Arboricity and Bipartite Subgraph Listing Algorithms

David Eppstein*

Department of Information and Computer Science
University of California, Irvine, CA 92717

Tech. Report 94-11

February 24, 1994

Abstract

In graphs of bounded arboricity, the total complexity of all maximal complete bipartite subgraphs is $O(n)$. We describe a linear time algorithm to list such subgraphs. The arboricity bound is necessary: for any constant k and any n there exists an n-vertex graph with $O(n)$ edges and $(n/\log n)^k$ maximal complete bipartite subgraphs $K_{k,\ell}$.

*Work supported in part by NSF grant CCR-9258355.

1 Introduction

A number of graph algorithms depend on finding all subgraphs of a certain type in a larger graph. For instance, in interval or chordal graphs, a decomposition into maximal cliques is key; such a decomposition can be constructed in linear time [4, 17]. Optimal triangulation construction [3] and certain planar graph computations [8] require a listing of all triangles. Related subgraph isomorphism problems also occur in a wide variety of practical applications [2, 5, 12, 9, 13, 14, 19].

For planar graphs, or more generally for graphs of bounded arboricity, the problem of listing clique subgraphs is well understood. Chiba and Nishizeki [6] show that there can be at most $O(n)$ cliques of a given size in such graphs, and they further describe linear time algorithms for listing these cliques. An alternative linear time algorithm and its parallelization is presented in [7].

Enumeration of incomplete subgraphs is less well understood, but some results are known. In planar graphs, a given graph G is limited to $O(n)$ occurrences as a subgraph if and only if G is 3-connected [10]; however the only such graphs for which it is known how to list the occurrences of G in linear time are cliques (as noted above) and wheels [10]. Chiba and Nishizeki studied C_4 subgraphs in graphs of bounded arboricity; there can be $\Omega(n^2)$ such 4-cycles but an implicit representation of them can be found in linear time [6].

In this paper we again consider listing subgraphs of bounded arboricity graphs. The *arboricity* $a(G)$ of a graph G is the minimum number of forests into which the edges of G can be partitioned [16]. Every planar graph has arboricity at most three [16]; many other classes of graphs enjoy a bounded arboricity, including geometric neighborhood graphs [15], graphs embeddable on some fixed surface, partial k-trees, and bounded degree graphs. Indeed all of these are sparse graphs, and a bound on arboricity is equivalent to a notion of hereditary sparsity. A decomposition of a planar graph into three forests can be found in linear time [7, 18]. For general graphs the exact arboricity can computed using matroid intersection techniques in time $O(mn \log n)$ [11]. However, for our applications we will be satisfied by an approximation to the arboricity, computed very easily in linear time (Lemma 3).

It is natural, given the known results on complete subgraphs, to try to extend these results to complete bipartite subgraphs. The problem of finding complete bipartite subgraphs arises for instance in connection with

certain methods of data compression for graphs [1]. However, there is no $O(n)$ bound on the number of such subgraphs. For instance $K_{2,x}$ appears $\binom{n-2}{x}$ times as part of the planar graph $K_{2,n-2}$. Further we must worry not only about the number of subgraphs, but their complexity, if we are to be finding subgraphs of nonconstant size. However in this example there is only one *maximal* complete bipartite subgraph, namely the graph itself, and of course it has complexity $O(n)$. We will see that this behavior is typical: for any graph of bounded arboricity, there are $O(n)$ maximal complete bipartite subgraphs, and they have total complexity $O(n)$. We further show that these subgraphs can be found in linear time.

This result also provides an implicit representation for non-maximal complete bipartite subgraphs, for any such subgraph is an induced subgraph of a maximal complete bipartite subgraph, and any induced subgraph of a maximal complete bipartite subgraph is itself complete bipartite. This generalizes the result of Chiba and Nishizeki on an implicit representation of $C_4 = K_{2,2}$ subgraphs. Indeed, their representation consists of a collection of $K_{2,x}$ subgraphs, however these are neither guaranteed to be maximal nor are all maximal $K_{2,x}$ subgraphs included in their representation. It is not clear how to generalize their techniques even to find all $K_{2,3}$ subgraphs. Our algorithm uses techniques that differ from theirs, and is more similar in spirit to that of [7].

2 Bounded Acyclic Orientations

Our algorithm is based on a technique from [7], used in that paper as a space-efficient data structure for testing whether an edge is part of the given graph. Given any undirected graph G, a *d-bounded orientation* of G is simply an orientation in which each vertex has out-degree at most d.

Lemma 1 (Chrobak and Eppstein [7]). *If a graph G has arboricity a, it has an a-bounded orientation.*

Proof: Partition G into a trees, each directed toward its root. □

Lemma 2 (Chrobak and Eppstein [7]). *If a graph G has a d-bounded orientation, and if we store for each vertex a list of out-edges in the orientation, then we can use this representation to test whether or not a potential edge (x, y) is part of the graph, in time $O(d)$.*

Proof: We simply check for each of the at most d out-edges of each of x and y, whether that edge is the one we are testing. If no edge matches, the query edge does not exist. □

An *acyclic orientation* is one in which there is no directed cycle. Chrobak and Eppstein [7] used acyclic orientations in their algorithm for clique enumeration. Acyclicity will also be needed in our bipartite subgraph enumeration algorithm. An advantage of acyclic orientations is that they are easy to construct.

Lemma 3 (Chrobak and Eppstein [7]). *If a graph G has a d-bounded orientation, it has a $2d$-bounded acyclic orientation which can be constructed in linear time without knowledge of d.*

Proof: Since G has at most dn edges, some vertex v has at most $2d$ neighbors. We remove the minimum degree vertex from G, leaving a smaller graph which has an induced d-bounded orientation and hence by induction a $2d$-bounded acyclic orientation. To form an orientation of G we direct all edges out of v.

We can implement this algorithm to run in linear time as follows. We sort the vertices of G into buckets by their degrees, and maintain at all times a sorted doubly linked list of nonempty buckets. At each step, we remove a vertex from the graph, possibly emptying its bucket. We also reduce the degrees of neighboring vertices by one, which may either create a new bucket or move a vertex to an existing bucket. Each of these steps can be implemented in constant time, and the number of such steps is bounded by the number of edges and vertices in the graph. □

We note that if a graph has arboricity a, the degree bound in Lemma 3 can be tightened from $2a$ to $2a - 1$, as there can be at most $a(n - 1) < an$ edges in G. Since our subgraph listing algorithms depend on the degree of the orientation, this improvement gives a small constant factor improvement in our bounds. As a converse to the lemmas above, a graph with a d-bounded acyclic orientation must have arboricity at most d, for we can partition its edges into d trees simply by choosing one out-edge per tree from each vertex. Thus the existence of a bounded orientation is equivalent to a bound on the arboricity.

3 Listing Bipartite Subgraphs

Given a set of vertices A, the *induced complete bipartite subgraph* $A \times B$ is found by letting B consist of all the common neighbors of A. This will not necessarily be a maximal complete bipartite subgraph in G: some superset of A may have the same set of common neighbors. Nevertheless all maximal complete bipartite subgraphs can be found by choosing the appropriate sets A. Also note that in a graph of arboricity a, at least one of A or B must be small (have $2a-1$ or fewer vertices) because otherwise the subgraph $K_{2a,2a}$ would have arboricity at least $a+1$ (if it had arboricity a or less, it would have a vertex of degree $2a-1$ or less as noted after Lemma 3).

We now show how to use bounded acyclic orientations as a data structure for finding common neighbors of sets of vertices. We say that a complete bipartite subgraph $A \times B$ is *generated* by a set A of vertices if B consists of all common neighbors of vertices in A. If $A \times B$ is maximal, it must be generated both by A and by B.

Lemma 4. *Given a graph with a d-bounded orientation, and given a collection of sets A_i with total size m, we can compute the graphs $A_i \times B_i$ generated by each A_i, in total time $O(d2^d n + d^2 m)$.*

Proof: For each set A_i, and each v in the set of common neighbors B_i, one of two cases can occur: either all edges from v to A_i are directed away from v and included in the list of out-edges of v, or some edge is directed from A_i to v and included in the lists of out-edges of A_i.

For the second case, in which an edge to v is included in the out-edges of A_i, there can be at most $d|A_i|$ edges and hence at most $d|A_i|$ neighbors. For each such neighbor v we count the edges directed from A_i to v by scanning in time $O(d|A_i|)$ all out-edges of A_i. Then we scan the at most d out-edges of v and count the edges to A_i (we can test each edge in time $O(1)$ by keeping a bit per vertex distinguishing members of A_i from the rest of G). If the total number of edges found equals $|A_i|$, we have found a member of B_i. This phase of the algorithm takes time $O(d^2|A_i|)$ for each A_i, and totals $O(d^2 m)$.

In the other case, we must find those members of B_i for which all edges are directed towards A_i. In such cases $|A_i| \leq d$ or no such member could exist. We create a sequence of pairs $(A_i, -i)$ for the sets A_i that have d or fewer vertices, together with pairs (S, v) for each subset of the out-neighbors of each vertex v of G. There are thus $O(m + 2^d n)$ pairs formed. We bucket sort them in $O(dm + d2^d n)$ time, after which the common neighbors for

this case can be found for each set A_i by examining the pairs (A_i, v) which appear adjacent to $(A_i, -i)$ in the sorted order. □

Lemma 5. *If G has a d-bounded acyclic orientation, then for any complete bipartite subgraph $A \times B$, one of A or B is a subset of the out-neighbors of some vertex v in G.*

Proof: Let v be the first vertex in A or B to appear in a topological ordering of the acyclic orientation. □

Theorem 1. *If a graph G has arboricity $a = O(1)$, we can list the maximal complete bipartite subgraphs of G in time $O(n)$.*

Proof: By Lemmas 1 and 3, we can find a $2a$-bounded acyclic orientation of G. By Lemma 5, each maximal complete bipartite subgraph must be generated by one of the at most $2^{2a}n$ subsets of out-neighbors of vertices. We can eliminate duplicate copies of the same sets using bucket sorting. Then by Lemma 4 (with $d = 2a$ and $m \leq (2a)2^{2a}n$) the bipartite subgraphs generated by these sets can be found in time $O(a^3 2^{2a} n)$.

Not all of these graphs will be maximal, but we can eliminate the non-maximal ones by counting the sizes of the sets B_i generated by Lemma 4 and by testing, for each A_i, whether some superset A_j has a B_j of the same size. To save time, we only test A_j satisfying $|A_j| = |A_i| + 1$. These tests can be performed by bucket sorting the m sets A_i, with their sizes, together with the $O(dm)$ sets formed by removing one point in each possible way from each A_i, after which A_i will appear in the list consecutively with the equal sets coming from its supersets. This step takes time $O(d^2 m) = O(a^3 2^{2a} n)$ matching the previous bound. □

Corollary 1. *In any graph of arboricity a, there are at most $2^{2a}n$ maximal complete bipartite subgraphs, and these subgraphs have a total of $O(a^2 2^{2a} n)$ vertices and $O(a^3 2^{2a} n)$ edges.*

4 Lower Bounds

Our results use the relatively weak assumption of bounded arboricity (a stronger assumption would be the absence of some graph minor). However one might think that weaker assumptions, such as having few edges, could

perhaps be used in our results. One might also question the necessity of our exponential dependence on a: our bounds have the form $O(2^{ca}n)$ rather than $O(a^c n)$. Here we show that both an arboricity bound and our exponential dependence are necessary: there are sparse graphs with large arboricity that have a number of maximal complete bipartite subgraphs superlinear in n, and there are graphs with bounded arboricity a that have a number of maximal complete bipartite subgraphs exponential in a.

First consider the clique K_n. It has arboricity $n-1$, and 2^{n-1} maximal bipartite subgraphs, one for each partition of the vertices. If we form a union of n/a cliques K_a we get a graph with arboricity a and $\Omega(2^a n/a)$ maximal bipartite subgraphs, having a total of $\Omega(2^a n)$ vertices and $O(a2^a n)$ edges. By including a clique $K_{\sqrt{n}+1}$ in a larger graph we get a graph with $O(n)$ edges and $2^{\sqrt{n}}$ maximal bipartite subgraphs, showing that sparsity alone is not sufficient for a subexponential bound on the number of such subgraphs.

Even if we only count complete bipartite graphs $K_{k,\ell}$ for which we have some fixed bound on k, sparsity alone is still not sufficient to prove a bound linear in n. The same clique example shows that there can be $\binom{n}{k} = \Omega(n^k)$ maximal complete bipartite subgraphs $K_{k,\ell}$ in a dense graph, or $\Omega(n^{k/2})$ in a sparse graph. We now tighten the latter bound. Consider the bipartite graph (A, B, E), where $|A| = n$, $|B| = 2^{k+1}\log_2 n$, and the edges connecting A to B are chosen independently at random with probability $1/2$. Then any k-tuple of vertices in A should expect to have $2\log_2 n$ common neighbors in B. For any other vertex in A outside the k-tuple, the probability of being adjacent to all those neighbors is $O(1/n^2)$, so with high probability this is true of no vertex and the k-tuple induces a maximal complete bipartite subgraph $K_{k,\ell}$. Thus with nonzero probability there are at least $\binom{n}{k}(1-o(1))$ such maximal complete bipartite subgraphs. By reducing the size of A we can find a graph with $O(n)$ edges and $\Omega((n/\log n)^k)$ maximal complete bipartite subgraphs $K_{k,\ell}$.

We leave as open problems the gap between our $\Omega(2^a n/a)$ lower bound and our $2^{2a}n$ upper bound on the number of maximal complete bipartite subgraphs, and the similar gaps on the numbers of vertices and edges in such subgraphs.

References

[1] P. Agarwal, N. Alon, B. Aronov, and S. Suri. Can visibility graphs be represented compactly? In *Proc. 9th ACM Symp. Computational Geometry*, pages 338–347, 1993.

[2] P. J. Artymiuk, P. A. Bath, H. M. Grindley, C. A. Pepperrell, A. R. Poirrette, D. W. Rice, D. A. Thorner, D. J. Wild, P. Willett, F. H. Allen, and R. Taylor. Similarity searching in databases of three-dimensional molecules and macromolecules. *J. Chemical Information and Computer Sciences*, 32:617–630, 1992.

[3] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In F. K. Hwang and D. Z. Du, editors, *Computing in Euclidean Geometry*, pages 23–90. World Scientific, 1992.

[4] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Sys. Sci.*, 13:335–379, 1976.

[5] A. D. Brown and P. R. Thomas. Goal-oriented subgraph isomorphism technique for IC device recognition. *IEE Proceedings I (Solid-State and Electron Devices)*, 135:141–150, 1988.

[6] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Computing*, 14:210–223, 1985.

[7] M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theoretical Computer Science*, 86:243–266, 1991.

[8] M. B. Dillencourt and W. D. Smith. A linear-time algorithm for testing the inscribability of trivalent polyhedra. In *Proc. 8th ACM Symp. Computational Geometry*, pages 177–185, 1992.

[9] Dong Hong, Wu Youshou, and Ding Xiaoqiag. An ARG representation for Chinese characters and a radical extraction based on the representation. In *9th IEEE Intl. Conf. Pattern Recognition*, volume 2, pages 920–922, 1988.

[10] D. Eppstein. Connectivity, graph minors, and subgraph multiplicity. *J. Graph Theory*, 17:409–416, 1993.

[11] H. N. Gabow and H. H. Westermann. Forests, frames, and games: algorithms for matroid sums and applications. In *20th ACM Symp. Theory of Computing*, pages 407–421, 1988.

[12] A. Guha. Optimizing codes for concurrent fault detection in microprogrammed controllers. In *Proc. IEEE Intl. Conf. Computer Design: VLSI in Computers and Processors (ICCD '87)*, pages 486–489, 1987.

[13] S. Y. T. Lang and A. K. C. Wong. A sensor model registration technique for mobile robot localization. In *Proc. 1991 IEEE Intl. Symp. Intelligent Control*, pages 298–305, 1991.

[14] R. Levinson. Pattern associativity and the retrieval of semantic networks. *Computers & Mathematics with Applications*, 23:573–600, 1992.

[15] G. L. Miller, S.-H. Teng, and S. A. Vavasis. A unified geometric approach to graph separators. In *32nd IEEE Symp. on Foundations of Computer Science*, pages 538–547, 1991.

[16] C. St. J. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36:445–450, 1961.

[17] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:266–283, 1976.

[18] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Symp. Discrete Algorithms*, pages 138–148, 1990.

[19] T. Stahs and F. Wahl. Recognition of polyhedral objects under perspective views. *Computers and Artificial Intelligence*, 11:155–172, 1992.