Listing All Maximal Cliques in Sparse Graphs in Near-optimal Time

David Eppstein, Maarten Löffler, and Darren Strash

Department of Computer Science, University of California, Irvine, USA

Abstract. The *degeneracy* of an n-vertex graph G is the smallest number d such that every subgraph of G contains a vertex of degree at most d. We show that there exists a nearly-optimal fixed-parameter tractable algorithm for enumerating all maximal cliques, parametrized by degeneracy. To achieve this result, we modify the classic Bron–Kerbosch algorithm and show that it runs in time $O(dn3^{d/3})$. We also provide matching upper and lower bounds showing that the largest possible number of maximal cliques in an n-vertex graph with degeneracy d (when d is a multiple of 3 and $n \geq d + 3$) is $(n-d)3^{d/3}$. Therefore, our algorithm matches the $\Theta(d(n-d)3^{d/3})$ worst-case output size of the problem whenever $n - d = \Omega(n)$.

1 Introduction

Cliques, complete subgraphs of a graph, are of great importance in many applications. In social networks cliques may represent closely connected clusters of actors [6,14,28,40] and may be used as features in exponential random graph models for statistical analysis of social networks [17,19,20,44,49]. In bioinformatics, clique finding procedures have been used to detect structural motifs from protein similarities [26,35,36], to predict unknown protein structures [45], and to determine the docking regions where two biomolecules may connect to each other [22]. Clique finding problems also arise in document clustering [3], in the recovery of depth from stereoscopic image data [29], in computational topology [52], and in e-commerce, in the discovery of patterns of items that are frequently purchased together [51].

Often, it is important to find not just one large clique, but all *maximal cliques*. Many algorithms are now known for this problem [1,7,9,10,11,23,28,32,41,43,47] and for the complementary problem of finding maximal independent sets [16,31,37,39,48]. One of the most successful in practice is the *Bron–Kerbosch algorithm*, a simple backtracking procedure that recursively solves subproblems specified by three sets of vertices: the vertices that are required to be included in a partial clique, the vertices that are to be excluded from the clique, and some remaining vertices whose status still needs to be determined [7,9,32,35,47].

All maximal cliques can be listed in polynomial time per clique [37,48] or in a total time proportional to the maximum possible number of cliques in an n-vertex graph, without additional polynomial factors [15,47]. In particular, a variant of the Bron–Kerbosch algorithm is known to be optimal in this sense [9,47]. Unfortunately this maximum possible number of cliques is exponential [42], so that all general-purpose algorithms for listing maximal cliques necessarily take exponential time.

We are faced with a dichotomy between theory, which states categorically that clique finding takes exponential time, and practice, according to which clique finding is useful and can be efficient in its areas of application. One standard way of resolving dilemmas such as this one is to apply *parametrized complexity* [13]: one seeks a parameter of instance complexity such that instances with small parameter values can be solved quickly. A parametrized problem is said to be *fixed-parameter tractable* if instances with size n and parameter value p can be solved in a time bound of the form $f(p)n^{O(1)}$, where f may grow exponentially or worse with p but is independent of n. With this style of analysis, instances with a small parameter value are used to model problems that can be solved quickly, while instances with a large parameter value represent a theoretical worst case that, one hopes, does not arise in practice.

The size of the largest clique does not work well as a parameter: the maximum clique problem, parametrized by clique size, is hard for $\mathbf{W}[1]$, implying that it is unlikely to have a fixed-parameter tractable algorithm [12], and Turán graphs $K_{\frac{n}{k};\frac{n}{k};\frac{n}{k};\dots}$ have $(n/k)^k$ maximal cliques of size k forcing any algorithm that lists them all to take time larger than any fixed-parameter-tractable bound. However, clique size is not the only parameter one can choose. In this paper, we study maximal clique finding parametrized by *degeneracy*, a frequently-used measure of the sparseness of a graph that is closely related to other common sparsity measures such as arboricity and thickness, and that has previously been used for other fixed-parameter problems [2,8,25,34]. We are motivated by the fact that sparse graphs often appear in practice. For instance, the World Wide Web graph, citation networks, and collaboration graphs have low arboricity [24], and therefore have low degeneracy. Empirical evidence also suggests that the *h*-index, a measure of sparsity that upper bounds degeneracy, is low for social networks [17]. As we show in Appendix A, protein–protein interaction networks have low degeneracy as well. Furthermore, planar graphs have degeneracy at most five [38], and the Barabási–Albert model of preferential attachment [4], frequently used as a model for large scale-free social networks, produces graphs with bounded degeneracy. We show that:

- A variant of the Bron–Kerbosch algorithm, when applied to *n*-vertex graphs with degeneracy d, lists all maximal cliques in time $O(dn3^{d/3})$.
- Every *n*-vertex graph with degeneracy d (where d is a multiple of three and $n \geq d+3$) has at most $(n-d)3^{d/3}$ maximal cliques, and there exists an *n*-vertex graph with degeneracy d that has exactly $(n-d)3^{d/3}$ maximal cliques. Therefore, our variant of the Bron–Kerbosch algorithm is optimal in the sense that its time is within a constant of the parametrized worst-case output size.

Our algorithms are fixed-parameter tractable, with a running time of the form $O(f(d)n)$ where $f(d) = d3^{d/3}$. Algorithms for listing all maximal cliques in graphs of constant degeneracy in time $O(n)$ were already known [10,11], but these algorithms had not been analyzed for their dependence on the degeneracy of the graph. We compare the parametrized running time bounds of the known alternative algorithms to the running time of our variant of the Bron–Kerbosch algorithm, and we show that the Bron–Kerbosch algorithm has a much smaller dependence on the parameter d. Thus we give theoretical evidence for the good performance for this algorithm that had previously been demonstrated empirically.
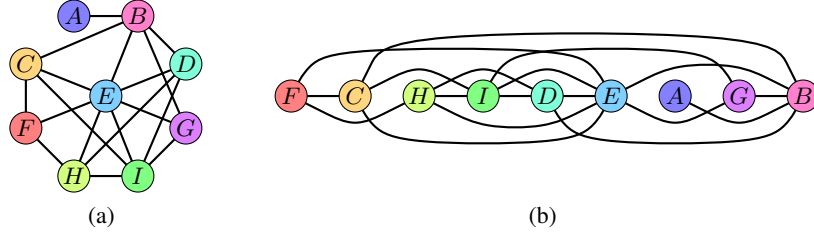
Fig. 1. (a) A graph with degeneracy 3. (b) A vertex ordering showing that the degeneracy is not larger than 3.

2 Preliminaries

We work with an undirected graph $G = (V, E)$, which we assume is stored in an adjacency list data structure. We let n and m be the number of vertices and edges of G, respectively. For a vertex v, we define $\Gamma(v)$ to be the set $\{w \mid (v, w) \in E\}$, which we call the *neighborhood* of v, and similarly for a subset $W \subset V$ we define $\Gamma(W)$ to be the set $\bigcap_{w \in W} \Gamma(w)$, which is the common neighborhood of all vertices in W.

2.1 Degeneracy

Our algorithm is parametrized on the *degeneracy* of a graph, a measure of its sparsity.

Definition 1 (degeneracy). *The degeneracy of a graph G is the smallest value d such that every nonempty subgraph of G contains a vertex of degree at most d [38].*

Figure 1(a) shows an example of a graph of degeneracy 3. Degeneracy is also known as the k-core number [5], width [21], and linkage [33] of a graph and is one less than the coloring number [18]. In a graph of degeneracy d, the maximum clique size can be at most $d + 1$, for any larger clique would form a subgraph in which all vertices have degree higher than d.

If a graph has degeneracy d, then it has a *degeneracy ordering*, an ordering such that each vertex has d or fewer neighbors that come later in the ordering. Figure 1(b) shows a possible degeneracy ordering for the example. Such an ordering may be formed from G by repeatedly removing a vertex of degree d or less: by the assumption that G is d-degenerate, at least one such vertex exists at each step. Conversely, if G has an ordering with this property, then it is d-degenerate, because for any subgraph H of G, the vertex of H that comes first in the ordering has d or fewer neighbors in H. Thus, as Lick and White [38] showed, the degeneracy may equivalently be defined as the minimum d for which a degeneracy ordering exists. A third, equivalent definition is that d is the minimum value for which there exists an orientation of G as a directed acyclic graph in which all vertices have out-degree at most d [11]: such an orientation may be found by orienting each edge from its earlier endpoint to its later endpoint in a degeneracy ordering, and conversely if such an orientation is given then a degeneracy ordering may be found as a topological ordering of the oriented graph.

Degeneracy is a robust measure of sparsity: it is within a constant factor of other popular measures of sparsity including arboricity and thickness. In addition, degeneracy, along with a degeneracy ordering, can be computed by a simple greedy strategy of repeatedly removing a vertex with smallest degree (and its incident edges) from the graph until it is empty. The degeneracy is the maximum of the degrees of the vertices at the time they are removed from the graph, and the degeneracy ordering is the order in which vertices are removed from the graph [30]. The easy computation of degeneracy has made it a useful tool in algorithm design and analysis [11,16].

We can implement this algorithm in $O(n+m)$ time by maintaining an array D, where $D[i]$ stores a list of vertices of degree i in the graph [5]. To remove a vertex of minimum degree from the graph, we scan from the beginning of the array until we reach the first nonempty list, remove a vertex from this list, and then update its neighbors' degrees and move them to the correct lists. Each vertex removal step takes time proportional to the degree of the removed vertex, and therefore the algorithm takes linear time.

By counting the maximum possible number of edges from each vertex to later neighbors, we get the following bound on the number of edges of a d-degenerate graph:

Lemma 1 (Proposition 3 of [38]). *A graph $G = (V, E)$ with degeneracy d has at most $d(n - \frac{d+1}{2})$ edges.*

2.2   The Bron–Kerbosch Algorithm

The Bron–Kerbosch algorithm [7] is a widely used algorithm for finding all maximal cliques in a graph. It is a recursive backtracking algorithm which is easy to understand, easy to code, and has been shown to work well in practice.

A recursive call to the Bron–Kerbosch algorithm provides three disjoint sets of vertices R, P, and X as arguments, where R is a (possibly non-maximal) clique and $P \cup X = \Gamma(R)$ are the vertices that are adjacent to every vertex in R. The vertices in P will be considered to be added to clique R, while those in X must be excluded from the clique; thus, within the recursive call, the algorithm lists all cliques in $P \cup R$ that are maximal within the subgraph induced by $P \cup R \cup X$. 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 R to P; in this recursive call, it restricts X to the neighbors of v, since non-neighbors cannot affect the maximality of the resulting cliques. When the recursive call returns, v is moved to X to eliminate redundant work by further calls to the algorithm. When the recursion reaches a level at which P and X are empty, R is a maximal clique and is reported (see Fig. 2). To list all maximal cliques in the graph, this recursive algorithm is called with P equal to the set of all vertices in the graph and with R and X empty.

Bron and Kerbosch also describe a heuristic called *pivoting*, which limits the number of recursive calls made by their algorithm. The key observation is that for any vertex u in $P \cup X$, called a *pivot*, any maximal clique must contain one of u's non-neighbors (counting u itself as a non-neighbor). Therefore, we delay the vertices in $P \cap \Gamma(u)$ from being added to the clique until future recursive calls, with the benefit that we make fewer recursive calls. Tomita et al. [47] show that choosing the pivot u from $P \cup X$ in order to maximize $|P \cap \Gamma(u)|$ guarantees that the Bron–Kerbosch algorithm has worst-case running time $O(3^{n/3})$, excluding time to write the output, which is worst-case optimal.

proc BronKerbosch(P, R, X)

1: **if** $P \cup X = \emptyset$ **then**
2:     report R as a maximal clique
3: **end if**
4: **for each** vertex $v \in P$ **do**
5:     BronKerbosch($P \cap \Gamma(v), R \cup \{v\}, X \cap \Gamma(v)$)
6:     $P \leftarrow P \setminus \{v\}$
7:     $X \leftarrow X \cup \{v\}$
8: **end for**

proc BronKerboschPivot(P, R, X)

1: **if** $P \cup X = \emptyset$ **then**
2:     report R as a maximal clique
3: **end if**
4: choose a pivot $u \in P \cup X$ {Tomita et al. choose u to maximize $|P \cap \Gamma(u)|$}
5: **for each** vertex $v \in P \setminus \Gamma(u)$ **do**
6:     BronKerboschPivot($P \cap \Gamma(v), R \cup \{v\}, X \cap \Gamma(v)$)
7:     $P \leftarrow P \setminus \{v\}$
8:     $X \leftarrow X \cup \{v\}$
9: **end for**

Fig. 2. The Bron–Kerbosch algorithm without and with pivoting

3   The Algorithm

In this section, we show that apart from the pivoting strategy, the order in which the vertices of G are processed by the Bron–Kerbosch algorithm is also important. By choosing an ordering carefully, we develop a variant of the Bron–Kerbosch algorithm that correctly lists all maximal cliques in time $O(dn3^{d/3})$. Essentially, our algorithm performs the outer level of recursion of the Bron–Kerbosch algorithm without pivoting, using a degeneracy ordering to order the sequence of recursive calls made at this level, and then switches at inner levels of recursion to the pivoting rule of Tomita et al. [47].

In the original Bron–Kerbosch algorithm, in each recursive call the vertices in P are considered for expansion one by one (see line 4 of BronKerbosch in Figure 2). The order in which the vertices are treated is not specified. We first analyze what happens if we fix an order $v_1, v_2, \ldots, v_n$ on the vertices of V, and use the same order consistently to loop through the vertices of P in each recursive call of the non-pivoting version of BronKerbosch.

Observation 1 *When processing a clique R in the ordered variant of Bron–Kerbosch, the common neighbors of R can be partitioned into the set P of vertices that come after the last vertex of R, and the set X of remaining neighbors, as shown in Figure 3.*

Our algorithm computes a degeneracy ordering of the given graph, and performs the outermost recursive calls in the ordered variant of the Bron–Kerbosch algorithm (without pivoting) for this ordering. The sets P passed to each of these recursive calls will have at most d elements in them, leading to few recursive calls within each of these
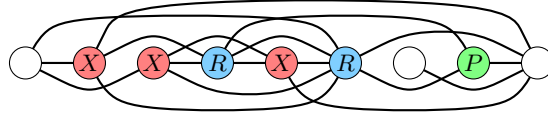
Fig. 3. Partitioning the common neighbors of a clique R into the set P of later vertices and the set X of remaining neighbors.

outer calls. Below the top level of the recursion we switch from the ordered non-pivoting version of the Bron–Kerbosch algorithm to the pivoting algorithm (with the same choice of pivots as Tomita et al. [47]) to further control the number of recursive calls.

proc BronKerboschDegeneracy(V, E)
1: **for** each vertex v_i in a degeneracy ordering $v_0, v_1, v_2, \ldots$ of (V, E) **do**
2:     $P \leftarrow \Gamma(v_i) \cap \{v_{i+1}, \ldots, v_{n-1}\}$
3:     $X \leftarrow \Gamma(v_i) \cap \{v_0, \ldots, v_{i-1}\}$
4:     BronKerboschPivot($P, \{v_i\}, X$)
5: **end for**

Fig. 4. Our algorithm.

Lemma 2. *The Bron–Kerbosch algorithm using the Tomita et al. pivoting strategy generates all and only maximal cliques containing all vertices in R, some vertices in P, and no vertices in X, without duplication.*

Proof. See Tomita et al. [47]. □

Theorem 1. *Algorithm* BronKerboschDegeneracy *generates all and only maximal cliques without duplication.*

Proof. Let C be a maximal clique, and v its earliest vertex in the degeneracy order. By Lemma 2, C will be reported (once) when processing v. When processing any other vertex of C, v will be in X, so C will not be reported. □

To make pivot selection fast we pass as an additional argument to BronKerboschPivot a subgraph $H_{P,X}$ of G that has $P \cup X$ as its vertices; an edge (u, v) of G is kept as an edge in $H_{P,X}$ whenever at least one of u or v belongs to P and both of them belong to $P \cup X$. The pivot chosen according to the pivot rule of Tomita et al. [47] is then just the vertex in this graph with the most neighbors in P.

Lemma 3. *Whenever* BronKerboschDegeneracy *calls* BronKerboschPivot *it can form $H_{P,X}$ in time $O(d(|P| + |X|))$.*

Proof. The vertex set of $H_{P,X}$ is known from P and X. Each edge is among the d outgoing edges from each of its vertices. Therefore, we can achieve the stated time bound by looping over each of the d outgoing edges from each vertex in $P \cup X$ and testing whether each edge meets the criterion for inclusion in $H_{P,X}$. □

The factor of d in the time bound of Lemma 3 makes it too slow for the recursive part of our algorithm. Instead we show that the subgraph to be passed to each recursive call can be computed quickly from the subgraph given to its parent in the recursion tree.

Lemma 4. *In a recursive call to* BronKerboschPivot *that is passed the graph $H_{P,X}$ as an auxiliary argument, the sequence of graphs $H_{P\cap\Gamma(v),X\cap\Gamma(v)}$ to be passed to lower-level recursive calls can be computed in total time $O(|P|^2(|P|+|X|))$.*

Proof. It takes $O(|P|+|X|)$ time to identify the subsets $P\cap\Gamma(v)$ and $X\cap\Gamma(v)$ by examining the neighbors of v in $H_{P,X}$. Once these sets are identified, $H_{P\cap\Gamma(v),X\cap\Gamma(v)}$ may be constructed as a subgraph of $H_{P,X}$ in time $O(|P|(|P|+|X|))$ by testing for each edge of $H_{P,X}$ whether its endpoints belong to these sets. There are $O(|P|)$ graphs to construct, one for each recursive call, hence the total time bound. $\qquad\square$

Lemma 5 (Theorem 3 of [47]). *Let T be a function which satisfies the following recurrence relation:*
$$T(p) \leq \begin{cases} \max_k\{kT(p-k)\}+dp^2 & \text{if } p > 0 \\ e & \text{if } p = 0 \end{cases}$$

Where p and k are integers, such that $p \geq k$, and d,e are constants greater than zero. Then, $T(p) \leq \max_k\{kT(p-k)\}+dp^2 = O(3^{p/3})$.

Lemma 6. *Let v be a vertex, P_v, be v's later neighbors, and X_v be v's earlier neighbors. Then* BronKerboschPivot(P_v, $\{v\}$, X_v) *executes in time $O((d+|X_v|)3^{|P_v|/3})$, excluding the time to report the discovered maximal cliques.*

Proof. Define $D(p,x)$ to be the running time of BronKerboschPivot(P_v, $\{v\}$, X_v), where $p = |P_v|$, and $x = |X_v|$. We show that $D(p,x) = O((d+x)3^{p/3})$. By the description of BronKerboschPivot, D satisfies the following recurrence relation:

$$D(p,x) \leq \begin{cases} \max_k\{kD(p-k,x)\}+c_1p^2(p+x) & \text{if } p > 0 \\ c_2 & \text{if } p = 0 \end{cases}$$

where c_1 and c_2 are constants greater than 0.

Since our graph has degeneracy d, the inequality $p+x \leq d+x$ always holds. Thus,

$$D(p,x) \leq \max_k\{kD(p-k,x)\}+c_1p^2(p+x)$$
$$\leq (d+x)\left(\max_k\left\{\frac{kD(p-k,x)}{d+x}\right\}+c_1p^2\right)$$
$$\leq (d+x)\left(\max_k\{kT(p-k)\}+c_1p^2\right)$$
$$= O((d+x)3^{p/3}) \quad \text{by letting } d=c_1, e=c_2 \text{ in Lemma 5}$$

$\qquad\square$

Theorem 2. *Given a n-vertex graph G with degeneracy d, our algorithm reports all maximal cliques of G in time $O(dn3^{d/3})$.*

Proof. For each initial call to BronKerboschPivot for each vertex v, we first spend time $O(d(|P_v| + |X_v|))$ to set up subgraph H_{P_v,X_v}. Over the entire algorithm we spend time

$$\sum_v O(d(|P_v| + |X_v|)) = O(dm) = O(d^2 n)$$

setting up these subgraphs. The time spent performing the recursive calls is

$$\sum_v O((d + |X_v|)3^{|P_v|/3}) = O((dn + m)3^{d/3}) = O(dn3^{d/3}),$$

and the time to report all cliques is $O(d\mu)$, where μ is the number of maximal cliques. We show in the next section that $\mu = (n - d)3^{d/3}$ in the worst case, and therefore we take time $O(d(n-d)3^{d/3})$ reporting cliques in the worst case. Therefore, the algorithm executes in time $O(dn3^{d/3})$. □

This running time is nearly worst-case optimal, since there may be $\Theta((n-d)3^{d/3})$ maximal cliques in the worst case.

4 Worst-case Bounds on the Number of Maximal Cliques

Theorem 3. *Let d be a multiple of 3 and $n \geq d + 3$. Then the largest possible number of maximal cliques in an n-vertex graph with degeneracy d is $(n-d)3^{d/3}$.*

Proof. We first show that there cannot be more than $(n-d)3^{d/3}$ maximal cliques. We then show that there exists a graph that has $(n-d)3^{d/3}$ maximal cliques.

An Upper Bound. Consider a degeneracy ordering of the vertices, in which each vertex has at most d neighbors that come later in the ordering. For each vertex v that is placed among the first $n - d - 3$ vertices of the degeneracy ordering, we count the number of maximal cliques such that v is the clique vertex that comes first in the ordering.

Since vertex v has at most d later neighbors, the Moon–Moser bound [42] applied to the subgraph induced by these later neighbors shows that they can form at most $3^{d/3}$ maximal cliques with each other. Since these vertices are all neighbors of v, v participates in at most $3^{d/3}$ maximal cliques with its later neighbors. Thus, the first $n - d - 3$ vertices contribute to at most $(n - d - 3)3^{d/3}$ maximal cliques total.

By the Moon–Moser bound, the remaining $d + 3$ vertices in the ordering can form at most $3^{(d+3)/3}$ maximal cliques. Therefore, a graph with degeneracy d can have no more than $(n - d - 3)3^{d/3} + 3^{(d+3)/3} = (n - d)3^{d/3}$ maximal cliques.

A Lower Bound. By a simple counting argument, we can see that the graph $K_{n-d,3,3,3,3,\ldots}$ contains $(n - d)3^{d/3}$ maximal cliques: Each maximal clique must contain exactly one vertex from each disjoint independent set of vertices, and there are $(n - d)3^{d/3}$ ways of forming a maximal clique by choosing one vertex from each independent set. Figure 5 shows this construction for $d = 6$. We can also see that this graph is d-degenerate since in any ordering of the vertices, the first vertex must have d or more later neighbors, and in any ordering where the $n - d$ disjoint vertices come first, these first $n - d$ vertices have exactly d later neighbors, and the last d vertices have fewer later neighbors. □

Relatedly, a bound of $(n - d + 1)2^d$ on the number of cliques (without assumption of maximality) in n-vertex d-degenerate graphs was already known [50].
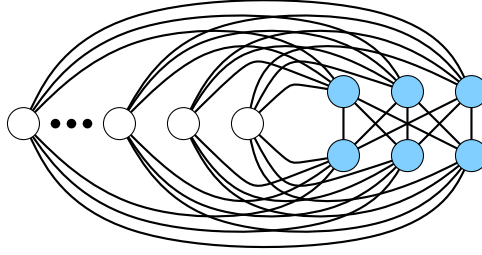
Fig. 5. The lower bound construction for $d = 6$, consisting of a Moon–Moser graph of size d on the right (blue vertices) and an independent set of $n - d$ remaining vertices that are each connected to all of the last d vertices.

5   Comparison with Other Algorithms

Chiba and Nishizeki [10] describe two algorithms for finding cliques in sparse graphs. The first of these two algorithms reports all maximal cliques using $O(am)$ time per clique, where a is the arboricity of the graph, and m is the number of edges in G. The *arboricity* is the minimum number of edge-disjoint spanning forests into which the graph can be decomposed [27]. The degeneracy of a graph is closely related to arboricity: $a \leq d \leq 2a - 1$. In terms of degeneracy, Chiba and Nishizeki's algorithm uses $O(d^2 n)$ time per clique. Combining this with the bound on the number of cliques derived in Section 4 results in a worst-case time bound of $O(d^2 n(n - d)3^{d/3})$. For constant d, this is a quadratic time bound, in contrast to the linear time of our algorithm.

Another algorithm of Chiba and Nishizeki [10] lists cliques of order l in time $O(la^{l-2}m)$. It can be adapted to enumerate all maximal cliques in a graph with degeneracy d by first enumerating all cliques of order $d + 1$, d, ... down to 1, and removing cliques that are not maximal. Applying their algorithm directly to a d-degenerate graph takes time $O(ld^{l-1}n)$. Therefore, the running time to find all maximal cliques is $\sum_{1 \leq i \leq d+1} O(ind^{i-1}) = O(nd^{d+1})$. Like our algorithm, this is linear when d is constant, but with a much worse dependence on the parameter d.

Chrobak and Eppstein [11] list triangles and 4-cliques in graphs of bounded degeneracy by testing all sets of two or three later neighbors of each vertex according to a degeneracy ordering. The same idea extends in an obvious way to finding maximal cliques of size greater than four, by testing all subsets of later neighbors of each vertex. For each vertex v, there are at most 2^d subsets to test; each subset may be tested for being a clique in time $O(d^2)$, by checking whether each of its vertices has all the later vertices in the subset among its later neighbors, giving a total time of $O(nd^2 2^d)$ to list all the cliques in the graph. However, although this singly-exponential time bound is considerably faster than Chiba and Nishizeki, and is close to known bounds on the number of (possibly non-maximal) cliques in d-degenerate graphs [50], it is slower than our algorithm by a factor that is exponential in d. Our new algorithm uses this same idea of searching among the later neighbors in a degeneracy order but achieves much greater efficiency by combining it with the Bron–Kerbosch algorithm.

6   Conclusion

We have presented theoretical evidence for the fast performance of the Bron–Kerbosch algorithm for finding cliques in graphs, as has been observed in practice. We observe that the problem is fixed-parameter tractable in terms of the degeneracy of the graph, a parameter that is expected to be low in many real-world applications, and that a slight modification of the Bron–Kerbosch algorithm performs optimally in terms of the degeneracy.

We explicitly prescribe the order in which the Bron–Kerbosch algorithm processes the vertices of the graph, something that has not been considered before. Without this particular order, we do not have a bound on the running time. It would be interesting to determine whether a random order gives similar results, as this would further explain the observed performance of implementations of Bron–Kerbosch that do not use the degeneracy order.

Acknowledgments

References

1. E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM J. Comput.* 2(1):1–6, 1973, doi:10.1137/0202001.

2. N. Alon and S. Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica* 54(4):544–556, 2009, doi:10.1007/s00453-008-9204-0.

3. J. G. Augustson and J. Minker. An analysis of some graph theoretical cluster techniques. *J. ACM* 17(4):571–588, 1970, doi:10.1145/321607.321608.

4. A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science* 286:509–512, 1999, doi:10.1126/science.286.5439.509.

5. V. Batagelj and M. Zaveršnik. An $O(m)$ algorithm for cores decomposition of networks, 2003, arXiv:cs/0310049.

6. N. M. Berry, T. H. Ko, T. Moy, J. Smrcka, J. Turnley, and B. Wu. Emergent clique formation in terrorist recruitment. *Proc. AAAI-04 Worksh. Agent Organizations*. AAAI Press, 2004, http://www.aaai.org/Papers/Workshops/2004/WS-04-02/WS04-02-005.pdf.

7. 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.

8. L. Cai, S. Chan, and S. Chan. Random separation: A new method for solving fixed-cardinality optimization problems. *Proc. 2nd Int. Worksh. Parameterized and Exact Computation (IWPEC 2006)*, pp. 239–250. Springer-Verlag, LNCS 4169, 2006, doi:10.1007/11847250_22.

9. F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theor. Comput. Sci.* 407(1-3):564 – 568, 2008, doi:10.1016/j.tcs.2008.05.010.

10. N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.* 14(1):210–223, 1985, doi:10.1137/0214017.

11. M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theor. Comput. Sci.* 86(2):243 – 266, 1991, doi:10.1016/0304-3975(91)90020-3.

12. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theor. Comput. Sci.* 141(1-2):109 – 131, 1995, doi:10.1016/0304-3975(94)00097-3.

13. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

14. N. Du, B. Wu, X. Pei, B. Wang, and L. Xu. Community detection in large-scale social networks. *Proc. 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, pp. 16–25, 2007, doi:10.1145/1348549.1348552.

15. D. Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms & Applications* 7(2):131–140, 2003, arXiv:cs.DS/0011009.

16. D. Eppstein. All maximal independent sets and dynamic dominance for sparse graphs. *ACM Trans. Algorithms* 5(4):A38, 2009, doi:10.1145/1597036.1597042, arXiv:cs.DS/0407036.

17. D. Eppstein and E. S. Spiro. The *h*-index of a graph and its application to dynamic subgraph statistics. *Proc. 11th Symp. Algorithms and Data Structures (WADS 2009)*, pp. 278–289. Springer-Verlag, LNCS 5664, 2009, doi:10.1007/978-3-642-03367-4_25.

18. P. Erdős and A. Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Hungarica* 17(1–2):61–99, 1966, doi:10.1007/BF02020444.

19. O. Frank. Statistical analysis of change in networks. *Statistica Neerlandica* 45(3):283–293, 1991, doi:10.1111/j.1467-9574.1991.tb01310.x.

20. O. Frank and D. Strauss. Markov graphs. *J. Am. Stat. Assoc.* 81(395):832–842, 1986, doi:10.2307/2289017.

21. E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM* 29(1):24–32, 1982, doi:10.1145/322290.322292.

22. E. J. Gardiner, P. Willett, and P. J. Artymiuk. Graph-theoretic techniques for macromolecular docking. *J. Chem. Inf. Comput. Sci.* 40(2):273–279, 2000, doi:10.1021/ci990262o.

23. L. Gerhards and W. Lindenberg. Clique detection for nondirected graphs: Two new algorithms. *Computing* 21(4):295–322, 1979, doi:10.1007/BF02248731.

24. G. Goel and J. Gustedt. Bounded arboricity to determine the local structure of sparse graphs. *WG 2006*, pp. 159–167. Springer-Verlag, LNCS 4271, 2006, doi:10.1007/11917496_15.

25. P. A. Golovach and Y. Villanger. Parameterized complexity for domination problems on degenerate graphs. *Proc. 34th Int. Worksh. Graph-Theoretic Concepts in Computer Science (WG 2008)* 5344:195–205, 2008, doi:10.1007/978-3-540-92248-3_18.

26. H. M. Grindley, P. J. Artymiuk, D. W. Rice, and P. Willett. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *J. Mol. Biol.* 229(3):707 – 721, 1993, doi:10.1006/jmbi.1993.1074.

27. F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1972.

28. F. Harary and I. C. Ross. A procedure for clique detection using the group matrix. *Sociometry* 20(3):205–215, 1957, doi:10.2307/2785673.

29. R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Patt. An. Mach. Int.* 11(11):1168–1180, 1989, doi:10.1109/34.42855.

30. T. R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley-Interscience, New York, 1995.

31. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Inf. Proc. Lett.* 27(3):119 – 123, 1988, doi:10.1016/0020-0190(88)90065-8.

32. H. C. Johnston. Cliques of a graph—variations on the Bron–Kerbosch algorithm. *Int. J. Parallel Programming* 5(3):209–238, 1976, doi:10.1007/BF00991836.

33. L. Kirousis and D. Thilikos. The linkage of a graph. *SIAM J. Comput.* 25(3):626–647, 1996, doi:10.1137/S0097539793255709.

34. T. Kloks and L. Cai. Parameterized tractability of some (efficient) *Y*-domination variants for planar graphs and *t*-degenerate graphs. *Proc. International Computer Symposium*, 2000, `http://hdl.handle.net/2377/2482`.

35. I. Koch. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.* 250(1-2):1 – 30, 2001, doi:10.1016/S0304-3975(00)00286-3.

36. I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *J. Comput. Biol.* 3(2):289–306, 1996, doi:10.1089/cmb.1996.3.289.

37. E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM J. Comput.* 9(3):558–565, 1980, doi:10.1137/0209042.

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

39. E. Loukakis and C. Tsouros. A depth first search algorithm to generate the family of maximal independent sets of a graph lexicographically. *Computing* 27(4):349–366, 1981, doi:10.1007/BF02277184.

40. R. D. Luce and A. D. Perry. A method of matrix analysis of group structure. *Psychometrika* 14(2):95–116, 1949, doi:10.1007/BF02289146.

41. K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. *Proc. 9th Scand. Worksh. Algorithm Theory*, pp. 260–272. Springer-Verlag, LNCS 3111, 2004.

42. J. W. Moon and L. Moser. On cliques in graphs. *Israel J. Math.* 3(1):23–28, 1965, doi:10.1007/BF02760024.

43. G. D. Mulligan and D. G. Corneil. Corrections to Bierstone's algorithm for generating cliques. *J. ACM* 19(2):244–247, 1972, doi:10.1145/321694.321698.

44. G. Robins and M. Morris. Advances in exponential random graph (p^*) models. *Social Networks* 29(2):169–172, 2007, doi:10.1016/j.socnet.2006.08.004.

45. R. Samudrala and J. Moult. A graph-theoretic algorithm for comparative modeling of protein structure. *J. Mol. Biol.* 279(1):287 – 302, 1998, doi:10.1006/jmbi.1998.1689.

46. C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID: a general repository for interaction datasets. *Nucleic Acids Res.* 34:D535–D539, 2006, doi:10.1093/nar/gkj109.

47. 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, 2006, doi:10.1016/j.tcs.2006.06.015.

48. S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.* 6(3):505–517, 1977, doi:10.1137/0206036.

49. S. Wasserman and P. Pattison. Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and p^*. *Psychometrika* 61(3):401–425, 1996, doi:10.1007/BF02294547.

50. D. R. Wood. On the maximum number of cliques in a graph. *Graphs and Combinatorics* 23(3):337–352, 2007, doi:10.1007/s00373-007-0738-8.

51. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining*, pp. 283–286. AAAI Press, 1997, http://www.aaai.org/Papers/KDD/1997/KDD97-060.pdf.

52. A. Zomorodian. The tidy set: a minimal simplicial set for computing homology of clique complexes. *Proc. 26th ACM Symp. Computational Geometry*, pp. 257–266, 2010, http://www.cs.dartmouth.edu/~afra/papers/socg10/tidy-socg.pdf.

A Appendix

A.1 The Degeneracy of Protein–Protein Interaction Networks

The Biological General Repository for Interaction Datasets (BioGRID) [46] [1] is a curated database containing the interactions between proteins that have been published in the literature, and those that have been discovered through high-throughput screening methods. Using this interaction data, we can form a graph by creating a vertex for each protein and an edge between two proteins that interact. Such a graph is called a protein–protein interaction (PPI) network.

We computed the degeneracy of seven PPI networks in version 3.0.65 of the BioGRID database. We chose to omit other networks in the BioGRID database because they contain only a handful of known proteins and interactions. The results are summarized in Table 1. Observe that, for each PPI network, the degeneracy is significantly lower than both the number of vertices in the graph and the maximum degree of the graph. We therefore conclude that the degeneracy of these PPI networks is low and that fixed-parameter algorithms parametrized by degeneracy can be expected to perform well on these graphs.

Table 1. Graph statistics for seven PPI networks from version 3.0.65 of the BioGRID database. For each network we list the number of vertices (n), the number of edges (m), the maximum degree (Δ), and the degeneracy (d).

Organism (PPI network)	n	m	Δ	d
Mus musculus	1455	1636	111	6
Caenorhabditis elegans	3518	6531	523	10
Arabisopsis thaliana	1745	3098	71	12
Drosophila melanogaster	7282	24894	176	12
Homo Sapiens	9527	31182	308	12
Schizosaccharomyces pombe	2031	12637	439	34
Saccharomyces cerevisae	6008	156945	2557	64

---

[1] http://thebiogrid.org/