The Bron-Kerbosch Algorithm with Vertex Ordering is Output-Sensitive

George Manoussakis*

*LIPARAD, Université de Versailles Saint-Quentin-En-Yvelines
gеorge.manoussakis@uvsq.fr

Abstract

The Bron-Kerbosch algorithm is a well known maximal clique enumeration algorithm. So far it was unknown whether it was output sensitive or not. In this paper we partially answer this question by proving that the Bron-Kerbosch Algorithm with vertex ordering, first introduced and studied by Eppstein, Löffler and Strash in “Listing all maximal cliques in sparse graphs in near-optimal time. International Symposium on Algorithms and Computation. Springer, Berlin, Heidelberg, 2010” is output sensitive.

1 Introduction

Finding communities, which are essentially densely connected subgraphs of a graph, is an important question and find many applications in various domains, such as biology [13], sociology [11], as well as physics [14], for instance. Modeling communities can be done using the notion of maximal cliques, that is, maximal subgraphs in which all vertices are pairwise connected.

This question has been extensively studied from a theoretical point of view. 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. [19] propose an algorithm enumerating all maximal cliques of a general n-order graph in time $O(3^{n/3})$ (excluding time to print the output). This is worst-case output size optimal in general graphs, as for instance the Moon-Moser graphs have $\Theta(3^{n/3})$ cliques [4, 18]. Thus, even printing the cliques of these graphs would require at least $\Omega(3^{n/3})$ time. Similarly, for k-degenerate graphs (degeneracy is a measure of sparsity), Eppstein et al. [9] prove a $O((n - k)3^{k/3})$ bound on the maximal number of maximal cliques and then show a fixed-parameter tractable algorithm running in time $O(kn3^{k/3})$. They later improve the complexity in [10]. The two algorithms of Tomita et al. and Eppstein et al. are both variations of the well known Bron-Kerbosch algorithm [3]. All these algorithms are described in the next section.

In the worst case, the number of cliques can be exponential, therefore any algorithm for this problem has exponential complexity. On the other hand, if the specific input graph has few cliques, it may be relevant to achieve better
time complexity than in the general case. Thus, this is why algorithms whose running time depends on the number of maximal cliques of the graph have also been considered. This family of algorithms is often referred to as output sensitive algorithms. Their time complexities can be divided into a preprocessing phase followed by an enumeration phase whose complexity is polynomial in the number of maximal cliques of the input graph. There has been quite a few such algorithms [5, 6, 7, 8, 12, 15, 16, 20]. Many of them are based on the reverse search technique of Avis and Fukuda [1].

Our contribution. It was unknown if a Bron-Kerbosch algorithm was output sensitive, which could explain somehow why they are quite efficient in practice. We show in this paper that the Bron-Kerbosch variant proposed by Eppstein et al. in [9, 10] is output sensitive. This is the first such result.

2 Preliminaries

2.1 The Bron-Kerbosch Algorithms
The Bron-Kerbosch algorithm [3] is a well known backtracking algorithm used to enumerate all maximal cliques in a graph. It is described in Figure 1. The algorithm has three disjoint sets of vertices $R$, $P$, and $X$ as arguments, where $R$ is a clique being constructed and where the set $P \cup X$ are the vertices that are adjacent to every vertex in $R$. The vertices in $P$ are candidate vertices to add to clique $R$, while those in $X$ must be excluded from the clique. The algorithm chooses a candidate $v$ in $P$ to add to the clique $R$ and makes a recursive call in which $v$ has been moved from $P$ to $R$. When the recursive call returns, $v$ is moved to $X$ to eliminate duplicate work. When the recursion reaches a level at which $P$ and $X$ are empty, $R$ is a maximal clique and is reported.

In the same paper, Bron and Kerbosch also describe a variant of their algorithm where a pivot is used, to limit the number of recursive calls made by the algorithm. Tomita et al. [19] show that choosing the pivot $u$ from $P \cup X$ in order to maximize $|P \cap N(u)|$ (with $N(u)$ the neighborhood of $u$ in the graph) guarantees that the Bron-Kerbosch algorithm has optimal worst-case running time $O(3^{n/3})$. We give their algorithm in Figure 2.

Later, Eppstein et al. [9] proposed a new algorithm for maximal clique enumeration, which is fixed-parameter tractable, with parameter the degeneracy $k$ of the input graph. Their algorithm essentially performs the outer level of recursion of the BronKerbosch algorithm without pivoting, using a degeneracy ordering to order the sequence of recursive calls made at this level. Then it switches at inner levels of recursion to the pivoting rule of Tomita et al [19]. We give their algorithm in Figure 3. The authors prove in the conference paper that their algorithm has complexity $O(nk3^{n/3})$, which is near worst-case optimal. They later improved the complexity in [10] and reached worst-case output size optimality. In this paper we actually show that their algorithm is output sensitive. We will not use exactly the same pivot rule as Tomita et al. We recall that they choose as pivot a vertex $u$ in $P \cup X$ which maximizes $|P \cap N(u)|$. Instead we will always choose a vertex (arbitrarily) in $P$ as long as $P$ is not empty. When $P$ becomes empty, then the algorithm terminates. Thus, Line 4 of their algorithm is equivalent to “choose a pivot vertex $u$ in $P$”. Note that this choice
of pivot only changes the complexity of the algorithm, but not its correctness, since this pivot rule is the one proposed by Bron and Kerbosch in [3].

1: **procedure** BronKerbosch(P, R, X)
2:     if P ∪ X = ∅ then
3:         report R as a maximal clique
4:     for each vertex v ∈ P do
5:         BronKerbosch(P ∩ N(v), R ∪ {v}, X ∩ N(v) )
6:     P ← P \ {v}
7:     X ← X ∪ {v}

Figure 1: The original maximal clique enumeration algorithm, presented by Bron and Kerbosch in [3]. No complexity bound is given.

1: **procedure** BronKerboschPivot(P, R, X)
2:     if P ∪ X = ∅ then
3:         report R as a maximal clique
4:     Choose a pivot vertex u in P ∪ X
5:     for each vertex v ∈ P \ N(u) do
6:         BronKerboschPivot(P ∩ N(v), R ∪ {v}, X ∩ N(v) )
7:     P ← P \ {v}
8:     X ← X ∪ {v}

Figure 2: A variation of the Bron-Kerbosch algorithm, with pivot, presented by Tomita et al. [19]. It has worst-case output size optimal complexity $O(3^{n/3})$ when u is chosen from $P \cup X$ to maximize $|P \cap N(u)|$.

2.2 Definitions

The graphs we consider are of the form $G = (V, E)$ with $V$ the vertex set and $E$ the edge set. If $X \subset V$, the subgraph of $G$ induced by $X$ is denoted by $G[X]$. 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 $G$. The closed neighborhood of $x$, denoted by $N[x]$, is defined as the set $\{N(x) \cup \{x\}\}$. Given an ordering $v_1, ..., v_n$ of the vertices of $G$, set $V_i$ is the vertices following $v_i$ including itself in this ordering, that is, the set $\{v_i, v_{i+1}, ..., v_n\}$. By $c(G)$ we denote the number of maximal cliques of $G$ and by $q$ the maximum clique size of the graph.

In our proofs we consider the call tree of the Bron-Kerbosch algorithm with degeneracy ordering, that is, the algorithm of Figure 3. The root of this tree is the call to the procedure **BronKerboschDeg(G)** in Line 1. A leaf of this tree corresponds to a final recursive call, after which there is either an output or no output. In our proofs we will denote the leaves of the call trees by $l_1, \ldots, l_r$ with $r$ an arbitrary finite integer. Moreover, to each such leaf $l_j$ (and more generally
procedure BronKerboschDeg\((G = V, E)\)

for each vertex \(v_i\) in a degeneracy ordering \(v_0, v_1, \ldots, v_n\) of \(G\) do

\[P \leftarrow N(v_i) \cap \{v_{i+1}, \ldots, v_n\}\]

\[X \leftarrow N(v_i) \cup \{v_1, \ldots, v_{i-1}\}\]

BronKerboschPivot\((P, \{v_i\}, X)\)

Figure 3: The Bron-Kerbosch algorithm with vertex ordering, presented by Eppstein et al. in [9]. It has complexity \(O(kn^{3k/3})\), which is near worst-case output size optimal.

to any vertex of the call tree of the algorithm), we associate the sets \(P_l, X_l\) and \(R_l\) corresponding to the input values of the last recursive call to the function. Set \(P\) corresponds to a set of candidate vertices of the graph while set \(X\) refers to non-candidate vertices. Set \(R\) is the clique being currently constructed (that we call result set through the proofs). In the proofs we will distinguish between two types of leaves of the call tree. On one side we have the set \(D\) of leaves with no output (that is where set \(R\) is not reported as a maximal clique). On the other side we will have set \(C\) of leaves with output. Note that a leaf of the call tree belongs to either \(C\) or \(D\).

Observe that the root of the call tree of \(\text{BronKerboschDeg}\) has \(n\) children if the input graph is of order \(n\). Each such child corresponds to a call to the procedure \(\text{BronKerboschPivot}(P, \{v_i\}, X)\) in Line 5 of procedure \(\text{BronKerboschDeg}\), for \(v_i\) a vertex in a degeneracy ordering. We denote these \(n\) subtrees as \(S_1, S_2, \ldots, S_n\).

We now define a family of induced subgraphs of \(G\), defined with respect of some ordering of its vertices. By \([n]\) we denote the set of integers \(\{1, 2, \ldots, n\}\).

Definition 1. Let \(G\) be a graph of order \(n\) and \(v_1, \ldots, v_n\) some ordering of its vertices. Graph \(G_i, i \in [n]\), is the induced graph \(G[N[v_i] \cap V_i]\).

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 \(k\) is the smallest integer such that there is an ordering \(v_1, \ldots, v_n\) of its vertices such that for all \(i \in [n]\), we have \(|N(v_i) \cap V_i| \leq k\). The degeneracy ordering can be computed in \(O(m)\) time [2]. The idea is essentially to remove iteratively vertices of minimum degree until the graph is empty. The order in which the vertices are removed yields the degeneracy ordering. In the following, when not specified, we consider the family of subgraphs \(G_i, i \in [n]\) constructed following a degeneracy ordering of the graph.

3 Proof

The proof is as follows. We characterize the sets \(R\) of the leaves of the call tree of Algorithm \(\text{BronKerboschDeg}\) described in Figure 3. There are two types of leaves, the one for which there is an output of some maximal clique of the graph and the ones for which there is not. By correctness of the algorithm, the set of leaves with output is of size \(c(G)\), with \(G\) the input graph. Therefore, to show that the algorithm is output sensitive, it is necessary to show that the set
of leaves with no output is bounded polynomially by some function of $c(g)$ (and possibly some other graph parameters). The main idea is to show that the sets $R$ of these leaves are somehow related to the maximal cliques of the graphs $G_i$ for $i \in [n]$. Then, using Lemma 3 we will be able to bound the number of these leaves with no output.

We first recall some results concerning the family of subgraphs $G_i$, $i \in [n]$, proved in [17]. They link somehow the maximal cliques of the input graph $G$ to the maximal cliques of the graphs $G_i$, $i \in [n]$. We think that the key lemmas of this paper are Lemmas 3 and 11. Lemma 3 bounds the number of maximal cliques of graphs $G_i$, $i \in [n]$ which are not maximal in $G$, while Lemma 11 shows how these maximal cliques appear in the execution of the algorithm. The output sensitivity of the algorithm is shown in Theorem 13.

Lemma 2. [17] Every maximal clique of $G$ is a subgraph of exactly one graph $G_i$, $i \in [n]$.

Lemma 3. [17] We have that $\sum_{j=1}^{n} c(G_i) \leq c(G)q$ where we recall that $c(G_i)$ is the number of maximal cliques of graph $G_i$, that $c(G)$ the number of maximal cliques of $G$, and that $q$ is the maximum clique size.

We now recall some general results concerning procedure BronKerboschDeg and more specifically its correctness and some properties on the input sets $P$, $R$ and $X$.

Theorem 4. [9] Procedure BronKerboschDeg generates all and only maximal cliques without duplication.

Lemma 5. [19] Procedure BronKerboschPivot generates all and only maximal cliques containing all vertices in $R$, some vertices in $P$, and no vertices in $X$, without duplication.

We show in the next corollary a first simple link between the algorithm and the family of graphs $G_i$, $i \in [n]$.

Corollary 6. When BronKerboschPivot is called in Line 5 of procedure BronKerboschDeg on vertex $v_i$, it enumerates all maximal cliques of graph $G_i$ which are maximal in graph $G$.

Proof. Observe that when BronKerboschPivot is called on vertex $v_i$, the set $\{\{v_i\} \cup P\}$ is exactly the vertex set of graph $G_i$. This, together with the correctness of procedure BronKerboschDeg shown in Theorem 4 and Lemma 5 yields the proof.

In the following lemmas, we characterize the $P$, $R$ and $X$ sets of the leaves of the call tree of algorithm BronKerboschDeg. Recall that the pivot we chose is an arbitrary vertex in $P$, as described in Section 2.1.

Lemma 7. For any leaf $l$ of the call tree of procedure BronKerboschDeg, its set $P$ is empty.

Proof. This holds otherwise we could still apply Line 4 of the algorithm and that would yield a contradiction by definition of $l$. 

5
Lemma 8. Vertex $v_i$ belongs to the $R$ set of every leaf of subtree $S_i$.

Proof. At the root of $S_i$, the recursive call is initially done with $R = \{v_i\}$. Since no vertices are removed from this $R$ set during the lower level recursive calls, but only added, then the lemma holds.

Lemma 9. Consider the root of some subtree $S_i$. Let $X_i$ be its non-candidate set and $P_i$ its candidate set. Let $l$ be a leaf of $S_i$ and $R_l$ its result set. Then no vertex of $X_i$ appears in $R_l$.

Proof. Since no vertices of $X_i$ are added to the $R$ sets (only vertices from $P_i$) during the lower level recursive calls, then the lemma holds.

Lemma 10. Let $l_i, l_j$ be two leaves of the call tree of Algorithm 3. Assume that $l_i, l_j \in D$ with $i < j$. Then $R_{l_i} \neq R_{l_j}$.

Proof. Assume by contradiction that $R_{l_i} = R_{l_j}$. Assume first that $l_i$ and $l_j$ belong to two different subtrees $S_i$ and $S_j$, respectively. Since the root of tree $S_i$ consists of a recursive call with $v_i$ as the $R$ set, then by Lemma 8 vertex $v_i$ belongs to all the $R$ sets of the subtree $S_i$. Similarly, the root of subtree $S_j$ consists of a recursive call with set $X$ such that $v_i \in X$ since we assumed $i < j$. Therefore by Lemma 9 $v_i$ cannot belong to $R_{l_j}$, which yields the contradiction.

Thus assume now that leaves $l_i$ and $l_j$ belong to the same subtree, say $S_i$ and that $R_{l_i} = R_{l_j}$. Let $t$ be the least common ancestor of $l_i$ and $l_j$ and let $P_t, R_t$ and $X_t$ be the sets associated to the recursive call of $t$. Then $t$ has at least two children $x$ and $y$ (ancestors of $l_i$ and $l_j$ respectively). Let $v_x$ (resp. $v_y$) be the vertex chosen in $P$ in the recursive call of node $t$ which creates node $x$ (resp. node $y$). Let $P_x, R_x, X_x$ and $P_y, R_y, X_y$ be the sets of nodes $x$ and $y$. Assuming that $v_x$ is chosen before $v_y$, when the recursive call which generates $x$ terminates, then vertex $v_x$ will be in the set $X_y$. Since no vertices of $X_y$ are added to the $R$ sets during the lower level recursive calls, vertex $v_x$ will not be in $R_{l_j}$ and this concludes the proof.

Lemma 11. Let $l$ be a leaf in $D$ and $R_l$ its associated set. Assume that $l$ belongs to subtree $S_i$ for some $i \in [n]$. Then $R_l$ forms a maximal clique in subgraph $G_i$.

Proof. Observe first that $R_l$ forms a clique in $G_i$. Assume by contradiction that it is not maximal in graph $G_i$. Then there exists a non empty set $K$ of vertices such that $K \subseteq V[G_i]$ and such that $R_l \cup K$ forms a maximal clique in $G_i$. Let $k$ be a vertex in $K$. First observe that at the root of $S_i$, the recursive call is initially done with some set candidate set $P$ such that $k \in P$, by definition of $G_i$. Moreover, $k$ belongs to the neighborhood of every vertex in $R_l$. Thus, for every recursive call corresponding to a vertex on the path of the call tree from the root of $S_i$ to $l$, vertex $k$ will never be excluded from the current candidate set since this set is updated by intersection operations with the neighborhoods of vertices in $R_l$. This yields a contradiction since by Lemma 10 set $P_l$ of leaf $l$ must be empty.

Theorem 12. Let $D$ be the set of non output leaves of the call tree of BronKerboschDeg and $C$ the set of output leaves. Then $|D| + |C| = \mathcal{O}(c(G)q)$ with $c(G)$ the number of maximal cliques of the input graph $G$ and $q$ its maximum clique size.
Proof. Let us consider first set $D$. By Lemma 11, the set $R_l$ of a leaf $l$ in $D$ forms a maximal clique in some graph $G_i$. Since this set corresponds to non output leaves, then by definition clique $R_l$ is maximal in $G_i$ but not in $G$. Moreover by Lemma 10, each such clique appears in exactly one subgraph $G_i$ for some $i \in [n]$ (since all the sets $R$ of leaves in $D$ are distinct). Concerning the maximal cliques of the input graph $G$, we know by Lemma 2 that each such clique appears in exactly one subgraph $G_i$ for some $i \in [n]$. Putting these two observations together, we get that $|D| + |C| = n \sum_{j=1}^{c(G_i)}$, where $c(G_i)$ is the number of maximal cliques of graph $G_i$. Finally, the proof holds by Lemma 3.

Theorem 13. The Bron-Kerbosch algorithm with vertex ordering as described in Figure 3 is output sensitive.

Proof. The call tree of the algorithm has $O(c(g)q)$ leaves, by Theorem 12. It has height at most $k$, the degeneracy of the input graph, since the $P$ sets of the roots of the subtrees $S_1, S_2, \ldots, S_n$ are of size at most $k$. Thus, there are at most $O(c(g)qk)$ recursive calls in total. Moreover, in each recursive call only polynomial time set operations are done. Therefore, overall, the complexity of the algorithm is polynomial in $c(g), k, q, n$, which yields the proof.

References

[1] D. Avis and K. Fukuda. Reverse search for enumeration. Discrete applied mathematics, 65(1-3):21–46, 1996.
[2] V. Batagelj and M. Zaversnik. An $O(m)$ algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003.
[3] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. Commun. ACM, 16(9):575–577, September 1973.
[4] F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. Theoretical Computer Science, 407(1-3):564–568, 2008.
[5] L. Chang, J. X. Yu, and L. Qin. Fast maximal cliques enumeration in sparse graphs. Algorithmica, 66(1):173–186, 2013.
[6] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. SIAM Journal on Computing, 14(1):210–223, 1985.
[7] C. Comin and R. Rizzi. An improved upper bound on maximal clique listing via rectangular fast matrix multiplication. arXiv preprint arXiv:1506.01082, 2015.
[8] 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.

7
[9] 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.

[10] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in large sparse real-world graphs. J. Exp. Algorithmics, 18:3:1–3:1:3.21, November 2013.

[11] S. Fortunato. Community detection in graphs. Physics Reports, 486(3):75–174, 2010.

[12] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. Information Processing Letters, 27(3):119–123, 1988.

[13] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. Journal of computational biology, 3(2):289–306, 1996.

[14] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. Phys. Rev. E, 80:056117, Nov 2009.

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

[16] 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, pages 27:1–27:8, 2017.

[17] G. Manoussakis. A new decomposition technique for maximal clique enumeration for sparse graphs. Theor. Comput. Sci., 770:25–33, 2019.

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

[19] 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.

[20] 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.