be done in \( O(k^2) \) using the array \( D_{n-k} \) constructed in Lemma 4. Assume that it is the case. Therefore \( G_{n-k}^* \) and \( G_{n-k+1}^* \) are the same graphs. Now we proceed exactly as for Theorem 5 except that we do not need to do any work for the last graph \( G_{n-k+1}^* \) since none of its maximal cliques are maximal in G. In this case we need \( O(k(n - k)3^{k/3}) \) time.

Conversely assume that all the vertices of \( G_{n-k+1}^* \) are not in the neighbourhood of \( v_{n-k} \). In this case, we proceed exactly as in Theorem 5. Since \( |V(G_{n-k})| < k \) we get the claimed complexity. Consider the graphs \( G_i \) for \( i = 1, ..., (n - k - 1) \) and \( G_{n-k+1}^* \). To sort their vertices we need time \( O((k(n - k)3^{k/3}) \). To report all their cliques and sort them we need time \( O((k(n - k)3^{k/3}) \). For these graphs, overall we need time \( O(k(n - k)3^{k/3}) \). For the graph \( G_n-k \), to sort its vertices we need time \( O((k - 1)\log(k - 1)) \). To list its cliques and sort them we need time \( O((k - 1)3^{k-1/3}) \). To check if they are in the generalized suffix tree and to insert them we need \( O((k - 1)3^{k-1/3}) \). Overall for \( G_{n-k} \) we need time \( O((k - 1)3^{k-1/3}) \). Therefore, for \( G \) we need time \( O((k - 1)3^{k-1/3}) + O(k(n - k)3^{k/3}) = O(k(n - k)3^{k/3}) \) as claimed.

References

[1] V. Batagelj and M. Zaversnik. An \( O(m) \) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003.

[2] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. Commun. ACM, 16(9):575–577, September 1973.

[3] A. Buchanan, J. L. Walteros, S. Butenko, and P. M. Pardalos. Solving maximum clique in sparse graphs: an \( O(nm + n2^{d/2}) \) algorithm for \( d \)-degenerate graphs. Optimization Letters, 8(5):1611–1617, 2014.
[4] F. Chung. Graph theory in the information age. Notices of the AMS, 57(6):726–732, 2010.

[5] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. Introduction to Algorithms. McGraw-Hill Higher Education, 2nd edition, 2001.

[6] A. T. White D. R. Lick. d-degenerate graphs. Canad. J. Math., 22:1082–1096, 1970.

[7] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In O. Cheong, K-Y. Chwa, and K. Park, editors, ISAAC (1), volume 6506 of Lecture Notes in Computer Science, pages 403–414. Springer, 2010.

[8] David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. J. Exp. Algorithmics, 18:3.1:3.1–3.1:3.21, November 2013.

[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.

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

[11] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, Complexity of Computer Computations, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

[12] G. Manoussakis. The clique problem on inductive k-independent graphs. ArXiv e-prints, October 2014.

[13] E. M. McCreight. A space-economical suffix tree construction algorithm. J. ACM, 23(2):262–272, April 1976.

[14] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. Theor. Comput. Sci., 363(1):28–42, October 2006.

[15] E. Ukkonen. On-line construction of suffix trees. Algorithmica, 14(3):249–260, 1995.

[16] P. Weiner. Linear pattern matching algorithms. In Switching and Automata Theory, 1973. SWAT ’08. IEEE Conference Record of 14th Annual Symposium on, pages 1–11, Oct 1973.