CSL851: Algorithmic Graph Theory **Fall 2013**

Lecture 3: Comparability Graphs

Lecturer: Prof. Amit Kumar *Scribes: Arpit Jain*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

This lecture develops the concepts of *comparability* and *co-comparability* graphs. Then, we define *perfect graphs* and their properties.

3.1 Comparability Graph

Definition 3.1 *A comparability graph is an undirected graph in which it is possible to orient each edge such that the resultant graph (G=(V, U)) has the following properties*

1. Anti-symmetry*: If edge $u \rightarrow v$ exists, then $v \rightarrow u$ does not.*
2. Transitivity*: If edges $u \rightarrow v$ and $v \rightarrow w$ exist, then so does $u \rightarrow w$.*

Note: A comparability graph with oriented edges has to be acyclic, or else it violates the anti-symmetry property.

In effect, comparability graphs capture *partial order* or *precedence* between pairs of vertices. For instance, an edge $u \rightarrow v$ may convey the constraint that the event u must precede the event v. In general, the problem of assigning directions to edges is not trivial. In the rest of the lecture, the orientations of the edges are assumed to be given.

3.1.1 Example

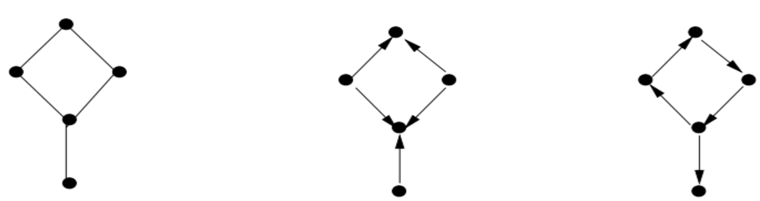

Figure 3.1: A comparability graph, an acyclic transitive orientation and orientation that is neither acyclic not transitive

3.1.2 Relationship to other graph classes

There is no general relation between comparability graphs and chordal graphs.

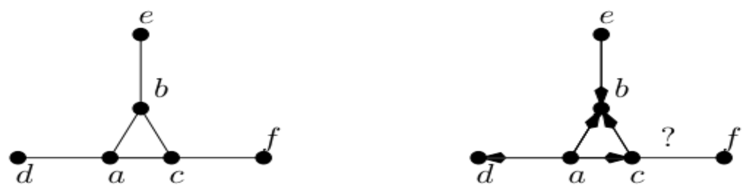

Figure 3.2: A chordal graph that is not a comparability graph

The above graph is chordal as it does not have an induced cycle of length ≥ 4.

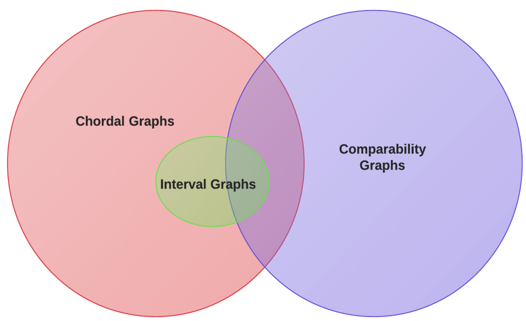

Figure 3.3: Relation between chordal and comparability graphs

3.2 Co-comparability Graphs

Definition 3.2 *G is co-comparability if G^C (the complement graph of G) is comparability.*

Lemma 3.3 *Every interval graph is co-comparability*

Proof: Consider an interval graph G=(V, E) and its complement G^C=(V, E'). Let $u, v \in V$. Now $(u, v) \in E'$ iff $(u, v) \notin E$. For each $(u, v) \in E'$, we should be able to orient it so that it satisfies the properties mentioned earlier. Since u and v represent intervals, there would be an edge between them in G^C iff they do not intersect. Orient the edge as follows: $u \to v$ if u starts and ends before v, or $v \to u$ otherwise.

The above orientation satisfies transitivity, since if interval u comes before v which comes before w, then u also comes before w. Also, it is anti-symmetric, because if interval u precedes v, then u starts and ends before v starts. Hence v cannot precede u. ■

Note: A co-comparability graph may not be interval. Actually, it may not even be chordal. For instance, refer to figure 4.

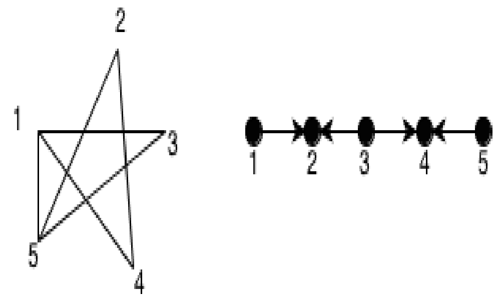

Figure 3.4: Co-comparability graph G and its complement, which is comparability. Note that G is not chordal

3.3 Perfect Graphs

Definition 3.4 *G is a perfect graph is for every induced subgraph H of G,*

$$\chi(H) = cliquesize(H)$$

where $\chi(H)$ denotes the chromatic number of H.

Note that not all graphs are perfect. For e.g. a cycle of length 5 is not perfect.

Theorem 3.5 *If G is perfect, then G^C is also perfect.*

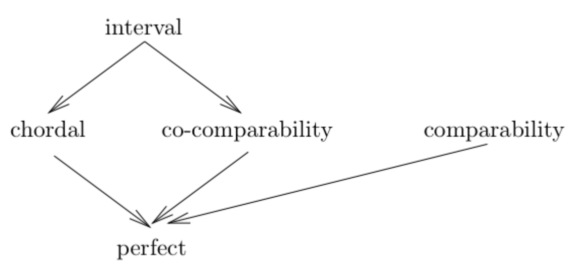

Figure 3.5: Relationship between graph classes

3.3.1 Perfect Graph Conjecture

G is perfect iff it does not contain an odd cycle (of length ≥ 5) or complement of odd cycle (of length ≥ 5) as an induced subgraph.

3.4 Comparability Graphs Revisited

Suppose G=(V, E) is a comparability graph (assume orientations are given). We want to find the largest clique in G.

Lemma 3.6 *Every directed path is a clique. Further, the longest path is the largest clique.*

Proof: We first prove that any path gives a clique. Consider a directed path $(v_1, v_2, ..., v_k)$. Because of transitivity, there are edges from v_1 to $v_3, ...v_k$, v_2 to $v_4, .., v_k$ and so on. Hence there are edges between every pair of vertices on the path (in the undirected version of the graph). Hence it is a clique.

Next we prove that any clique gives a path. Note that a clique is a DAG, with an edge between every pair of vertices. Also it has a *topological sort* where each edge goes from left to right. Hence it gives a path. ■

3.4.1 Finding the longest path

To find the longest path, we arrange the vertices in a topological sort and use dynamic programming. Let $L(v_i)$ denote the longest path starting at vertex v_i.

$$L(v_i) = max\{L(w) + 1, \text{ where } w \in \text{ Nbr}(v_i)\}$$

3.4.2 Chromatic Number

We want to find the chromatic number of a comparability graph. Continuing with the formulation for the longest path, consider a situation where $L(v_i) \leq L(v_j)$. In such a case, there cannot be an edge from v_i to v_j. If there was such an edge, $L(v_i) > L(v_j)$.

This leads to an algorithm where we can form sets of vertices, where each set contains vertices with $L(v)$ value $1, 2, .., \omega$. Since vertices in same set form an independent set, they can be colored with the same color.

3.4.3 Independent Set (for Bipartite Graphs)

Consider the problem on a bipartite graph. We know that

$$\text{Max independent set } \equiv \text{Min vertex cover } \geq \text{size of any matching}$$

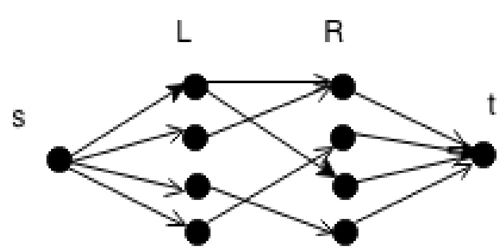

Figure 3.6: Construction for max independent set on G=(L, R)

We carry out the above construction, where the edges from s to L and R to t have capacity 1, while edges from L to R have capacity ∞; and solve the min cut problem on the above graph. Let the min cut be (S, T) and define $R_1 = S \cap R, L_1 = S \cap L, R_2 = T \cap R, L_2 = T \cap L$. Note that in the min cut, there cannot be edges from L_1 to R_2 (because they have infinite capacity). There are two types of edges - those which go from s to L_2 and those which go from R_1 to t.

Claim: $R_1 \cup L_2$ is a vertex cover. If there it is not a vertex cover, then there exists an edge in $R_2 \cup L_1$. But as argued above, there cannot be such an edge.

Not that max flow = Size of min cut = $|R_1 \cup L_2|$ = size of min vertex cover. Hence for a bipartite graph, the independent set can be found efficiently by taking the complement of min vertex cover obtained by the above procedure.