CS762: Graph-Theoretic Algorithms
Lecture 7: More on Chordal Graphs. Comparability Graphs.
January 21, 2002

Scribe: mike newman

Abstract

We finish discussing chordal graphs. In particular we give a characterization as intersection graphs of subtrees of a tree, which nicely generalizes an analogous formulation of interval graphs. We also summarize some results for chordal graphs. We introduce comparability graphs, and efficient algorithms for recognition, maximum clique and vertex colouring.

1 Introduction

We have formally seen two equivalent characterizations of chordal graphs: they contain no chordless cycle, and they have a perfect elimination order. We now show a third characterization: chordal graphs are exactly those graphs that can be represented as the intersection graph of some collection of subtrees of a tree. More precisely:

Theorem 1 *A graph G with n vertices is chordal if and only if there exists a tree T and subtrees (of T) $T_1, \ldots, T_n$ such that $(v_i, v_j) \in E(G)$ exactly when $T_i \cap T_j \neq \emptyset$.*

This is the principal result on chordal graphs we will prove.

We also introduce the class of comparability graphs.

2 Intersection graphs of subtrees of a tree

Definition 2 *Given a tree T and a set of trees $\{T_i : 1 \leq i \leq n\}$ such that each T_i is a subtree of T, we define a graph as follows. Let $V = \{v_1, \ldots, v_n\}$ and $E = \{v_i v_j : T_i \cap T_j \neq \emptyset$. Then $G = (V, E)$ is the intersection graph of the subtrees $\{T_i\}$.*

There is a potential ambiguity in the above definition. We merely required that two subtrees intersect, meaning that they have (at least) one common vertex. We might be more restrictive and insist that in order to generate an edge in G, the subtrees must intersect in (at least) one edge. We might also insist, for instance, that no subtree is contained in another. In fact these concerns are somewhat immaterial, as shown by the following lemma.

Lemma 3 *Let $T, T_1, \ldots, T_n$ be given, and let G be their intersection graph. Then there exists $T', T'_1, \ldots, T'_n$ such that their intersection graph G' is isomorphic to G, and for every $i \neq j$ either $T'_i \cap T'_j$ is empty or $T'_i \cap T'_j$ contains an edge. Furthermore, for each T'_i, there is some edge $e_i \in T'_i$ such that $e_i \notin T'_j$ for every $j \neq i$.*

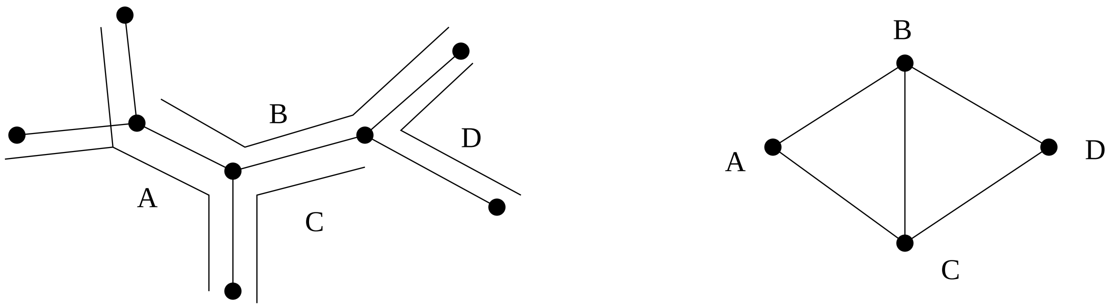

Figure 1: A set of subtrees of a tree, and the intersection graph they generate.

The proof is quite straightforward, and is left as an exercise.

Note that the converse of Lemma 3 is false. That is, if we modify Definition 2 to read as follows, then we obtain a class of graphs that is strictly larger.

Definition 4 *Given a tree T and a set of trees $\{T_i : 1 \leq i \leq n\}$ such that each T_i is a subtree of T, we define a graph as follows. Let $V = \{v_1, \ldots, v_n\}$ and $E = \{v_iv_j : T_i \cap T_j \cap E(T) \neq \emptyset$. Then $G = (V, E)$ is the edge-intersection graph of the subtrees $\{T_i\}$.*

Lemma 3 shows the containment, and Figure 2 shows the strictness. We leave the details as an exercise, but note that the subtrees in Definition 4 do *not* satisfy the Helly property (see Definition 5). Of course, every graph of Definition 4 is a subgraph of some graph of Definition 2.

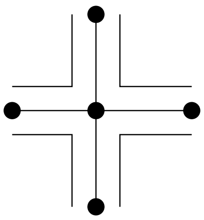

Figure 2: A representation of C_4 as an edge-intersection graph of subtrees of a tree. This cannot be done with a vertex-intersection graph of subtrees of a tree.

So we may assume that subtrees that intersect do so in an edge and that no subtree contains another. In practice, these will not often be needed: the purpose of the above lemma is to demonstrate that these assumptions do not alter the class of graphs. In the remainder we will not insist on either of these conditions.

Before actually proving Theorem 1, we motivate it with the following observation. We showed as an exercise that interval graphs on n vertices can always be represented by a set of closed intervals such that no two endpoints coincide and every endpoint is an integer between 1 and $2n$, inclusively. In fact what we were showing was that interval graphs are precisely intersection graphs of subpaths of a path. We simply take as vertex set of P the integers from 1 to $2n$, in that order. For each interval I_i, the endpoints are "vertices" of P; let P_i be the path in P between the two endpoints of I_i. Then two intervals intersect if and only if the corresponding subpaths intersect. So Theorem 1 generalizes a natural characterization of interval graphs to chordal graphs.

It is not immediately clear that intersection graphs of subtrees of trees are a more general class than intersection graphs of subpaths of paths. This will be a consequence of the proof, but we can see this in a very straightforward way directly. Take any tree T, and root it at an arbitrary

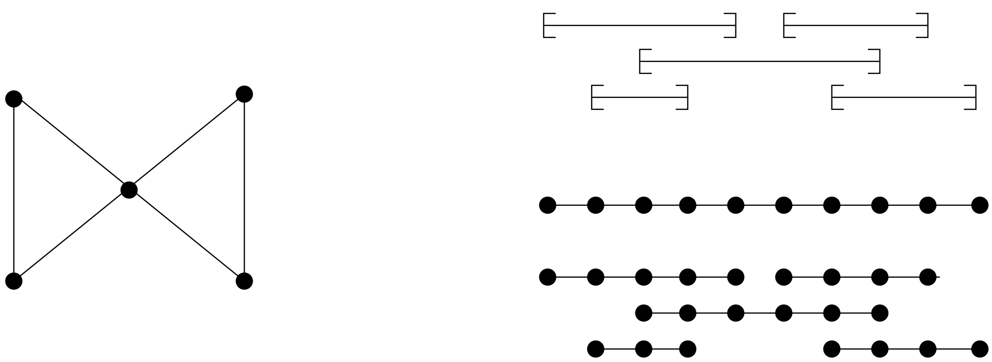

Figure 3: The bowtie graph, an interval representation, and the corresponding subpath intersection representation.

vertex r. Let $T_w = w$ for every vertex w that is a leaf, and for each vertex v that is not a leaf, let T_v be the subtree induced by v and its children. Then T is the intersection graph of the subtrees $\{T_v : v \in V(T)\}$. In fact, it is an intersection graph of subtrees of itself. Thus our first example of a chordal graph that is not an interval graph serves as an example of a graph that is an intersection graph of subtrees of a tree but not of subpaths of a path.

Figure 4: A chordal graph that is not an interval graph, and a representation of it as an intersection graph of subtrees of itself (taking r to be the root in the given construction).

3 Proof of Theorem 1

We will deal with the case of vertex intersection, i.e., we will not insist that intersecting subtrees intersect in (at least) one edge.

It will be useful to remark that a collection of subtrees of a tree satisfies the *Helly condition*, meaning that pairwise intersection implies common intersection. More precisely:

Definition 5 *Let $\mathcal{A}$ be a collection of sets. Then $\mathcal{A}$ is said to satisfy the* Helly property *if for every subcollection $\mathcal{A}' \subset \mathcal{A}$ with $A_i \cap A_j \neq \emptyset$ for all $A_i, A_j \in \mathcal{A}'$, then $\bigcap_{A \in \mathcal{A}'} A \neq \emptyset$.*

To see that this is in fact true for subtrees of a tree, consider the case where T_1, T_2, T_3 are three subtrees of T that intersect pairwise but have no common intersection. Let $v_{i,j}$ be a vertex in $T_i \cap T_j$ for $1 \leq i < j \leq 3$; note that these three vertices must be distinct. Then since trees are connected, there is a (simple) path P_1 between $v_{1,2}$ and $v_{1,3}$ contained in T_1. Likewise there are paths P_2 between $v_{1,2}$ and $v_{2,3}$ in T_2 and P_3 between $v_{1,3}$ and $v_{2,3}$ in T_3. Together, these three paths form a closed walk in T. Note that $v_{1,2} \notin P_3$ (and similarly) so that this closed walk contains a (simple) circuit of length at least three. This is impossible as T is a tree.

We first prove that every intersection graph of subtrees of a tree is chordal.

Proof: Let a tree T be given, and some collection of subtrees of T, $T_1, \ldots, T_n$. We use induction on $|T|$, the number of vertices in T.

If $|T| = 1$, then T is a single vertex, so each $T_i = T$. Thus the intersection graph is complete, which is chordal.

If $|T| \geq 2$, then T has a leaf v. There are two possibilities. If no subtrees intersect in exactly v, then if we delete v from T and any T_i that contain it, intersecting subtrees will remain intersecting, and nonintersecting subtrees will remain nonintersecting. Thus the intersection graph is the same with or without v. But by deleting v, we reduce $|T|$, so the induction hypothesis applies and the intersection graph is chordal. If some subtrees intersect in exactly v, then (since T is a tree and v is a leaf) one of the subtrees must consist of the single vertex v; call this subtree T_n. Note that any other subtree that intersects T_n must contain v, and thus all subtrees that intersect T_n intersect pairwise. Thus in the intersection graph, the neighbours of the vertex corresponding to T_n form a clique. By induction, there is a perfect elimination order for the intersection graph with v deleted; to this, we append the vertex corresponding to T_n. By the above remarks, this is a perfect elimination order for the intersection graph on T. □

Now we show that every chordal graph can be obtained as the intersection graph of subtrees of some tree.

Proof: We build the tree T step by step, adding a new subtree for each vertex as we go.

Let G be a chordal graph with perfect elimination order $\{v_1, \ldots, v_n\}$. We start with $T = T_1 = K_1$, a single vertex. For $i = 2, 3, \ldots, n$, note that the predecessors of v_i form a clique. Therefore the subtrees corresponding to the predecessors of v_i intersect pairwise. By the Helly property, there is (at least) one common vertex p in all of these subtrees. Add to T a new vertex q that is adjacent to exactly p, and add q and the edge pq to each subtree of a predecessor of v_i. Finally set $T_i = \{q\}$. □

4 Final comments on chordal graphs

We now have three equivalent definitions of chordal graphs. They are the graphs with perfect elimination orders, they are graphs that contain no chordless cycle, and they are intersection graphs of subtrees of a tree. There are many other characterizations. The following is described in [Gol80]; we state it without proof. Note that although we did not state it this way, we had previously shown one direction, that every minimal vertex seperator of a chordal graph is a clique.

Proposition 6 *A graph G is chordal if and only if every minimal vertex separator induces a complete subgraph of G.*

We have previously seen linear time algorithms for finding a maximal independent set, a maximal clique, and an optimal vertex colouring for chordal graphs. In fact, the algorithm was simply the greedy algorithm.

We can recognize chordal graphs in linear time [RTL76, TY84]; the same is true of interval graphs [BL76, KM89, HM91, HPV96]. For chordal graphs, finding a maximum cut is NP-hard [BJ00]. It is not clear what the complexity is for interval graphs. Hamiltonian cycles can be found in linear time for interval graphs [CPL93]; $O(m+n)$ and $\Theta(n \log n)$ algorithms are in [Kei85] and [MMS90]. It is not clear what the complexity is for chordal graphs.

5 Comparability graphs

Given an undirected graph, we may orient the edges in some way. If an orientation has the property that it gives rise to no directed cycle, then we say that the orientation is *acyclic*. Note that the underlying graph may contain cycles. If the orientation has the property that for any vertices x, y, z with $x \to y$ and $y \to z$, we also have $x \to z$, then it is said to be *transitive*.

Definition 7 *A graph G that has an orientation that is acyclic and transitive is a* comparability graph*. A (comparability) graph with an orientation that is acyclic and transitive is a* partially ordered set*, or* poset*.*

Note that comparability graphs may also have orderings which aren't acyclic and transitive.

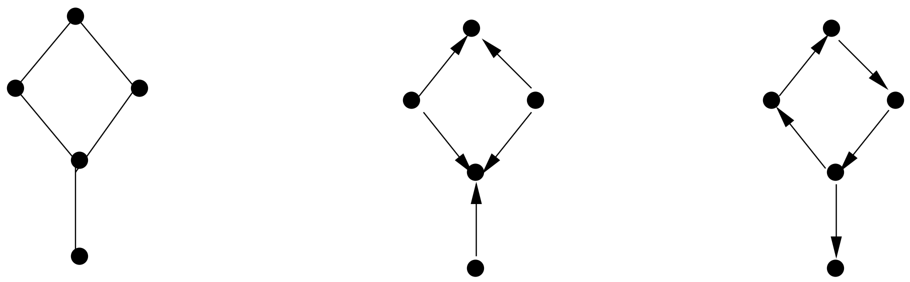

Figure 5: A comparability graph, an acyclic transitive orientation, and an orientation that is neither acyclic nor transitive.

Golumbic gives a linear algorithm for recognizing comparability graphs [Gol80]; see also [MS89, Gol77]. Recall that if G is an interval graph, then the complement of G is a comparability graph. In fact, the ordering is immediate: two intervals are not adjacent if they are disjoint, so we may orient each edge in the complement from the leftmost interval to the rightmost one.

6 Colouring comparability graphs

We begin with a simple observation. Let P be a directed path in a poset. Then by the transitivity of the ordering, every pair of vertices in P is joined by an arc, so the vertices of P form a clique in G. Though this is a simple observation, it is the basis for a linear algorithm for finding $\omega(G)$ and $\chi(G)$ when G is a comparability graph.

Given a poset, we define a function h on the vertices by the following:

$$h(v) = 1 + \max\{h(w) : w \to v\}$$

with the convention that $h(v) = 1$ if v has no incoming edges.

This is well defined if there are no directed cycles, which is always the case in a poset. It can be computed in linear time by traversing the vertices in a topological order. A topological order is an ordering such that all edges go "forward"; more precisely, a topological order of a directed graph is an ordering "$\prec$" of the vertex-set such that the existence of an edge (v_i, v_j) implies $v_i \prec v_j$.

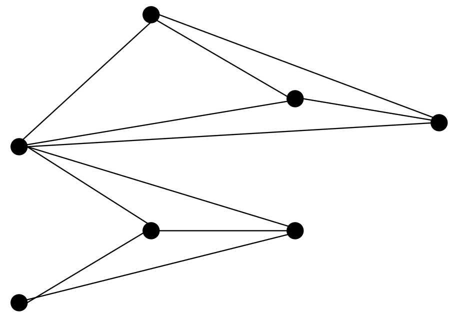

Figure 6: A poset, with vertices arranged in columns according to the function h. Edges are oriented from left to right.

Note that if $v \to w$ then $h(w) \geq h(v) + 1$. In other words, for any fixed i, the set of vertices $\{v : h(v) = i\}$ is an independent set. Let k be the maximum value attained by h. Then we may colour each vertex with colour $h(v)$, giving a proper k-colouring. Now if $h(v) > 1$ for some vertex v, then by definition it must have a predecessor w with $h(w) = h(v) - 1$ (this need not be true for *all* of its predecessors). Thus there is a directed path of length k. But this path is in fact a complete subgraph, meaning that G contains a k-clique. Note that in Figure 6 there is a 4-path, which induces a 4-clique and the height function gives a 4-colouring.

Now if we combine the remarks in the previous paragraph, we have:

$$\chi(G) \leq k \leq \omega(G).$$

But in *any* graph, we always have $\chi(G) \geq \omega(G)$, so in fact in a comparability graph we have

$$\chi(G) = k = \omega(G).$$

We have just established the following result.

Theorem 8 *In a poset G the values of $\chi(G)$ and $\omega(G)$ (as well as an optimal colouring and a maximal clique) can be computed in linear time.*

References

[BJ00] H.L. Bodlaender and K. Jansen. On the complexity of the maximum cut problem. *Nordic Journal of Computing*, 7(1):14, 2000.

[BL76] K.S. Booth and G.S. Lueker. Testing for the consecutive one's property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput Sytem Sci.*, 13:335–379, 1976.

[BLS99] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey.* SIAM Monographs on Discrete Mathematics and Applications, 1999.

[CPL93] M.-S. Chang, S.-L. Peng, and J.-L. Liaw. Deferred-query – an efficient approach for problems on interval and circular-arc graphs. *Algorithms and data structures, WADS'93*, pages 223–233, 1993.

[Gol77] M.C. Golumbic. The complexity of comparability graph recognition and colouring. *Computing*, 18:199–208, 1977.

[Gol80] M.C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[HM91] W.-L. Hsu and T.H. Ma. Substitution decomposistion on chordal graphs and applications. *ISA'91 algorithms, Lecture Notes in Computer Science*, 557:52–60, 1991.

[HPV96] M. Habib, C. Paul, and L. Viennot. lexBFS, a partition refining technique. Application to transitive orientation, interval graph recognition and consecutive one's testing. Unpublished manuscript, 1996.

[Kei85] J.M. Keil. Finding hamiltonian circuits in interval graphs. *Information Processing Letters*, 20(4):201–206, 1985.

[KM89] N. Korte and R.H. Mohring. An incremental linear time algorithm for recognizing chordal graphs. *SIAM J. Computing*, 18:68–81, 1989.

[MMS90] G.K. Manacher, T.A. Mankus, and C.J. Smith. An optimum $\theta(n \log n)$ algorithm for finding a canonical Hamiltonian path and a canonical Hamiltonian circuit in a set of intervals. *Information Processing Letters*, 25:205–211, 1990.

[MS89] J.H. Muller and J.P. Spinrad. Incremental modular decomposistion. *J. Assoc. Comput. Mach.*, 36:1–19, 1989.

[RTL76] D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic aspects of vertex elimination in a graph. *SIAM J. Computing*, 5:266–283, 1976.

[TY84] R.E. Tarjan and M. Yannakakis. Simple linear time algorithms to test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM J. Computing*, 13:566–579, 1984.