CS762: Graph-Theoretic Algorithms
Lecture 8: Comparability Graphs
January 23, 2002

Scribe: Mohammad Ali Safari

Abstract

Comparability graphs are a well-known class of graphs that have a close relationship with interval graphs. A comparability graph with its edge orientation is called a poset. Many known NP-complete problems have polynomial time algorithms for posets. In this lecture we solve some well-known NP-complete problems on posets and show the relationship between comparability graphs, interval graphs, and chordal graphs.

1 Introduction

An undirected graph G is a *comparability* graph iff there is an acyclic and transitive orientation for its edges. G is *co-comparability* graph iff $\overline{G}$ is comparability graph. We know that if a graph is an interval graph then it is co-comparability graph, but the converse is not correct, i.e. there is a co-comparability graph which is not interval graph(see Fig.1).

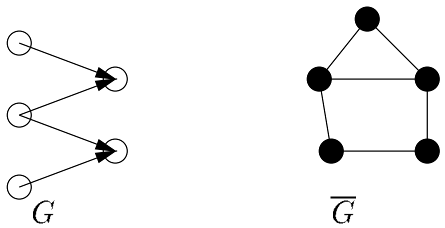

Figure 1: G is comparability graph but $\overline{G}$ is not an interval graph.

A comparability graph with its edge orientation is called a *partially ordered set* or *poset*. A comparability graph may have several edge orientations, each of which is a poset.

In the next section we present some polynomial time algorithms for some well-known NP-complete problems on posets. In section 4 we show the relationship among comparability graphs, interval graphs and chordal graphs.

2 Some algorithms

2.1 Maximum Clique

Posets are directed acyclic graphs (DAG) and so have a *topological ordering*. We can compute this ordering in linear time. If we have h levels in the topological ordering, then there is a path of length h from a vertex in the h^{th} level to some vertices in the first level. Because the graph is a poset, this path will form a clique (see Fig.2). This is the maximum clique, because every clique will have at most one vertex from each layer, and so can not have more than h vertices. It is clear that our algorithm runs in linear time.

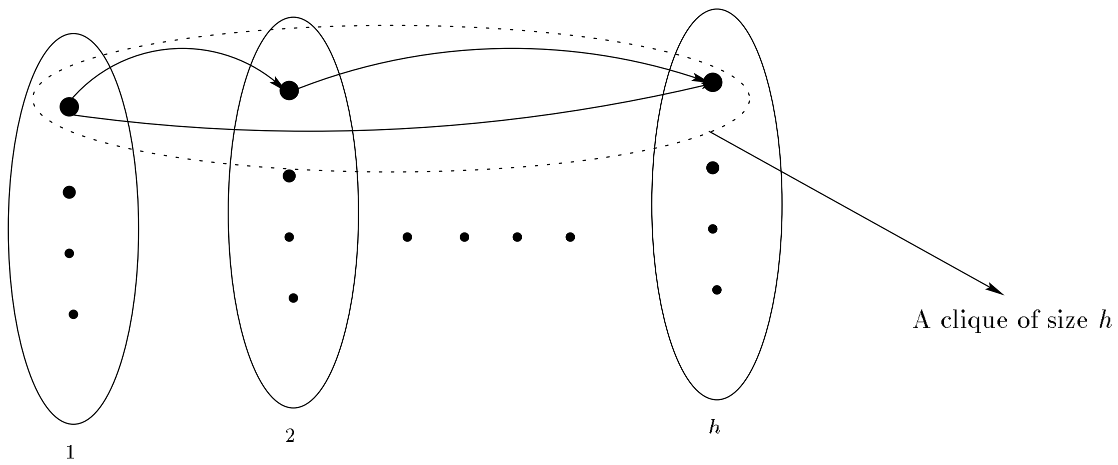

Figure 2: A poset with its maximum clique.

Also, we can color vertices of a poset with exactly h colors(one color for each level). So $\omega(G) = \chi(G) = h$ for the poset G.

2.2 Maximum Weighted Clique

Let us assign weights to the vertices of the poset. We define the weight of the clique containing vertices $v_1, v_2, \cdots, v_m$ as $W(v_1) + W(v_2) + \cdots + W(v_m)$, where $W(v)$ is weight of vertex v. We want to find a clique with maximum weight over all cliques.

Using dynamic programming technique, we solve this problem for a poset. Suppose that the poset G with topological ordering $v_1, v_2, \cdots, v_n$ is given. We define $C(v)$ for a vertex v as the weight of the maximum clique containing v as the vertex with the maximum order in the topological ordering. We can calculate C as follow:

for $i \leftarrow 1$ to n

Initialize $C(v_i)$ to be 0.

for all predecessors v_j of v_i do

$$C(v_i) \leftarrow Max\{C(v_i), W(v_i) + Max(0, C(v_j))\}$$

Finally, the weight of the maximum clique is $Max(C(v_1), C(v_2), \cdots, C(v_n))$.

Claim 1 *Our algorithm computes a maximum weighted clique.*

Proof: Suppose that the maximum weighted clique containing v_i as the vertex with the most order is $v_{i_1}, v_{i_2}, \cdots, v_{i_k}$ where $v_{i_k} = v_i$. If $k = 1$ it is clear that our algorithm works correctly. Otherwise, for $k \geq 2$, all v_{i_j}'s for $j < k-1$ are predecessors of $v_{i_{k-1}}$ and so $v_{i_1}, v_{i_2}, \cdots, v_{i_{k-1}}$ should be the maximum weighted clique containing $v_{i_{k-1}}$ as the vertex with the most order. So, its weight is $C(v_{i_{k-1}})$ and therefore the weight of the first clique is $W(v_i) + C(v_{i_{k-1}})$ which is calculated in our algorithm. $\square$

We can easily verify that our algorithm works in linear time. In addition, with some small changes in algorithm, we can keep track of vertices of the maximum clique and also generate the maximum weighted clique itself.

2.3 Maximum Independent Set

Suppose that the number of layers in the topological ordering of our poset G is h. All vertices in a layer will form an independent set. One may guess that $\alpha(G)$[1] equals to the maximum number of vertices in a layer. The graph in Fig.3 shows that this is not correct. But, we can still find the maximum independent set of a poset in polynomial time.

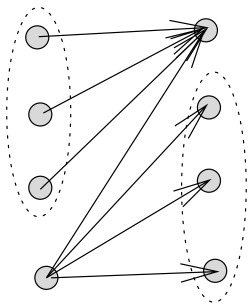

Figure 3: The maximum independent set of a comparability graph

Definition: Let P be a poset. A *chain* in P is a set of pairwise comparable elements (i.e., a totally ordered subset).

Definition: Let P be a poset. An antichain in P is a set of pairwise incomparable elements.

Theorem 1 (Dilworth 1950) *The minimum number of chains needed to cover a poset is equal to the maximum cardinality of an antichain in that poset.*

Using Dilworth theorem, instead of $\alpha(G)$ we calculate $\kappa(G)$[2] . To calculate $\kappa(G)$, we add two vertices s and t and edges $s \to x$ and $y \to t$ for all sources x and sinks y in graph. Then we assign the maximum capacity of each vertex to 1 and run maximum flow on this graph. It is easy to see that the maximum flow on this network equals to $\kappa(G)$. This can be done in polynomial time.

2.4 More on algorithms for posets

There are polynomial time algorithms for some problems such as maximum independent set, maximum clique, minimum clique cover, generating all maximal cliques, counting number of maximum cliques.

But, some problems such as Hamiltonian Cycle and Dominating Set are still NP-complete on this class of graphs. See [GClml] for an overview. Some good refrences related to this subject are [MS95, Spi85, Gol77, PLE71, CH95, Spi92, McC95, MS89, BL75, Gol80, EPL72, CK92, Kri76, Dew81, PLH83, CP84, GJ79].

[1] $\alpha(G)$ equals to the cardinality of the maximum independent set.

[2] $\kappa(G)$ equals to the minimum number of cliques needed to cover all vertices of G.

3 The dimension of a poset

Every partial order (X, P) can be extended to a *linear order* (also called a *topological order*). Let $\mathcal{L}(P)$ be the set of all linear orders of P. A subset $L \subset \mathcal{L}$ that satisfies $\cap_{l \in L} l = P$ is called a *realizer* of P, and its size is $|L|$. The *dimension* of a poset P is the size of the minimum realizer of P.

Example: The poset P in Fig.4 has dimension 2. A minimum realizer of P is also shown. Notice that the subposet P' which is circled also has dimension 2 and it must appear above element a in one of the linear orders and below element a in the other.

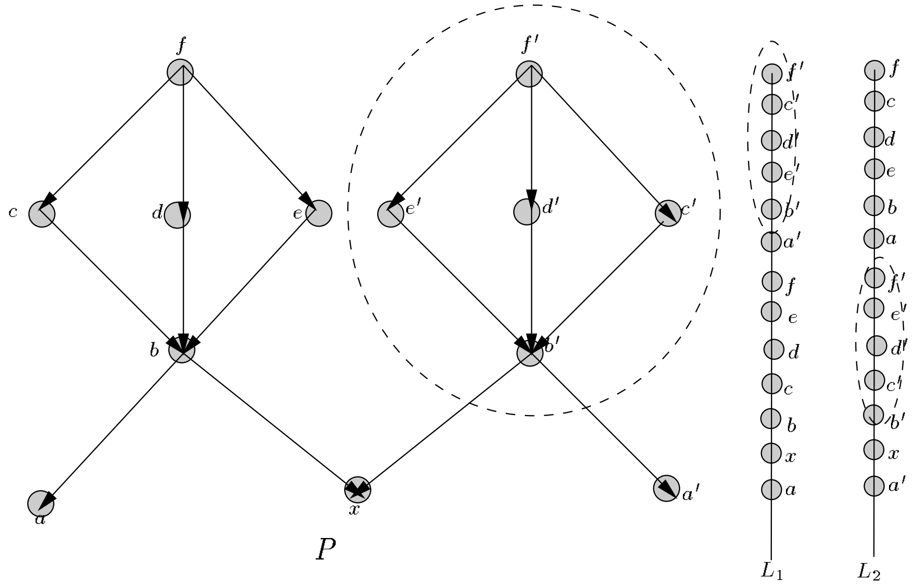

Figure 4: A poset P with dimension 2. We have $P = L_1 \cap L_2$. Note that transitive edges are not shown.

We can represent a linear order using $O(n)$ space, so it is clear that we can represet a poset with dimension D using $O(D \times n)$ space. Therefore, for posets of constant dimension it is a good way of representing them.

A subset of a poset is a subset of its elements with the same order (i.e. an induced subset). The dimension of a subset of a poset is less than or equal to the dimension of the poset itself.

We conclude this section with two theorems on dimension of posets.

Theorem 2 *(Dushnik and Miller [1941]) Let G be a comparability graph of a poset P. Then $Dim(P) \leq 2$ iff G and $\overline{G}$ are both comparability graphs.*

Proof: This was left as an excercise. □

Theorem 3 *(Trotter, Moore, and Sumner [1976]) If two poset P and Q have the same comparability graph G, then $Dim(P) = Dim(Q)$.*

The problem of calculating the dimension of posets is mentioned as a major open problem in the Garey and Johnson book[GJ79]. In 1982, Yannakakis proved that the problem is NP-hard. In addition he proved that deciding whether $Dim(P) \leq k$ is NP-complete, for every $k \geq 3$.

4 Back to interval graphs

So far, we know that every interval graph is chordal and a co-comparability graph. What about the converse? Is every chordal and co-comparability graph an interval graph? The following theorem shows it.

Theorem 4 *The following three statements are equivalent.*

(i) G is an interval graph.

(ii) G has no induced $4-$cycle and is a co-comparability graph.

(iii) The maximal cliques of G can be ordered consecutivevly, i.e. we can arrange them as $C_1, C_2, \cdots, C_k$ such that $\forall v \in V$, the set $\{i \in \{1, 2, \cdots, k\} : v \in C_i\}$ has no gaps.

Proof:

- We know $(i) \Rightarrow (ii)$.
- $(iii) \Rightarrow (i)$ is straightforward:

  Set $I(v) = [min\{i : v \in C_i\}, max\{i : v \in C_i\}]$. Because of (iii) it is clear that if $I(v) = [a, b]$ then $v \in C_i$ iff $i \in [a, b]$.

  If $(u, v) \in G$, then u and v are both in at least one clique C_i and so $i \in I(u) \cap I(v)$. Reversely if $i \in I(u) \cap I(v)$, both u and v are in the clique C_i and so $(u, v) \in G$.
- $(ii) \Rightarrow (iii)$

To show this, we define a directed graph H whose vertices $V(H)$ are all maximal cliques of G and $C \to C'$ in H iff there is an edge $u \to w$ in $\overline{G}$ such that $u \in C$ and $v \in C'$.

Claim 2 *H is transitive.*

Proof: Suppose that (C_1, C_2) and (C_2, C_3) are in H, but (C_1, C_3) is not in H(see Fig.5). There are vertices $a \in C_1$, $b_1, b_2 \in C_2$, $c \in C_3$ such that $(a, b_1), (b_2, c) \in \overline{G}$.

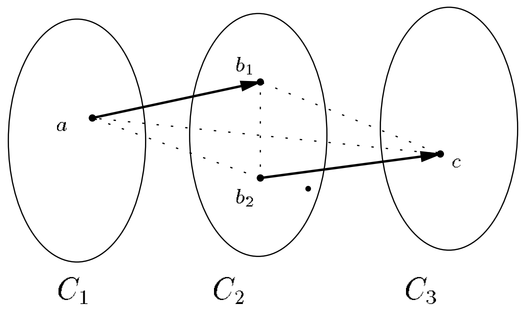

Figure 5: H should be transitive.

If $b_1 = b_2$ then because $\overline{G}$ is transitive, we can deduce that $(a, c) \in \overline{G}$ and so $(C_1, C_3) \in H$, which is not possible. Therefore, $b_1 \neq b_2$.

If $(c, a) \in \overline{G}$, because both (b_2, c) and (a, b_1) are in $\overline{G}$ then $(b_2, b_1) \in \overline{G}$, where b_1 and b_2 are in one clique of G and there can not be any edge between them in $\overline{G}$. Also, it is clear that (a, c) is not in $\overline{G}$. So we can deduce that $(a, c) \in G$.

If $(b_1, c) \in \overline{G}$, because $(a, b_1) \in \overline{G}$ then $(a, c) \in \overline{G}$ and then (C_1, C_3) will be in H. If $(c, b_1) \in \overline{G}$, because $(b_2, c) \in \overline{G}$ then (b_2, b_1) will be in $\overline{G}$. So we can deduce that $(b_1, c) \in G$. Similarly $(a, b_2) \in G$. Now we have the induced 4-cycle (a, c, b_1, b_2) in G which is a contradiction. □

Claim 3 *H is acyclic.*

Proof: Because H is transitive, if H is not acyclic then we have a two-sided edge between two vertices of H, i.e. $C_1 \leftrightarrow C_2$(see Fig.6).

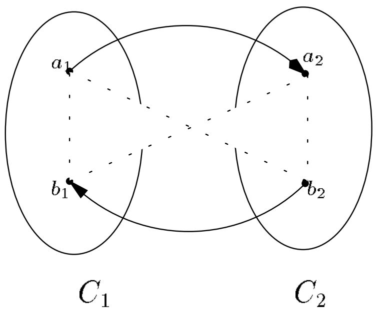

Figure 6: H is acyclic.

Suppose that $(a_1, a_2), (b_2, b_1) \in \overline{G}$ where $a_i, b_i \in C_i$. $a_1 \neq b_1$, because if $a_1 = b_1$ then we have $(b_2, a_1), (a_1, a_2) \in \overline{G}$ and so $(b_2, a_2) \in \overline{G}$ which is not possible. Similarly $a_2 \neq b_2$. There is no edge between a_2 and b_1 in $\overline{G}$. Because if we have $(a_2, b_1) \in \overline{G}$ then $(a_1, b_1) \in \overline{G}$ and if we have $(b_1, a_2) \in \overline{G}$ then (b_2, a_2) will be in $\overline{G}$, but both cases are impossible. Similarly, there is no edge between a_1 and b_2 in $\overline{G}$. So (a_1, b_2) and (a_2, b_1) are both in G and therefore (a_1, a_2, b_1, b_2) is an induced 4-cycle in G. The resulting contradiction shows that H is acyclic. □

We proved that H is transitive and acyclic, so it has a topological ordering. Suppose that $C_1, C_2, \cdots, C_m$ is such a topological ordering.

Claim 4 *This ordering satisfies (iii).*

Proof: Suppose not, then there exist $u \in G$ and numbers $i < j < k$ such that $u \in C_i$ and $u \in C_k$, but u is not in C_j. For all $w \in C_j$, u has not any edge to or from w in $\overline{G}$, because if there were an edge from u to w then there would be an edge from C_k to C_j in H and so C_k can not be after C_j in the topological ordering. The case of edge from w to u is similar. Therefore, there must be an edge between u and w in G. It means that u is adjcent to all vertices in C_j and therefore adding it to C_j will yield a larger clique which contradicts our assumption of the maximality of C_j. □

□

References

[BL75] K.S. Booth and G.S. Lueker. Linear algorithms to recognize interval graphs and test for consecutive ones property. In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, pages 255–265, 1975.

[CH95] Alain Cournier and Michel Habib. A linear algorithm to build modular decomposition trees. Unpublished, 1995.

[CK92] Yang Cai and M.C. Kong. Generation of all maximal cliques and related problems for certain perfect graphs. *Congressus Numerantium*, 90:33–55, 1992.

[CP84] D.G. Corneil and Y. Pearl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9:27–40, 1984.

[Dew81] A.K. Dewdnay. Fast Turing reductions between problems in NP. Technical Report 71, University of Western Ontario, 1981.

[EPL72] Shimon Even, Amir Pnueli, and Abraham Lempel. Permutation graphs and transitive graphs. *Journal of the Association for Computing Machinery*, 19:400–410, 1972.

[GClml] Graph Class/Comarability Graph,
online at http://www.cs.ualberta.ca/˜ stewart/GRAPH/comparability/comparability.html.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, NY, 1979.

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

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

[Kri76] M.S. Krishnamoorthy. An NP-hard problem in bipartite graphs. *SIGACT News*, 7:26, 1976.

[McC95] Ross M. McConnell. An $O(n^2)$ incremental algorithm for modular decomposition of graphs and 2-structures. *Algorithmica*, 14:229–248, 1995.

[MS89] John H. Muller and Jeremy Spinrad. Incremental modular decomposition. *Journal of the ACM*, 36:1–19, 1989.

[MS95] Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. Unpublished, 1995.

[PLE71] Amir Pnueli, Abraham Lempel, and Shimon Even. Transitive orientation of graphs and identification of permutation graphs. *Canadian Journal of Mathematics*, 23:160–175, 1971.

[PLH83] J. Pfaff, R. Lasker, and S.T. Hedetniemi. NP-completeness of connected and total domination and irredundance for bipartite graphs. Technical Report 428, Clemson University, 1983.

[Spi85] Jeremy P. Spinrad. On comparability and permutation graphs. *SIAM Journal of Computing*, 14:658–670, 1985.

[Spi92] Jeremy P. Spinrad. P4-trees and substitution decomposition. *Discrete Applied Mathematics*, 39:263–291, 1992.