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

Introduction to the Problem

A *clique*, or complete subgraph, is an important graph feature in application areas ranging from social networks to bioinformatics. Often, it is important to find not just one large clique, but all maximal cliques.

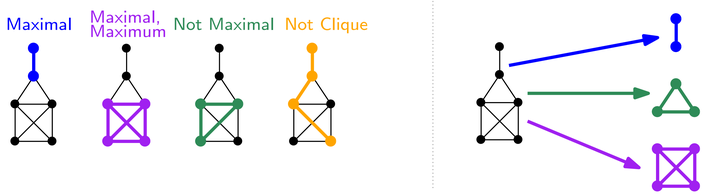


Figure: Maximal cliques are cliques that cannot be extended through the addition of vertices. We want to list all the maximal cliques of a graph.

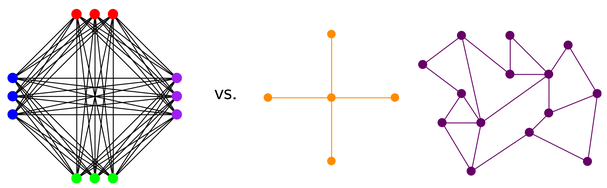


Figure: Listing all maximal cliques on an n-vertex graph must take exponential time in the worst case because there may be up to $3^{n/3}$ cliques to list [2]! However, sparse graphs have fewer cliques. For example, n-vertex planar graphs have $O(n)$ cliques.

Goal: We desire an algorithm to list all maximal cliques that works fast on sparse graphs in theory and in practice.

The Bron–Kerbosch Algorithm

The Bron–Kerbosch algorithm [1] is a simple, recursive algorithm that is easy to implement and works well in practice.

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

Figure: The algorithm uses three sets of vertices to generate all maximal cliques: a partial clique R, candidates for clique expansion P, and vertices which have already been evaluated and are forbidden X.

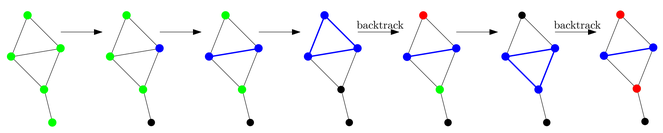


Figure: The Bron–Kerbosch algorithm in action.

Pivoting

The branching factor of the Bron–Kerbosch algorithm can be reduced by choosing a vertex u, called a *pivot*, and only evaluating candidates which are non-neighbors of u. Choosing the pivot with the minimum number of candidate non-neighbors gives us a worst-case optimal $O(3^{n/3})$-time algorithm [3].

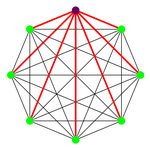
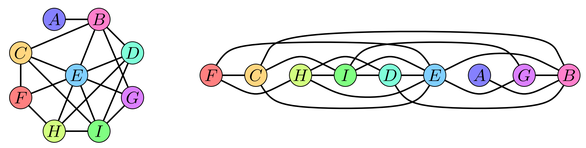


Figure: The purple pivot vertex has many candidate neighbors. We delay the evaluation of these neighbors until the purple vertex is added to the partial clique.

Sparse Graphs

One measure of sparsity is the *degeneracy* of a graph. The degeneracy of a graph is the minimum integer d such that every subgraph contains a vertex with degree at most d. If a graph has degeneracy d, then we can order the vertices so that each vertex has at most d later neighbors in the ordering.



(a) (b)

Figure: (a) A graph with degeneracy 3 , and (b) an ordering showing the graph has degeneracy at most 3.

Our Algorithm

At the top level, we choose vertex candidates in degeneracy order, and then execute the Bron–Kerbosch algorithm with pivoting.

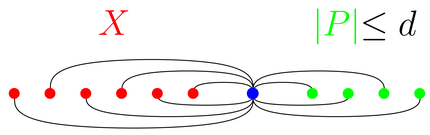


Figure: By choosing vertices in the degeneracy order, we ensure that there are few candidates for clique expansion, limiting the recursion depth.

By choosing vertices in this order, we get a $O(d^2 n 3^{d/3})$-time algorithm. Computing the pivot is the bottleneck of this algorithm, which we can improve upon by maintaining a subgraph throughout the recursion.

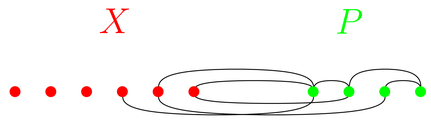
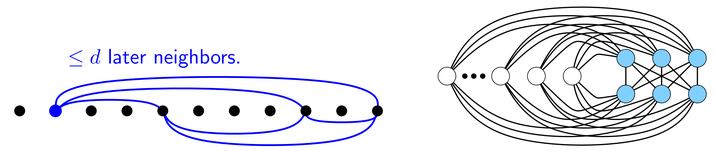


Figure: If we maintain a subgraph consisting of edges that have an end-vertex in the candidate set, our algorithm has running time $O(dn3^{d/3})$.

Near-Optimality of Our Algorithm

The worst-case output size is $\Omega(d(n-d)3^{d/3})$. Therefore our algorithm is worst-case optimal whenever $n - d = \Omega(n)$.



(a) (b)

Figure: (a) Graphs with degeneracy d have cliques of size at most $d+1$. (b) We show that d-degenerate graphs have at most $(n-d)3^{d/3}$ maximal cliques.

Experimental Results

We made new C implementations of the Bron–Kerbosch algorithm (BK), the variant with pivoting (BK-pivot), and two versions of our algorithm: without maintaining the subgraph (BK-hybrid) and with the subgraph (BK-degen). We then compared our results with a state-of-the-art algorithm that uses fancy bit tricks (Uno).

graph	d	BK	BK-pivot	BK-hybrid	BK-degen	Uno
foldoc	12	4.2sec	9sec	< 1sec	1sec	< 1sec
patents	24	> 5min	> 5min	**4.3sec**	**5.3sec**	2.2sec
internet	25	19.4sec	10.3sec	**< 1sec**	**< 1 sec**	**<1sec***
condmat	29	> 3min	65sec	**1.6sec**	**2.61sec**	< 1sec
eatRS	34	19.8sec	53sec	12.3sec	9.12sec	14.9sec
polblogs	36	> 3min	2sec	1.5sec	1.2sec	1.8sec
hep-th	37	> 5min	69.6sec	**22.6sec**	**17.2sec**	**41.5sec***
astro	56	> 3min	12.3sec	**1.4sec**	**3.14sec**	< 1sec
yeast	64	> 3min	81sec	44.3sec	**20.5sec**	**121.1sec***
days-all	73	> 5min	379.1sec	206.5sec	**51.4sec**	10min 25sec
ND-www	155	> 5min	> 5min	**27.8sec**	**41.11sec**	**9.7sec***

* The algorithm did not correctly report all maximal cliques.

Conclusion

Our algorithm is near-optimal in theory, and works fast in practice.

[1] **Bron, C., Kerbosch, J.**: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM 16(9), 575–577 (1973)

[2] **Moon, J.W., Moser, L.**: On cliques in graphs. Israel J. Math. 3(1), 23–28 (1965)

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

University of California · Irvine

Center for Algorithms and Theory of Computation
Donald Bren School of Information and Computer Sciences
University of California, Irvine

Supported by: Office of Naval Research award N00014-08-1-1015.

Scalable Methods for the Analysis of Network-Based Data