Advanced Topics in Graph Algorithms

Technical Report

Lecturer: **Ron Shamir**
Edited by: **Sariel Har-Peled**

Based on Notes from the Course
Advanced Topics in Graph Algorithms

Contents

List of Figures

Chapter 1

Lecture 1

Advanced Topics in Graph Algorithms

Spring Semester, 1994

Lecturer	:	`Ron Shamir`
Scribe	:	`Kahn Shaul`
Lecture 1	:	`April 21`

1.1 Introduction to Graph Theory

1.1.1 Basic Definitions and Notations

A directed graph G consists of a finite set V and an irreflexive binary relation on V. We call the elements of V, *vertices*. The binary relation may be represented either as a collection E of ordered pairs, or as a function from V to its power set: $Adj : V \to P(V)$.

We call $Adj(v)$ the *adjacency set* of vertex v, and we call the ordered pair $(v, w) \in E$ an *edge*. Clearly : $(v, w) \in E \leftrightarrow w \in Adj(v)$. In this case we say that w is *adjacent* to v, and v and w are *endpoints* of the edge (v, w). The assumption of irreflexivity implies that $(v, v) \notin E$ $(v \in V)$, or equivalently, $v \notin Adj(v)$ $(v \in V)$. $Adj(v)$ is sometimes denoted by $N(v)$ and is called the *open neighborhood* of v. The *closed neighborhood* of v is defined :

$$N[v] = v + Adj(v)$$

We will usually drop the parentheses and the comma when denoting an edge. Thus $xy \in E$ and $(x, y) \in E$ will have the same meaning.

In an undirected graph every edge appears in both directions.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are called *isomorphic*, denoted $G \cong G'$, if there is a bijection $f : V \to V'$ satisfying for all $x, y \in V$,

$$(x, y) \in E \leftrightarrow (f(x), f(y)) \in E'$$

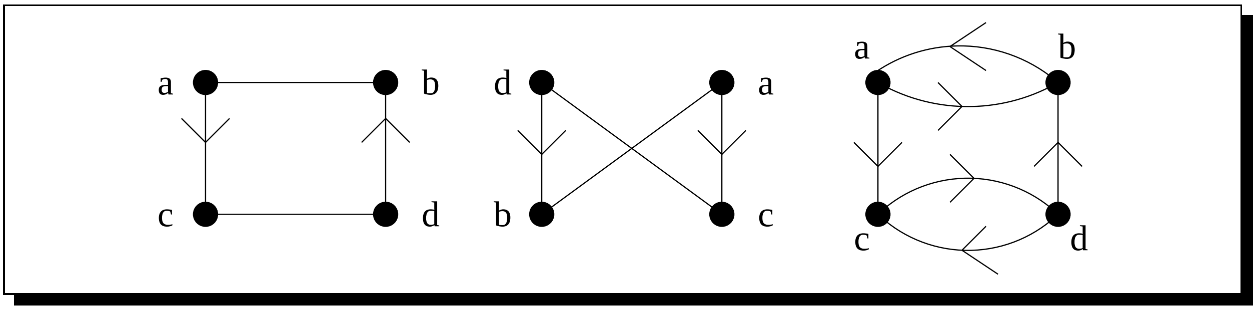


Figure 1.1: Three pictures of the same graph.

Let $G = (V, E)$ be a graph with vertex set V and edge set E. The graph $G^{-1} = (V, E^{-1})$ is said to be the *reversal* of G, where $E^{-1} = \{(x, y) \mid (y, x) \in E\}$, that is,

$$xy \in E^{-1} \leftrightarrow yx \in E \ (x, y \in V).$$

We define *symmetric closure* of G to be the graph $\hat{G} = (V, \hat{E})$, where $\hat{E} = E \cup E^{-1}$.

A graph $G = (V, E)$ is called *undirected* if its adjacency relation is symmetric i.e., if $E = E^{-1}$ or $E = \hat{E}$.

A graph is called an *oriented graph* if $E \cap E^{-1} = \emptyset$.

Orientation of an undirected graph is a choice of a direction on every edge in it.

For example : see Figure 1.2.

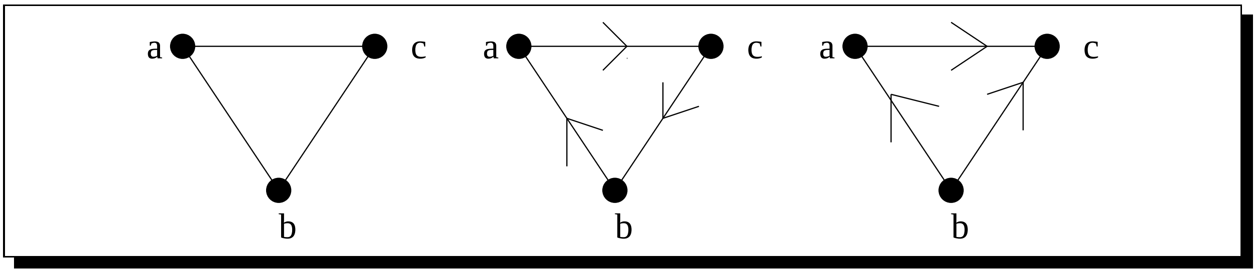


Figure 1.2: Some orientations of a triangle.

Let $G = (V, E)$ be an undirected graph. We define the *complement* of G to be the graph $\overline{G} = (V, \overline{E})$, where

$$\overline{E} = \{(x, y) \in V \times V \mid x \neq y \textit{ and } (x, y) \notin E\}$$

A graph is *complete* if every two distinct vertices are adjacent. The complete graph on n vertices is usually denoted by K_n, and is called a *clique* of size n.

A (partial) *subgraph* of a graph $G = (V, E)$ is defined to be any graph $H = (V', E')$ satisfying $V' \subseteq V$ and $E' \subseteq E$.

Given a subset $A \subseteq V$ of the vertices, we define the *subgraph induced by* A to be $G_A = (A, E_A)$, where $E_A = \{xy \in E \mid x \in A \textit{ and } y \in A\}$.

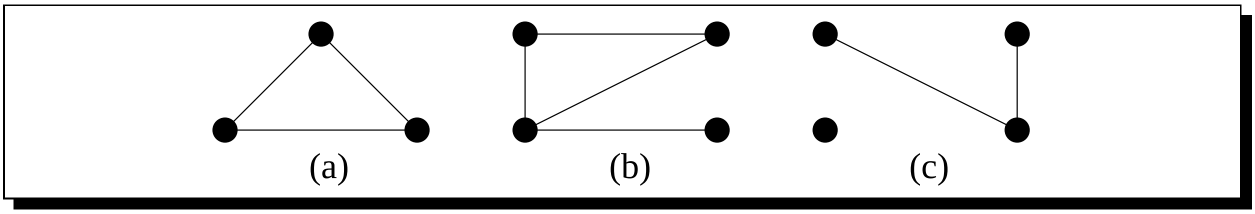


Figure 1.3: (a) The graph K_3.(b) is the complement of (c) and vice versa.

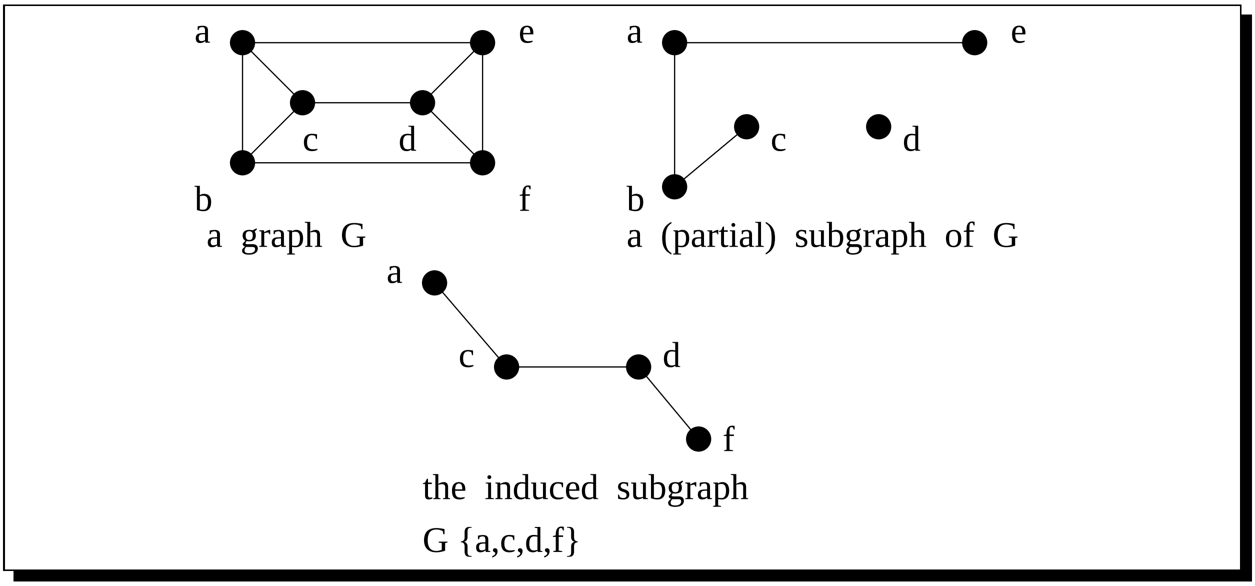


Figure 1.4: Examples of subgraphs.

Let $G = (V, E)$ be an undirected graph. Consider the following definitions :

A subset $A \subseteq V$ of r vertices is an *r-clique@r-clique* if it induces a complete subgraph, i.e., if $G_A \cong K_r$. A single vertex is a 1-clique.

A clique K in G is *maximal* if there is no clique of G which properly contains K as a subset.

A clique is *maximum* if there is no clique of G of larger cardinality.

$\omega(G)$ is the number of vertices in a maximum clique of G, and is called the *clique number* of G.

A *clique cover* of size k is a partition of the vertices $V = A_1 + A_2 + ... + A_k$ such that each A_i is a clique.

$k(G)$ is the size of a smallest possible clique cover of G. It is called the *clique cover number* of G.

A *stable set* or *independent set* is a subset X of vertices no two of which are adjacent.

$\alpha(G)$ is the number of vertices in an independent set of maximum cardinality. It is called the *stability number* of G.

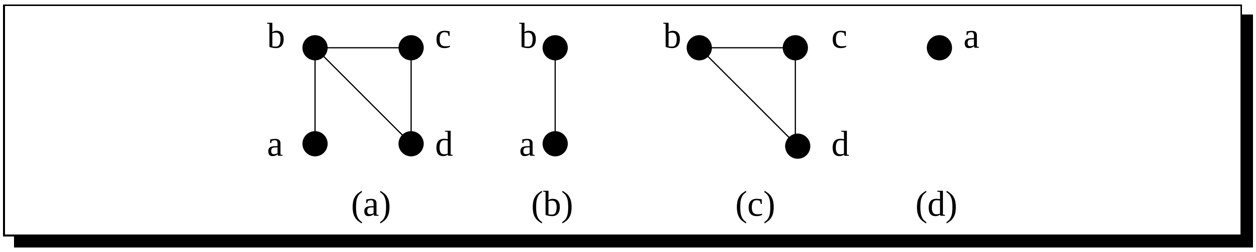


Figure 1.5: (a) A graph G.(b) A maximal clique in G.(c) A maximum clique of G.(d) A 1-clique.

A proper *ccoloring@c-coloring* is a partition of the vertices $V = X_1 + X_2 + ... + X_c$ such that each X_i is an independent set. In such a case, the vertices of X_i are "painted" with the color i and adjacent vertices will receive different colors. If G has a proper c-coloring, we say that G is *ccolorable@c-colorable.*

$\chi(G)$ is the smallest possible c for which there exist *a* proper c-coloring of G, it is called the *chromatic number* of G.

In the graph G of figure 1.5 (a) : $\{b, c, d\}$ and $\{a\}$ are a clique cover of size 2. $\{a, c\}$ is an independent set in G.

Hence, $\omega(G) = 3$; $k(G) = 2$; $\alpha(G) = 2$; $\chi(G) = 3$

It is always true that:

$$\omega(G) \leq \chi(G) \textit{ and } \alpha(G) \leq k(G)$$

Since every vertex of a maximum clique must be contained in a different color class in any minimum proper coloring, and every vertex of a maximum stable set must be contained in a different clique in any minimum clique cover.

The following relations are sometimes called *duality*:

$$\omega(G) = \alpha(\overline{G}) \text{ and } \chi(G) = k(\overline{G}).$$

Let $G = (V, E)$ be an arbitrary graph, consider the following definitions :

Chain: A sequence of vertices $[v_0, v_1, v_2, ...v_l]$ is a chain of length l in G if

$$v_{i-1}v_i \in E \text{ or } v_iv_{i-1} \in E \text{ for } i = 1...l.$$

Path: A sequence of vertices $[v_0, v_1, v_2, ...v_l]$ is a path from v_0 *to* v_l in G provided that $v_{i-1}v_i \in E$ for $i = 1..l$. A path or a chain in G is called *simple* if no vertex occurs more than once in the sequence. It is called trivial if $l = 0$. It is called *chordless* if the graph induced by $\{v_0, ...v_l\}$is a path of length l. We denote the chordless path on n vertices by P_n.

Connected graph: A graph G is connected if between any two vertices there exists a chain in G joining them.

Strongly connected graph : A graph G is strongly connected if for any two vertices x *and* y there exists a path in G from x *to* y.

Cycle : A sequence of vertices $[v_0, v_1, v_2, ...v_l, v_0]$ is called a cycle of length $l+1$ (or closed path) if $v_{i-1}v_i \in E$ *for* $i = 1...l$ *and* $v_lv_0 \in E$.

Simple cycle: A cycle $[v_0, v_1, v_2, ...v_l, v_0]$ is a simple cycle if $v_i \neq v_j$ *for* $i \neq j$.

Chordless cycle: A simple cycle $[v_0, v_1, v_2, ...v_l, v_0]$ is chordless if $v_iv_j \notin E$

for every i, j such that $\{(i - j) > 1(mod\ l)\}$. A chordless cycle on n vertices is denoted by C_n.

Bipartite graph: An undirected graph $G = (V, E)$ is bipartite if its vertices can be partitioned into two disjoint independent sets $V = S_1 + S_2$ i.e., every edge has one endpoint in S_1 and one in S_2. We usually denote such graphs by (S_1, S_2, E) to distinguish the two parts.

Complete bipartite graph: A bipartite graph $G = (S_1, S_2, E)$ is complete if for every $x \in S_1$ *and* $y \in S_2$ we have $xy \in E$. We denote by $K_{m,n}$ the complete bipartite graph (S_1, S_2, E) with $\mid S_1 \mid= m$ and $\mid S_2 \mid= n$.

Examples for using the above notation : $K_3 = C_3$ is called a *triangle* ; $\overline{C}_4 = 2K_2$; $K_{n,n} = \overline{2K_n}$; $C_5 = \overline{C}_5$.

1.1.2 Intersection Graphs

Let F be a family of nonempty sets. The *intersection graph* of F is obtained by representing each set in F by a vertex and connecting two vertices by an edge iff their corresponding sets intersect. Examples :

1. Interval graph : The intersection graph of a family of intervals on a linearly ordered set (like the real line) is called an *interval graph*. If these intervals have a unit length, then we call the graph a *unit interval graph*.

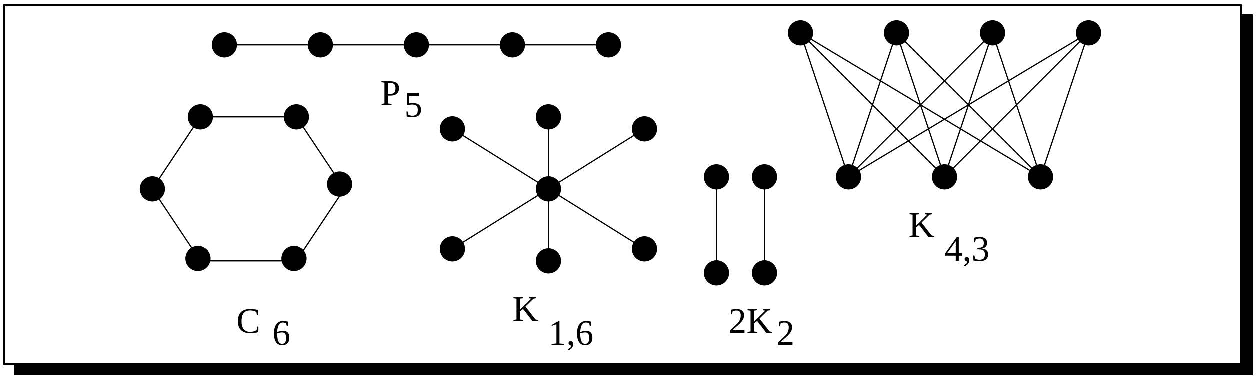


Figure 1.6: Graph examples.

2. Proper interval graph : A proper interval graph is constructed from a family of intervals on a line such that no interval properly contains another.

In 1969 Roberts showed that a graph is a unit interval graph iff it is a proper interval graph.

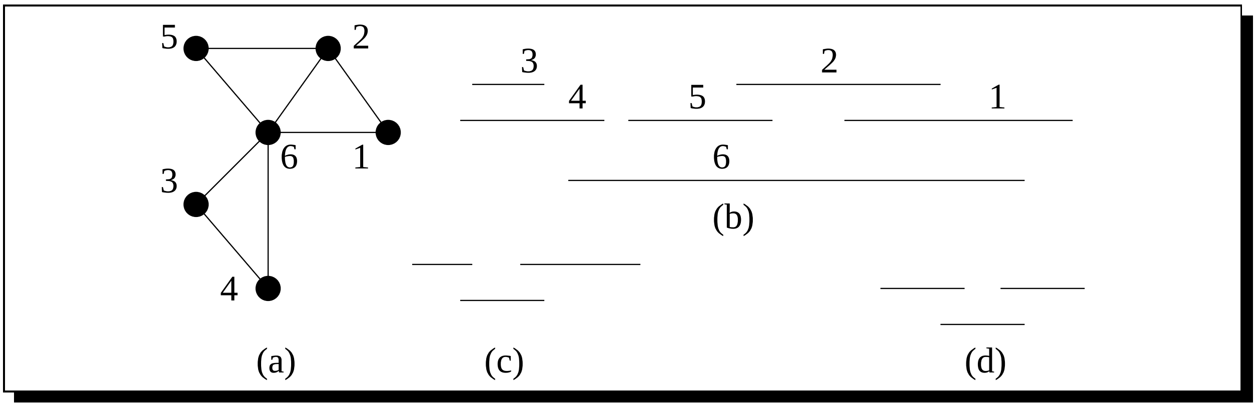


Figure 1.7: (a) A graph G.(b) An interval representation of G.(c) A proper interval representation for P_3.(d) A unit interval representation for P_3.

1.1.3 Circular-arc Graphs

Consider the following generalization of the notion of intervals on a line. If we join the two ends of our line, thus forming a circle, the intervals will become arcs on the circle. Allowing arcs to slip over and include the point of connection, we obtain a class of intersection graphs called the circular-arc graphs, which properly contains the interval graphs. In other words : a graph is a circulation arc-graph if it is the intersection graph of a family of arcs on a circle.

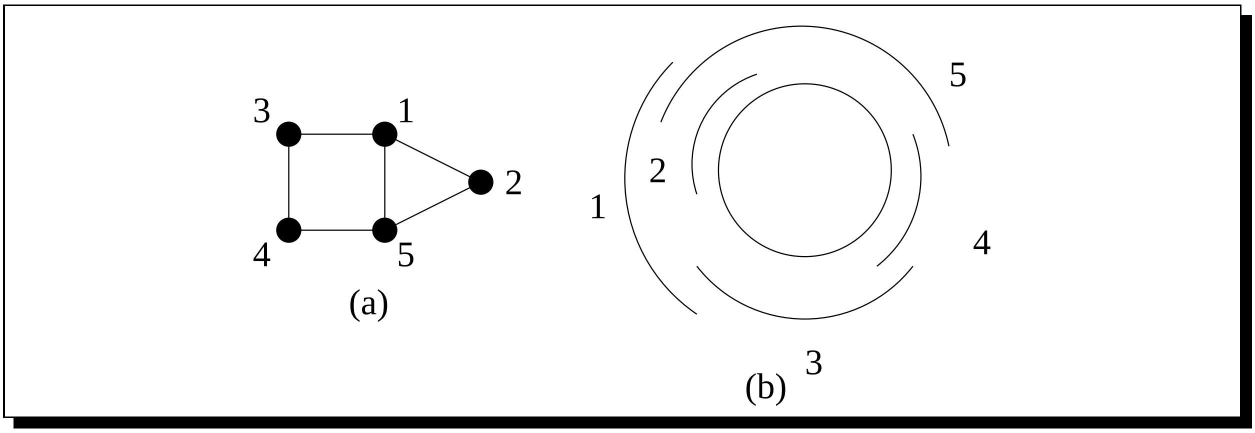


Figure 1.8: (a) A graph G.(b) A circular-arc representation of G.

Example 1.1 *The traffic flow at the corner of Holly, Wood and Vine is pictured in the next page in figure 1.9 (a). Two lanes are called compatible if traffic can flow on both of them simultaneously without collisions. Certain lanes are compatible with one another, such as c and j or d and k, while others are incompatible, such as b and f. In order to avoid collisions, we wish to install a traffic light system to control the flow of vehicles. Each lane will be assigned an arc on a circle representing the time interval during which it has a green light. Incompatible lanes must be assigned disjoint arcs. The circle may be regarded as a clock representing an entire circle which will be continually repeated.*

An arc assignment for our example is given in figure 1.9 (c). In general if G is the intersection graph of the arcs of such an assignment (see the figure 1.9 (b)), and if H is the compatibility relation defined on the pairs of lanes, then clearly G is a (partial) subgraph of H. In our example, the compatible pairs $(d, k), (h, j)$, and (i, j) are in H but are not in G.

A *proper circular arc graph* is the intersection graph of a family of arcs none of which properly contains another. In a different generalization of interval graphs Renz [1970] characterized the intersection graphs of paths in a tree. Walter [1972], Buneman [1974], and Gavril [1974] carried this idea further and showed that the intersection graphs of subtrees of a tree are exactly the triangulated graphs (the graphs with no induced C_k for $k \geq 4$).

A *permutation diagram* consists on n points on each of two parallel lines, and n straight line segments matching the points. The intersection graph of the line segments is called a *permutation graph.*

If the $2n$ points are located randomly around a circle, then the matching segments will be chords of the circle, and the resulting class of intersection graphs properly contains the permutation graphs. This class is called the class of *circle graphs.*

A simple argument shows that every proper circular-arc graph is also a circle graph. We may assume that no pair of arcs together covers the entire circle. For each arc on the circle

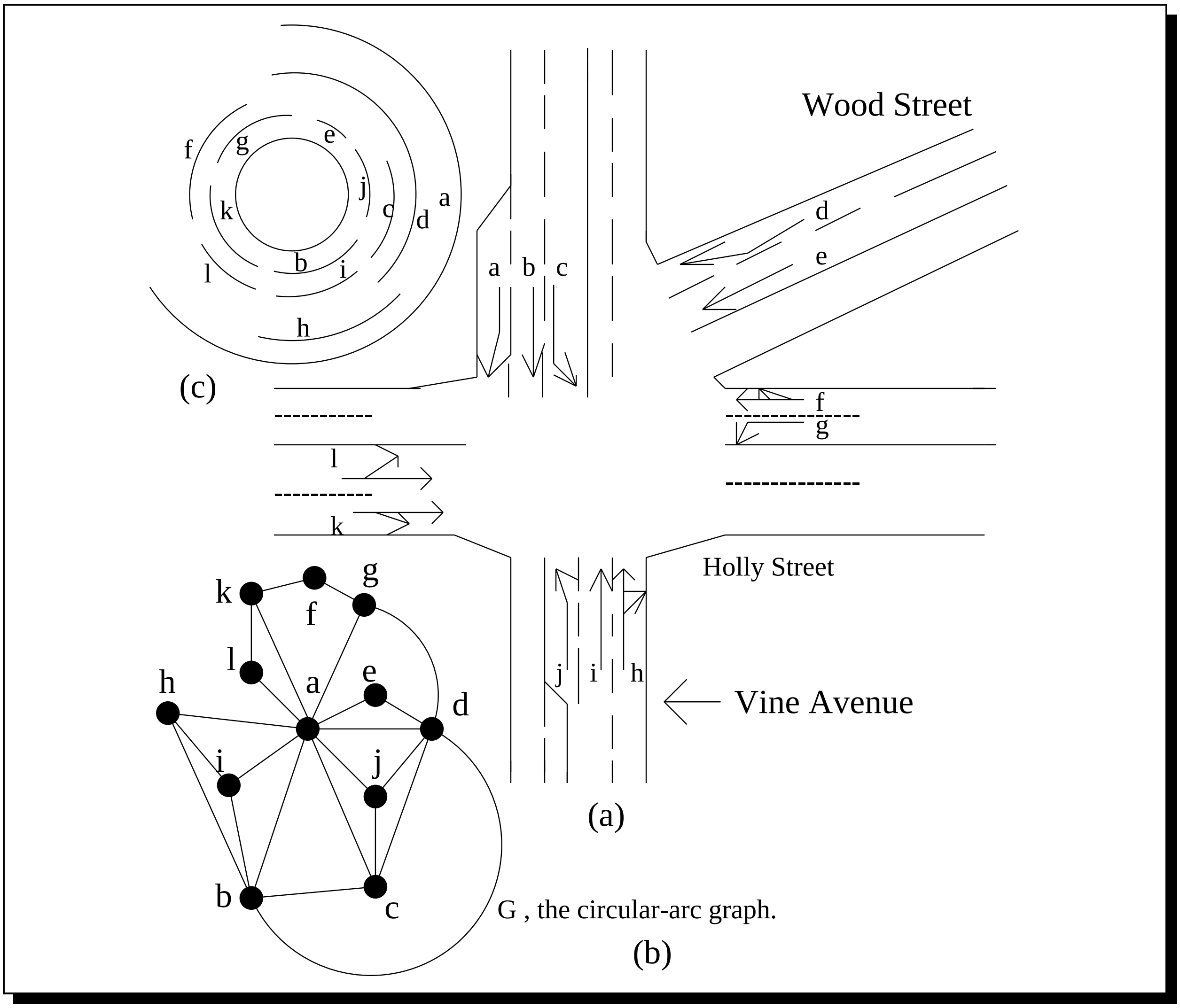


Figure 1.9: (a) The traffic flow. (b) The intersection graph of the arcs. (c) The clock cycle.

draw the chord connecting its two endpoints. Clearly two arcs overlap iff their corresponding chords intersect.

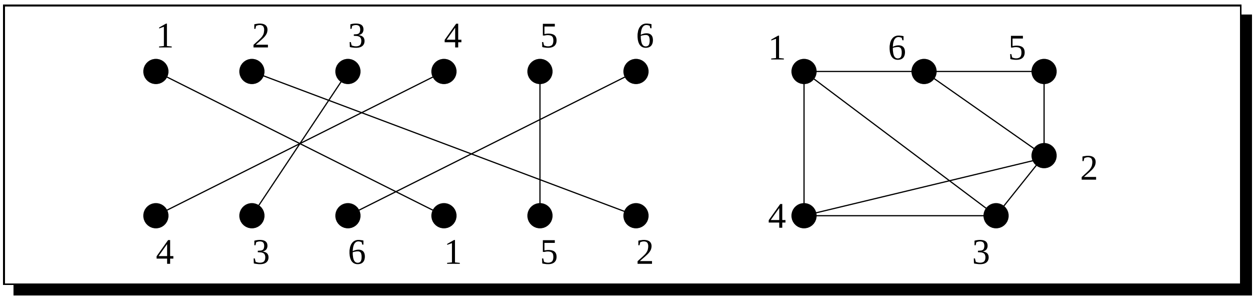


Figure 1.10: The matching diagram of [4,3,6,1,5,2], and the corresponding permutation graph.

1.1.4 Interval Graphs

An undirected graph G is called an interval graph if its vertices can be put into one-to-one correspondence with a set of intervals f of a linearly ordered set (like the real line) such that two vertices are connected by an edge of G iff their corresponding intervals have nonempty intersection. We call f an *interval representation* for G.

Application to scheduling :

Consider a collection $c = c_i$ of courses being offered by a major university. Let T_i be the time interval during which course c_i is to take place. We would like to assign courses to classrooms so that no two courses meet in the same room at the same time. This problem can be solved by properly coloring the vertices of the graph $G = (C, E)$ where :

$$c_i c_j \in E \leftrightarrow T_i \cap T_j \neq \emptyset.$$

Each color corresponds to a different classroom. The graph G is obviously an interval graph, since it is represented by time intervals. This example is especially interesting because efficient linear-time algorithms are known for coloring interval graphs with a minimum number of colors. (The minimum coloring problem is NP-complete for general graphs).

An application of interval Graph

Suppose $c_1, c_2, ..., c_n$ are chemical compounds which must be refrigerated under closely monitored conditions. If compound c_i must be kept at a constant temperature between t_i and t'_i degrees, how many refrigerators will be needed to store all the compounds ?

Let G be the interval graph with vertices $c_1, c_2, ..., c_n$ and connect two vertices whenever the temperature intervals of their corresponding compounds intersect. Intervals on a line satisfy the *Helly property* : if a set $S = \{T_i ... T_k\}$ of intervals such that, $T_i \cap T_j \neq \emptyset$ for every $i \neq j$ then $\bigcap_{t_i \in S} T_i \neq \emptyset$. Hence, if $\{c_{i1}, c_{i2}, ..., c_{ik}\}$ is a clique of G, then the intervals

$\{[t_{ij}, t'_{ij}] \mid j = 1, 2, ..., k\}$ will have a common point of intersection, say t. A refrigerator set at a temperature of t will be suitable for storing all of them. Thus, a solution to the problem will be obtained by finding a minimum clique cover of G. This is a polynomial problem on interval graph, although it is NP-hard for arbitrary graphs.

Interval graphs satisfy many interesting properties, one of them is that interval graph is an *hereditary property*: An induced subgraph of an interval graph is an interval graph.

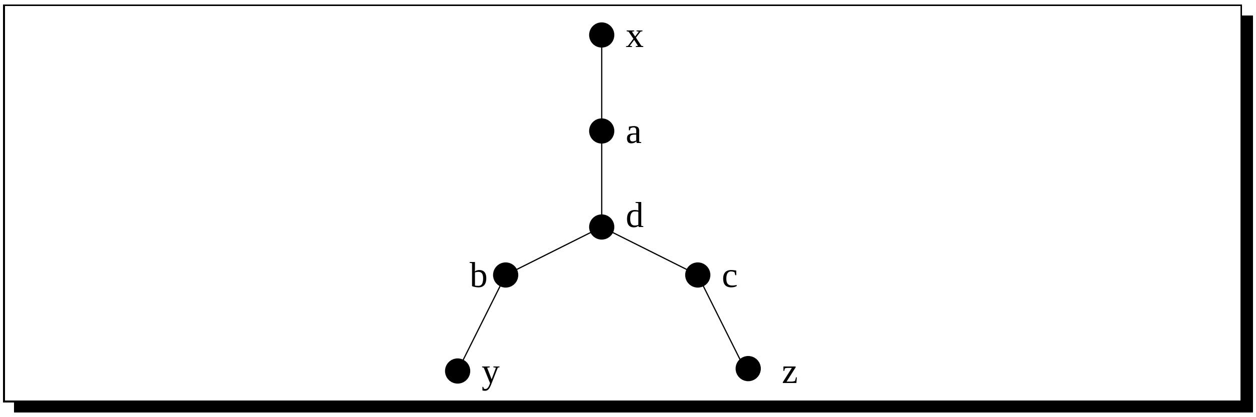


Figure 1.11: A triangulated graph which is not an interval graph.

Hereditary properties abound in graph theory, such as planarity and bipartiteness. The next property of interval graphs is also an hereditary property.

Triangulated graph property :

Every simple cycle of length strictly greater than 3 possesses a chord. Graphs which satisfy this property are called *triangulated graphs.*

Theorem 1.2 *(Hajös[1958]) An interval graph satisfies the triangulated graph property.*

Proof: Suppose the interval graph G contains a chordless cycle $[v_0, v_1, v_2, ...v_{l-1}, v_0]$ with $l > 3$. Let I_k denote the interval corresponding to V_k. For $i = 1, 2, ..., l-1$ choose a point $p_i \in (I_{i-1} \cap I_i)$. Since I_{i-1} and I_{i+1} do not overlap, the P_i constitute a strictly increasing or strictly decreasing sequence. Therefore, it is impossible for I_0 and I_{l-1} to intersect, contradicting the criterion that $V_0 V_{l-1}$ is an edge of G. ∎

Not every triangulated graph is an interval graph. Consider the tree T given in figure 1.11, which certainly has no chordless cycles. The intervals $I_a, I_b,$ *and* I_c of a representation for T would have to be disjoint, and I_d would properly include the middle interval, say I_b. Where, then could we put I_y so that it intersects I_b but not I_d ? Clearly we would be stuck.

Transitive orientation property.

Each edge can be assigned a one-way direction, in such a way that the resulting oriented graph $G = (V, E)$ satisfies the following condition :

$$(\ ab \in E \textit{ and } bc \in E\ ) \textit{imply } ac \in E \ (\forall a, b, c \in V) \quad \star$$

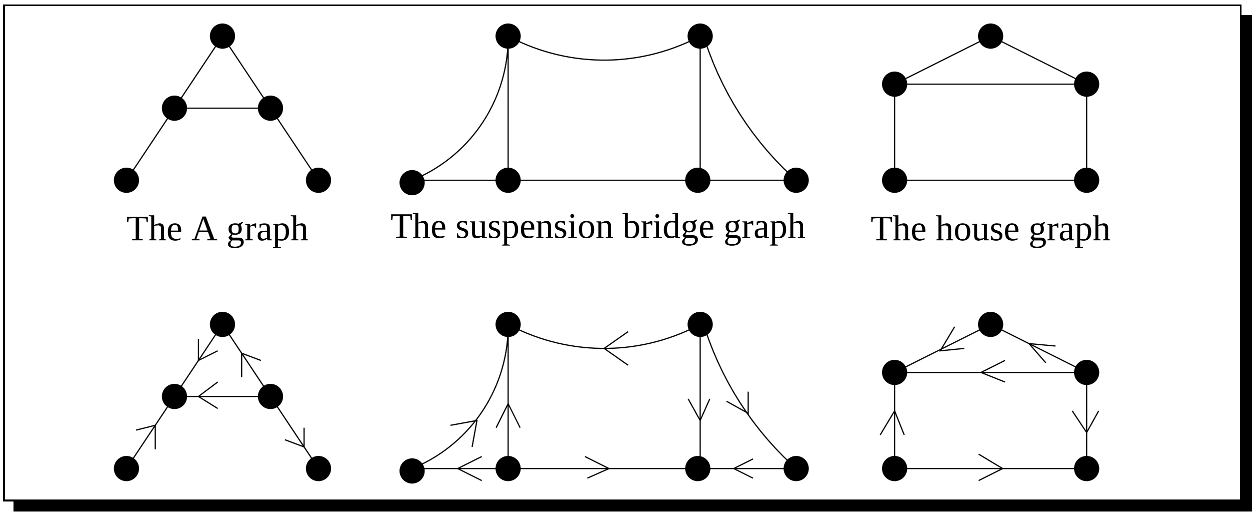


Figure 1.12: Transitive orientations of Three comparability graphs.

An undirected graph which is transitively orientable is sometimes called a *comparability graph.* Figure 1.12 shows a transitive orientation of the A graph and of the suspension bridge graph. The odd length chordless cycles C_5, C_7, C_9... and the bull's head graph (see Figure 1.13) cannot be transitive oriented.

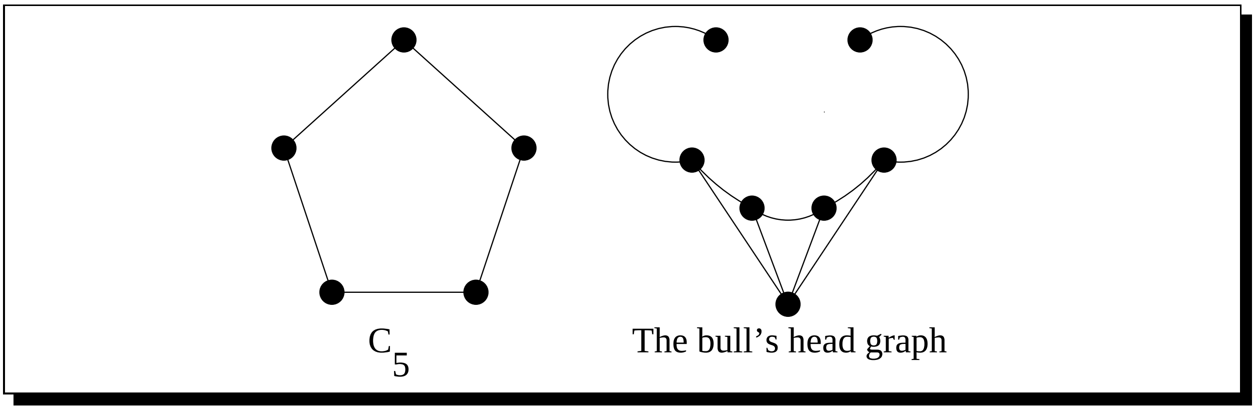


Figure 1.13: Two graphs which are not transitively orientable.

Theorem 1.3 *(Ghouila - Houri [1962]). The complement of an interval graph satisfies the transitive orientation property.*

Proof: Let $\{I_v \mid v \in V\}$ be an interval representation for $G = (V, E)$. Define an orientation F of the complement $\overline{G} = (V, \overline{E})$ as follows :

$$xy \in F \leftrightarrow I_x < I_y \quad (\forall xy \in \overline{E}).$$

Here $I_x < I_y$ means that the interval I_x lies entirely to the left of the interval I_y. (Remember, they are disjoint). Clearly $(\star)$ is satisfied, since $I_x < I_y < I_z$ implies $I_x < I_z$. Thus, F is a transitive orientation of $\overline{G}$. ■

Theorem 1.4 (Gilmore and Hoffman 1964) *An undirected graph G is an interval graph iff G is a triangulated graph and its complement $\overline{G}$ is a comparability graph.*

We will see the proof in a later lecture.

We have seen already that every interval graph satisfies the two conditions. Looking back each of the graph in Figure 1.12 can be properly colored using three colors and each contains a triangle. Therefore for these graphs, their chromatic number equals their clique number. We will see later in the course that any triangulated graph and any comparability graph also satisfies the following properties :

$\chi-$*Perfect property*

For each induced subgraph G_A of G, $\chi(G_A) = \omega(G_A)$.

The chordless cycles C_5, C_7, C_9 are not χ - perfect. A dual notion of χ - perfection is the following:

alphaperfect@$\alpha-$*Perfect property*

For each induced subgraph G_A of G, $\alpha(G_A) = k(G_A)$.

Theorem 1.5 *The perfect graph theorem (Lovász [1972]).*

A graph is $\chi-$perfect iff it is $\alpha-$perfect.

We will prove this theorem in lecture 2.

1.1.5 Line graphs of bipartite graphs

Let $G = (V, E)$. The *line graph* or the *edge graph* of G denoted $L(G)$ is the graph with $V(L(G)) = E(G)$ and

$$(e1, e2) \in E(L(G)) \leftrightarrow e1 \textit{ and } e2 \textit{ have a common endpoint in } G.$$

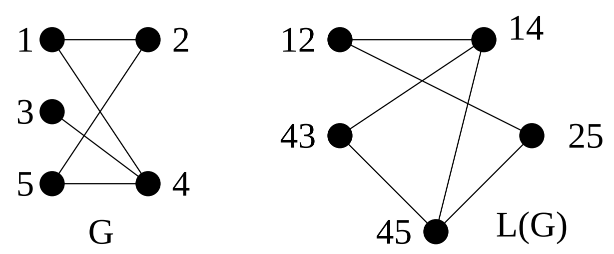


Figure 1.14: (a) A graph G.(b) The line graph of G.

Being a line graph of a bipartite graphs is an hereditary property. (Removing a vertex in $L(G)$ is the same as removing the corresponding edge in G, which remains bipartite).

An Independent set in $L(G)$ corresponds to a matching (a set of independent edges) in G. A Clique in $L(G)$ corresponds to a set of pairwise adjacent edges in G. If G is bipartite it contains no triangles, so every set of pairwise adjacent edges in G) is a star. Therefore, if G is bipartite, a clique cover of $L(G)$ corresponds to a vertex cover of G.

Let us call a graph $LGBG$ if it is the line graph of a bipartite graph. König (1931) proved that the edge chromatic number of a bipartite graph equals its maximum degree. By the above, this is equivalent to saying that if G is $LGBG$ then $\chi(G) = \omega(G)$. The König-Hall theorem (1931) states that the size of a maximal matching in a bipartite graph equals its minimum vertex cover. This is equivalent to saying that $\alpha(G) = K(G)$ if G is $LGBG$. Together with the fact that being $LGBG$ is hereditary, any of these two statements imply that $LGBG'S$ are perfect. From the perfect graph theorem the two statements are equivalent.

Chapter 2

Lecture 2

Advanced Topics in Graph Algorithms

Spring Semester, 1994

Lecturer : Ron Shamir
Scribe : Erez Hartuv
Lecture 2 : April 28

2.1 Perfect Graphs

In this lecture the graphs we consider are undirected. When we refer to a graph G we mean the Graph $G(V, E)$, with vertex set V and edge set E.

2.1.1 Definition of perfect graph

The following three conditions define perfection properties of a graph $G(V, E)$.

- (P1) $\omega(G_A) = \chi(G_A), \ \forall A \subseteq V$
- (P2) $\alpha(G_A) = k(G_A), \ \forall A \subseteq V$
- (P3) $\omega(G_A)\alpha(G_A) \geq |A|, \ \forall A \subseteq V$

By duality, a graph G satisfies (P1) iff its complement $\overline{G}$ satisfies (P2).

In this lecture we'll prove the *perfect graph theorem* which says that (P1),(P2) and (P3) are equivalent. A graph which satisfies one of the perfection properties satisfies all (by the theorem) and is called *perfect graph*.

Remark: The equivalence holds for finite graphs only.

Lemma 2.1 *Let the graph $G(V,E)$ satisfy: $\alpha(G) = k(G)$. If $\overline{K}$ is a minimum clique cover, and S is a maximum independent set, then each clique in $\overline{K}$ has exactly one vertex in common with S.*

Proof: By definition, no clique can intersect an independent set in more than one element. Therefore the intersection of S and each clique in $\overline{K}$ may be empty, or one element. Remember that each vertex is contained in some clique. If there is a clique whose intersection with S is empty then, $|\overline{K}| > |S|$, *contradiction* to the assumption of equality. ■

Lemma 2.2 *Let G be an induced subgraph of H then:*

1. $\omega(H) \geq \omega(G)$
2. $\chi(H) \geq \chi(G)$
3. $\alpha(H) \geq \alpha(G)$
4. $k(H) \geq k(G)$

Lemma 2.3 *If $D \subseteq V$ intersects every maximum independent set of G in exactly one vertex then: $\alpha(G_{V-D}) = \alpha(G) - 1$*

Proof: Denote $\alpha = \alpha(G)$. Let U be a maximum independent set of G. By the assumption $U \cap D = \{i\}$. Therefore $U - \{i\}$ is an independent set of G_{V-D}. Therefore $\alpha(G_{V-D}) \geq \alpha - 1$. If $\alpha(G_{V-D}) > \alpha - 1$ then $\alpha(G_{V-D}) = \alpha$ (because of the monotonicity shown in Lemma 2.2). It follows that G_{V-D} contains a maximum independent set of size α, therefore G contains this maximum independent set of size α, which does not intersect D, *contradiction.* ■

2.1.2 Some Definitions and Properties

Given two graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$, we define their *union* and *join.*

Definition *Union*: $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$

Definition *Join*: $G_1 + G_2 = (V_1 \cup V_2, E = E_1 \cup E_2 \cup \{uv | u \in V_1, v \in V_2\})$

Lemma 2.4 *if G_1, G_2 are perfect, then their union and join are perfect.*

Proof: Denote the parameters of each graph G_i as: $\chi_i,\ \omega_i,\ \alpha_i,\ k_i \quad i = 1,2$
The parameters value for $G_1 \cup G_2$ are:

$$\omega = \max(\omega_1, \omega_2) \qquad \chi = \max(\chi_1, \chi_2)$$

$$k = k_1 + k_2 \qquad \alpha = \alpha_1 + \alpha_2$$

The parameters value for $G_1 + G_2$ are:

$$\omega = \omega_1 + \omega_2 \qquad \chi = \chi_1 + \chi_2$$

$$k = \max(k_1, k_2) \qquad \alpha = \max(\alpha_1, \alpha_2)$$

These relations together with the perfection of $G1$, $G2$ imply the claim. ■

Definition $G \circ x$: Let $G(V, E)$ be an undirected graph with vertex $x \in V$. The graph $G \circ x$ is obtained from G by adding a new vertex x' which is connected to all the neighbors of x: $G \circ x = (V \cup \{x'\}, E' = E \cup \{x'u | u \in Adj(x)\})$.

Lemma 2.5 $\forall x \neq y \quad (G \circ x) - y = (G - y) \circ x$

Proof: Let x' denote the new vertex (the copy of x). Both graphs has the same vertex set: $V + x' - y$. In both graphs, uv is an edge iff one of the following exist:

1. $uv \in E, u \neq y, v \neq y$

2. $v = x', ux \in E, u \neq y$

■

Definition More generally, if $x_1, \ldots, x_n$ are the vertices of G and $h = (h_1, \ldots, h_n)$ is a vector of non-negative integers, then $H = G \circ h$ is constructed by substituting for each x_i a stable set of h_i vertices $x_i^1, \ldots, x_i^{h_i}$ and joining x_i^s with x_j^t iff x_i and x_j are adjacent in G. We say that H is obtained from G by *multiplication of vertices*.

Remark. The definition allows $h_i = 0$, in which case H includes no copy of x_i. Thus, every induced subgraph of G can be obtained by multiplication of G by appropriate (0,1)-valued vector. For example $G = G \circ (1, \ldots, 1)$.

2.1.3 Perfect Graph Theorem

We now reach the main part of the lecture, in which we prove two lemmas that will help to prove the perfect graph theorem.

Lemma 2.6 *(Berge[1961]). Let H be obtained from G by multiplication of vertices.*

1. *If G satisfies (P1), then H satisfies (P1).*

2. *If G satisfies (P2), then H satisfies (P2).*

Proof: The proof is by induction on the number of vertices in G. The lemma is true if G has only one vertex. We shall assume that (1) and (2) are true for all graphs with fewer vertices than G. Let $H = G \circ h$. If one of the coordinates of h equals zero, say $h_i = 0$, then H can be obtained from $G - x_i$ by multiplication of vertices.

But, if G satisfies (P1) [respectively (P2)], then $G - x_i$ also satisfies (P1) [respectively (P2)] ($G - x_i$ is an induced subgraph of G, therefore (P1) on G implies (P1) on all induced subgraphs of G). Then by the induction hypothesis H satisfies (P1) [respectively (P2)].

Thus, we may assume that each coordinate $h_i \geq 1$, and since H can be built up from a sequence of single-copy multiplications, it is sufficient to prove the result for $H = G \circ x$. Let x' denote the added "copy" of x.

Assume that G satisfies (P1). Since x and x' are nonadjacent, $\omega(G \circ x) = \omega(G)$. Let G be colored using $\omega(G)$ colors. Color x' the same color as x. This will be a coloring of $G \circ x$ in $\omega(G \circ x)$ colors. Hence, $G \circ x$ satisfies (1).

Next assume that G satisfies (P2). We must show that $\alpha(G \circ x) = k(G \circ x)$. Let $\overline{K}$ be a (minimum) clique cover of G with $|\overline{K}| = k(G) = \alpha(G)$, and let K_x be the clique in $\overline{K}$ containing x. There are two cases.

Case 1: x is contained in a maximum stable set S of G, i.e., $|S| = \alpha(G)$. In this case $S \cup \{x'\}$ is a stable set of $G \circ x$, so

$$\alpha(G \circ x) = \alpha(G) + 1. \tag{2.1}$$

Since $\overline{K} \cup \{\{x'\}\}$ covers $G \circ x$, we have that

$$k(G \circ x) \overset{\overline{K}\cup\{\{x'\}\}\ covers\ G\circ x}{\leq} k(G) + 1 \overset{(P2)}{=} \alpha(G) + 1 \overset{equation\ 2.1}{=} \alpha(G \circ x) \overset{always}{\leq} k(G \circ x).$$

Thus, $\alpha(G \circ x) = k(G \circ x)$.

Cases 2: No maximum stable set of G contains x. In this case,

$$\alpha(G \circ x) = \alpha(G). \tag{2.2}$$

(G is an induced subgraph of $G \circ x$, therefore $\alpha(G \circ x) \geq \alpha(G)$. If $\alpha(G \circ x) > \alpha(G)$ it means that we switched one vertex from a maximum stable set in G with x, x', to obtain a maximum stable set in $G \circ x$ with cardinality greater by one. But that means x must be an element in a maximum stable set in G, contradicting case 2 assumption).

Since each clique of $\overline{K}$ intersects a maximum stable set exactly once (Lemma 2.1), this is true in particular for K_x. But x is not a member of any maximum stable set. Therefore, $D = K_x - \{x\}$ intersects each maximum stable set of G exactly once, so by Lemma 2.3

$$\alpha(G_{V-D}) = \alpha(G) - 1. \tag{2.3}$$

This implies that

$$k(G_{V-D}) \overset{induction}{=} \alpha(G_{V-D}) \overset{equation\ 2.3}{=} \alpha(G) - 1 \overset{equation\ 2.2}{=} \alpha(G \circ x) - 1.$$

Taking a clique cover of G_{V-D} of cardinality $\alpha(G \circ x) - 1$ along with the extra clique $D \cup \{x'\}$, we obtain a cover of $G \circ x$ of cardinality $\alpha(G \circ x)$. Therefore we have a clique cover with the lowest possible number of cliques, because always $k \geq \alpha$, so

$$k(G \circ x) = \alpha(G \circ x).$$

■

Lemma 2.7 *(Fulkerson[1971], Lovász[1972]). Let G be an undirected graph each of whose proper induced subgraphs satisfies (P2), and let H be obtained from G by multiplication of vertices. If G satisfies (P3), then H satisfies (P3).*

Proof: Let G satisfy (P3) and choose H to be a graph having the smallest possible number of vertices which can be obtained from G by multiplication of vertices but which fails to satisfy (P3) itself. Thus,

$$\omega(H)\alpha(H) < |X|, \tag{2.4}$$

where X denotes the vertex set of H, yet (P3) does hold for each proper induced subgraph of H.

As in the proof of the preceding lemma, we may assume that each vertex of G was multiplied by at least 1 and that some vertex u was multiplied by $h \geq 2$. Let $U = \{u^1, \dots u^h\}$ be the vertices of H corresponding to u. The vertex u^1 plays a distinguished role in the proof. By the minimality of H, (P3) is satisfied by H_{X-u^1}, which gives

$$\begin{aligned} |X| - 1 = |X - u^1| &\overset{(P3)}{\leq} \omega(H_{X-u^1})\alpha(H_{X-u^1}) \\ &\leq \omega(H)\alpha(H) \\ &\overset{equation\ 2.4}{\leq} |X| - 1 \end{aligned}$$

Thus, equality holds throughout, and we can define

$$p = \omega(H_{X-u^1}) = \omega(H),$$

$$q = \alpha(H_{X-u^1}) = \alpha(H),$$

and

$$pq = |X| - 1. \tag{2.5}$$

Since H_{X-U} is obtained from $G-u$ by multiplication of vertices, and using the assumption that every proper induced subgraph of G satisfies (P2), Lemma 2.6 (Berge) implies that H_{X-U} satisfies (P2). Thus, $k(H_{X-U}) = \alpha(H_{X-U}) \leq \alpha(H_{X-u^1}) = q$. Therefore, H_{X-U} can be covered by a set of q complete subgraphs of H, say $K_1, \ldots, K_q$ (this is not necessarily a minimum cover). Without lost of generality we may assume that these q cliques are pairwise disjoint and that $|K_1| \geq |K_2| \geq \ldots \geq |K_q|$. Obviously,

$$\sum_{i=1}^{q} |K_i| = |X - U| = |X| - h \overset{equation\ \ 2.5}{=} pq - (h-1)$$

Since $|K_i| \leq p$, at most $h-1$ of the cliques fail to contribute p to the sum. Hence,

$$|K_1| = |K_2| = \ldots = |K_{q-h+1}| = p.$$

Let H' be the subgraph of H induced by $X' = K_1 \cup \ldots \cup K_{q-h+1} \cup \{u^1\}$. Thus

$$|X'| = p(q-h+1) + 1 \overset{h\geq 2}{<} pq + 1 \overset{equation\ \ 2.5}{=} |X|. \tag{2.6}$$

Therefore H' is a proper induced subgraph of H. So by the minimality of H, H' satisfies (P3):

$$\omega(H')\alpha(H') \geq |X'|. \tag{2.7}$$

But $p = \omega(H) \geq \omega(H')$, so

$$\alpha(H') \overset{equation\ \ 2.7}{\geq} |X'|/p \overset{equation\ \ 2.6}{>} q - h + 1$$

Let S' be a stable set of H' of cardinality $q - h + 2$. Certainly $u^1 \in S'$, for otherwise S' would contain two vertices of a clique (by the definition of H'). Therefore, $S = S' \cup U = S' + (U - \{u'\})$ is a stable set of H with $(q-h+2) + (h-1) = q+1$ vertices, contradicting the definition of q. ■

Theorem 2.8 *The Perfect Graph Theorem (Lovász [1972]). For an undirected graph $G = (V, E)$, the following statements are equivalent:*

- *(P1)* $\omega(G_A) = \chi(G_A), \ \forall A \subseteq V$
- *(P2)* $\alpha(G_A) = k(G_A), \ \forall A \subseteq V$
- *(P3)* $\omega(G_A)\alpha(G_A) \geq |A|, \ \forall A \subseteq V$

Proof: The three statements hold for a one vertex graph. We may assume that the theorem is true for all graphs with fewer vertices than G.

(P1) $\Rightarrow$ (P3). Suppose we can color G_A in $\omega(G_A)$ colors. Every optimal coloring (a partition to stable sets) $C_1, \ldots, C_m$ satisfies:

$$|A| = \sum_{i=1}^{m} |C_i| \leq m \max_i |C_i| \overset{at\ most\ \alpha(G_A)\ vertices\ of\ a\ given\ color}{\leq} \chi(G_A)\alpha(G_A) \overset{(P1)}{=} \omega(G_A)\alpha(G_A)$$

Therefore,

$$|A| \leq \omega(G_A)\alpha(G_A)$$

(P3) $\Rightarrow$ (P1). Let $G = (V, E)$ satisfy (P3). Then each proper induced subgraph of G satisfies (P3), and by induction satisfies (P1) (and also (P2)). It is sufficient to show that $\omega(G) = \chi(G)$.

If we had a stable set S of G such that $\omega(G_{V-S}) < \omega(G)$, we could then paint S in one color and paint G_{V-S} in $\omega(G) - 1$ other colors, and we would have $\omega(G) = \chi(G)$.

Now let's assume that such S does not exist. It means that for every stable set S of G G_{V-S} has a clique $K(S)$ of size $\omega(G)$. Let $\mathcal{I}$ be the collection of all stable sets of G. Keep in mind that $\forall S \in \mathcal{I} \;\; \exists\, \omega-clique\; K(S)$ in G_{V-S} and in particular $S \cap K(S) = \emptyset$.

Construct a (0,1)-matrix M with rows corresponding to vertices, and a column i corresponding to each independent set $S_1, \ldots, S_{|\mathcal{I}|}$. $M_{ij} = 1$ iff vertex i belongs to $K(S_j)$. The matrix is illustrated in figure 2.1.

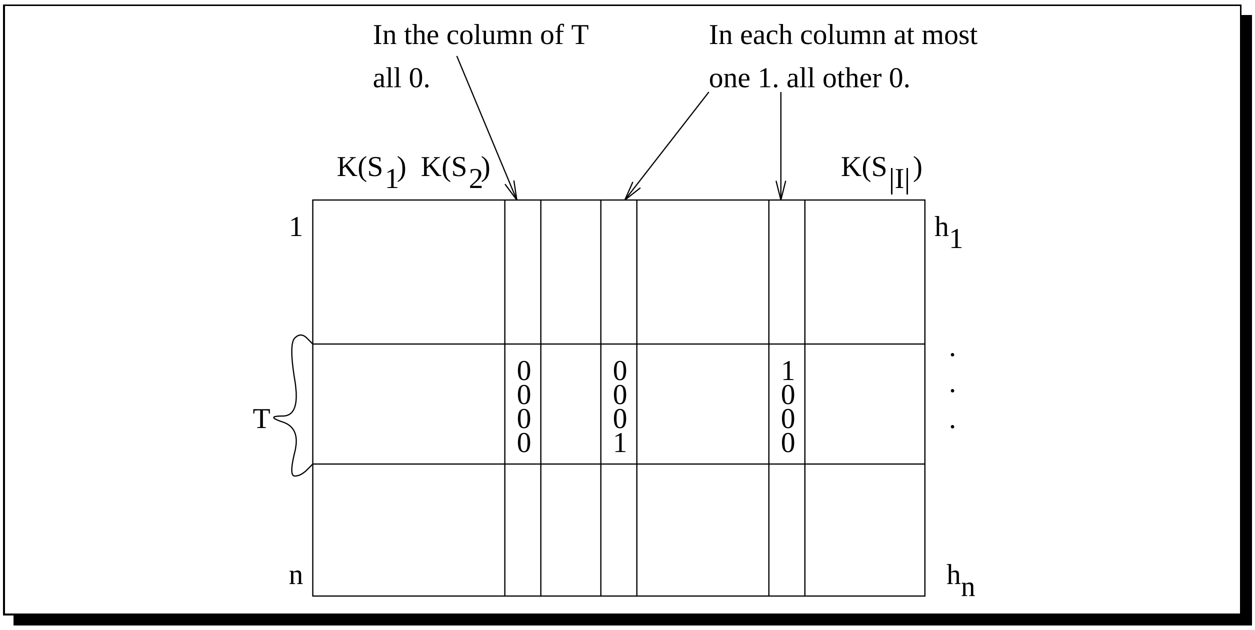


Figure 2.1: The matrix M: each row correspond to a vertex in V, and each column to an $\omega-clique\; K(S)$

For each $x_i \in V$, let h_i denote the number of cliques $K(S)$ which contain x_i(=the number of ones in the i'th row). Let $H = (X, F)$ be obtained from G by multiplying each x_i by h_i [if $h = (h_1, \ldots, h_n)$, then $H = G \circ h$]. On the one hand, by Lemma 2.7,

$$\omega(H)\alpha(H) \geq |X|. \tag{2.8}$$

On the other hand,

$$|X| = \sum_{x_i \in V} h_i \overset{\substack{\text{summing the matrix M} \\ \text{over rows and columns.}}}{=} \sum_{S \in \mathcal{I}} |K(S)| \overset{\substack{\text{every clique in } \mathcal{I} \\ \text{is of size } \omega.}}{=} \omega(G)|\mathcal{I}|. \tag{2.9}$$

Since every clique in $G \circ x$ contains at most one "copy" of vertex $x \in G$, we have $\omega(G \circ x) \leq \omega(G)$. Therefore,

$$\omega(H) \leq \omega(G). \tag{2.10}$$

If a maximum stable set contain some "copy" of x_i then it will contain *all* its "copies" (i.e. all h_i "copies"). To find $\alpha(H)$, we sum up the h_i's of the vertices which induce a stable set T (in G) , and do so to all the stable sets of G which are collected in $\mathcal{I}$, searching the one with a maximum sum (note that all maximum stable sets in H are obtained in this way). Hence,

$$\alpha(H) = \max_{T \in \mathcal{I}} \sum_{x_i \in T} h_i$$

(Note that the maximum stable set in H does not necessarily correspond to the maximum stable set in G. The multiplicities may play a role too).

For each T, the sum above can be computed in two ways (figure 2.1):

1. Move row by row, but only for rows of vertices in T, and sum up the ones in each row($= h_i$).

2. Move column by column, and sum up the the ones in each column which correspond to vertices in T.

Hence,

$$\max_{T \in \mathcal{I}} \sum_{x_i \in T} h_i = \max_{T \in \mathcal{I}} [\sum_{S \in \mathcal{I}} |T \cap K(S)|].$$

T is a stable set and $K(S)$ is a clique, so

$$T \cap K(S) \leq 1.$$

$K(T)$ is a clique in G_{V-T}, so

$$T \cap K(T) = 0.$$

In the second way of summing, at least one term will be zero, and all terms are at most one (compare figure 2.1). So we conclude:

$$\alpha(H) = \ldots = \max_{T \in \mathcal{I}} [\sum_{S \in \mathcal{I}} |T \cap K(S)|] \leq |\mathcal{I}| - 1. \tag{2.11}$$

Let us summarize what we have shown:

$$|X| = \omega(G)|\mathcal{I}| \qquad \textit{equation } 2.9$$

$$\alpha(H) \leq |\mathcal{I}| - 1. \qquad \textit{equation } 2.11$$

$$\omega(H) \leq \omega(G). \qquad \textit{equation } 2.10$$

Then,

$$\alpha(H)\omega(H) \leq \omega(G)[|\mathcal{I}| - 1] = |X|\frac{|\mathcal{I}| - 1}{|\mathcal{I}|} < |X|$$

contradicting equation 2.8.

(P2) $\Leftrightarrow$ (P3). G satisfies (P2) $\overset{duality}{\Leftrightarrow}$ $\overline{G}$ satisfies (P1) $\overset{proved\ above}{\Leftrightarrow}$ $\overline{G}$ satisfies (P3) $\overset{duality}{\Leftrightarrow}$ G satisfies (P3). ■

Corollary 2.9 *A graph G is perfect iff its complement $\overline{G}$ is perfect.*

Proof: By duality, G satisfies (P2) $\Leftrightarrow$ $\overline{G}$ satisfies (P1) . By the Perfect Graph Theorem each of these conditions is sufficient and necessary for perfection. ■

Corollary 2.10 *A graph G is perfect iff every graph H obtained from G by multiplication of vertices is perfect.*

Proof: One direction is trivial. The other direction follows immediately from Lemma 2.6 and the Perfect Graph Theorem. ■

The last two corollaries are useful in checking perfection: we can choose to check the complement, and we can reduce non-adjacent vertices with identical neighborhoods and check on the resulting simpler graph.

Chapter 3

Lecture 3

Advanced Topics in Graph Algorithms		
©Tel-Aviv University		
Spring Semester, 1994		
Lecturer	:	Ron Shamir
Scribe	:	Dekel Tsur
Lecture 3	:	May 5

3.1 Perfect Graphs

3.1.1 p-Cririical Graphs

Definition An undirected graph G is called *p-critical* if G is not perfect, but every proper induced subgraph of G is perfect.

Note: If G is p-critical then for all vertices x of G

$$\alpha(G-x) = k(G-x),\ w(G-x) = \chi(G-x)$$

Theorem 3.1 *If G is a p-critical graph on n vertices, then:*

1. $\alpha(G)w(G) = n-1$
2. *for all vertices x of G, $\alpha(G) = k(G-x)$ and $w(G) = \chi(G-x)$*

Proof: By the Perfect Graph theorem, since G is p-critical we have $\alpha(G)w(G) < n$ and $\alpha(G-x)w(G-x) \geq n-1$ for all vertices x. Since α and w are monotone,

$$n > \alpha(G)w(G) \geq \alpha(G-x)w(G-x) \geq n-1$$

Hence, $\alpha(G)w(G) = n-1$, $\alpha(G) = \alpha(G-x) = k(G-x)$ and $w(G) = w(G-x) = \chi(G-x)$ ∎

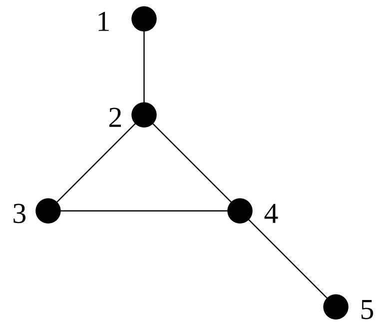
3.1.2 A Polyhedral Characterization of Perfect Graphs

Let A be an $m \times n$ matrix. We define the two polyhedra

$$P(A) = \{\mathbf{x} | A\mathbf{x} \leq \mathbf{1},\ \mathbf{x} \geq \mathbf{0}\}$$

$$P_I(A) = convex - hull(\{\mathbf{x} | \mathbf{x} \in P(A),\ \mathbf{x}\, integral\})$$

where $\mathbf{x}$ is an n-vector, $\mathbf{1}$ is the m-vector of all ones, and $\mathbf{0}$ is the n-vector of all zeros. Clearly $P_I(A) \subseteq P(A)$, and for $(0,1)$ matrix A without zero columns, $P(A)$ and $P_I(A)$ are bounded in the unit hypercube in R^n. The *clique matrix* of a graph $G = (V, E)$ is a $(0,1)$ matrix where if $V = \{1, 2, \ldots n\}$ and $C_1, C_2, \ldots C_m$ are all the maximal cliques of G numbered arbitrarily, then $A_{ij} = 1$ iff $j \in C_i$. See Figure 3.1 for an example.



$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figure 3.1: A graph and its clique matrix

Theorem 3.2 (Chvátal, 75) *Let A be the clique matrix of an undirected graph G. Then G is perfect iff $P(A) = P_I(A)$.*

Polyhedron P is called *integral* if all the coordinates of all its extreme points are integers, i.e. if $P(A) = P_I(A)$. In other words, theorem 3.2 is: G is perfect iff $P(A)$ is integral polyhedron. In particular, this implies that optimization problems which can be posed as *integer* programming problems with constraints $P_I(A)$, can be solved in polynomial time on perfect graphs, by solving the corresponding *linear* program.

3.1.3 The Strong Perfect Graph Conjecture

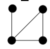
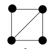
It is easy to see that the odd cycle C_{2k+1} (for $k \geq 2$) is a p-critical graph. By the Perfect Graph theorem its complement $\overline{C_{2k+1}}$ is also p-critical. C_{2k+1} and $\overline{C_{2k+1}}$ (called *odd hole* and *odd anti-hole* respectively) are the only known p-critical graphs. In 1960 C. Berge conjectured that there are no other p-critical graphs. This conjecture is called the *strong perfect graph conjecture* (SPGC).

The strong perfect graph conjecture can be rephrased in several equivalent forms:

(1) G is perfect iff it does not contain induced subgraph isomorphic to an odd hole or an odd anti-hole.

(2) G is perfect iff every cycle of length ≥ 5 in G or $\overline{G}$ contains a chord.

(3) The only p-critical graphs are C_{2k+1} and $\overline{C_{2k+1}}$ (for $k \geq 2$).

The strong perfect graph conjecture remains an open question. However, the conjecture was proved on several classes of graphs like planar graphs, circular arc graphs, graphs with $w \leq 3$, graphs with maximal degree ≤ 6, and also $K_{1,3}$ - free, K_4 - free,  -free, and  -free graphs. (a graph is *X-free* if it does not contains induced subgraph isomorphic to X).

3.2 Triangulated Graphs

3.2.1 Introduction

Definition An undirected graph G is called *triangulated* if every cycle of length > 3 contains a chord. In other words G is triangulated if it does not contain an induced subgraph isomorphic to C_n for $n > 3$.

Being triangulated is a hereditary property, meaning that every induced subgraph of a triangulated graph is also triangulated. Triangulated graphs have also been called *chordal*, *rigid-circuit*, *monotone transitive* and *perfect elimination* graphs. For an example of triangulated graph see Figure 3.2.

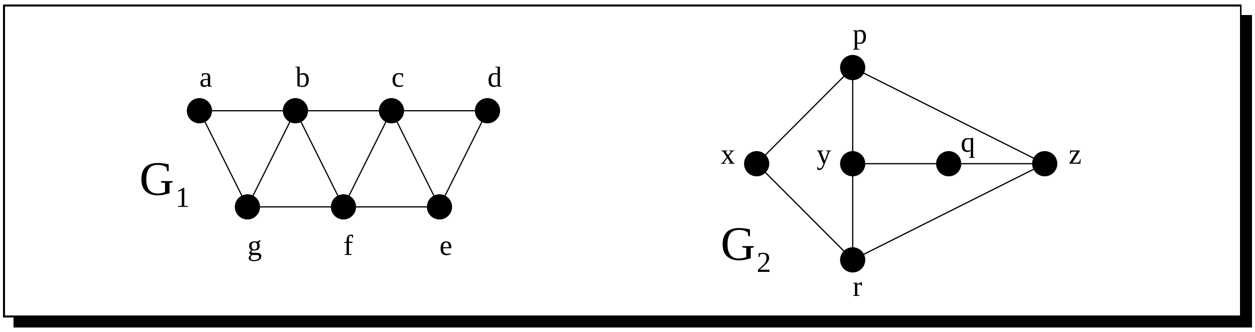


Figure 3.2: Two graphs, the left one is triangulated, and the right one is not.

3.2.2 Characterizing Triangulated Graphs

Definition A vertex x in G is called *simplicial* if $Adj(x)$ induces complete graph in G (i.e.,

$Adj(x)$ is a clique).

We will prove that every triangulated graph has a simplicial vertex. Since being triangulated is a hereditary property, it gives us an iterative procedure for identifying triangulated graphs: Search for simplicial vertex, remove it from the graph and repeat until there is no simplicial vertex (in that case, the graph is not triangulated) or there are no vertices left (and then the graph is triangulated).

Definition Let G be an undirected graph and let $\sigma = [v_1, \ldots, v_n]$ be an ordering of the vertices. σ is called *perfect elimination order* (PEO) if every v_i is a simplicial vertex in the induced subgraph $G_{\{v_i,\ldots,v_n\}}$. In other words, each set $X_i = \{v_j \in Adj(v_i) | j > i\}$ is complete.

In Figure 3.2, $\sigma = [a, g, b, f, c, e, d]$ is one of the perfect elimination orders of G_1. In contrast to this, G_2 has no simplicial vertex, so obviously it has no perfect elimination order.

Definition A subset $S \subset V$ is called a *vertex separator* for nonadjacent vertices a, b (or *a-b separator*), if in G_{V-S} a and b are in different connected components. If no proper subset of S is an a-b separator, S is called *minimal vertex separator.*

The example will be again on the graphs in Figure 3.2. In G_2, the set $\{y, z\}$ is a minimal vertex separator for p and q, whereas $\{x, y, z\}$ is a minimal vertex separator for p and r. In G_1 all the minimal vertex separators have 2 vertices which are neighbors.

Theorem 3.3 *(Dirac 61, Fulkerson and Gross 65)*
For undirected graph G the following statements are equivalent:

(1) G is triangulated.

(2) G has a perfect elimination order. Moreover, any *simplicial vertex can start a perfect order.*

(3) Every minimal vertex separator induces a complete subgraph of G.

Proof:

(3) $\rightarrow$ (2) Let $[a, x, b, y_1, \ldots, y_k, a]$ $(k \geq 1)$ be a simple cycle. Any minimal a-b separator must contain vertices x and y_i for some i, but from (3) it follows that $xy_i \in E$, which is a chord of the cycle.

(1) $\rightarrow$ (3) Let S be an a-b separator. We will denote G_A, G_B the connected components of G_{V-S} containing a and b respectively. Since S is minimal, every vertex $x \in S$ is a neighbor of a vertex in A and a vertex in B. For any $x, y \in S$ there exist minimal paths $[x, a_1, \ldots, a_r, y] (a_i \in A)$ and $[x, b_1, \ldots, b_k, y] (b_i \in B)$. Since $[x, a_1, \ldots, a_r, y, b_1, \ldots, b_k, x]$

is a simple cycle of length ≥ 4, it contains a chord. For every i,j $a_ib_j \notin E$ (S is a-b separator) and also $a_ia_j \notin E, b_ib_j \notin E$ (by the minimality of the paths) Necessarily $xy \in E$ is the chord.

■

To prove the rest of the theorem, we will first prove the following lemma:

Lemma 3.4 *(Dirac 61) Every triangulated graph G has a simplicial vertex. If G is not complete then it has two nonadjacent simplicial vertices.*

Proof: If G is complete the lemma is trivial. We will prove the lemma for noncomplete graphs by induction on the number of vertices in G. The lemma is trivial for graphs with 1 or 2 vertices. Assume the statement is correct for graphs with $n-1$ vertices and let G be a triangulated graph with n vertices. Let a,b be two nonadjacent vertices in G, and let S be minimal a-b separator. Denote by G_A, G_B the connected components of G_{V-S} which contain a,b respectively. We shall look at G_{A+S}:

1. If G_{A+S} is complete, then each of its vertices is a simplicial vertex in G_{A+S}. Particularly, every vertex in A which is simplicial in G_{A+S} is also simplicial in G (as there are no edges between A and B).

2. If G_{A+S} is not complete, by induction it contains two nonadjacent simplicial vertices. Since S is complete (we already proved $1 \to 3$ in Theorem 3.3) at least one of the simplicial vertices is in A and like before this vertex is simplicial in G.

We showed that A contains a simplicial vertex in G. In the same manner, B contains a simplicial vertex in G. These two vertices are not adjacent. ■

Now we will continue the proof of Theorem 3.3:

(1) $\to$ (2) By Lemma 3.4, G contains a simplicial vertex x. $G_{V-\{x\}}$ is triangulated and contains less vertices, so by induction it has a perfect elimination order $[\sigma_{i_1},\ldots,\sigma_{i_{n-1}}]$. Since x is simplicial $[x,\sigma_{i_1},\ldots,\sigma_{i_{n-1}}]$ is a perfect elimination order in G.

(2) $\to$ (1) Let C be a simple cycle in G. Let x be the *first* vertex from C to appear in the perfect elimination order. In the induced graph before eliminating x, the two neighbors of x in the cycle still exist, and since x is simplicial in that graph, these two vertices form a chord in the cycle. Hence, every cycle in G contains a chord.

■

```
assign label 0 to each vertex
for i ← n to 1 by −1 do
    select: pick an unnumbered vertex v with largest label
    σ(i) ← v
    update: for each unnumbered vertex w ∈ Adj(v) do
        add 1 to label(w) end for
end for
```

Figure 3.3: Code for MCS algorithm

3.2.3 Recognizing Triangulated Graphs

If a graph G is triangulated and is not complete, it has two nonadjacent simplicial vertices. Also every induced subgraph of G is also triangulated. Therefore it is possible to choose *any* vertex v and decide it will be last in the PEO (it is always possible to choose another simplicial vertex instead of v). Afterwards, it is possible to choose a vertex u adjacent to v and put it in $(n-1)$st position. It is possible to continue choosing vertices *backwards* to the perfect elimination order. Using the above idea will give us a linear time algorithm for recognizing triangulated graphs. The following linear time algorithm (called Maximum Cardinality Search) by Tarjan and Yannakakis (1985), receives a graph G given by adjacency list for each vertex, and outputs an order σ of the vertices, with the property that if G is triangulated then σ is a PEO. We will show later how to check if this vertex order is a perfect elimination order (i.e. if the graph is triangulated) in linear time. The code is in Figure 3.3, and a running example is given in Figure 3.4.

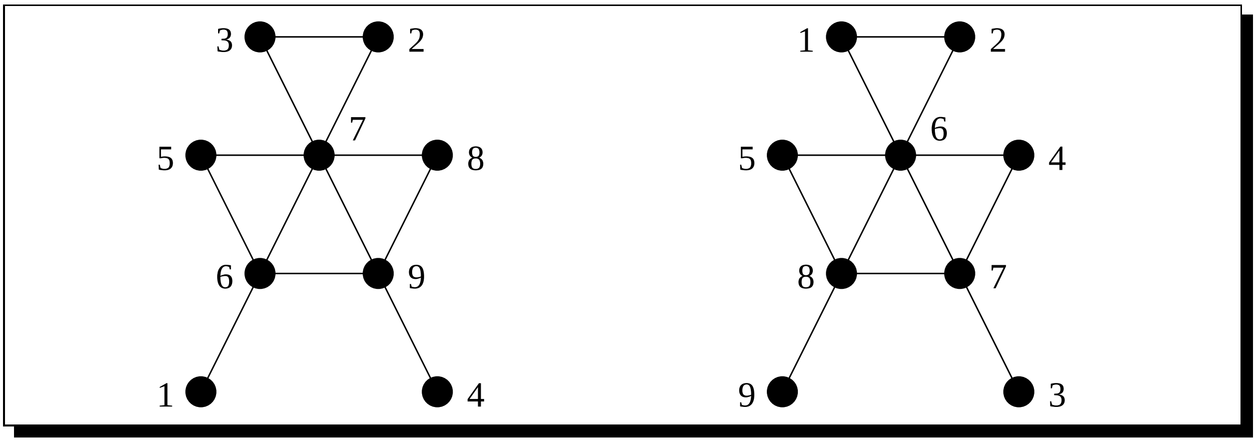


Figure 3.4: Two different possible running of the MCS algorithms.

Time Complexity

Let S_i be the set of all vertices v that have not been numbered yet, and have label$(v) = i$. Each S_i is represented by a doubly linked list. The set of active S_i-s (all those S_i-s which were non-empty at some point) will also be represented by a doubly linked list in increasing order of i,with a pointer to the non-empty set with largest label. For each vertex we will store its label i and a pointer to its position in S_i. When v receives a number we take v out of its list and move each neighbor of v one list upwards. This takes $O(1 + degree(v))$, therefore the overall time complexity is $O(|V| + |E|)$.

Chapter 4

Lecture 4

Advanced Topics in Graph Algorithms

Spring Semester, 1994

Lecturer	:	Ron Shamir
Scribe	:	Izik Pe'er
Lecture 4	:	May 12

4.1 Recognizing Triangulated Graphs

4.1.1 Generating a PEO

In lecture 3 we have introduced the *MCS* algorithm for generating a PEO for a triangulated graph, and proved that it is a linear time algorithm. We shall prove the correctness of this algorithm.

Theorem 4.1 *(Tarjan and Yannakakis 85)*
A graph $G = (V, E)$ is triangulated if and only if the MCS *algorithm produces a permutation σ* [1] *which is a PEO.*

Proof:

$\Rightarrow$ Trivial.

$\Leftarrow$ We shall prove the claim by induction on the number of vertices n.

[1] We shall refer to each permutation σ simply as a one-to-one function from $\{1, \ldots, n\}$ to V, so that $\sigma(1)$ is the first vertex in the order, $\sigma(2)$ is the second, etc. Surely, for each vertex v, $\sigma^{-1}(v)$ is v's serial number in the order.

The theorem is trivially true for $n = 1$. Assume correctness for every graph with less than n vertices. Let $G(V, E)$ be a triangulated graph with n vertices, and let σ be the permutation generated by the *MCS* algorithm with input G. According to the induction hypothesis it suffices to show that the vertex $v = \sigma(1)$ is simplicial.

Claim 4.2 *G cannot contain an induced chordless path $\mu = [u = v_0, v_1, \ldots, v_k, w]$ for $k \geq 1$ so that*

$$\forall i, 1 \leq i \leq k : \sigma^{-1}(v_i) < \sigma^{-1}(u) < \sigma^{-1}(w) \tag{4.1}$$

Proof: Assume, by contrary, that such a path exists. Let μ be a path satisfying 4.1, such that $\sigma^{-1}(u)$ is maximal. u was numbered before v_k, despite the fact that w is a neighbor of v_k and not of u. According to the maximum cardinality rule, this implies there exists a vertex x, such that x is a neighbor of u and not of v_k, and x was numbered before u, as in figure 4.1.

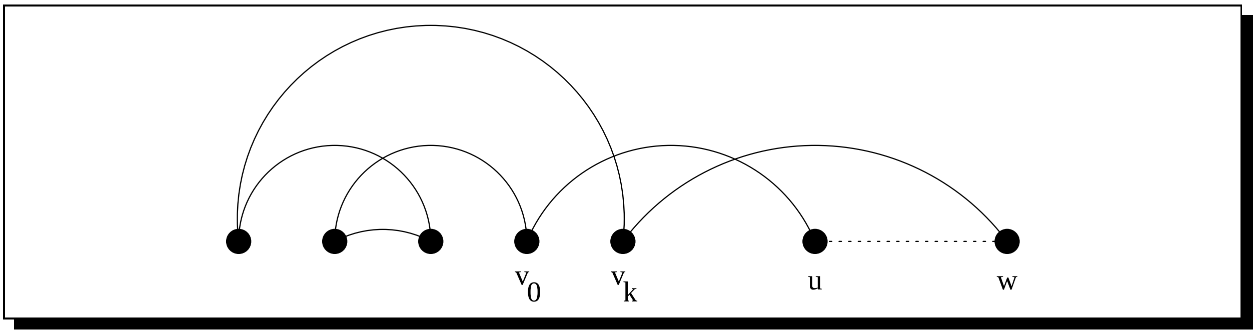


Figure 4.1: The path μ. Vertices appear ordered from left to right according to the *MCS* algorithm output.

Let j be the maximal index such that v_j is a neighbor of x. Surely j is well defined, since $u = v_0$ is a neighbor of x, and $j < k$, by the definition of x. It follows from the maximality of j that the induced path $\mu' = [x, v_j, \ldots, v_k, w]$ is chordless ($xw \notin E$, otherwise $\mu' = [x, v_j, \ldots, v_k, w, x]$ is a chordless cycle of length ≥ 4, contradicting G's chordality).

If $\sigma^{-1}(x) < \sigma^{-1}(w)$ (as in figure 4.3), then for each $j \leq i \leq k$, $\sigma^{-1}(v_i) < \sigma^{-1}(u) < \sigma^{-1}(x)$, so the path μ' meets the condition 4.1, and contradicts the maximality of $\sigma^{-1}(u)$, since $\sigma^{-1}(x) > \sigma^{-1}(u)$. Hence $\sigma^{-1}(x) > \sigma^{-1}(w)$ (as in figure 4.4), and the path $\mu'' = [w, v_k, \ldots, v_j, x]$ satisfies 4.1, contradicting the maximality of $\sigma^{-1}(u)$.

It follows that there is no path satisfying 4.1. ■

Back to the proof of Theorem 4.1:

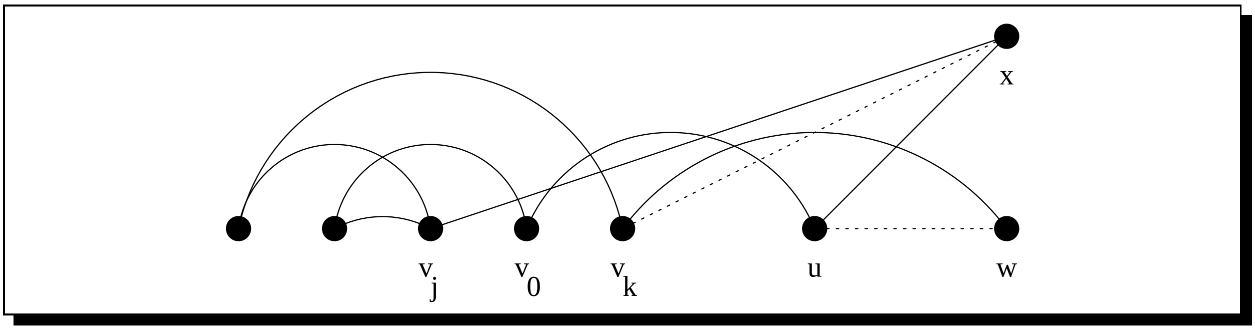


Figure 4.2: The vertices x and v_j

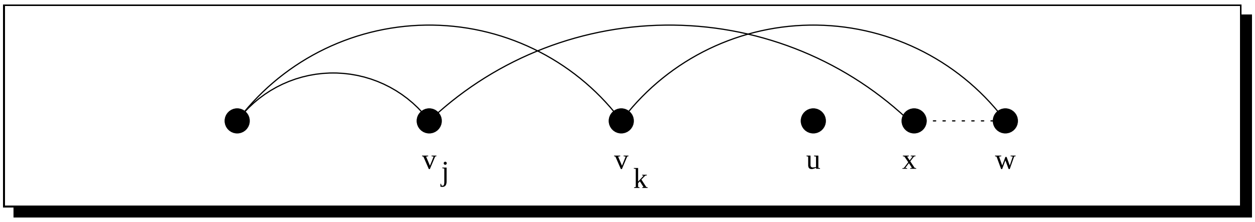


Figure 4.3: The path μ'

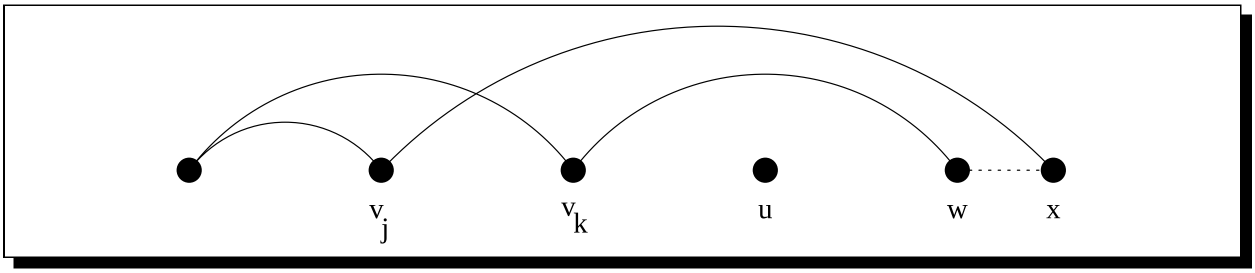


Figure 4.4: The path μ''

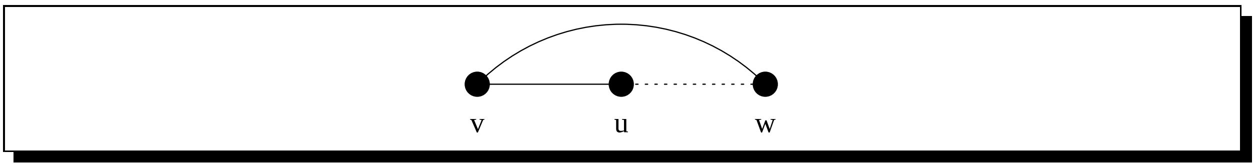


Figure 4.5: A forbidden structure in a chordal graph

Denote $v = \sigma^{-1}(1)$, and assume, by contrary that, v is not a simplicial vertex. Then there exist $u, w \in Adj(v)$ such that $uw \notin E$, and $\sigma^{-1}(u) < \sigma^{-1}(w)$ (as in figure 4.5. It follows that the path $[u, v, w]$ satisfies 4.1 of Claim 4.2, yielding a contradiction.

■

4.1.2 Testing an Elimination Scheme

In order to recognize whether a graph is triangulated, we can use the *MCS* algorithm to get an elimination scheme, but we have yet to describe how we can test whether this elimination scheme is a PEO.

Naive Algorithm

Simulate the elimination process, and for each vertex to be eliminated, check whether its noneliminated neighbors are a clique. This strategy results in a total time complexity of $O(|V| \cdot |E|)$.

Efficient Algorithm

We can improve complexity if we collect a list of desired neighbors for each vertex, and check all the members in this list only once.

begin
1. **for each** vertex v **do** $A(v) \leftarrow \emptyset$
2. **for** $i \leftarrow 1$ to $n-1$ **do**
3. $\quad v \leftarrow \sigma(i)$
4. $\quad X \leftarrow \{x \in Adj(v) | \sigma^{-1}(v) < \sigma^{-1}(x)\}$
5. $\quad$ **if** $X \neq \emptyset$ **then**
6. $\quad\quad u \leftarrow \sigma(min\{\sigma^{-1}(x) | x \in X\})$
7. $\quad\quad$ concatenate $X \setminus \{u\}$ to $A(u)$

$\quad$ **end if**

8. $\quad$ **if** $A(v) \setminus Adj(v) \neq \emptyset$ **then return** *FALSE*

end for

9. **return** *TRUE*

end

Figure 4.6: Code for PEO testing algorithm

Example

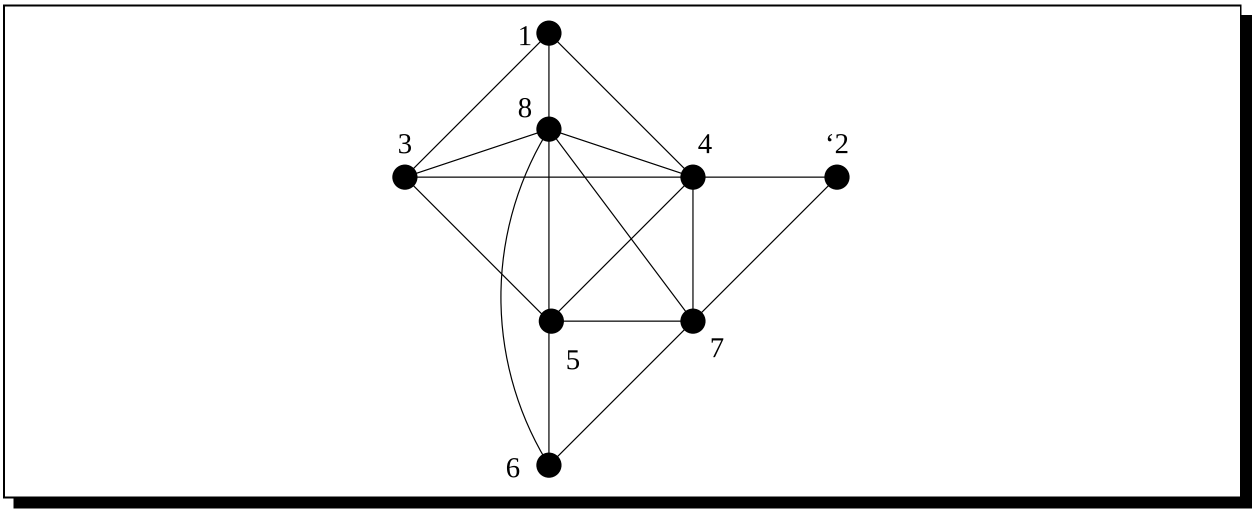


Figure 4.7: An input chordal graph and elimination order

When applying the algorithm in 4.6 to the input graph and elimination order in 4.7, the algorithm does the following iterations:

Iteration 1		$u = 3$	$A(3) = \{4, 8\}$	$X = 3, 4, 8$
Iteration 2	$X = 4, 7$	$u = 4$	$A(4) = \{7\}$	$A(2) \subseteq Adj(2)$
Iteration 3	$X = 4, 5, 8$	$u = 4$	$A(4) = \{7, 5, 8\}$	$A(3) \subseteq Adj(3)$
Iteration 4	$X = 5, 7, 8$	$u = 5$	$A(5) = \{7, 8\}$	$A(4) \subseteq Adj(4)$
Iteration 5	$X = 6, 7, 8$	$u = 6$	$A(6) = \{7, 8\}$	$A(5) \subseteq Adj(5)$
Iteration 6	$X = 7, 8$	$u = 7$	$A(7) = \{8\}$	$A(6) \subseteq Adj(6)$

Correctness of the Algorithm

Theorem 4.3 *The algorithm in figure 4.6 returns* TRUE *if and only if* σ *is a PEO.*

Proof:

$\Leftarrow$ Suppose the algorithm returns *FALSE* on iteration number $\sigma^{-1}(u)$. This may happen only if in stage 8 at that iteration there exists an unnumbered vertex $w \in A(u) \setminus Adj(u)$. The vertex w was added to $A(u)$ at stage 7 of a prior iteration, number $\sigma^{-1}(v)$. It follows that $\sigma^{-1}(v) < \sigma^{-1}(u) < \sigma^{-1}(w)$, for $u, w \in Adj(v)$, but since $uw \notin E$ (as in figure 4.5), σ is not a PEO.

$\Rightarrow$ Suppose σ is not a PEO, and the algorithm returns *TRUE*. Let v be a vertex with maximal $\sigma^{-1}(v)$, such that the set $X_v = \{w | w \geq Adj(v), \sigma^{-1}(v) < \sigma^{-1}(w)\}$ does not induce a clique. The vertex v is well defined, because if all of the X_v's are cliques, then σ is a PEO. Let u be the vertex defined in stage 6 of iteration $\sigma^{-1}(v)$. In stage 7 of this iteration, $X_v \setminus \{u\}$ is added to $A(u)$. Since the algorithm returns *TRUE* in stage 8 of iteration number $\sigma^{-1}(u)$, the condition of stage 8 (i.e. The condition $A(u) \subseteq Adj(u)$) is not met, so every $x \in X_v \setminus \{u\}$ is a neighbor of u.

Since $\sigma^{-1}(v)$ is chosen to be the maximal vertex for which X_v is not a clique, and v is prior to u, X_u is a clique, so all u's neighbors in $X_v \setminus \{u\}$ are adjacent to each other (as in figure 4.8, hence X_v is a clique, and a contradiction follows.

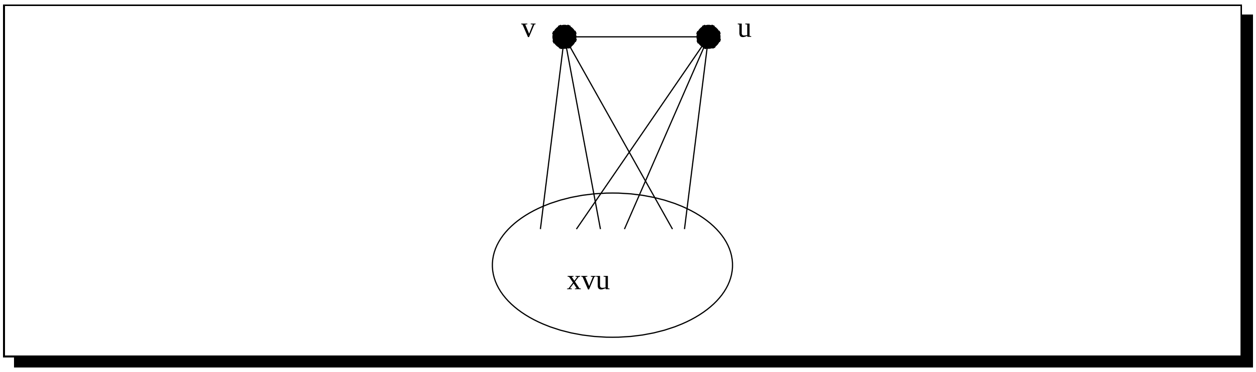


Figure 4.8: $X_v \setminus \{u\}$ is a clique

■

Complexity of the Algorithm

Theorem 4.4 *The algorithm 4.6 runs in time* $O(|V| + |E|)$.

Proof: We shall hold σ and σ^{-1} as vectors, and for each v we shall hold $Adj(v)$ and $A(v)$ as lists, allowing duplicates to be concatenated to $A(v)$. Stage 8 of iteration $\sigma^{-1}(v)$ can be implemented in $O(|Adj(v)| + |A(v)|)$ by holding a bit vector $Test$ of length n, and the condition $A(v) \setminus Adj(v) = \emptyset$ is implemented in the following way :

1. for each $w \in Adj(v)$, set the bit $Test(w)$ to 1.
2. for each $w \in A(v)$, if $Test(w)$ is 0, return *FALSE*.
3. for each $w \in Adj(v)$, clear the bit $Test(w)$ to 0.
4. return *TRUE*

The resulting overall complexity is $O(|V| + \sum v \in V|Adj(v)| + \sum v \in V|A(v)|)$, where $|A(v)|$ is the last (maximal) size of $A(v)$, including the duplicates. For each $v \in V$, $Adj(v)$ adds vertices only to one list $A(u)$, and the number of added vertices is at most $|Adj(v) - 1|$. Therefore $\sum v \in V|A(v)| < \sum v \in V|Adj(v)| = O(|E|)$ and the overall time complexity is $O(|V| + |E|)$. ■

Corollary 4.5 *There is a linear time algorithm for checking whether a given graph is triangulated.*

4.2 Triangulated Graphs as Intersection Graphs

In Lecture 1 it was shown that every interval graph is triangulated, and that interval graphs are characterized as intersection graphs. Is there such a characterization for triangulated graphs as well? We shall see that a graph G is triangulated if and only if G is an intersection graph of a family of subtrees of a tree. The tree containing all the subtrees is called the *host tree*.

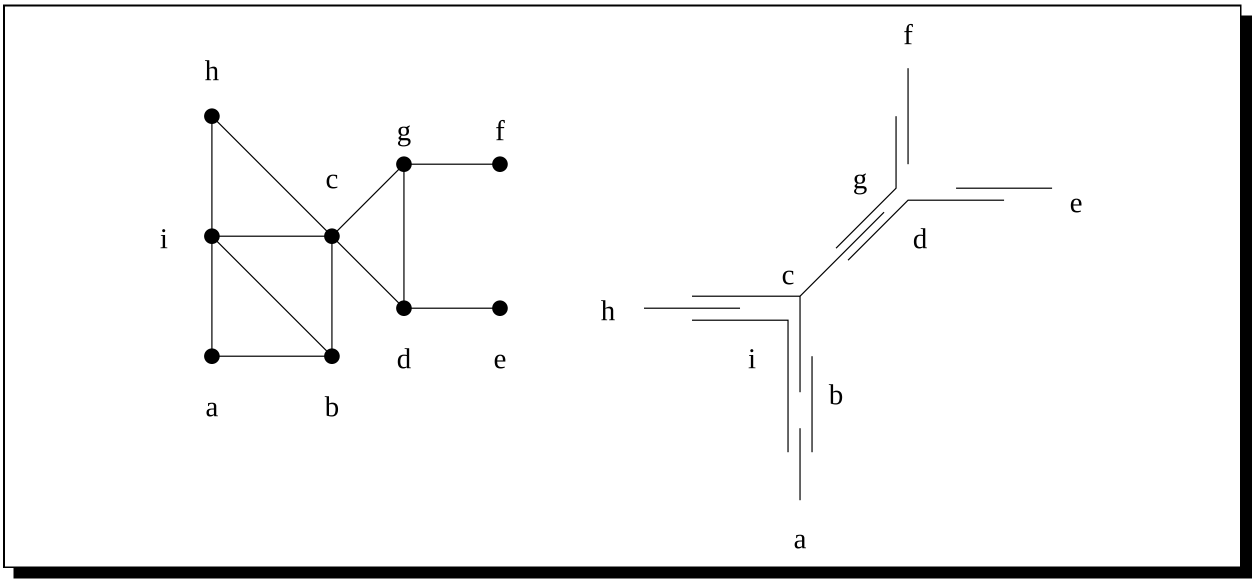


Figure 4.9: The intersection graph of a set of subtrees of a host tree is triangulated

4.2.1 Evolutionary Trees

An application for this characterization of triangulated graphs is the evolutionary tree, also called the phylogenetic tree. This model is formulated in the study of evolution, where we wish to determine the way a group of species evolved in time to their current properties. We use the following assumptions :

- There exists a single ancient species, from which all the other species have evolved.
- Each ancient species might have split into two subspecies, because of a mutation changing a property of one of the subspecies.
- Mutations are rare, so each mutation can occur only once in the tree.

We can therefore define the evolutionary tree, which is a rooted binary tree. In this tree, every vertex corresponds to a species (every leaf corresponds to a current species). Because of the third assumption for every property the set of all the species having that property corresponds to a set of tree vertices that induces a *connected subgraph*, i.e. a subtree.

We can build a graph with vertices that match properties. Two vertices will be adjacent if their properties appear together in a certain species (as in figure 4.10). This graph must be *triangulated*.

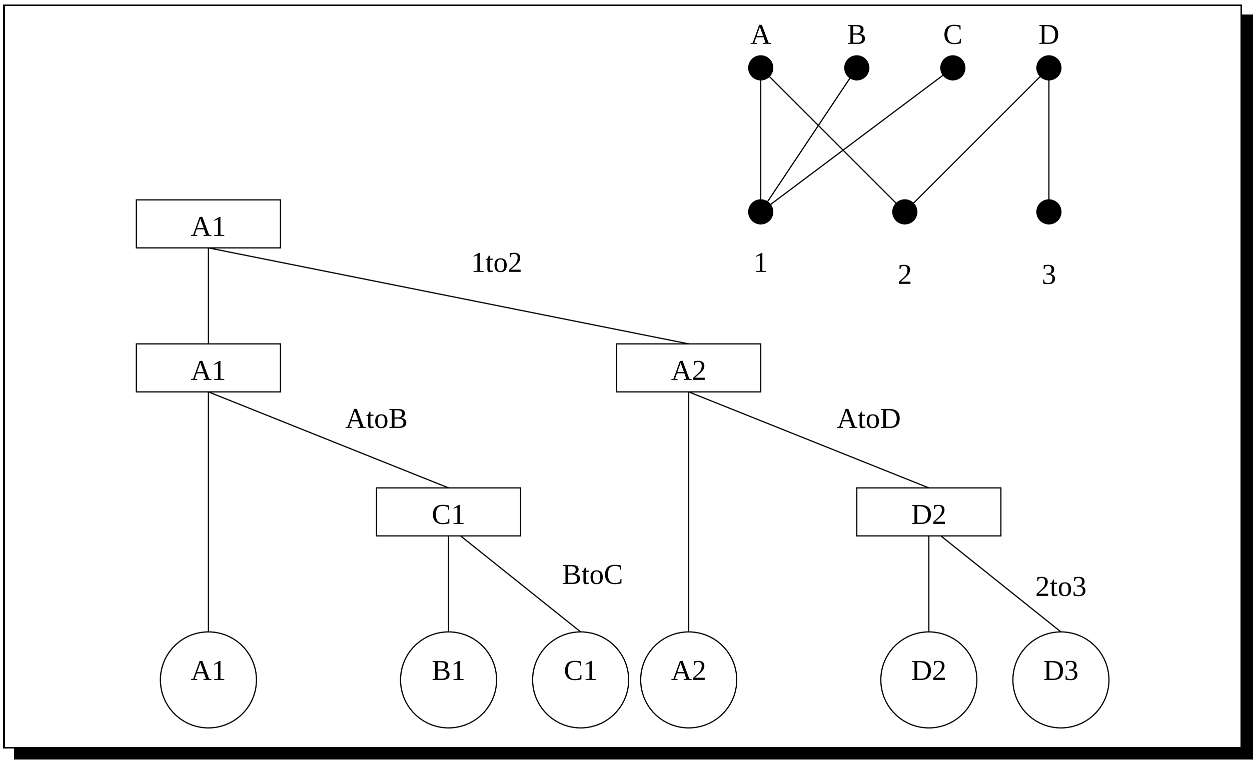


Figure 4.10: An evolutionary tree describing the properties $A, B, C, D, 1, 2, 3$, and the corresponding triangulated intersection graph

4.2.2 Triangulated Graphs as Intersection Graphs

Definition Let $S = \{T_i\}_{i \in I}$ be a family of subsets of a set T. S has the *Helly* property if and only if for every $J \subseteq I$, if $\forall i, j \in J : T_i \cap T_j \neq \emptyset$ then $\bigcap_{j \in J} T_j \neq 0$. (figure)

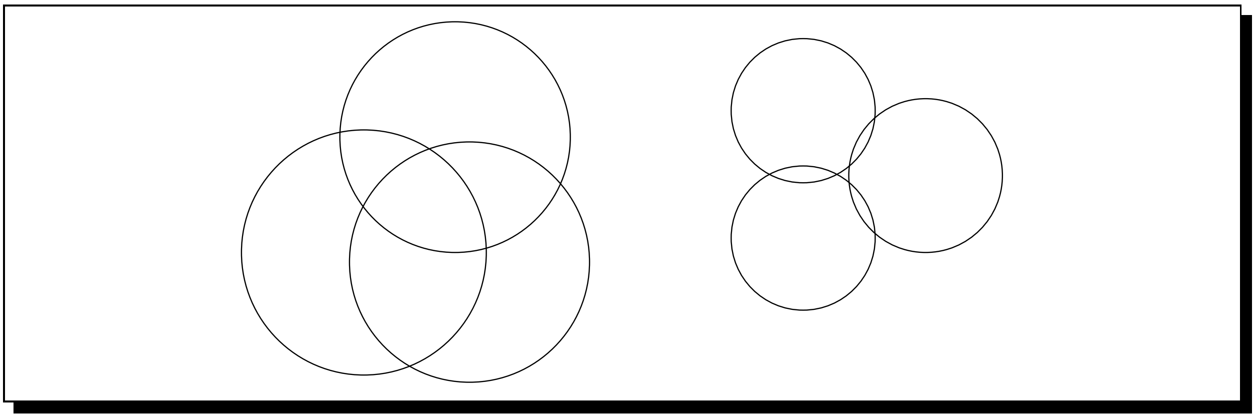


Figure 4.11: The left family has the Helly property, while the right family does not

Lemma 4.6 *Suppose that $\{T_i\}_{i \in S}$ is a set of subtrees of T, and that there exist points a, b, c on T, such for each $i \in S$, T_i contains at least two of them, then $\bigcap_{i \in S} T_i \neq \emptyset$*

Proof: Denote: P_1 the simple path from a to b in T, P_2 the simple path from b to c in T, and P_3 the simple path from c to a in T. Since T is a tree, $P_1 \cap P_2 \cap P_3 \neq \emptyset$ (otherwise $P_1 - P_2 - P_3$ contains a cycle). Every T_i contains at least one of the paths $P_i, i = 1, 2, 3$. Hence, $\bigcap_{i \in S} T_i \supseteq P_1 \cap P_2 \cap P_3 \neq \emptyset$.

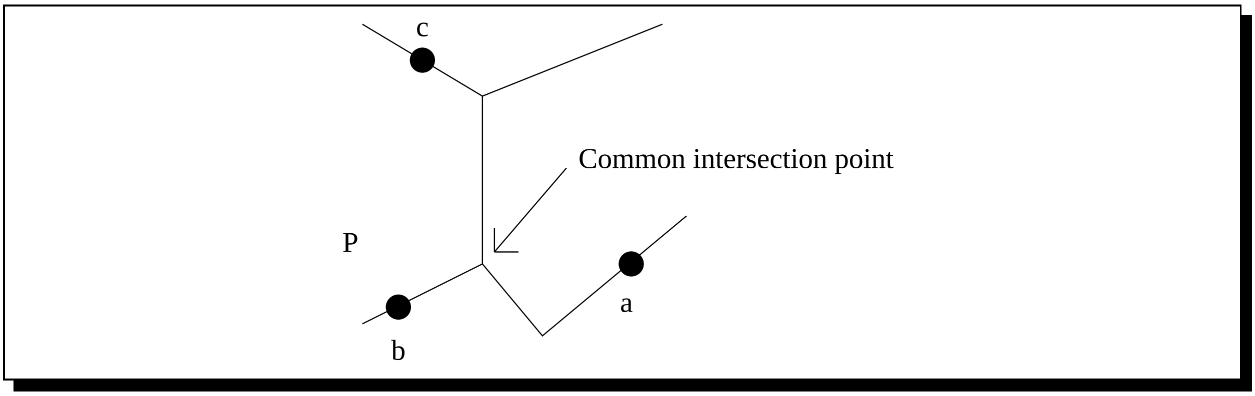


Figure 4.12: The paths P_1, P_2, and P_3 must intersect

■

Theorem 4.7 *A family of subtrees of a host tree has the* Helly *property.*

Proof: We shall prove the claim by induction on k, the size of the family J. For $k = 2$ the theorem is trivially true. Assume correctness for $|J| \leq k$. Let J be a family of $k+1$ trees : $\{T_1, \ldots, T_{k+1}\}$, each pair of which having a non-empty intersection. By the induction hypothesis:

- $\bigcap_{j=1}^{k} T_j \neq \emptyset$ so there exists $a \in \bigcap_{j=1}^{k} T_j$.
- $\bigcap_{j=2}^{k+1} T_j \neq \emptyset$ so there exists $b \in \bigcap_{j=2}^{k+1} T_j$.
- In addition, by assumption, $T_1 \cap T_{k+1} \neq \emptyset$ so there exists $c \in T_1 \cap T_{k+1}$.

Lemma 4.6 applied for $\{T_1, \ldots, T_{k+1}\}$ and the points a, b, c, yields $\bigcap_{j=1}^{k+1} T_j \neq \emptyset$. ∎

Theorem 4.8 *(Walter 72, Gavril 74, Buneman 74)*
Let $G(V, E)$ be an undirected graph. The following statements are equivalent :

(1) G is triangulated.

(2) G is the intersection graph of a family of subtrees of a tree.

(3) There exists a tree $T = (\bar{K}, \bar{E})$ (as in figure 4.13) such that $\bar{K}$ is the set of maximal cliques in G, and for every $v \in V$, if $\bar{K}_v = \{C \in \bar{K} | v \in C\}$, then $T_{\bar{K}_v}$, the subgraph induced by $\bar{K}_v$, is connected (hence a subtree).

Proof:

(3) $\Rightarrow$ (2) Let T be a tree satisfying (3). We shall prove that $vw \in E \Leftrightarrow T_{\bar{K}_v} \cap T_{\bar{K}_w} \neq \emptyset$:

$\Rightarrow$ If $v, w \in V$, $vw \in E$, then for some maximal clique $C \in \bar{K}$, $v, w \in C$ so $T_{\bar{K}_v} \cap T_{\bar{K}_w} \neq \emptyset$.

$\Leftarrow$ If $T_{\bar{K}_v} \cap T_{\bar{K}_w} \neq \emptyset$ then there are maximal cliques that contain both w and v, so $vw \in E$.

It follows that G is an intersection graph of the family $\{T_{\bar{K}_v}\}_{v \in V}$ of subtrees of T.

(2) $\Rightarrow$ (1) Let $\{T_{\bar{K}_v}\}_{v \in V}$ be a family of subtrees of a tree T, for which $vw \in E \Leftrightarrow T_{\bar{K}_v} \cap T_{\bar{K}_w} \neq \emptyset$. Suppose that G contains an induced chordless cycle $[v_0, v_1, \ldots, v_{k-1}, v_0]$, for $k > 3$. Let the corresponding subtrees be $T_0, T_1, \ldots T_{k-1}$, respectively. It is obvious that $|i - j| \leq 1 (mod k) \Leftrightarrow T_i \cap T_j \neq \emptyset$. Let a_i be an arbitrary point in $T_i \cap T_{i+1}$, for $i = 0, \ldots, k-1$ (throughout, indices are mod k). We would like to prove that the a_i's form a cycle,

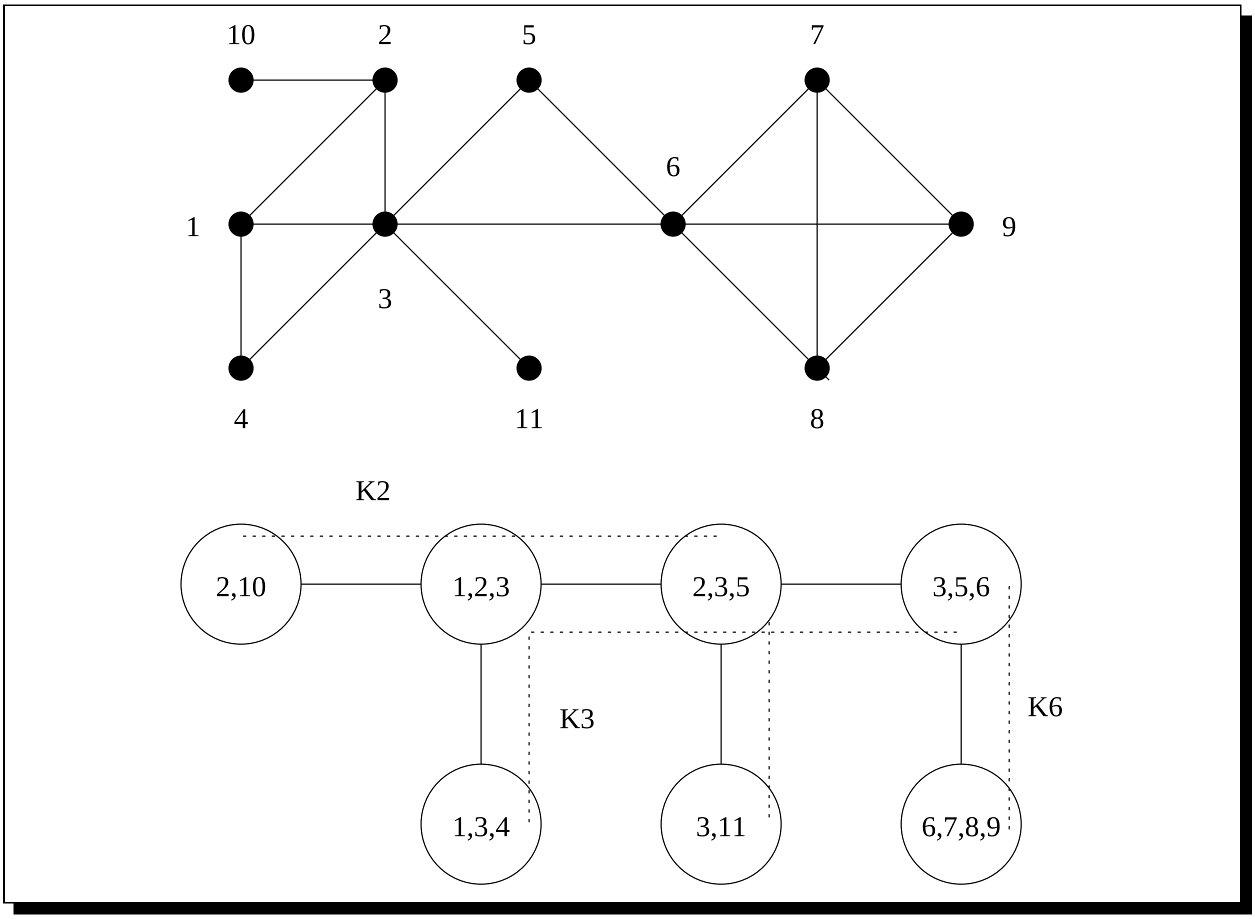


Figure 4.13: The tree $T = (\bar{K}, \bar{E})$

but we have to "throw away" possible branches, which are not on the main cycle, but that the a_i's may lie on. Let b_i be the last common point in the simple paths $a_i \to a_{i-1}$ and $a_i \to a_{i+1}$. The path $a_i \to a_{i-1}$ is in T_i and the path $a_i \to a_{i+1}$ is in T_{i+1}, therefore $b_i \in T_i \cap T_{i+1}$.

Denote by P_i the simple path from b_{i-1} to b_i. It follows that $P_i \subseteq T_i$, hence $P_i \cap P_j = \emptyset$ for every $|i-j| > 1$. On the other hand, $P_i \cap P_{i+1} = \{b_i\}$ for every i. We have shown that $\bigcup_{i=1}^{k} P_i$ is a simple cycle in T, contradicting T being a tree.

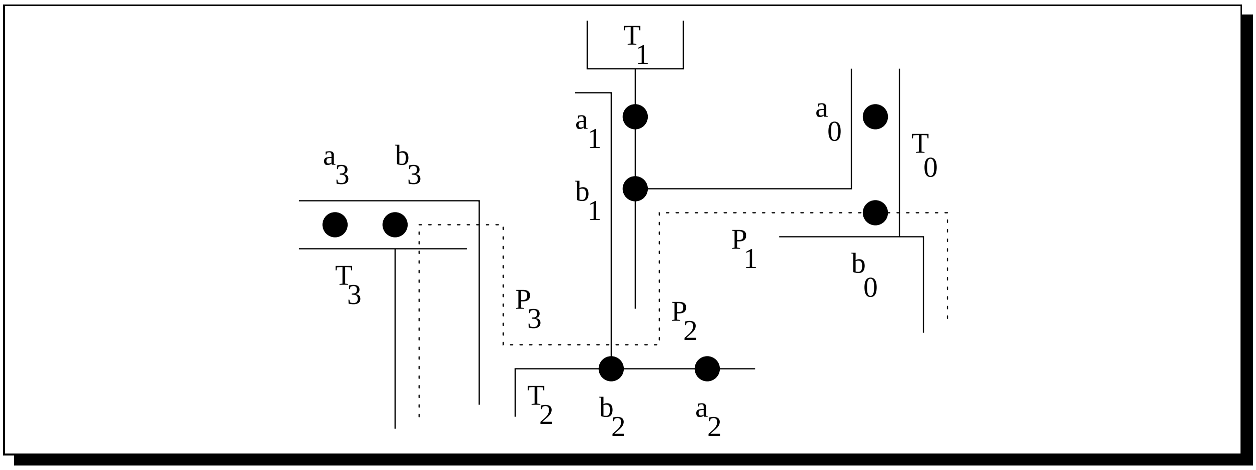


Figure 4.14: The paths $P_0, \ldots, P_{k-1}$ form a cycle

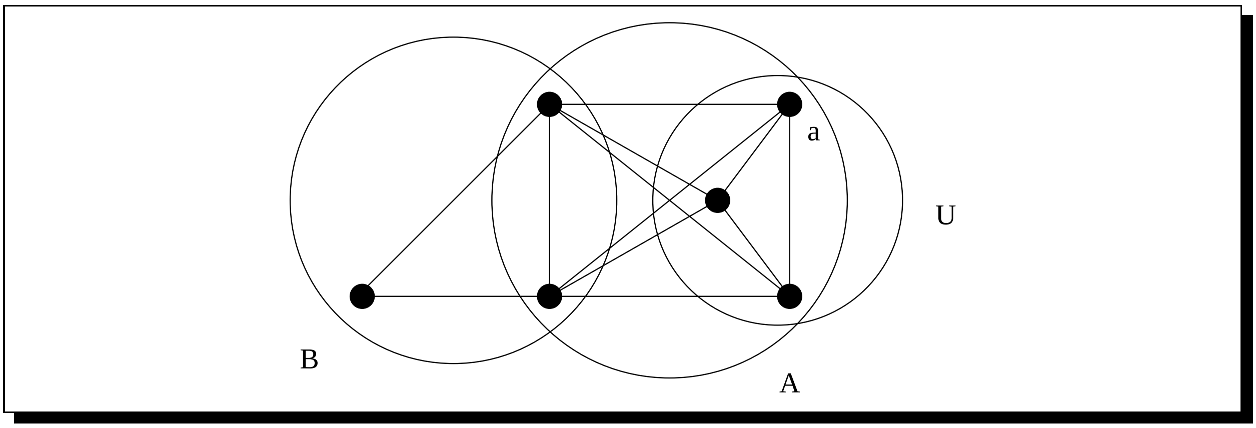


Figure 4.15: The case $B \neq Y$

(1) $\Rightarrow$ (3) We shall prove the claim by induction on n, the number of vertices in G. Assume correctness for every graph with less then n vertices.

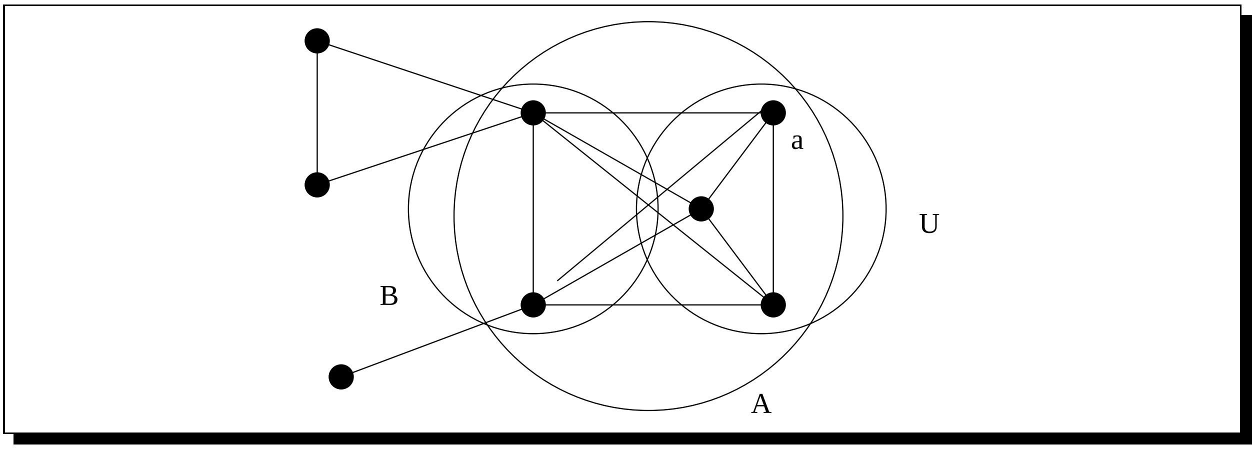


Figure 4.16: The case $B = Y$

- If G is the complete graph then $\bar{K} = V$, and T contains one vertex making the claim trivial (this applies also for $n = 1$).
- If G is not connected with connected components $G_1, \ldots, G_k$, then by the induction hypothesis for each G_i there is a host tree T_i satisfying (3), for $i = 1, \ldots, k$. For each $i = 1, \ldots, k-1$, connect T_i to T_{i+1} by adding an arbitrary edge. The resulting graph is a tree T satisfying (3).
- If G is a connected, not complete, triangulated graph, then let a be a simplicial vertex in G. The set $A = \{a\} \cup Adj(a)$ is a maximal clique in G. Denote $U = \{u \in A | Adj(u) \subset A\}$ and $Y = A \setminus U$.
  - $U \neq \emptyset$ because $a \in U$.
  - $Y \neq \emptyset$ because G is not complete.
  - $V \setminus A \neq \emptyset$ because G is connected and not complete.

  The graph $G' = G_{V \setminus U}$ is triangulated and contains less than n vertices, therefore, by the induction hypothesis there is a tree $T' = (\bar{K}', E')$, whose vertices correspond to the maximal cliques in G', in a way that for each vertex $u \in V \setminus U$, the set $\bar{K}'_v = \{C \in \bar{K}' | v \in C\}$ induces a connected subtree of T'.

  Let us examine the connection between $\bar{K}$ and $\bar{K}'$, the sets of maximal cliques in G and in G', respectively. Let B be the maximal clique in G' that contains Y. There are two cases :

$B = Y$ In that case T can be generated from T' by renaming B as A. For each $y \in Y$: $\bar{K}_y = \bar{K}'_y + \{A\} \setminus \{B\}$ and $\bar{K}_y$ induces the same subtree as $\bar{K}'_y$, because all we did is renaming B as A (as in figure 4.16).

$B \neq Y$ In that case T can be generated from T' by adding a vertex A, and connecting it to B. For each $y \in Y$: $\bar{K}_y = \bar{K}'_y + \{A\}$ and $\bar{K}_y$ induces a connected subtree (as in figure 4.15).

In both cases, for each vertex $u \in U$, $\{A\} = \bar{K}_u$, and for each vertex $v \in V \setminus A$, $\bar{K}_v = \bar{K}'_v$, and these sets induce connected subtrees in T, respectively.

■

Chapter 5

Lecture 5

Advanced Topics in Graph Algorithms

Spring Semester, 1994

Lecturer : Ron Shamir
Scribe : Lea Epstein
Lecture 5 : May 19

5.1 Triangulated Graphs Are Perfect

5.1.1 Proving This Property

In some graph families, the minimum graph coloring and the maximum clique problem, can be simplified using the principle of *separation into pieces*, as follows:

Theorem 5.1 (Berge, 1973) *Let G be a connected, undirected graph, let S be a vertex separator in G, and let $G_{A_1}, G_{A_2}, ..., G_{A_t}$ be the connected components of G_{V-S}. If S is a clique (not necessarily maximal), then*

$$\chi(G) = max_i \chi(G_{S+A_i})$$

$$w(G) = max_i \ w(G_{S+A_i})$$

Proof: Clearly $\chi(G) \geq \chi(G_{S+A_i})$ for each i, (because G_{S+A_i} is an induced subgraph of G) , so
$\chi(G) \geq k = max_i \ \chi(G_{S+A_i})$ On the other hand, G can be colored using exactly k colors, as follows:

- First, color G_S, (S is a clique, so there is only one way to do it).

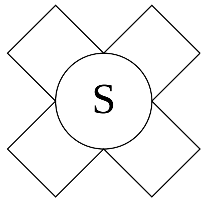


Figure 5.1: The Graph

- Then, independently extend the coloring to each piece G_{S+A_i}. Each piece can be colored with at most k colors, and S is always colored using $|S|$ colors , so it is possible to extend the coloring on S to G_{S+A_i} (renaming the colors if necessary).
- There will be no contradictions because if $v \in A_i$, $u \in A_j$, $i \neq j$ have the same color they are disconnected (because S separates them).

Next, consider the case of clique. Again $w(G) \geq w(G_{S+A_i})$ for each i, so
$w(G) \geq max_i\ w(G_{S+A_i}) = m$.
Let X be a maximum clique of G, i.e. $|X| = w(G)$. It is impossible that two vertices of X lie in G_{A_i} and G_{A_j}, $i \neq j$, since they are not connected . Thus, X is wholly contained in one of the pieces, say G_{S+A_r} . Hence $m \geq w(G_{S+A_r}) \geq |X| = w(G)$.
Therefore, $w(G) = m$. ∎

Corollary 5.2 *Let S be a separating set in a connected graph $G = (V, E)$, and let G_{A_1},...,G_{A_t} be the connected components of G_{V-S}. If S is a clique, and if each subgraph G_{S+A_i} is perfect, then G is perfect.*

Proof: The proof is by induction on n, the number of vertices in the graph.

Claim 5.3 *It is sufficient to prove $\chi(G) = w(G)$*

Proof: It is trivial if $n = 1$. Suppose $|V(G)| = n$ and the claim is true for all graphs with less then n vertices. Suppose S is a separating set, and a clique in G, and each G_{S+A_i} is perfect. Pick $v \in V$, and let $G' = G - \{v\}$. If $v \in S$, define $S' = S - \{v\}$. In G' S' is also a clique , and also each $G_{S'+A_i}$ is perfect (because this is an induced subgraph of a perfect graph). If $v \in A_i$, S is still a clique , and $G_{S+A_i-\{v\}}$ is perfect. In both cases, from the induction hypothesis , G' is perfect . This implies that for every proper induced subgraph H , $\chi(H) = w(H)$. ∎

Now prove that $\chi(G) = w(G)$.

$\chi(G) = max_i\ \chi(G_{S+A_i})$ (using theorem 5.1).
$max_i\ \chi(G_{S+A_i}) = max_i\ w(G_{S+A_i})$ (because every G_{S+A_i} is perfect),
$w(G) = max_i\ w(G_{S+A_i})$ (using theorem 5.1 again),

Thus, $\chi(G) = w(G)$. ∎

Theorem 5.4 *(Berge 1960, Hajnal-Suranyi 1958)*
Every triangulated graph is perfect .

Proof: By induction on the number of vertices in the graph. Let G be a triangulated graph. We shall assume it is connected, otherwise we consider each component individually.

- If G is complete, it is perfect.
- Otherwise let S be a minimal vertex separator for a pair a, b of nonadjacent vertices. By theorem 3.3, S is a clique. Each of the subgraphs G_{S+A_i} (defined in theorem 5.1) is triangulated (a hereditary property) and thus by induction hypothesis using corollary 5.2, G is perfect.

■

5.1.2 Other Results

Definition A *Meyniel graph* is a graph in which there are at least two chords in every odd cycle of length ≥ 5.

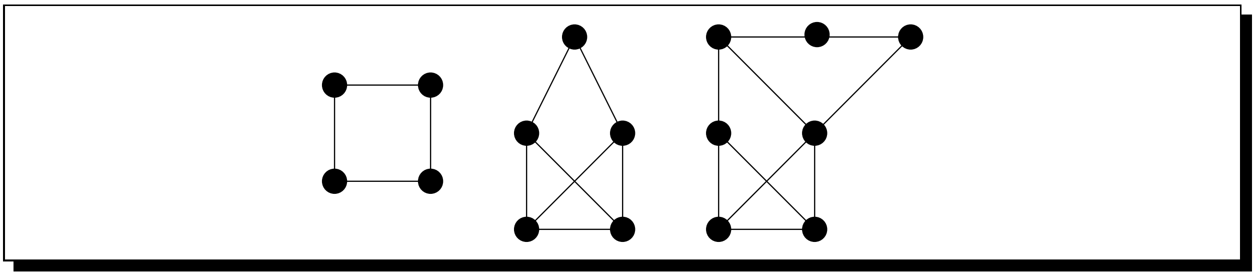


Figure 5.2: Some examples of Meyniel graphs (which are not chordal!)

Theorem 5.5 *(Meyniel 1976) Every Meyniel graph is perfect.*
(The proof can be found in Golumbic's book section 6.4).

Definition G is *weakly chordal graph* if G and $\bar{G}$ do not contain an induced C_k , $k \geq 5$.

Theorem 5.6 *(Hayward 1984) Weakly Chordal graphs are perfect.*

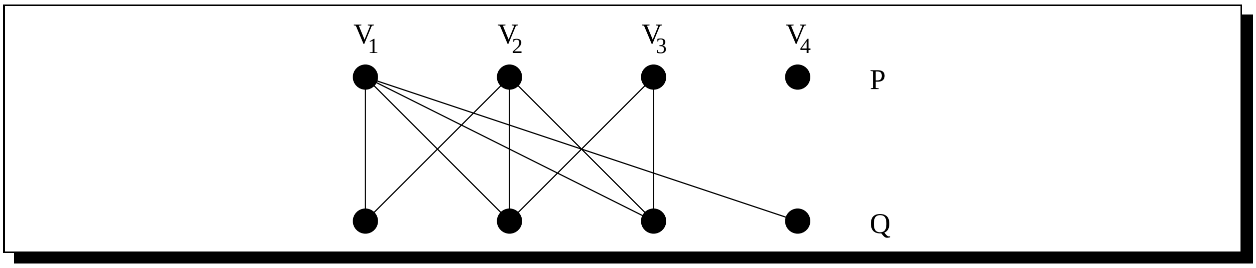


Figure 5.3: Example-A Chain Graph

5.2 Computing the Minimum Fill In

5.2.1 The Problem

The Minimum fill-in problem is defined as follows: Given a graph, find the minimum number of edges (fill in) whose addition makes the graph triangulated. i.e. for $G = (V, E)$ find F, such that $|F|$ is minimal and $G' = (V, E + F)$ is triangulated. This problem has to do with the Gaussian elimination of symmetric matrices. (When looking for an elimination with a maximal number of zeros in the matrix in each step - the problem arises in numerical algebra). In the book of Garey and Johnson, the minimum fill in problem (called there Chordal Graph Completion) is mentioned as a major open problem, shortly after the book was published, Yannakakis showed that the problem is NP complete.

5.2.2 Fill In is NP-Hard

Chain Graphs

Definition A bipartite graph $G = (P, Q, E)$ is a *chain graph* if one of the parts, say P, can be ordered v_1,v_2,..,v_k so that $Adj(v_1) \supseteq Adj(v_2) \supseteq ... \supseteq Adj(v_k)$ (the chain property).

This property is hereditary, (if we take out a vertex from P, the chain of containment doesn't change, if we take out a vertex from Q, the containment order does not change).

Claim 5.7 *If a bipartite graph is a chain graph, both parts of the graph have the chain property.*

Proof: We have to find an order for Q. Let $Q_i = \cap_{j \leq i} Adj(v_j)$ In this way, $Q_1 \supseteq Q_2 \supseteq ... \supseteq Q_k$. Let $R_i = Q_i - Q_{i+1}$, $i = 1, ..., k-1$, $R_k = Q_k$. Then $R_1 + R_2 + ... + R_k$ is a partition of Q. Impose a linear order $\prec$ on the vertices in Q such that if $u \in R_i$,$v \in R_j$ and $i < j$ then $v \prec u$. Thus if $v \prec u$, $u, v \in Q$ then either $u, v \in R_i$ and $Adj(u) = Adj(v) = \{v_1, ..., v_i\}$, or $u \in R_i$,$v \in R_j$ and $i < j$, in which case $Adj(v) = \{v_1, ..., v_j\} \supseteq \{v_1, ..., v_i\} = Adj(u)$. ■

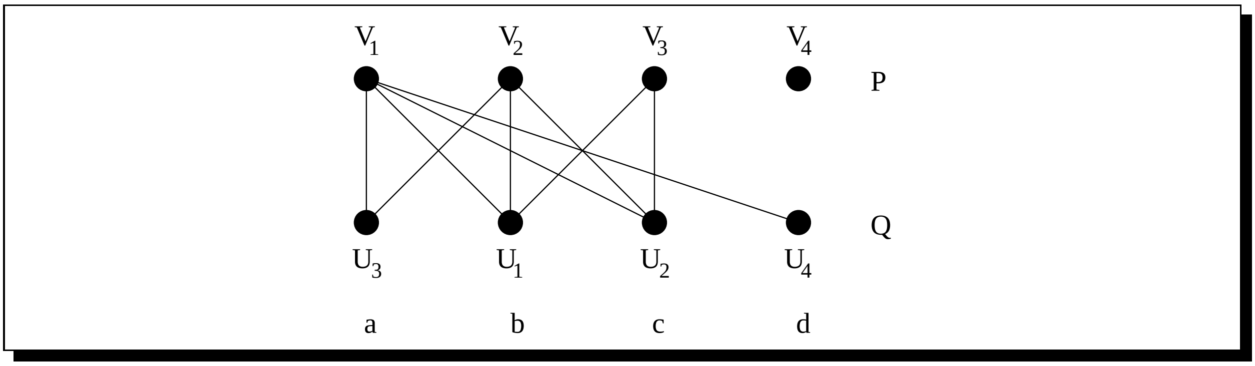


Figure 5.4: Ordering Q, in the same chain graph

For the example in Figure 5.4, we have:

$$Q_1 = \{a, b, c, d\};\, R_1 = \{d\}$$

$$Q_2 = \{a, b, c\};\, R_2 = \{a\}$$

$$Q_3 = \{b, c\};\, R_3 = \{b, c\}$$

$$Q_4 = \phi;\, R_4 = \phi$$

Definition Two edges ij,kl are independent if the graph induction on $\{i, j, k, l\}$ induces $2K_2$. (see figure 5.5)

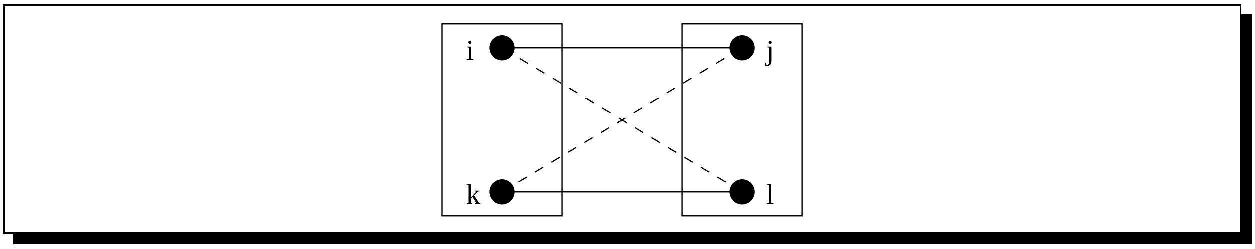


Figure 5.5: Independent edges (broken lines denote non-edges)

Theorem 5.8 *A bipartite graph G is a chain graph iff G does not contain two independent edges.*

Proof: ($\Longrightarrow$) Let ij and kl be independent edges. Then $Adj(i) \not\subseteq Adj(k)$, $Adj(i) \not\supseteq Adj(k)$ and G is not a chain graph.
($\Longleftarrow$) If this isn't a chain graph, there must be two vertices: $i, k \in P$ so that

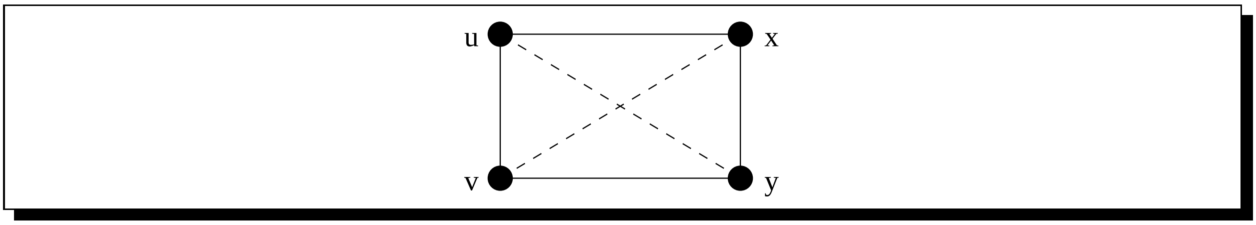


Figure 5.6: Independent edges that form an induced C_4

$Adj(i) \not\subseteq Adj(k)$ and $Adj(i) \not\supseteq Adj(k)$
there exist: $j \in Adj(i) - Adj(k)$ and $l \in Adj(k) - Adj(i)$
and the edges ij and kl are independent. ∎

Definition For a bipartite graph $G = (P, Q, E)$ we define a new graph $C(G) = (N, E')$, where $N = P \cup Q$, $E' = E + \{uv \mid \ u, v \in P\} + \{uv \mid \ u, v \in Q\}$. (i.e. we place the independent set on each side of the clique. See figure 5.7).

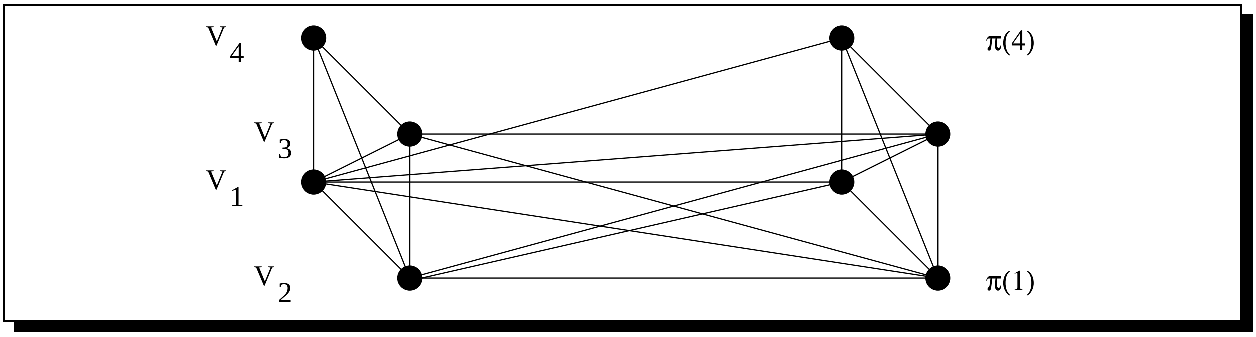


Figure 5.7: $C(G)$ for the graph G of Figure 5.3

Lemma 5.9 *G is a chain graph iff $C(G)$ is triangulated.*

Proof: ($\Longrightarrow$) If G is a chain graph, by lemma G contains two independent edges: ux , vy .

Thus, $\{u, v, y, x\}$ is an induced C_4 in $C(G)$.
($\Longleftarrow$) Let G be a chain graph, let π be an ordering of P , such that

$$Adj(\pi(1)) \supseteq Adj(\pi(2)) \supseteq ... \supseteq Adj(\pi(p))$$

where $p = |P|$. Since the property of being a chain graph is hereditary, it suffices to show that $C(G)$ has a simplicial vertex. The neighborhood $Adj'(\pi(p))$ of $\pi(p)$ in $C(G)$ is

$$Adj'(\pi(p)) = Adj_G(\pi(p)) \ + \ P \ - \ \pi(p)$$

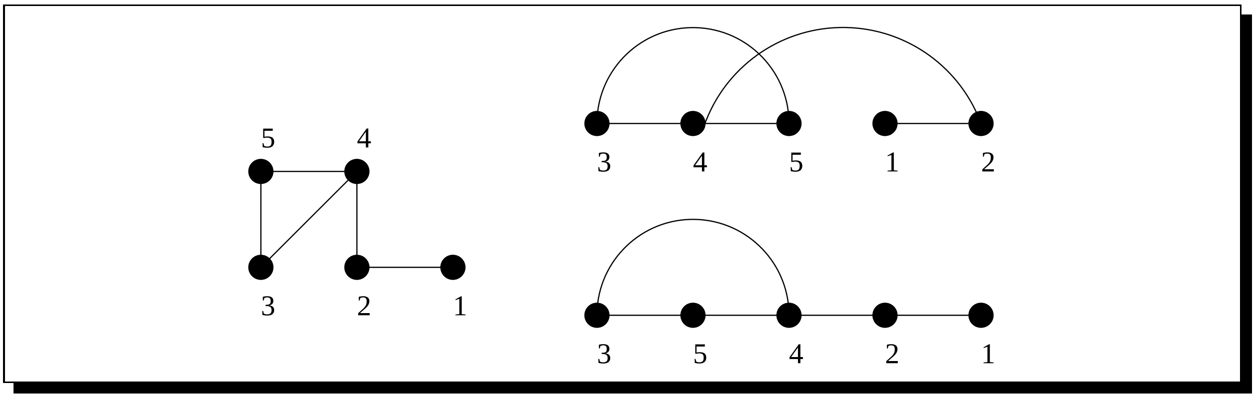


Figure 5.8: a graph (left) with two possible layouts of cost 8 (center) and 6 (right)

$Adj_G(\pi(p)) \subseteq Q$ and Q is a clique, so $Adj_G(\pi(p))$ is a clique in $C(G)$. $P - \pi(p) \subseteq P$ and P is a clique, so $P - \pi(p)$ is a clique in $C(G)$. Also, since $Adj(\pi(p)) \subseteq Adj(v)$ for every $v \in P$, all the nodes of P are adjacent to all nodes of $Adj(\pi(p))$. Therefore $Adj'(\pi(p))$ is a clique in $C(G)$, and $\pi(p)$ is a simplicial vertex of $C(G)$. ∎

Optimal Linear Arrangement (OLA)

A linear arrangement (or a layout) of a graph $G = (N, E)$ is a $1 - 1$ mapping $\pi : N \to \{1, 2, .., |N|\}$. For an edge $e = uv$ of G , let $\delta(e, \pi) = |\pi^{-1}(u) - \pi^{-1}(v)|$. The cost $c(\pi)$ of the linear arrangement π is $c(\pi) = \sum_{e \in E} \delta(e, \pi)$. The *optimal linear arrangement problem* is to decide, given a graph G and an integer k, whether there exists a linear arrangement π of G with cost $c(\pi) \leq k$. This problem is NP-complete.

Chain Graph Completion (CGC)

Given a bipartite graph $G = (P, Q, E)$ and a constant k, is there a set of non-edges F such that $|F| \leq k$ and $(P, Q, E + F)$ is a chain graph?

Theorem 5.10 *(Yannakakis , 1981)*
Chain graph completion is NP-complete.

Proof: Reduction from OLA: Given an instance of OLA: a graph $G = (N, E)$ and an integer k , we build an instance of CGC: a bipartite graph $G' = (P, Q, E')$ as follows: $P = N$, and Q contains two vertices e_1 and e_2 for every edge e of G, and a set $R(v)$ of $n - d(v)$ vertices for every vertex v of N, where $n = |N|$ and $d(v)$ is the degree of v in G. If $e = (u, v)$ is an edge of G, then the nodes e_1, e_2 that correspond to e are adjacent to u and v. The nodes in $R(v)$ are adjacent to v. In Fig 5.9 we see an example of this construction.

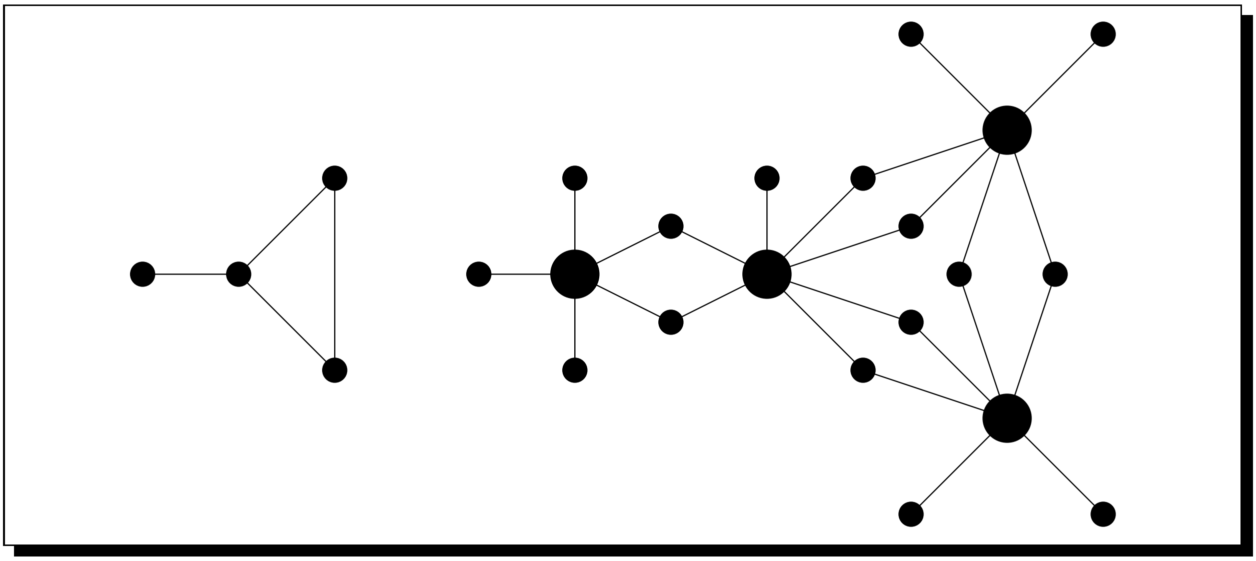


Figure 5.9: the graph G (left) and G' (right)

Let $A = [\frac{n^2(n-1)}{2}] - 2m$, where m is the number of edges in G. The constant for CGC will be $k' = k + A$. Let $l(G)$ be the minimum cost of a linear arrangement of G, and $h(G')$ the minimum number of edges whose addition to G' gives a chain graph. We shall prove that $h(G') = l(G) + A$ Thus $l(G) \leq k \iff h(G') \leq k + A$. which will prove the correctness of the reduction. First observe that an ordering π of N specifies uniquely a minimal set $H(\pi)$ of edges whose addition makes G' into a chain graph with the neighborhoods of the nodes in P ordered according to π. For every node $x \in Q$, let $\sigma(x) = max\{i \mid (x, \pi(i)) \in E'\}$. Then

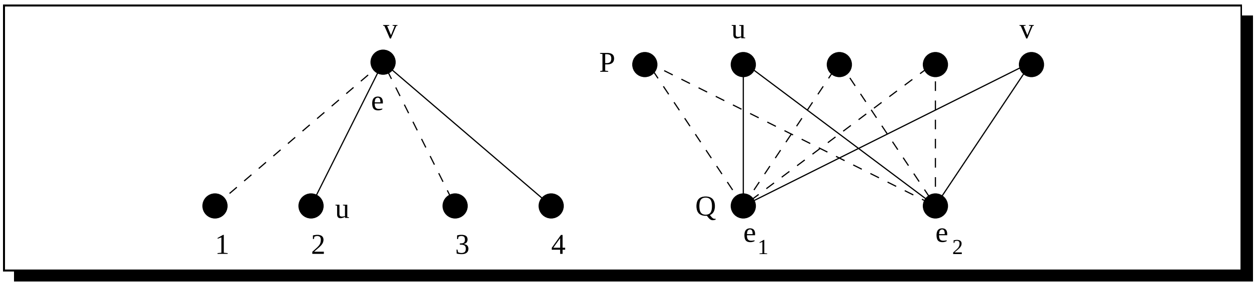


Figure 5.10: Example-Edges we have to add

$H(\pi) = \{(x, \pi(j)) \mid x \in Q, j < \pi^{-1}(x)\} - E'$.
Conversely, suppose that F is a set of edges such that $G'(F) = (P, Q, E' \cup F)$ is a chain graph and let π be an ordering of the nodes in P according to their neighborhoods in $G(F')$. It is easy to see that $F \supseteq H(\pi)$, and therefore if F is a minimal augmentation then $F = H(\pi)$. Let $h(\pi) = |H(\pi)|$. In order to complete the proof, it suffices to show that for every ordering π of N, $h(\pi) = c(\pi) + [\frac{n^2(n-1)}{2}] - 2m$,where $c(\pi)$ is the cost of the linear arrangement π of G.

Let π be an ordering of N. For every $v \in N$ and $x \in R(v)$, (see Figure 5.11) $H(\pi)$ contains $\pi^{-1}(v) - 1$ edges incident to x. Let $e = uv$ be an edge of G, and suppose without loss of generality that $\pi^{-1}(u) < \pi^{-1}(v)$. The number of edges of $H(\pi)$ incident to each of the two nodes e_1,e_2 that correspond to e is

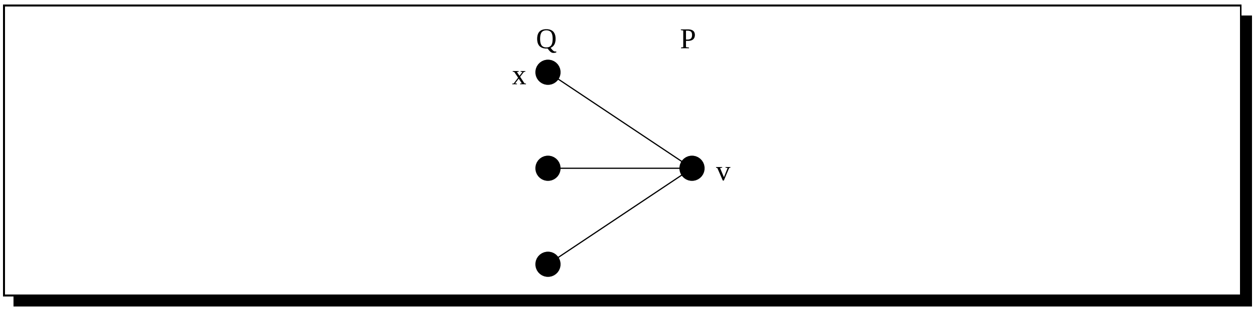


Figure 5.11: x and v

$$\pi^{-1}(v) - 2 = \pi^{-1}(u) + [\pi^{-1}(v) - \pi^{-1}(u)] - 2$$
$$= \pi^{-1}(u) + \delta(e, \pi) - 2;$$

thus the number of edges of $H(\pi)$ incident to e_1 and e_2 is
$2(\pi^{-1}(v) - 2) = \pi^{-1}(v) - 2 + \pi^{-1}(u) + \delta(e, \pi) - 2$
$= \pi^{-1}(u) + \pi^{-1}(v) + \delta(e, \pi) - 4$. Consequently:

$$h(\pi) = \sum_{v \in N} \sum_{x \in R(v)} [\pi^{-1}(v) - 1] \tag{5.1}$$

$$+ \sum_{e=uv \in E} [\pi^{-1}(v) + \pi^{-1}(u) + \delta(e, \pi) - 4] \tag{5.2}$$

$$= \sum_{v \in N} (n - d(v))(\pi^{-1}(v) - 1) \tag{5.3}$$

$$+ \sum_{v \in N} d(v)(\pi^{-1}(v)) + \sum_{e \in E} \delta(e, \pi) - 4m \tag{5.4}$$

$$= \sum_{v \in N} n[\pi^{-1}(v) - 1] + \sum_{v \in N} d(v) + c(\pi) - 4m \tag{5.5}$$

$$= c(\pi) + [\frac{n^2(n-1)}{2}] - 2m \tag{5.6}$$

the last equality follows

$$\sum_{v \in N} d(v) = 2m, \tag{5.7}$$

and

$$\sum_{v \in N} [\pi^{-1}(v) - 1] = 0 + 1 + 2 + ... + (n-1) = [\frac{n(n-1)}{2}] \tag{5.8}$$

■

The result for Fill in

Theorem 5.11 *Minimum Fill in is NP-hard*

Proof: Reduction from CGC: Given $G = (P, Q, E)$ and a constant k we build the graph $C(G)$, and give the same k (as an instance of the minimum fill in problem). By Lemma 5.8 Adding edges to $C(G)$ creates a triangulated graph iff adding the same edges to G creates a chain graph. ■

5.2.3 Problems

- **Maximum Chordal subgraph:** Remove minimum number of edges from a graph, until you get a triangulated graph. This problem is still open. There exists an algorithm for finding a maximal fill in (not maximum), it's time complexity is $O(mn)$ where $m = |E|$, $n = |V|$ (Ohtsuki, 1976).
- **Approximating the fill in**: Find in polynomial time a fill-in set (i.e. a set F' such that $(V, E+F')$ is chordal) such that if F is an optimal fill in, $[\frac{|E|+|F'|}{|E|+|F|}] \leq f(n)$. The best f(n) known today is $O(\sqrt{k} log^4 n)$ for graph with max degree k (Klein et al., 1990). Can one get better approximation, or prove lower bounds on the achievable approximation, assuming $P \neq NP$?
- **parametric fill-in problem:** How fast can we solve the fill in problem for fixed k? Simple enumeration of all possible sets of size $\leq$ takes $O(n^{2k}m)$. Two better algorithms are known (Kaplan,Shamir,Tarjan 94) of complexity $O(c_k m)$ and $O(k^5 mn + C_k)$. Both C_k and c_k are, of course, exponential in k.

Chapter 6

Lecture 6

Advanced Topics in Graph Algorithms

Spring Semester, 1994

Lecturer : Ron Shamir
Scribe : Rotem Erlich
Lecture 6 : May 26

6.1 Some Optimization Algorithms on Triangulated Graphs

6.1.1 Computing the chromatic number and all maximal cliques

Theorem 6.1 (Fulkerson and Gross 65) *Let $G = (V, E)$ be a triangulated graph, and let σ be a perfect elimination order for G. Every maximal clique in G is of the form: $X_u \cup \{u\}$ where $X_u = \{x \in Adj(u) | \sigma^{-1}(u) < \sigma^{-1}(x)\}$.*

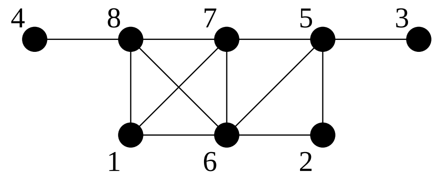


Figure 6.1: An example for maximal cliques in a graph

Example: In the chordal graph in Figure 6.1, vertices are numbered according to p.e.o and we have: $X_1 \cup \{1\} = \{1, 6, 7, 8\}$, $X_2 \cup \{2\} = \{2, 5, 6\}, X_3 \cup \{3\} = \{3, 5\}, X_4 \cup \{4\} =$

$\{4,8\}, X_5 \cup \{5\} = \{5,6,7\}$ - maximal cliques, and: $X_6 \cup \{6\} = \{6,7,8\}, X_7 \cup \{7\} = \{7,8\}$ - non maximal cliques.

Proof: Let C be an arbitrary maximal clique in G, and let v be the first vertex according to the order σ which is contained in C. Let $w \in G, \sigma^{-1}(w) > \sigma^{-1}(v)$. If $w \notin X_v \cup \{v\}$, then $wv \notin E$, therefore $w \notin C$, so we get: $C \subseteq X_v \cup \{v\}$.

On the other hand, since G is triangulated $X_v \cup \{v\}$ is a clique, and because C is maximal we get: $C = X_v \cup \{v\}$. ∎

Corollary 6.2 *A triangulated graph on n vertices has at most n maximal cliques, with equality if and only if the graph has no edges.*

Proof: Every maximal clique is characterized uniquely by the first vertex in σ which is contained in it. ∎

From Theorem 6.1, a naive algorithm for calculating the maximum clique in a triangulated graph is as follows:

- Find $\{v\} \cup X_v$ for each v.
- Filter out the non-maximal cliques.

A more efficient algorithm is based on the algorithm TEST for testing whether a given vertices permutation σ is a PEO as shown in lecture no. 4. In that algorithm X_v is calculated for each v.

Theorem 6.3 *When algorithm TEST operates on a chordal graph and a p.e.o. σ, $\{u\} \cup X_u$ is not a maximal clique iff X_u is concatenated to $A(u)$.*

Proof: Reminder: In the phase of eliminating v (phase $\sigma^{-1}(v) = i$ of TEST), $X_v - \{u\}$ is concatenated to A(u) if u is the first neighbor of v after it in σ.

($\Longrightarrow$) If $X_u \cup \{u\}$ is not a maximal clique then there exists a vertex w, $\sigma^{-1}(w) < \sigma^{-1}(u)$ such that $\{u\} \cup X_u \subset \{w\} \cup X_w$. Let us take v such that $v = \sigma(\max\{\sigma^{-1}(w) | \sigma^{-1}(w) < \sigma^{-1}(u)\})$.

In phase $\sigma^{-1}(v)$, $X_u = X_v - \{u\}$ is concatenated to A(u).

($\Longleftarrow$) Assume that at phase $\sigma^{-1}(v)$ $X_u = X_v - \{u\}$ is concatenated to A(u). Then $X_v = X_u \cup \{u\}$. So $X_v \cup \{v\}$ is a clique and $X_u \cup \{u\} \subset X_v \cup \{v\}$, therefore $X_u \cup \{u\}$ is not maximal. ∎

See the modified algorithm in figure 6.2.

Theorem 6.4 *The above algorithm correctly calculates the chromatic number and all maximal cliques of a triangulated graph $G = (V, E)$ in $O(|V| + |E|)$ time.*

The proof is similar to that of the algorithm in lecture no. 4.

1. $\chi \leftarrow 1$
2. for all vertices v do $S(v) \leftarrow 0$
3. for $i \leftarrow 1$ to n do
   begin
4. $v \leftarrow \sigma(i)$.
5. $X \leftarrow \{x \in \text{Adj}(v) | \sigma^{-1}(v) < \sigma^{-1}(x)\}$
6.   if $\text{Adj}(v) = \emptyset$ then print $\{v\}$.
7.   if $X = \emptyset$ then goto 13
8. $u \leftarrow \sigma(\min\{\sigma^{-1}(x) | x \in X\})$
9. $S(u) \leftarrow \max\{S(u), |X| - 1\}$
10.   if $S(v) < |X|$ then do
       begin
11.   print $\{v\} \cup X$
12. $\chi = \max\{\chi, 1 + |X|\}$
       end
13. end
14. print "The chromatic number is ",χ.

Figure 6.2: Chromatic Number Calculation

6.1.2 Finding α and k

Next we tackle the problem of finding the stability number $\alpha(G)$ of a triangulated graph G. Better yet, since G is perfect, the clique cover size $k(G) = \alpha(G)$.

Let σ be a perfect elimination scheme for $G = (V, E)$. We define inductively a sequence of vertices $y_1, y_2, \ldots, y_t$ in the following manner:

(1) $y_1 = \sigma(1)$

(2) $y_i = \sigma(\min\{\sigma^{-1}(x) | \sigma^{-1}(x) > \sigma^{-1}(y_{i-1})$ and $x \notin Xy_1 \cup Xy_2 \cup \ldots \cup Xy_{i-1}\})$

y_i is the first vertex in σ which appears after y_{i-1} in σ, and which is not in $Xy_1 \cup Xy_2 \cup \cdots \cup Xy_{i-1}$. All vertices following y_t are in $Xy_1 \cup Xy_2 \cup \cdots \cup Xy_t$.

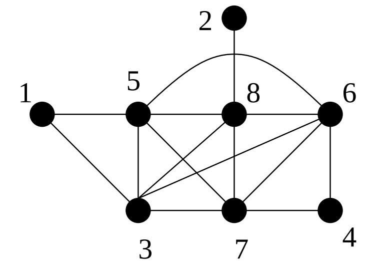


Figure 6.3: An example for finding maximal stable set

For example, in the graph in Figure 6.3:

$y_1 = 1, \ Xy_1 = \{3, 5\}$

$y_2 = 2, \ Xy_2 = \{8\}$

$y_3 = 4, \ Xy_3 = \{6, 7\}$

Here $\{y_1, y_2, y_3\}$ is a maximum stable set, and:

$y_1 \cup Xy_1 = \{1, 3, 5\}, \ y_2 \cup Xy_2 = \{2, 8\}, \ y_3 \cup Xy_3 = \{4, 6, 7\}$ are a minimum clique cover.

Theorem 6.5 *(Gavril, 1972) The set $\{y_1, y_2, \ldots, y_t\}$ is a maximum stable set of G, and the collection of sets $Y_i = \{y_i\} \cup Xy_i, \ i = 1, 2, \ldots, t$ comprises a minimum clique cover of G.*

Proof: The set $y_1, y_2, \ldots, y_t$ is stable since if $y_i y_j \in E$ for $i < j$, then $y_j \in Xy_i$, which cannot be. Thus $\alpha(G) \geq t$.

On the other hand, since $\{Y_1, Y_2, \ldots, Y_t\}$ is a set of cliques which together cover G, we get $k(G) \leq t$.

Since $\alpha(G) \leq k(G)$ we get: $\alpha(G) = k(G) = t$, and we have produced the desired maximum stable set and minimum clique cover. ■

The method above, with a slight modification in the algorithm of section 6.1.1, finds a maximum stable set and a minimum clique cover in time complexity of $O(|V| + |E|)$.

6.2 Comparability Graphs

6.2.1 Some Definitions

An undirected graph $G = (V, E)$ is a *comparability graph* if there exists an orientation (V, F) of G satisfying

$F \cap F^{-1} = \emptyset$,

$F + F^{-1} = E$,

$F^2 \subseteq F$, where $F^2 = \{ac | ab, bc \in F$ for some vertex $b\}$

F is also called a *transitive orientation* of G, and G is also called *transitively orientable* (TRO).

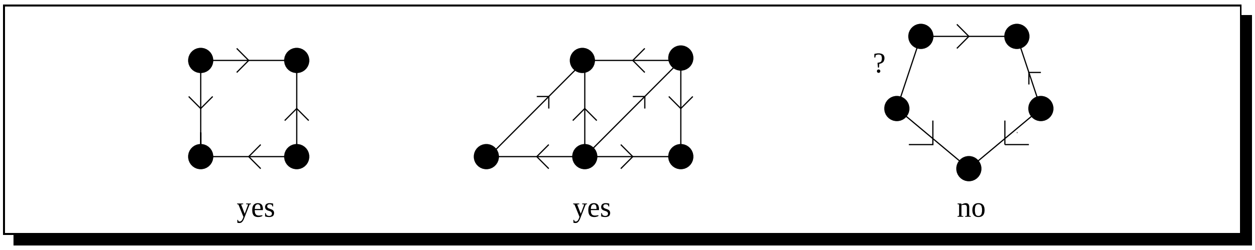


Figure 6.4: some comparability graphs

The relation F is a strict partial ordering (irreflexive, transitive). Comparability graphs are also known as *partially orderable* graphs.

Define the binary relation Γ on the edges of an undirected graph $G = (V, E)$ as follows: $ab\Gamma a'b' \iff a = a'$ and $bb' \notin E$ or $b = b'$ and $aa' \notin E$.

We say that *ab directly forces a'b'* whenever $ab\Gamma a'b'$.

The reflexive, transitive closure of Γ is Γ^*. It is easily shown that Γ^* is an equivalence relation on E, and hence partitions E into what we shall call the *implication classes* of G.

Thus edges ab and cd are in the same implication class iff there exists a sequence of edges $ab = a_0b_0\Gamma a_1b_1\Gamma \ldots \Gamma a_kb_k = cd$, $k \geq 0$. Such a sequence is called a Γ-*chain* from ab to cd, and we say that ab (eventually) *forces cd* whenever $ab\Gamma^*cd$.

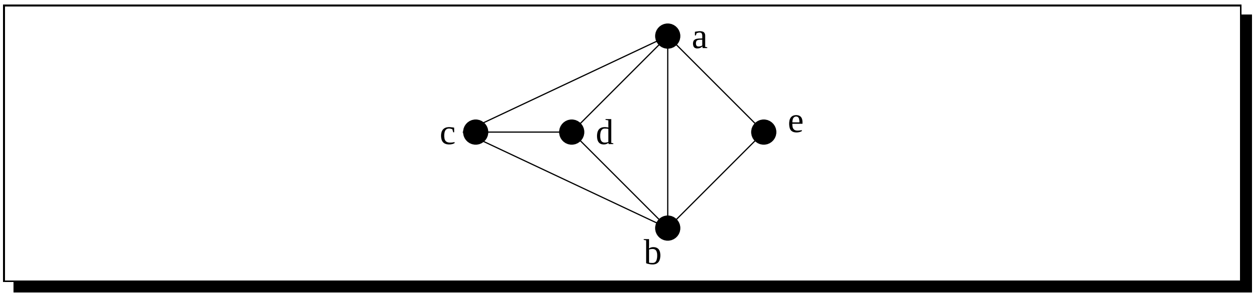


Figure 6.5: An example for implication classes

For example, the 8 implication classes in the graph of figure 6.5 are: $A_1 = \{ab\}$, $A_2 = \{cd\}$, $A_3 = \{ac, ad, ae\}$, $A_4 = \{bc, bd, be\}$, $A_1^{-1} = \{ba\}$, $A_2^{-1} = \{dc\}$, $A_3^{-1} = \{ca, da, ea\}$, $A_4^{-1} = \{cb, db, eb\}$.

Notice that $ba\Gamma dc \iff ab\Gamma cd$ and $ba\Gamma^* dc \iff ab\Gamma^* cd$, by definition. Notice also that $ab\Gamma ab$ (by our convention that G contains no self-loops), but *not* $ab\Gamma ba$ necessarily.

Let $I(G)$ denote the collection of implication classes of G. We define $\hat{I}(G) = \{\hat{A} | A \in I(G)\}$, where $\hat{A} = A \cup A^{-1}$ (the symmetric closure of $\hat{A}$). The members of $\hat{I}(G)$ are called the *color classes* of G.

6.2.2 Implication Classes

Theorem 6.6 *Let A be an implication class of an undirected graph G. If G has a transitive orientation F, then either $F \cap \hat{A} = A$ or $F \cap \hat{A} = A^{-1}$ and, in either case, $A \cap \hat{A} = \emptyset$.*

Proof: We defined Γ in order to capture the fact that, for any transitive orientation F of G, if $ab\Gamma a'b'$ and $ab \in F$, then $a'b' \in F$. Applying this property repeatedly, we obtain $F \cap A = \emptyset$ or $A \subseteq F$.

If $F \cap A = \emptyset \Rightarrow A \subseteq F^{-1} \Rightarrow A^{-1} \subseteq F \Rightarrow F \cap \hat{A} = A^{-1}$

If $A \subseteq F \Rightarrow A^{-1} \subseteq F^{-1} \Rightarrow F \cap A^{-1} = \emptyset$ (since $F \cap F^{-1} = \emptyset$) $\Rightarrow F \cap \hat{A} = A$.

In both cases $A \cap A^{-1} = \emptyset$ (since $F \cap F^{-1} = \emptyset$). ■

We shall prove later that the converse of Theorem 6.6 is also valid, namely if $A \cap A^{-1} = \emptyset$ for every implication class A, then G has a transitive orientation.

Note that the existence of an arbitrary union of implication classes $F = \cup_i A_i$ satisfying $F \cap F^{-1} = \emptyset$ and $F + F^{-1} = E$ does not necessarily mean that F is a transitive orientation of G. For example, in a simple triangle abc, $\{ab\}$, $\{bc\}$ and $\{ca\}$ are implication classes satisfying the condition but the resulting orientation is cyclic.

Lemma 6.7 *If $ab\Gamma^* cd$, then there exists a Γ-chain from ab to cd of the form*

$$ab = a_0 b_0 \Gamma a_1 b_0 \Gamma a_1 b_1 \Gamma a_2 b_1 \dots \Gamma a_k b_k = cd$$

Proof: In the Γ-chain in the lemma the indices change for the a_i-s in the odd "Γ"-s, and for the b_i-s in the even ones. Let us look at any Γ-chain between ab and cd. If in that chain there is a group of 3 or more sequential edges with the same index changed - in particular in that group the first and last edges directly force each other. We shall take for "our" chain those first and last edges - so we got only 2 edges in a row with the same index changed. We shall do that for every such group of 3 or more edges in a row, and will get the desired Γ-chain. ■

A Γ-chain satisfying the lemma is called a *canonical Γ-chain*.

6.2.3 The Triangle Lemma

Lemma 6.8 (The Triangle Lemma) *Let A, B and C be implication classes of an undirected graph $G = (V, E)$ with $A \neq B$ and $A \neq C^{-1}$ and having edges $ab \in C$, $ac \in B$ and $bc \in A$*

- *(i) If $b'c' \in A$ then $ab' \in C$ and $ac' \in B$*
- *(ii) If $b'c' \in A$ and $a'b' \in C$, then $a'c' \in B$*
- *(iii) No edge in A touches the vertex a*

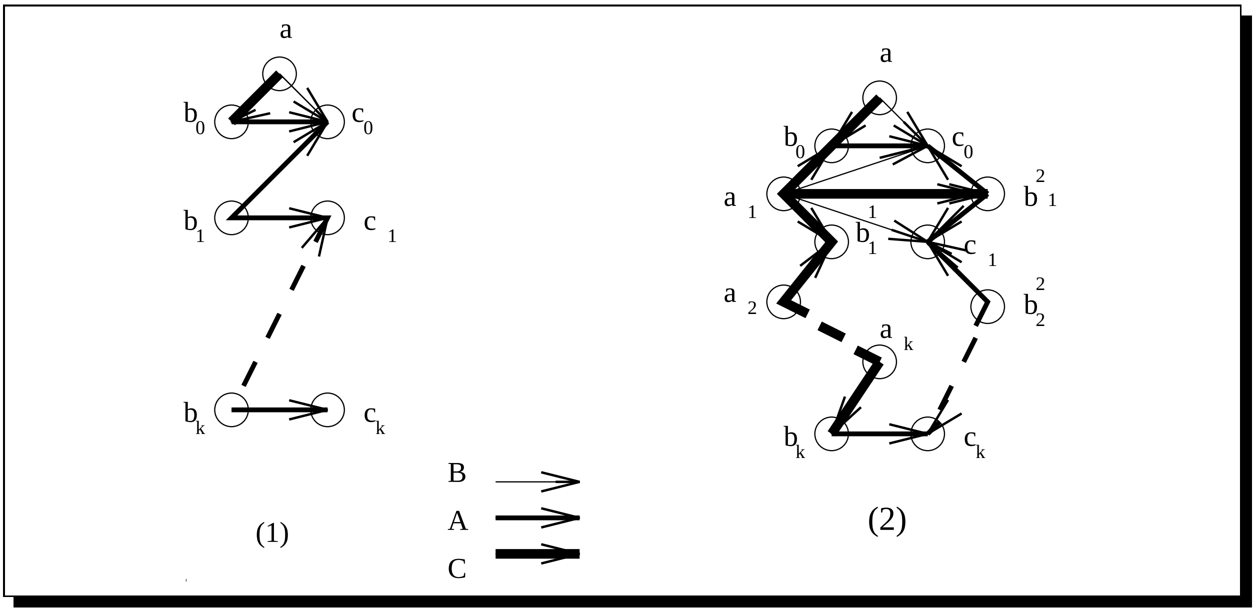


Figure 6.6: The triangle lemma

Proof: If $b'c' \in A$, by Lemma 6.7 there exists a canonical Γ-chain

$$bc = b_0c_0\Gamma b_1c_0\Gamma b_1c_1 \ldots \Gamma b_kc_k = b'c'$$

(see Figure 6.6(1)). Let $a = a_0$.

$a_0c_0 \in B$ doesn't force $b_1c_0 \in A \Rightarrow a_0b_1 \in E$.

$a_0b_0 \in C, b_0b_1 \notin E$ (since $b_0c_0\Gamma b_1c_0$) $\Rightarrow a_0b_1 \in C$.

$a_0b_1 \in C, b_1c_1 \in A \Rightarrow$ $a_0c_1 \in E$ (otherwise $b_1a_0\Gamma b_1c_1$, contradiction to $C^{-1} \neq A$).

$c_0c_1 \notin E, a_0c_0 \in B \Rightarrow a_0c_1 \in B$. By induction on the same argument we get: $a_0b_k \in C, a_0c_k \in B$. This proves (i). In particular (i)$\Rightarrow$(iii).

Next, assume $b'c' \in A, a'b' \in C$. Consider a new canonical Γ-chain

$$ab = a_0b_0\Gamma a_1b_0\Gamma a_1b_1\Gamma \ldots a_lb_l = a'b'$$

(see Figure 6.6(2)). If $ca_1 \notin E$ then $ba_1\Gamma bc$ but $C^{-1} \neq A$ therefore $ca_1 \in E$.

$aa_1 \notin E, ac \in B \Rightarrow a_1c \in B$.

The triangle a_1bc has the same characteristics as the original triangle abc. By induction we get that the triangle $a_lbc = a'bc$ has also the same characteristics. From (i) (with: a' instead of a) we get $a'c' \in B$. ■

Theorem 6.9 *Let A be an implication class of an undirected graph $G = (V, E)$. Exactly one of the following alternatives holds:*

- *(i) $A = \hat{A} = A^{-1}$*
- *(ii) $A \cap A^{-1} = \emptyset, A$ and A^{-1} are transitive orientations of $\hat{A}$.*

Proof: (i) Assume $A \cap A^{-1} \neq \emptyset$. Let $ab \in A \cap A^{-1}$. So $ab\Gamma^*ba$. According to the definition of Γ^* for any $cd \in A, cd\Gamma^*ab$ iff $dc\Gamma^*ba$. Since Γ^* is an equivalence relation and $ab\Gamma^*ba$, $cd\Gamma^*dc$ and therefore $dc \in A$. Thus $A = \hat{A}$.

(ii) Assume $A \cap A^{-1} = \emptyset$ and let $ab, bc \in A$. We shall now prove $ac \in A$. Now $ac \notin E \Rightarrow ab\Gamma cb \Rightarrow cb \in A \Rightarrow bc \in A^{-1}$, a contradiction. Thus $ac \in E$. Let B be the implication class of G containing ac, and suppose $A \neq B$.

Since $A \neq A^{-1}$ and $ab \in A$, the triangle lemma (iii) on triangle abc (with $bc \in A, ac \in B, ab \in C = A$) gives us that there is no edge in A touching a.

But $ab \in C = A$ touches a. A contradiction.

Therefore $ac \in A \Rightarrow A$ is transitive $\Rightarrow A^{-1}$ is transitive.

Finally, A is an implication class of $\hat{A}$, so by Theorem 6.6 A and A^{-1} are the only transitive orientations of $\hat{A}$. ■

Corollary 6.10 *Each color class of an undirected graph G either has exactly two transitive orientations, one being the reversal of the other, or has no transitive orientation. If in G there is a color class having no transitive orientation, then G is not a comparability graph.*

6.2.4 Decomposition of Graphs

Let H_0 be a graph with n vertices $v_1, v_2, \ldots, v_n$ and let $H_1, H_2, \ldots, H_n$ be n disjoint graphs. The *composition graph* $H = H_0[H_1, H_2, \ldots, H_n]$ is formed as follows:

For all $1 \leq i, j \leq n$, replace vertex v_i in H_0 with the graph H_i and make each vertex H_i adjacent to each vertex of H_j whenever v_i is adjacent to v_j in H_0.

Formally, for $H_i = (V_i, E_i)$ we define $H = (V, E)$ as follows:

$$V = \cup_{i\geq 1} V_i$$

$E = \cup_{i\geq 1} E_i \cup \{xy | x \in V_i,\ y \in V_j \text{ and } v_i v_j \in E_0\}$

We may also denote $E = E_0[E_1, E_2, \dots, E_n]$

H_0 is called the *outer factor* of H.

$H_i, i = 1, \dots, n$ are called the *inner factors* of H.

Theorem 6.11 *Let $G = G_0[G_1, G_2, \dots, G_n]$ where the G_i-s are disjoint undirected graphs. Then G is a comparability graph iff each G_i $(0 \leq i \leq n)$ is a comparability graph.*

Proof:

$(\Longrightarrow)$ It is obvious that if one of the G_i-s is not a TRO graph then G is not TRO.

$(\Longleftarrow)$ Let $F_0, F_1, \dots, F_n$ be transitive orientations of $G_0, G_1, \dots, G_n$, respectively. Let us show that $F = F_0[F_1, F_2, \dots, F_n]$ is transitive: Let $ab, bc \in F$: if $ab, bc \in F_i$ then $ac \in F_i$ by transitivity. If $ab \in F_i,\ bc \in F_j,\ j \neq i$ then we cannot have $i \neq 0,\ j \neq 0$ since inner factors are disjoint. If $ab \in F_i$ $bc \in F_0$ then $a \in G_i$ so, since every vertex in G_i is connected to every vertices in the factor in which c is contained in, then $ac \in F_0$. If $ab \in F_0$ $bc \in F_i$ then $c \in G_i$ so $ac \in F_0$,]

and in both cases we get transitivity. ∎

A graph is called *decomposable* if it can be expressed as a non-trivial composition of some of its induced subgraphs.

(For each graph there is the trivial composition: $G = k_1[G]$.)

Formally, $G = (V, E)$ is decomposable if there exists a partition $V = V_1 + V_2 + \cdots + V_r$ of the vertices into nonempty pairwise disjoint subsets with $1 < r < |V|$ such that $G = G_R[G_{V1}, G_{V2}, \dots, G_{Vr}]$ for any set of representatives $R = \{x_1, x_2, \dots, x_r\}, x_i \in V_i$.

Such a partition is said to *induce a proper decomposition of* G.

Chapter 7

Lecture 7

Advanced Topics in Graph Algorithms
©Tel-Aviv University
Spring Semester, 1994

Lecturer	:	Ron Shamir
Scribe	:	Sariel Har-Peled
Lecture 7	:	June 2

7.1 Uniquely Partially Orderable Graphs

Definition Given a graph H_0 with n vertices labeled $\{v_1, \ldots, v_n\}$ and n disjoint graphs $H_1, \ldots, H_n$, we denote by $H = H_0\Big[H_1, H_2, \ldots, H_n\Big]$ the *composition* graph, resulting from replacing each vertex in H_0, $v_i, 1 \leq i \leq n$ by the graph H_i, and replacing all edges $ij \in E(H_0)$ by the edges connecting all the vertices of H_i with the vertices of H_j. That is $V(H) = \cup_{i=1}^{n} V_i$ and $E(H) = \cup_{i=1}^{n} E_i \cup \{(v,u) | v \in V_i, u \in V_j \ \textit{and} \ (i,j) \in E_0\}$.

The graph H_0 is called the *outer factor* of H, and the graphs $H_1, \ldots, H_n$ are called the *inner factors* of H.

Example 7.1 *See figure 7.1 for an undirected example of a composition graph. See figure 7.2 for a directed example of a composition graph.*

Theorem 7.2 *Given $n+1$ undirected graphs $G_0, \ldots, G_n$, let $G = G_0\Big[G_1, \ldots, G_n\Big]$ denote their composition graph. Then, G is a comparability graph if and only if, all the graphs $G_i, 0 \leq i \leq n$ are comparability graphs.*

Proof: ($\Longleftarrow$) Clearly, if one of the $G_i, i = 0, \ldots, n$ is not a comparability graph, then G can not be a comparability graph.

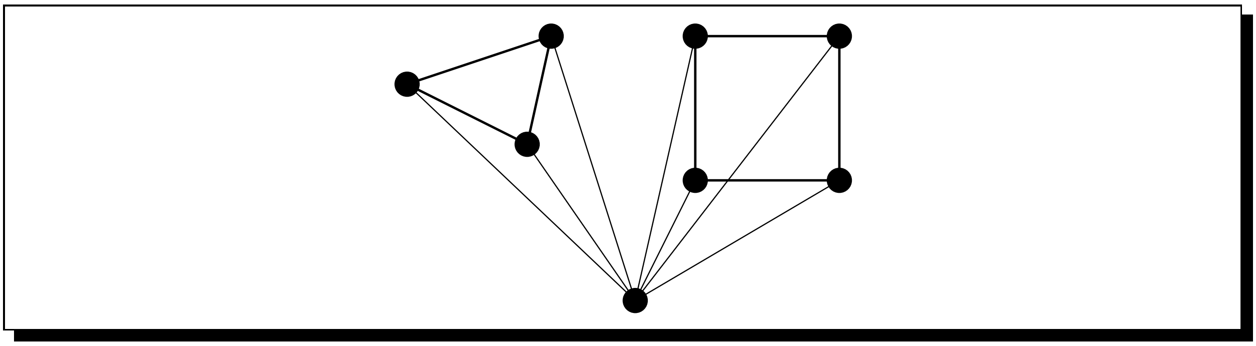


Figure 7.1: The composition graph of $P_3[K_3, K_1, C_4]$

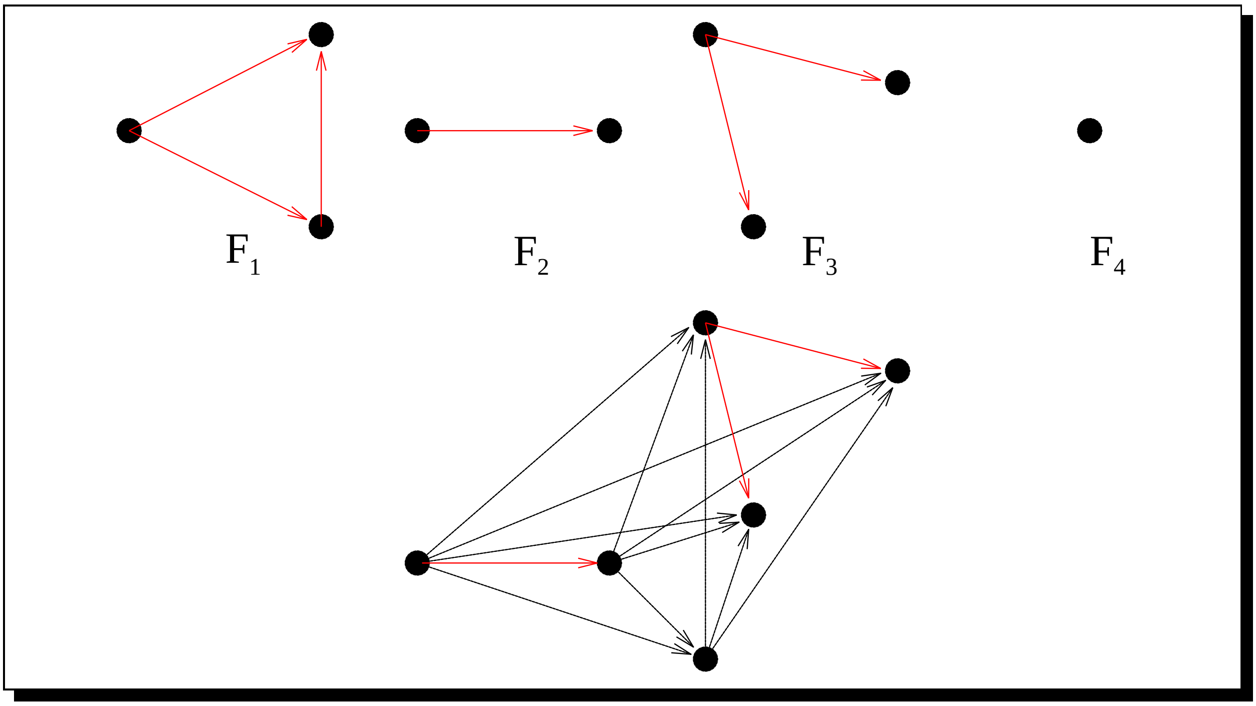


Figure 7.2: The composition graph of $F_1[F_2, F_3, F_4]$

($\Longrightarrow$) Let $F_0, \ldots, F_n$ be transitive orientations of the graphs $G_0, \ldots, G_n$, let $F = F_0\Big[F_1, \ldots, F_n\Big]$ be an orientation of G. Given two edges $a \to b, b \to c \in F$. We have to show that $a \to c \in F$. One of the following hold:

- The vertices $a, b, c \in V_i, 1 \leq i \leq n$. Since F_i is a transitive orientation it follows that $a \to c \in F$.
- The vertices $a, b \in V_i, c \in V_j, i \neq j$. Then, it follows that $i \to j \in F_0$, but then $a \to c \in F$.
- The vertices $a \in V_i$ $b, c \in V_j, i \neq j$, then $i \to j \in F_0$, thus $a \to c \in F$.
- The vertices $a \in V_i, b \in V_j, c \in V_k$, then $i \to j, j \to k \in F_0$, since F_0 is transitive it follows $i \to k \in F_0$, thus $a \to c \in F$.

Thus F is a transitive orientation of G. ■

Definition A graph $G = (V, E), V = \{v_1, \ldots, v_n\}$ is *decomposable* if it can be presented as a non-trivial composition of induced subgraphs of G. The trivial decompositions of G are $G = K_1\Big[G\Big]$, $G = G\Big[\{v_1\}, \ldots, \{v_n\}\Big]$. A graph which has only trivial decompositions is called *prime.*

If there exists a partition of the vertex set of G: $V = V_1 + V_2 + \cdots + V_r$, $1 < r < |V|$, such that $G = G_R\Big[G_{V_1}, \ldots, G_{V_r}\Big]$, where R is a set of r representatives from $V_1, \ldots, V_r$, then we say the the partition *induces a proper decomposition* of G.

Example 7.3 *See figure 7.3 for an example of various decompositions of a graph.*

Theorem 7.4 *Let $G_0, \ldots, G_n$ be n disjoint, undirected graphs, and let $G = G_0[G_1, \ldots, G_n]$ be their composition. Let $\hat{A}$ be a color class of G, then exactly one of the following holds:*

- $\hat{A} \subseteq E_j$ *for a single value of* $1 \leq j \leq n$.
- $\hat{A} \cap E_j = \emptyset$ *for all* $1 \leq j \leq n$.

Proof: Any color-class in G is, by definition, a connected subgraph of G. Assume that $\hat{A} \cap E_j \neq \emptyset$ for a vertex $1 \leq j \leq n$. Look at an edge $ab \in \hat{A} \cap E_j$, and let $cd \in E(G)$ be an edge, such that $cd \Gamma ab$. There are three possibilities:

1. The edge $cd \in E_k, k \neq j$. This is not possible, because internal edges of distinct inner factors do not share a vertex.

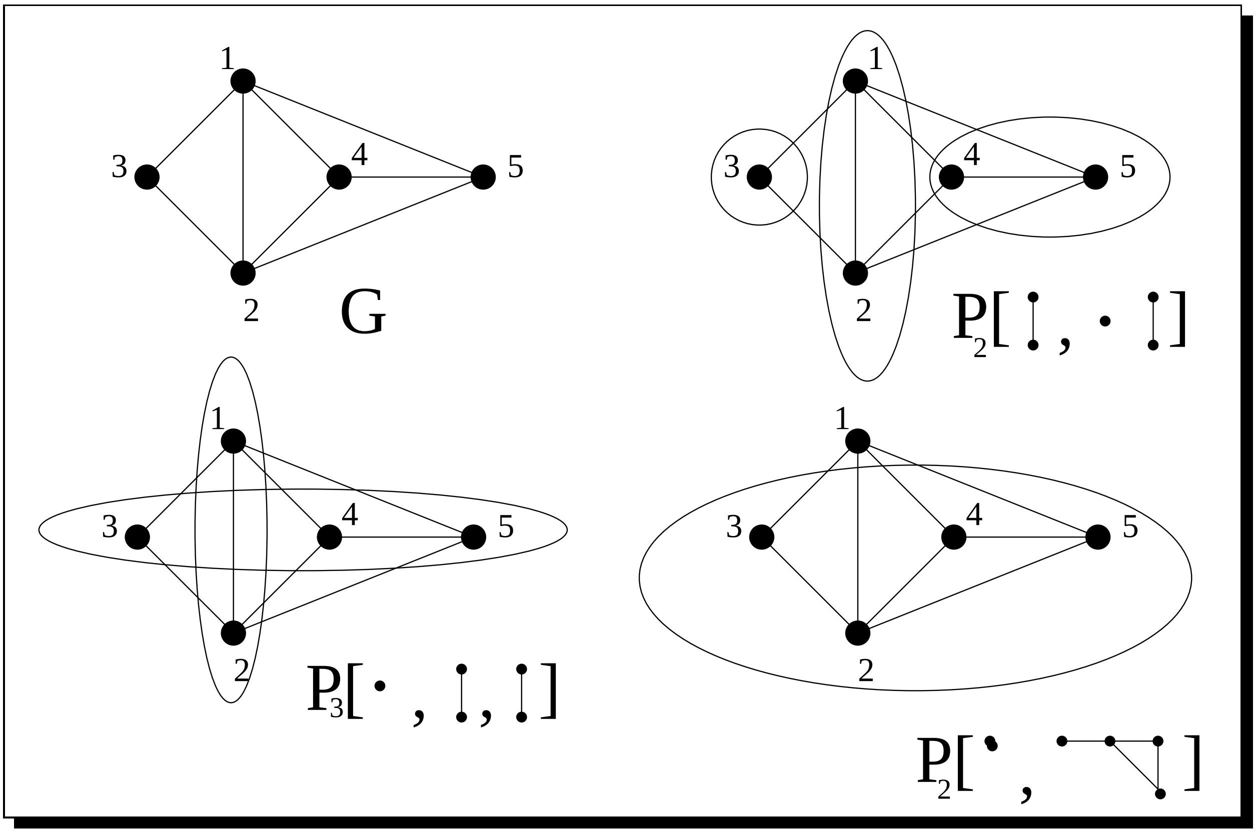


Figure 7.3: The decomposition of a graph

2. The edge cd is an external edge. This is not possible, because then the set $C = \{a, b\} \cup \{c, d\}$ having *three* vertices, induce a triangle in G, contradicting the fact that $cd\Gamma ab$.

Thus the only remaining possibility is that $cd \in E_j$, and by the connectivity of $\widehat{A}$, it follows that $\widehat{A} \subseteq E_j$. ∎

Definition Let $G = (V, E)$ be an undirected graph. A set $Y \subseteq V$ is called a *module* or a *partitive set* if for all $x \in V \setminus Y$ either $Y \cap Adj(x) = \emptyset$ or $Y \subseteq Adj(x)$. The set Y is a *non-trivial* module if $1 < |Y| < |V|$.

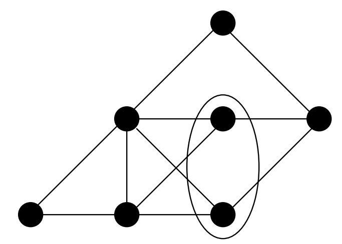


Figure 7.4: Module in a graph

Example 7.5 *See figure 7.4 for an example of a module in a graph.*

Lemma 7.6 *The graph G possess a non-trivial module $Y \subset V$, if and only if G is decomposable.*

Proof: ($\Longleftarrow$) Let $G = G_0[G_1, \ldots, G_n]$ is a decomposition of G, with $G_i = (V_i, E_i)$ a non-trivial inner factor. If $x \in G_j, j \neq i$ then either $V_i \subseteq adj(x)$ (if $ij \in E_0$) or $v_i \cap Adj(x) = \emptyset$ (if $ij \notin E_0$). Hence G_i is a module.

($\Longrightarrow$) If Y is a non-trivial module of G, then the division -

$$V = \{v_1\} + \{v_2\} + \cdots + \{v_k\} + Y$$

induce a proper decomposition of G. ∎

Corollary 7.7 *A set of vertices $Y \subset V$ in the graph $G = (V, E)$ with $1 < |Y| < |V|$ is a module, if and only if there exists a proper decomposition of $G = G_0[G_1, \ldots, G_n]$, such that, there is an inner factor of G, $G_i, 1 \leq i \leq n$, such that $G_i = G_Y$.*

Claim 7.8 *If Y is a set of vertices spanned by a color-class $\widehat{A}$ in the undirected graph $G = (V, E)$, then Y is a module.*

Proof: If $Y = V$ or $|Y| = 1$, then the set Y is a trivial module of G.

Else, let a be any vertex in $V \setminus Y$, and assume that $b \in Y \cap Adj(a)$, then $ab \in E \setminus \widehat{A}$. Since Y is defined by edges there exists a vertex $c \in Y$ such that $bc \in \widehat{A}$. The edge $ac \in E$ exists (else, $ba \in \widehat{A}$). The edge $ac \notin \widehat{A}$ (else, $a \in Y$).

Given any vertex $d \in Y$, let $de \in \widehat{A}$ be an edge connected to d. Now, using the triangle lemma, it follows that $ad \in E$, thus $Y \subseteq Adj(a)$. ■

Claim 7.9 *An undirected graph $G = (V, E)$ has at most one color-class which spans all of V.*

Proof: Assume that $\widehat{A}, \widehat{B}$ are two distinct color-classes that both span V. Given any vertex $b \in V$, there are edges $ab \in \widehat{B}, bc \in \widehat{A}$. Since $\widehat{A} \neq \widehat{B}$ the edge $ac \in E$.

Let $\widehat{C}$ be the color-class that contain the edge ac. If $\widehat{C} \neq \widehat{A}, \widehat{C} \neq \widehat{B}$, from the triangle lemma (iii), it follows that no edge in $\widehat{A}$ is incident to the vertex a. A contradiction.

Thus $\widehat{C} = \widehat{A} \neq \widehat{B}$, and by the same argument no edge from $\widehat{B}$ is incident to the vertex c. A contradiction. ■

Definition A comparability graph is *uniquely partially orderable* graph (abbreviated **UPO** graph), if it has only two transitive orientations, where each orientation is the reversal of the other.

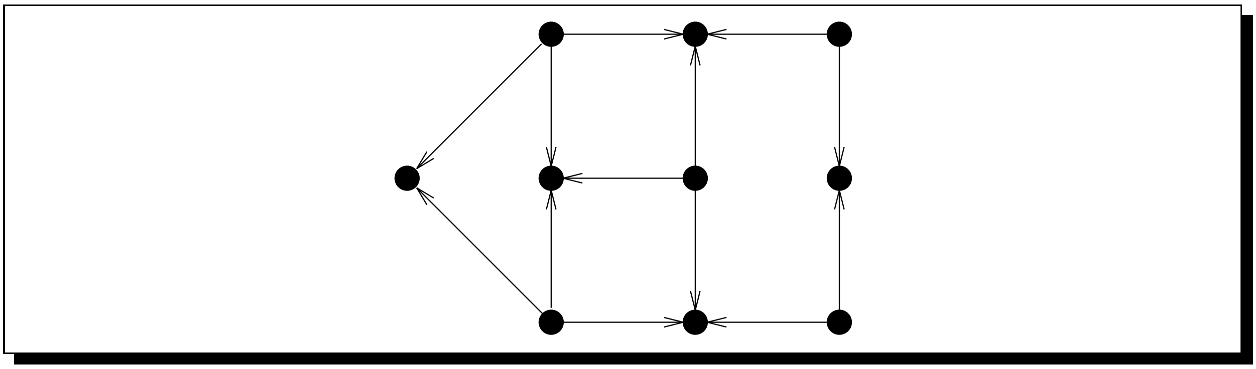


Figure 7.5: Uniquely partially orderable graph

Example 7.10 *See figure 7.5 for an example of a uniquely orderable graph*

Corollary 7.11 *A graph G is* **UPO iff** *G have a single color-class.*

Theorem 7.12 (Shevrin-Filippov '70) *Let G be a connected comparability graph. The following conditions are equivalent:*

1. *The graph G is* **UPO**.

2. *Every non-trivial module in G is an independent set.*

3. *In any proper decomposition of G, every inner-factor is an independent set (Explicitly, all the edges in G are external).*

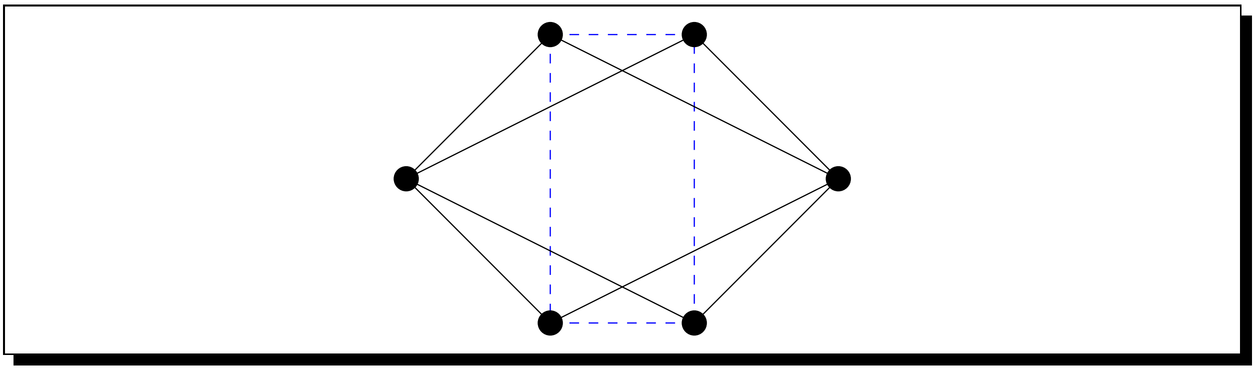


Figure 7.6: A graph with a color class that induce all vertices

Proof: (2) $\iff$ (3) Follow immediately from the natural equivalence between a module and a inner factor in a decomposition, as stated in corollary 7.7.

If there is a non-trivial module Y, which is not stable, there is a decomposition of G, with inner factor G_Y, which is not stable.

If G_Y is an inner factor in a given proper decomposition, then Y is a module, thus if G_Y is not stable, then the set Y is a non-stable module.

(1) $\Longrightarrow$ (3) If G is **UPO**, then in G there is a color-class, $\widehat{A}$, which contains all the edges of G. Thus by theorem 7.4, it follows that all the edges are external in any non-trivial decomposition of G (because no non-trivial inner factor can contain all the edges of $\widehat{A}$, thus such an inner factor, has a n empty intersection with $\widehat{A} = E(G)$).

(2) $\Longrightarrow$ (1) Assume, for the sake of contradiction, that G is not **UPO**. Then G has more than one color-classes (Since G is a comparability graph). From claim 7.9 it follows that there is a color-class which spans a proper subset Y of V. By claim 7.8 it follows that Y is a module of G. But Y is not a stable set (Y is defined as these vertices that are spanned by a color-class, which is connected). A contradiction. ∎

Corollary 7.13 *A prime comparability graph G is* **UPO**.

Proof: If G is not decomposable, then it must be connected. Condition (3) in theorem 7.12 holds trivially thus G is **UPO**. ∎

7.2 Characterizations and Recognition Algorithms

Definition For the undirected graph $G = (V, E)$, a partition of E into k sets: $E = \widehat{B_1} + \widehat{B_2} + \cdots + \widehat{B_k}$, is called a *G-decomposition* if B_i is an implication class of the graph $G = (V, \widehat{B_i} + \widehat{B_{i+1}} + \cdots + \widehat{B_k})$. See figure 7.7 for an example.

A sequence of edges $[x_1y_1, \ldots, x_ky_k]$ is a *decomposition scheme* for G, if there exists a G-decomposition $E = \widehat{B_1} + \widehat{B_2} + \cdots + \widehat{B_k}$, such that $x_iy_i \in \widehat{B_i}$, $i = 1, \ldots, k$.

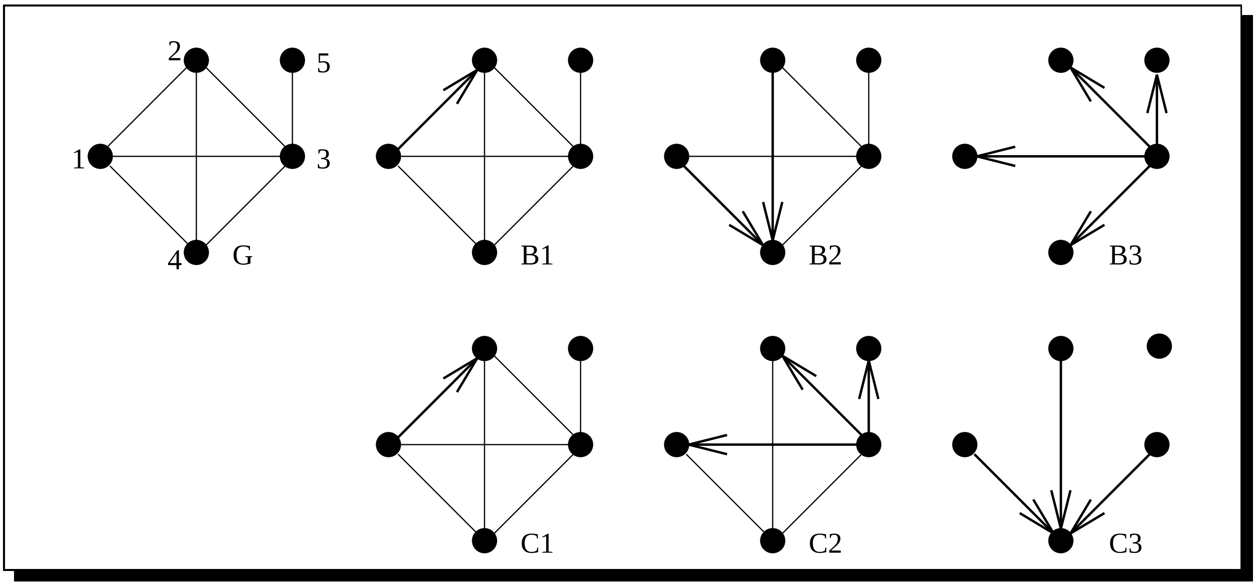


Figure 7.7: A graph G-decomposition

Example 7.14 *The following are examples for the decomposition schemes for figure 7.7:*

- *For the top G-decomposition:* $\left[12, \begin{matrix} 24 \\ 14 \end{matrix}, \begin{matrix} 35 \\ 13 \\ 23 \\ 43 \end{matrix} \right].$
- *For the bottom G-decomposition:* $\left[12, 13, 43 \right]$

Remark 7.15 *A G-decomposition of a graph can have several decomposition-schemes, but the decomposition scheme uniquely defines the G-decomposition. We will present algorithms to calculate both.*

See figure 7.8 for a description of the G-decomposition algorithm.

```
Algorithm: construct G-decomposition
    Input: A graph G = (V, E).
    Output: A G-DECOMPOSITION: A NUMBER k
            k SETS OF EDGES OF G: B̂_1, ..., B̂_k.
  begin
    Let E_1 ← E, i ← 1.

  Loop:
    Arbitrarily pick an edge e_i = x_i y_i ∈ E_i.

    Enumerate the implication class B_i of E_i containing x_i y_i.

    Let E_{i+1} ← E_i \ B̂_i.

    if E_{i+1} = ∅ then
      Let k ← i;
      stop.
    else
      Increase i by 1;
      go to Loop
    end if
  end    construct G-decomposition
```

Figure 7.8: Algorithm to construct G-decomposition

Theorem 7.16 (Golumbic 77) *Let A be an implication class of an directed graph $G = (V, E)$. Let D be an implication class in $E \setminus \widehat{A}$. One of the following holds:*

1. *D is an implication class in E, and A is an implication class in $E \setminus \widehat{D}$.*

2. *$D = B + C$, where C, B are implication classes in E, and G contain a triangle with edge taken from each implication class: A, B, C.*

Proof: In the graph remaining after the removal of $\widehat{A}$ several implication classes of G may had been united. Let D be an implication class in $E \setminus \widehat{A}$. Assume that D is a union of several implication classes of E, including B and C.

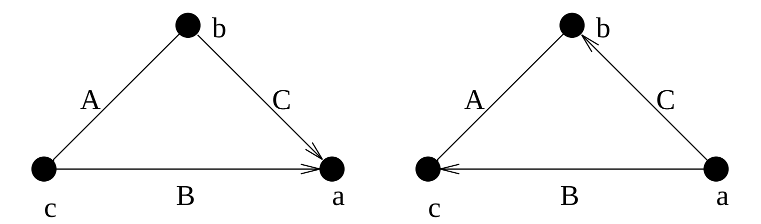


Figure 7.9: A triangle forcing the merge of two implication classes

Since B, C are two distinct implication classes in E, that were united in $E \setminus \widehat{A}$, there exist edges $ab, ac \in E \setminus \widehat{A}$ such that $ab \in C, ac \in B, bc \in \widehat{A}$ or $ba \in C, ca \in B, bc \in \widehat{A}$ (see figure 7.9). Assume without limiting generality that the former holds, since the second case is the inverse of the first case.

If $B = C^{-1}$ then $ba \in B$. It is known that $ac \in B$ but $bc \notin B$. Thus B is not transitive, and using theorem 6. it follows that $C^{-1} = \widehat{B}$, explicitly $B = C$. A contradiction.

Thus $\widehat{B} \cap \widehat{C} = \emptyset$ and thus the triangle $\triangle abc$ have three-colors.

Is it possible that D is a union of more than 2 implication classes? Assume that there is another implication class F, distinct from B, C, in $E \setminus \widehat{A}$ that is contained in D. From the above discussion, it follows that there is a three color triangle with $\widehat{A}, F, B$ as its colors (See figure 7.10). Now, using the triangle lemma it follows that triangles containing edges from both $\widehat{A}, \widehat{B}$, have their third edge from the same color-class. Thus $D = B + C$.

For the other case, suppose D is a single implication class in E. If A is not an implication class in $E \setminus \widehat{D}$, then there is a triangle with A and D as two of its edges. This triangle exist because where removing D from E, the implication class A was merged with another implication class. Let A_1 denote the implication class in $E - \widehat{D}$ that contains the third edge of the triangle (see figure 7.11).

But when removing A from E, this triangle implies the merge of the implication classes A_1, D into a single implication class. This contradicts the fact that D is an implication class in $E \setminus \widehat{A}$. Thus A is an implication class in $E \setminus \widehat{D}$. ∎

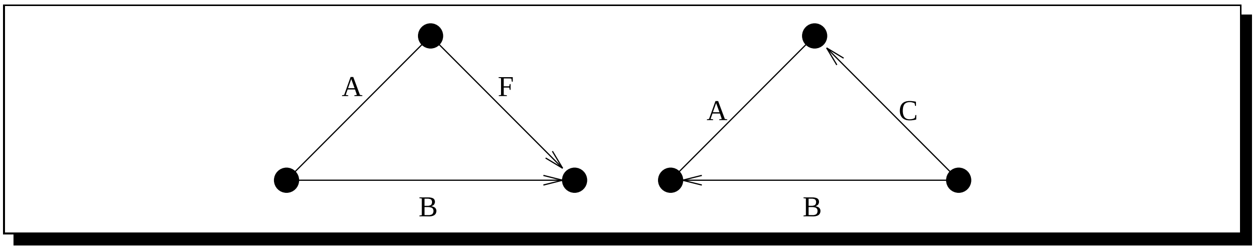


Figure 7.10: The classes $\widehat{F}, \widehat{C}$ must be equal.

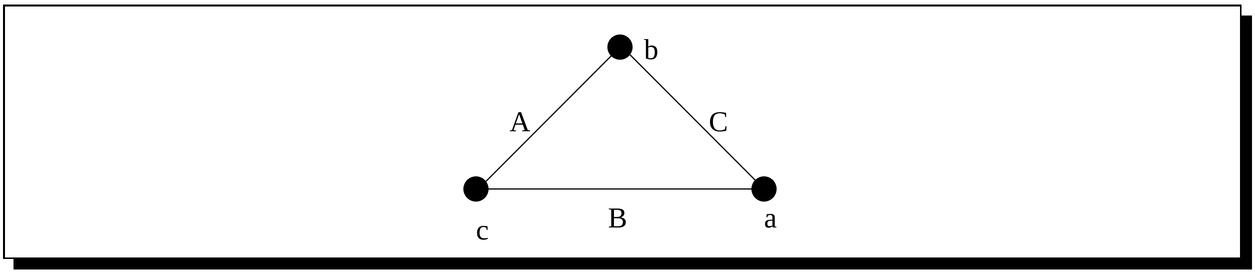


Figure 7.11: $\widehat{A} + \widehat{A_1}$ is an implication class in $E \setminus \widehat{D}$.

Corollary 7.17 *If A is an implication class, such that $A = \widehat{A}$, then there is no triangle with three colors such that $\widehat{A}$ is one of its colors.*

We leave this proof as an exercise. This claim implies, that for any triangle containing $\widehat{A}$, the two other edges in the triangle are colored by the same color.

Corollary 7.18 *If A is an implication of $G = (V, E)$ such that $A = \widehat{A}$ then all the other implication classes of G are implication classes in $E \setminus \widehat{A}$.*

Proof: By theorem 7.16 if $E \setminus \widehat{A}$ contains two implication classes from E that where merged, then these two class with $\widehat{A}$ induce a three colored triangle in G, which is impossible by Corollary 7.17. ■

Theorem 7.19 (characterizations of comparability graphs) (Gilmore and Hoffman 1964, Ghouilla-Houri 1962)

Let $G = (V, E)$ be an undirected graph with G-decomposition, $E = \widehat{B_1} + \cdots + \widehat{B_k}$. The following conditions are equivalent:

1. *G is a comparability graph.*

2. *For all implication classes A of E, $A \cap A^{-1} = \emptyset$.*

3. *$B_i \cap B_i^{-1} = \emptyset$ for $i = 1, \ldots, k$.*

4. *Any circuit $v_1v_2, v_2v_3, \ldots, v_qv_1$ in E (not necessarily simple), such that $v_{q-1}v_1 \notin E$, $v_qv_2 \notin E$ and $v_{i-1}v_{i+1} \notin E, i = 2, \ldots, q-1$ has* even *length,*

   Define a short chord *or a* triangular chord *in a cycle to be a chord connecting two vertices which are distance 2 along the cycle. Condition* 4 *is equivalent to requiring that every odd circuit in G contains a short chord.*

Proof: (1) $\Longrightarrow$ (2) Proved in theorem 6.6.

(2) $\Longrightarrow$ (3) By induction of k. For $k = 1$, B_1 is an implication class in G and it is given that $B_1 \cap B_1^{-1} = \emptyset$.

We assume the claim holds for all G-decomposition of graphs that include less than k sets. Specifically, the claim holds for $E \setminus \widehat{B_i}$. Let D by an implication class of $E \setminus \widehat{B_i}$. By theorem 7.16 two cases are possible:

- The class D is an implication class of E, then by the induction hypothesis it follows $D \cap D^{-1} = \emptyset$.
- $D = B + C$ where B, C are implication classes in E. Then, $D \cap D^{-1} = (B \cup C) \bigcap (B^{-1} \cup C^{-1}) = (B \cap B^{-1}) \cup (C \cap C^{-1}) = \emptyset$, because $B \cap C = \emptyset$ and $B \cap B^{-1} = \emptyset$ and $C \cap C^{-1} = \emptyset$ by the induction hypothesis.

(3) $\Longrightarrow$ (1) Let $E = \widehat{B_1} + \cdots + \widehat{B_k}$ a G-decomposition of E such that $B_i \cap B_{i-1} = \emptyset$. By theorem 6.9 it follows that B_1 is transitive. If $k = 1$ then we are done, else assume correctness to all G-decomposition of less than k sets. It follows that $F = B_2 + \cdots + B_k$ is a transitive orientation of $E \setminus \widehat{B_1}$. We have to prove that $B_1 + F$ is transitive orientation of G. Let $ab, bc \in B_1 + F$. If both edges belong to B_1 or to F, then the edge ac exists and must belong to the corresponding set by the fact that both B_1, F are transitive. Thus, assume that $ab \in B_1$ and $bc \in F$. Since $B_1 \cap B_1^{-1} = \emptyset$, if $ac \notin E$ then $ab\Gamma cb$ thus $cb \in B_1$ thus $bc \in \widehat{B_1}$ but $bc \in F$ by assumption, a contradiction. Thus $ac \in E$.

Assume that $ac \notin B_1 + F$, then $ca \in B_1 + F$, there are two possibilities :

- If $ca \in B_1$, since $ab \in B_1$ then $cb \in B_1$, a contradiction.
- If $ca \in F$, since $bc \in F$ then $ba \in F$, a contradiction.

Thus $ac \in B_1 + F$. In the same manner, if we assume that $ab \in F, bc \in B_1$ it follows that $ac \in B_1 + F$. Thus $B_1 + \cdots + B_k$ is a transitive orientation of E.

(1) $\Longleftrightarrow$ (4) Assume that G is not a comparability graph. By the fact that (1) $\Longleftrightarrow$ (2) it follows that there exists $v_1v_2 \in A \cap A^{-1} \neq \emptyset$. By the lemma ensuring the existence of a canonical implication chain, there is a canonical implication chain such that -

$$v_1v_2\Gamma v_3v_2\Gamma v_3v_4\Gamma v_5v_4 \cdots \Gamma v_qv_{q-1}\Gamma v_qv_{q+1} = v_2v_1.$$

The length of the chain is odd (because of its special structure), and there is no short chord between two vertices in cycle, because then $v_i v_{i+1} \not\Gamma v_{i+2} v_{i+1}$.

For the other direction, assume that there is an odd cycle in the graph with no short chords, then it holds -

$$v_1 v_2 \Gamma v_2 v_3 \Gamma v_3 v_4 \cdots \Gamma v_q v_{q-1} \Gamma v_q v_1 \Gamma v_2 v_1$$

is a canonical implication chain. Thus $v_1 v_2 \in A \cap A^{-1} \neq \emptyset$ and G is not a comparability graph. ∎

Definition Given an undirected graph $G = (V, E)$ we define a graph $G' = (V', E')$ in the following manner -

$$V' = \{(i,j),(j,i) | ij \in E\}$$

$$E' = \{(x,y) \leftrightarrow (y,z) | xz \notin E\}$$

(Note that by our convention $xx \notin E$ so $(x,y) \leftrightarrow (y,x)$ for every $xy \in E$.) An orientation of G' such that $(x \to y, y \to z)$ imply that $xz \in E$ is called *quasi-transitive* orientation of G.

Example 7.20 *See figure 7.12 for an example of a graph and the resulting quasi-transitive graph. The figure omits some the edges of the form (ij, ji).*

Theorem 7.21 (Ghouilla-Houri, 1962) *The following claims are equivalent:*

1. *The graph G is a comparability graph*
2. *The graph G have a quasi-transitive orientation.*
3. *The graph G' is bipartite.*

Proof: (1) $\Longrightarrow$ (2) Trivial, since G has a transitive orientation and this orientation is also quasi-transitive.

(2) $\Longrightarrow$ (1) Let F be a quasi transitive of G, and let C be an odd cycle. The cycle C must contain two adjacent edges in F: $y \to z, x \to y$ and then because of the quasi-transitive property $xz \in E$. Thus (1) follows from theorem 7.19 since we found a short chord.

(1) $\Longleftrightarrow$ (3) We will use GH characterization (Any odd cycle contains a short-chord). The vertices of a cycle in G' correspond to adjacent edges in G (with common vertex), that induce a cycle in G with no short chords. Thus G' is bipartite $\Longleftrightarrow$ G is a comparability graph. ∎

By the above theorem it is possible to check if a graph is a comparability graph, as follows: Construct the graph G', and check if it is bipartite. The size of G' is : $|V'| = 2|E|$, and -

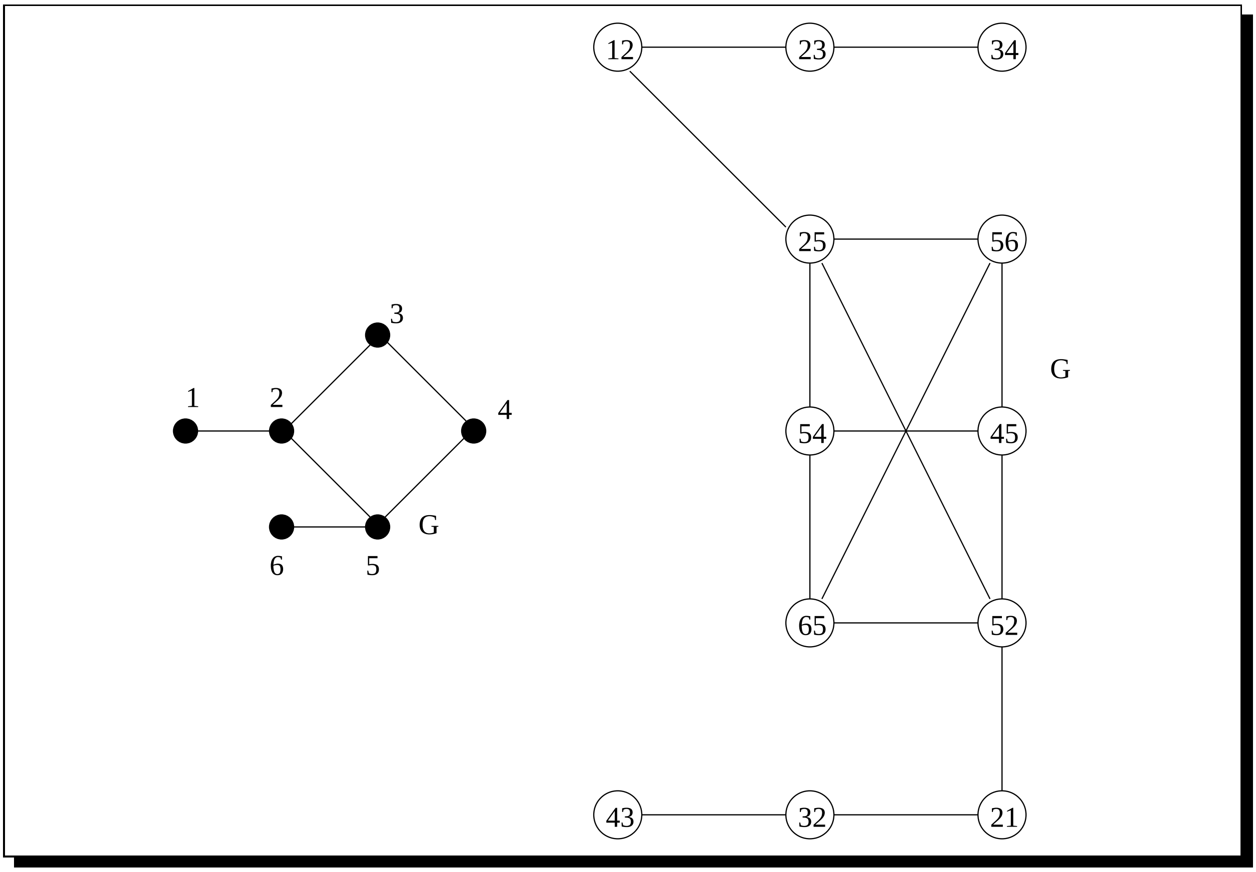


Figure 7.12: A graph G and its quasi-transitive graph G' (in G', edges $(x, y) \leftrightarrow (y, x)$ are not drawn)

$$|E'| = |E| + 2\sum_{v \in V} \left| \left\{ w, x \;\middle|\; wv \in E, vx \in E, wx \notin E \right\} \right| = O(|V|^3)$$

Thus the complexity of constructing the graph is $O(|V|^3)$. Check if G' is bipartite can be performed in linear time in the size of G', i.e. in $O(|V|^3)$.

One can obtain a transitive orientation of G by taking the orientation of edges whose vertices belong to one part of the bipartite graph G'. In particular, G is **UPO** *iff* G' is connected and bipartite. We shell give more efficient algorithms for recognizing comparability graph later in the course.

Chapter 8

Lecture 8

Advanced Topics in Graph Algorithms
©Tel-Aviv University
Spring Semester, 1994

Lecturer	:	Martin Golumbic
Scribe	:	Shimon Shahar
Lecture 8	:	June 5

8.1 Some interesting graph families characterized by intersection

8.1.1 Introduction

The topic of this lecture is intersection graphs. We shall focus mainly on F-graphs and tolerance graphs.

Definition Let S = $\{T_i | i \in I\}$ be a family of subsets of a set T. The *intersection graph* of S is $G = (S, E)$, where

- S = $\{T_i\}_j \in_I$
- E = {(i,j) $|T_i \cap T_j \neq \emptyset\}$.

It was proved (*Marczewski [1945]*) that if we allow S to be an arbitrary family, then we obtain all the undirected graphs. Usually we are interested in families of sets with specific topological properties. Moreover we should notice that some graphs were defined as intersection graphs (like line graphs), and others were defined by their properties, and only afterwards were proved to be an intersection graph (for example triangulated graphs, were originally studied as C_k-free graphs (k > 3) and only later characterized as intersection of subtrees).

8.1.2 Permutation graphs

Definition A *matching diagram* consists of n points, on each of two parallel lines, and n straight line segments matching the points. The intersection graph of the line segments is called a *permutation graph* (See Fig. 8.1). If the endpoints on the line are numbered 1,2,. . .n, then the endpoints on the other side of the line are numbered by a permutation on {1,2. . .,n} denoted σ.

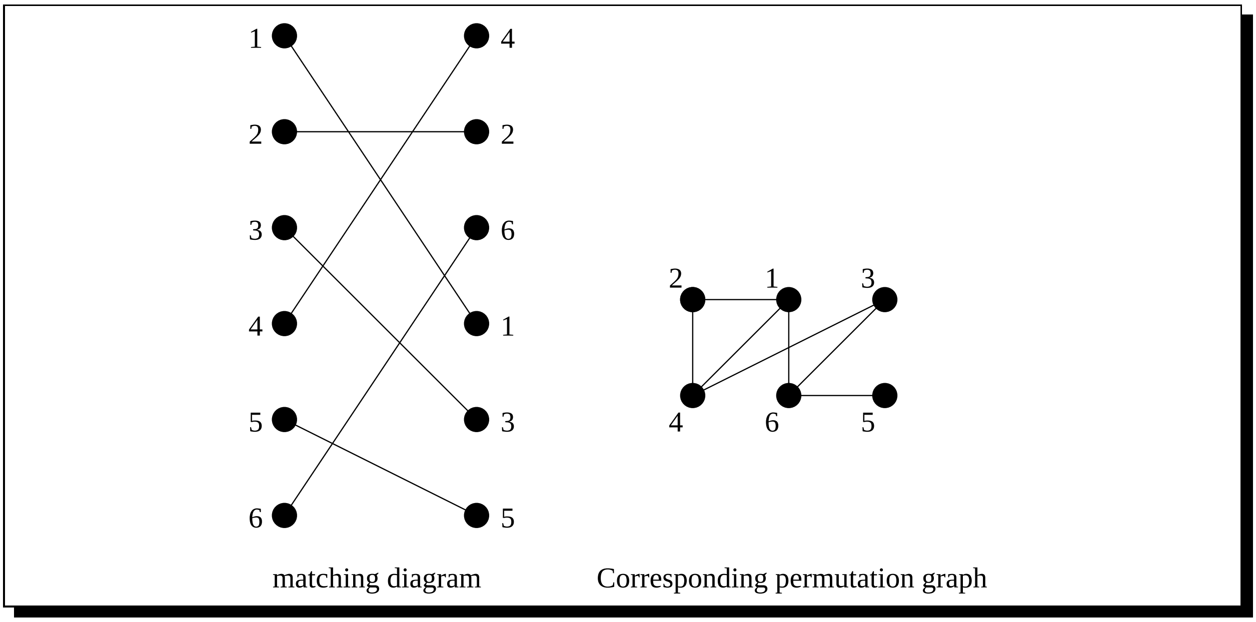


Figure 8.1: A permutation graph for the permutation [4,2,6,1,3,5]

Remark 8.1 *It is easy to build the permutation graph directly from the permutation.*
$\forall i, j$ *[$i < j$ and $\sigma(i) > \sigma(j)$]* $\Leftrightarrow (i, j) \in E$.

Theorem 8.2 *A permutation graph $G = (V, E)$ is transitively orientable.*

Proof: Let σ be the permutation of $\{1, \ldots, n\}$ generating $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$. For every $(i, j) \in E$, define F as follows :
$(i, j) \in F$ iff $i < j$.
We shall prove that $D = (V, F)$ is transitively orientable.
Let $(i, j) \in F, (j, k) \in F$, then by Remark 8.1, since $(i, j) \in E$, then $\sigma^{-1}(j) < \sigma^{-1}(i)$, and $\sigma^{-1}(k) < \sigma^{-1}(j)$. Hence $\sigma^{-1}(k) < \sigma^{-1}(i)$, and therefore $(i, k) \in F$.
which implies F is a transitive orientation (See fig. 8.2). ■

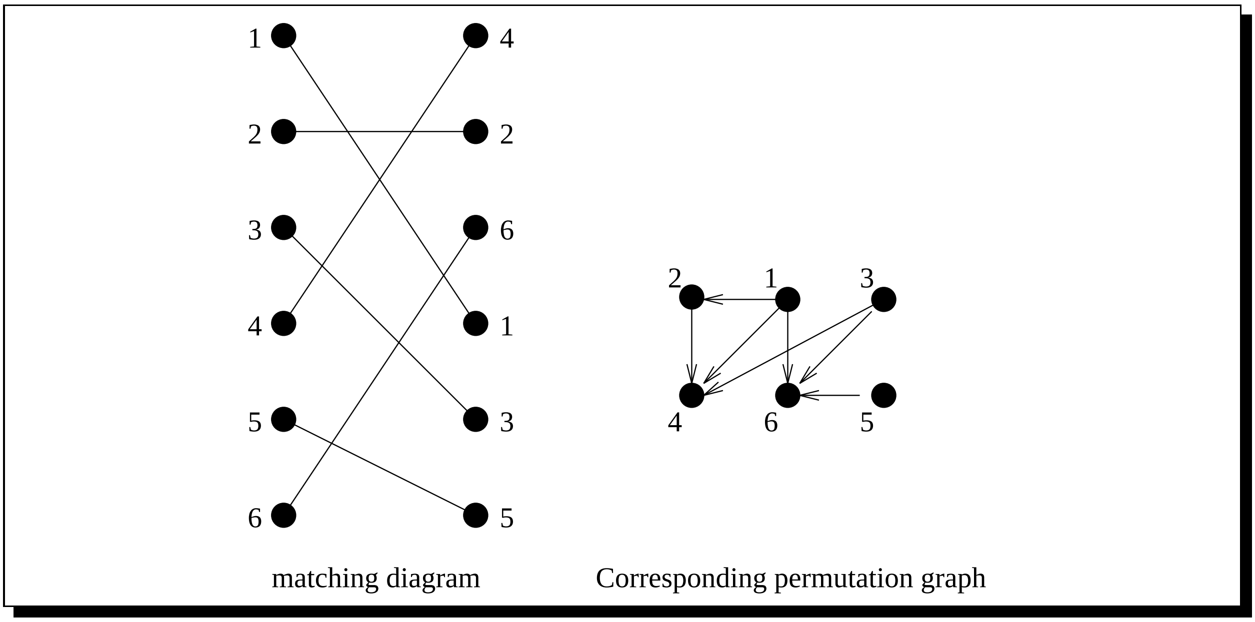


Figure 8.2: An orientation on a permutation graph

Definition If σ is a permutation, we say that $\bar{\sigma}$ is the opposite permutation, meaning that $\sigma(i) = \bar{\sigma}(n+1-i)$. For example σ = [5 6 7 2 3 8 1 9 4],$\bar{\sigma}$ = [4 9 1 8 3 2 7 6 5].

Theorem 8.3 *If G is a permutation graph generated by σ, then $\bar{G}$ is also a permutation graph, generated by $\bar{\sigma}$.*

Proof: We shall prove that the graph $H = (V, F)$ generated by $\bar{\sigma}$ is $\bar{G}$.

For every i,j$\in$V, $(i,j) \in E$ iff (i-j)$(\sigma^{-1}(i) - \sigma^{-1}(j)) < 0$. But since $\bar{\sigma}^{-1}(i) < \bar{\sigma}^{-1}(j)$ iff $\sigma^{-1}(i) > \sigma^{-1}(j)$ and (i,j)$\in$F iff (i-j)$(\bar{\sigma}^{-1}(i) - \bar{\sigma}^{-1}(j))$ >0 , we obtain that (i,j)$\in$E $\Leftrightarrow$ (i,j)$\notin$F, so H=$\bar{G}$. ∎

Theorem 8.4 *(Pnueli Lempel Even 1971) G is a permutation graph iff both G and $\bar{G}$ are transitively orientable.*

Proof:

$\Rightarrow$ Trivial from the previous two theorems.

$\Leftarrow$ Wasn't proved in class.

∎

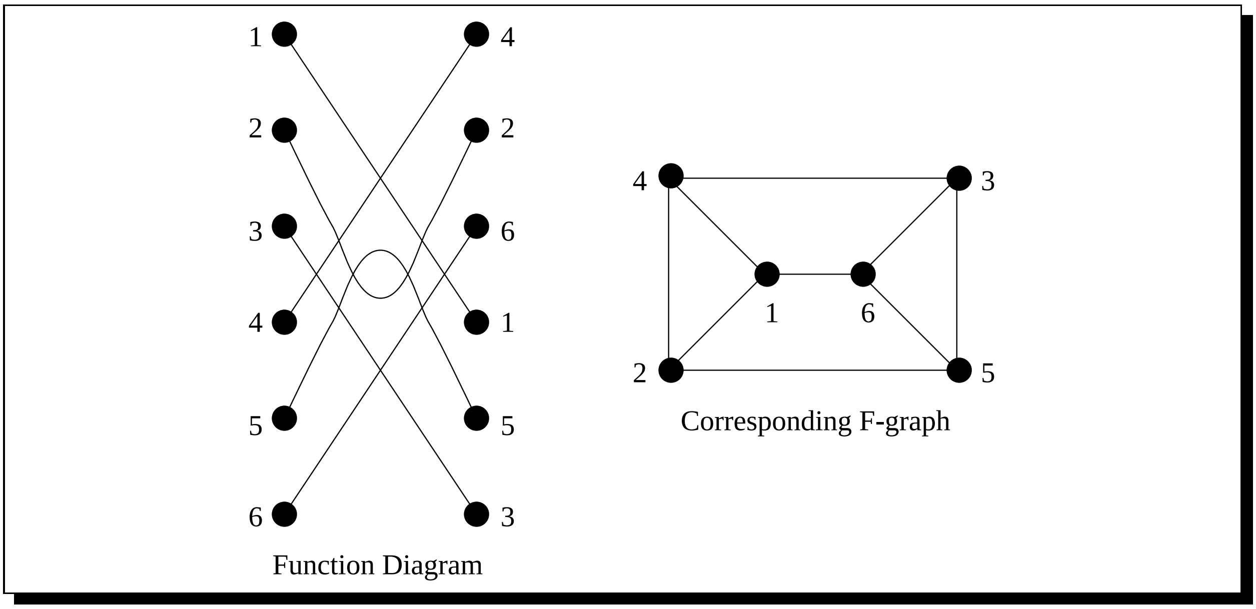


Figure 8.3: An F-graph

8.1.3 F-Graphs

Definition An *F-diagram* consists of n points, on each of two parallel lines, and n continuous function curves ending in the points. The intersection graph of the functions segments is called a function graph, or an *F-graph* (See Fig. 8.3). Note the requirement that each curve segment is a function (See Fig. 8.4 for negative example).

"The air controllers strike"

Given two collections of cities lying on two parallel lines, and a collection of flight paths between them, assign an altitude to each flight so that no crashes occur. It is easy to see that if we take the intersection graph of the paths, we obtain an F-graph (assuming that no flight is going backwards). We should assign different altitude only for those flights that intersect. Therefore we can consider each altitude to be a color, and we should solve the coloring problem for that F-graph (See Fig. 8.5). Note that if all flight paths are straight lines, then the resulting graph is a permutation graph.

A composition of permutation diagram.

We can think of F-graph as if it was constructed from few partial orders (each of them is a permutation on the vertices). Therefore we can construct the F-graph from the composition of these partial orders. (See Fig. 8.6).

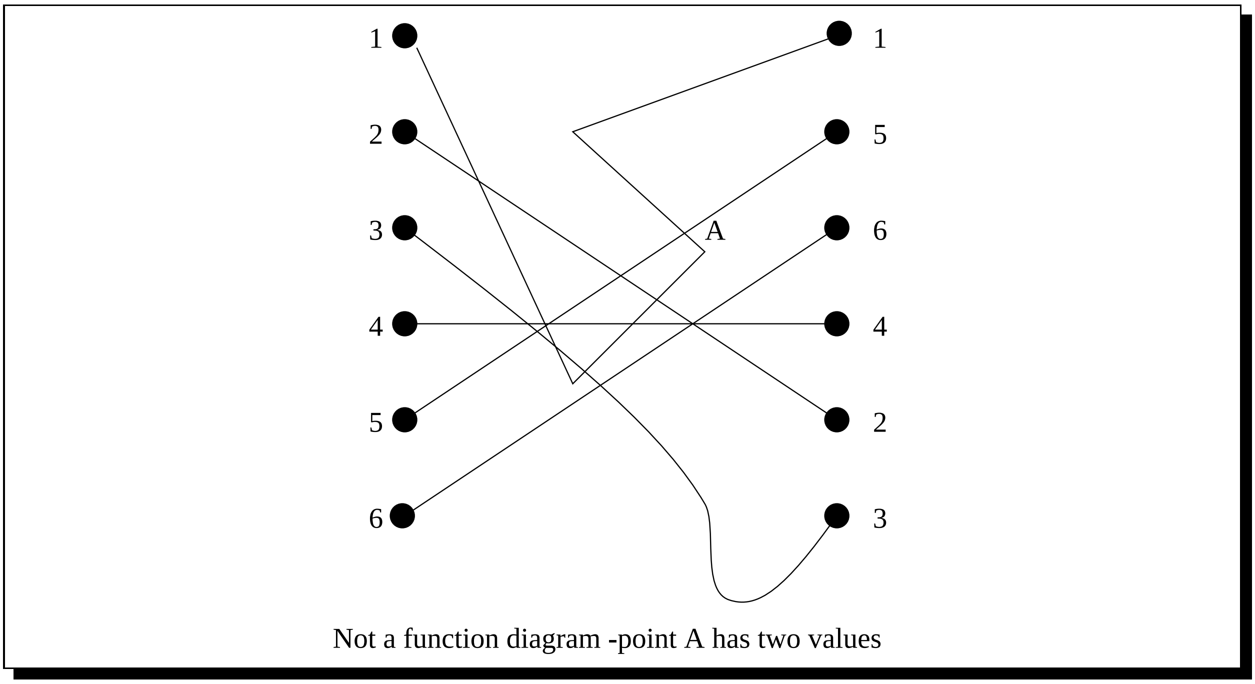


Figure 8.4: Not an F-graph,point A have two values of the function.

Theorem 8.5 *(Golumbic Rotem Urrutia 1982)*
Let G be an undirected graph. The following statements are equivalent.

(1) G is an F-graph.

(2) G is the intersection graph of a composition of permutation diagrams.

(3) $\bar{G}$ is transitively orientable.

Proof:

(2) $\Rightarrow$ (1) Trivial, each piecewise-linear path (connecting the point numbered i in all the orders), is a continuous function. Therefore it is an F-Graph.

(1) $\Rightarrow$ (3) We shall define $H = (V, F)$ an orientation of $\bar{G}$ as follows : (i,j) $\in$F if the i-th path in the diagram is completely below the j-th path. More formally :

(i,j)$\in$F if $\forall$ x $f_i(x) < f_j(x)$ and (j,i) if $\forall$ x $f_j(x) < f_i(x)$, otherwise the edge is not in F.

If $(i, j) \in E$, than obviously the function $f_i(x) - f_j(x)$ changes sign at least once. Therefore (i,j) $\notin F$, and vice versa. So $F = \bar{E}$. We should now prove that F is transitive orientation for $\bar{G}$.

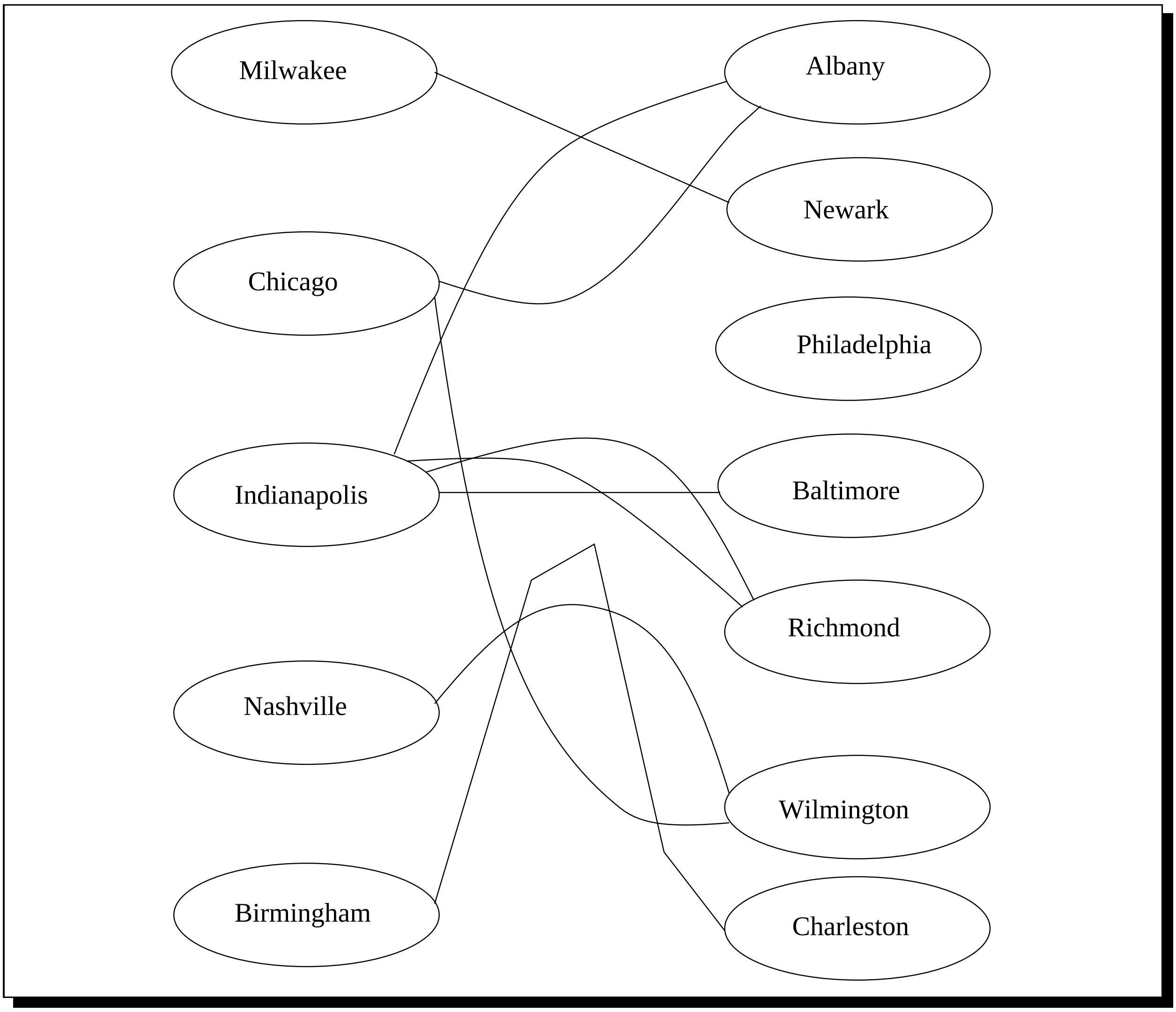


Figure 8.5: The air controllers strike problem.

If $(i, j) \in F$, and $(j, k) \in F$, then : $\forall x f_i(x) < f_j(x) < f_k(x)$.
$\Rightarrow (i, k) \in F$.

(3) $\Rightarrow$ (2) Let F be a transitive orientation of $\bar{G}$. We can choose $[l_1, l_2, \ldots l_k]$ to be a realizer of F. For each line order l_i we assign the points of its permutation in the corresponding order on the i-th horizontal line (See fig. 8.6). For i=1,2,...,n connect all points corresponding to i by straight line segment. The resulting structure is a function diagram.

Now, by the definition of the realizer, (i,j)$\in$F iff for every order l_k i,j appear in the same order (i.e. either for all k $i <_{l_k} j$ or for all k $j <_{l_k} i$). Hence (i,j)$\notin$F iff the paths of i and j intersect. Hence G is an F-graph.

■

8.1.4 Tolerance graphs

Definition Let $I = \{I_1, I_2, \ldots I_n\}$ be a collection of line segments, and let $T = \{t_1, t_2, \ldots t_n\}$ a set of positive numbers. The *tolerance graph* $G = (V, E)$ of (I,T) is defined as follows :
$V = \{v_1, \ldots, v_n\}$
$E = \{(i, j) || I_i \cap I_j| \geq min(t_i, t_j)\}$ (See Fig. 8.7)

Interval graphs as a subset of tolerance graphs.

The family of tolerance graphs is a generalization of the interval graphs. They are usually used for more general scheduling problems, where we allow things to happen simultaneously.

Given an interval graph $G = (V, E)$ whose realization is $I = \{I_1, I_2, \ldots, I_n\}$, we can construct an isomorphic tolerance graph in the following manner :
$T = (V, E)$ the tolerance graph of (I, X):
define $X(i, j) = |I_i \cap I_j|$, if $I_i \cap I_j \neq \emptyset$
$C = min\{X_(i, j)|(i, j) \in E\}$ Now let I=$\{I_1, I_2, \ldots, I_n\}$ and T=$\{t_1, t_2, \ldots, t_n\}$ where $t_i = C/2$ for all i. We claim that G is the tolerance graph of (I,T). Two line segments intersect in the original interval graph, iff the intersection length is bigger than C/2, therefore the edge sets of the two graphs coincide.

The following lemma will be used in the next theorem :

Definition G=(V,E) is an *interval containment* graph if there exists a set of intervals $\{I_u\}_{u \in V}$ such that (u,v)$\in$E$\Leftrightarrow$ $[I_u isin I_v$ or vice versa].

Lemma 8.6 *G is a permutation graph if and only if it is a interval containment graph.*

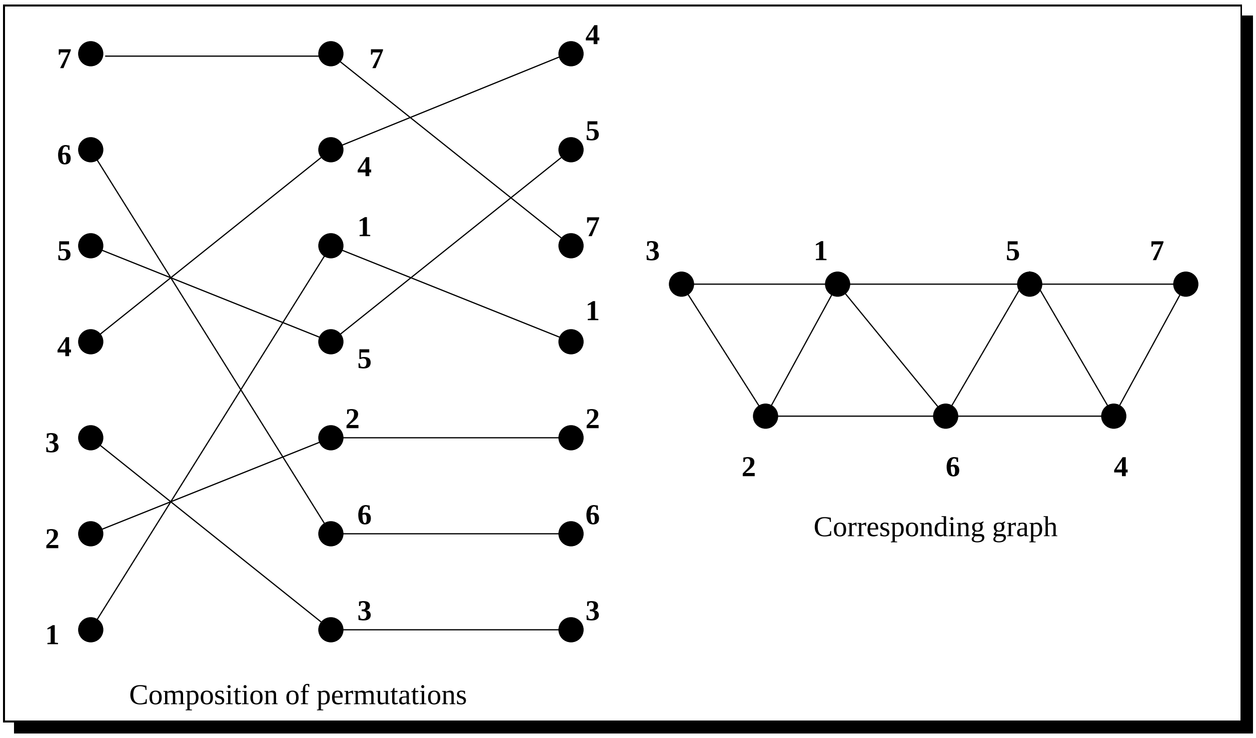


Figure 8.6: The composition of permutation diagrams.

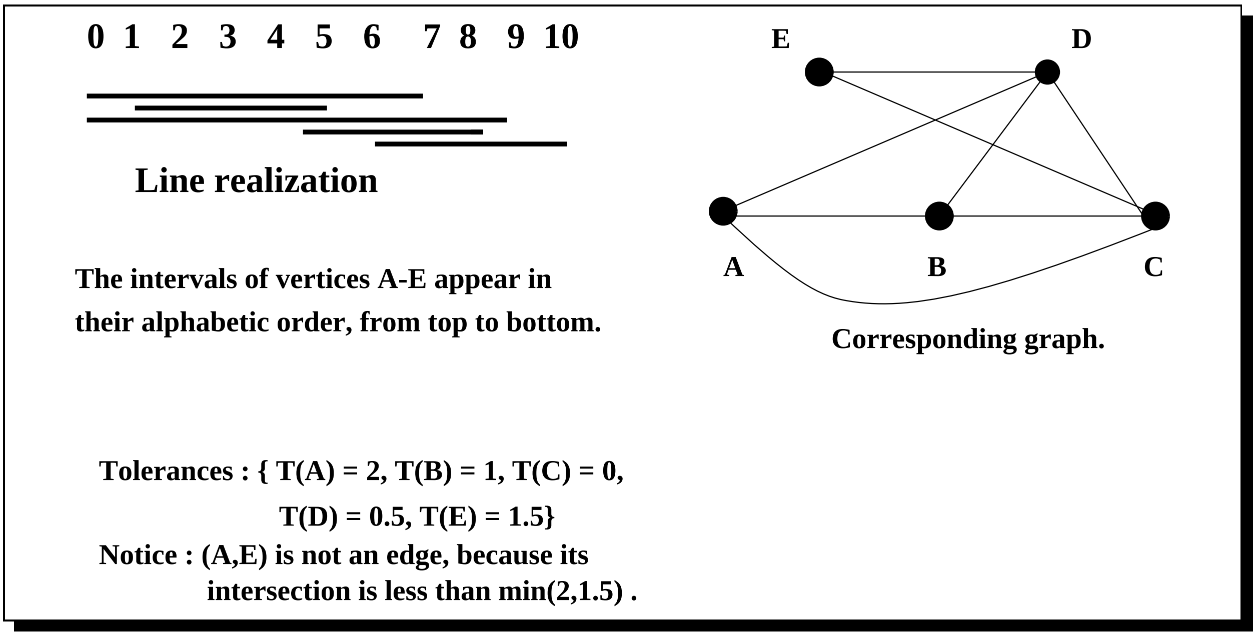


Figure 8.7: A tolerance graph.

Proof: Both sides wasn't proved in class. ∎

Theorem 8.7 *G is a tolerance graph with $t_i = |I_i|$, if and only if G is a permutation graph.*

Proof:

$\Rightarrow$ The proof is based on a the previous lemma. So all we have to do is to prove that G is an interval containment graph.

Let G be a tolerance graph with t_i defined as above. If $(i,j) \in E$ then either I_j contains I_i, or vice versa. Using the previous theorem, our theorem is proved.

$\Leftarrow$ If G is a permutation graph, than it is a line containment graph, therefore we can realize it with a tolerance graph with tolerances defined as above.

∎

8.1.5 Bounded and unbounded tolerance graphs.

If for a vertex i $t_i > |I_i|$, than we can enlarge t_i as much as we want, without affecting the graph. Any interval which intersects I_i will have a common intersection with I_i not longer than t_i, so in this case an edge (i,j) will be in the graph if and only if $|I_i \cap I_j| \geq t_j$. V_i will be called a vertex with *unbounded tolerance*. If G is a tolerance graph, in which no v_i is of unbounded tolerance, we call G a *bounded tolerance graph*.

Theorem 8.8 *Let G be a bounded tolerance graph, then is an F-graph.*

Proof: Wasn't proved in class. ∎

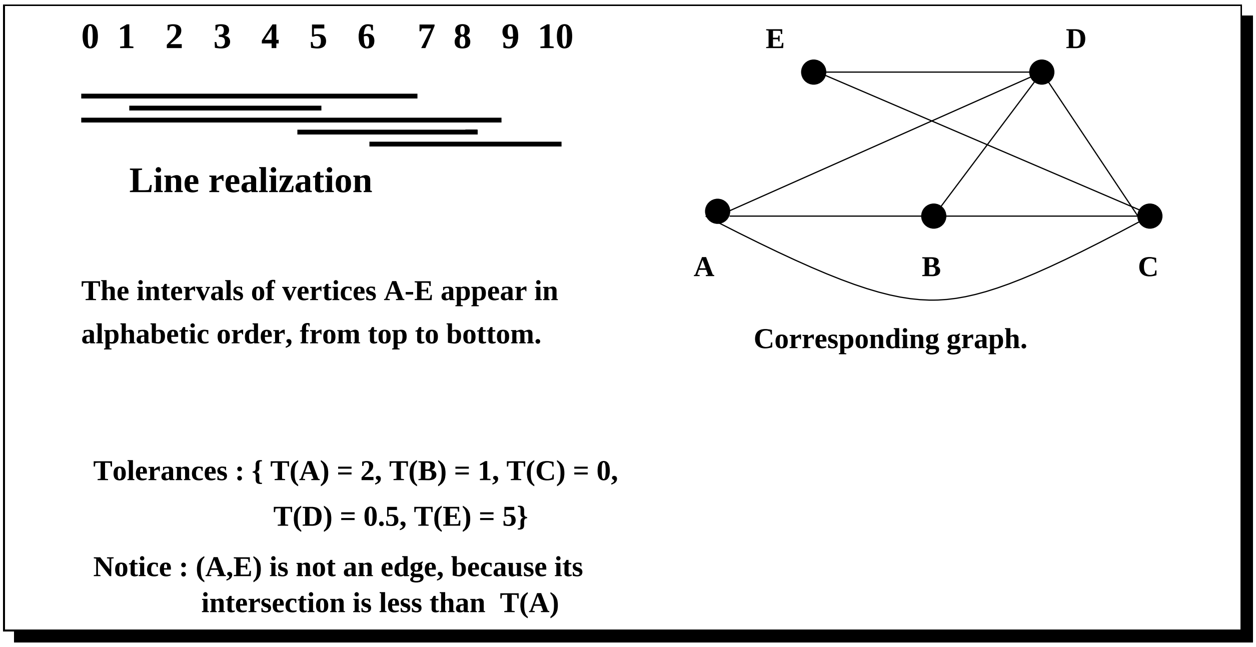


Figure 8.8: An unbounded tolerance graph.

Chapter 9

Lecture 9

Advanced Topics in Graph Algorithms		
©Tel-Aviv University		
Spring Semester, 1994		
Lecturer	:	Ron Shamir
Scribe	:	Muafaq Tannous
Lecture 9	:	June 9

9.1 Comparability Graph Recognition

In lecture 6 we discussed characterization of comparability graphs, and we proved a theorem - TRO theorem - that characterizes comparability graphs. We described an algorithm that constructs G-decomposition. By combining the TRO theorem with the G-decomposition construction we obtain an algorithm that recognizes comparability graphs and generates a transitive orientation the graph if it is TRO. The algorithm is shown in figure 9.1

0. initialize $i \longleftarrow 1, E_i \longleftarrow E, F \longleftarrow \emptyset$
1. arbitrarily pick an edge $x_i y_i \in E_i$
2. enumerate the implication class B_i of E_i containing $x_i y_i$
   if $B_i \cap B_i^{-1} = 0$ then add B_i to F
   else stop (G is not TRO)
3. $E_{i+1} \longleftarrow E_i - \hat{B}_i$
4. if $E_{i+1} = 0$ then output F and stop
   else $i \longleftarrow i + 1$ go to 1

Figure 9.1: Code of The TRO Algorithm

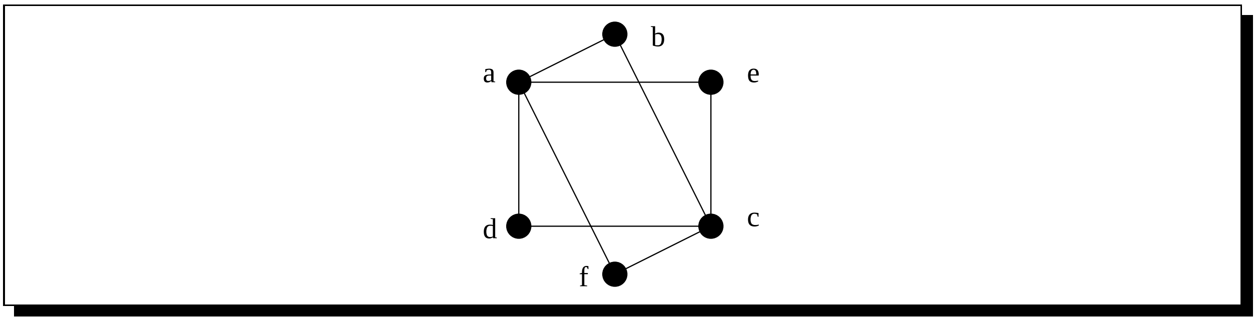


Figure 9.2: Example 1

Example: Let us run the algorithm on the graphs shown in figures 9.2, 9.3 and 9.4. The run of the algorithm on the first graph will be as the following :

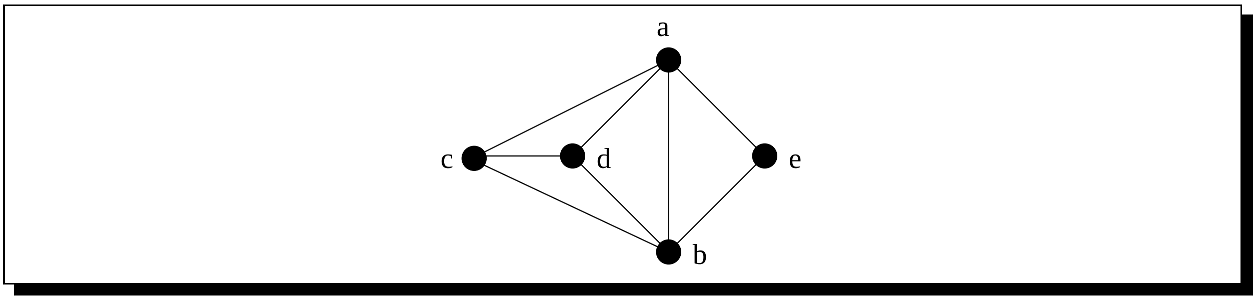


Figure 9.3: Example 2

Initialize : $E_1 = E = \{ab, ba, ae, ea, af, fa, ad, da, bc, cb, ce, ec, cf, fc, ea, ae\}, F = \emptyset$. After the first iteration we have $x_1y_2 = ab, B_1 = \{ab, ad, af, ae, cb, cf, ce, cd\}, F = B_1, E_2 = \emptyset$, so the graph is TRO.

The execution of the algorithm on the second graph will be the following :

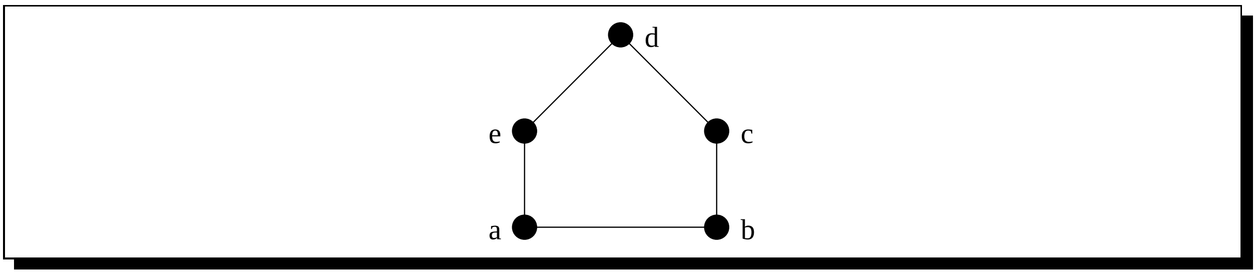


Figure 9.4: Example 3

Initialize :

$$E_1 = \{ac, ca, ae, ea, ad, da, ab, ba, cb, bc, eb, be, cd, dc, db, bd\}, F = \emptyset.$$

After the first iteration we have:

$$x_1y_1 = ac, B_1 = \{ac, ab, ad, ae\}, F = \{ac, ab, ad, ae\}, E_2 = \{dc, cd, ba, ab, be, eb, de, ed\}.$$

After the second iteration

$$x_2y_2 = bc, B_2 = \{bc, bd, ba, be\}, E_2 = \{cd\}, F = \{ac, ab, ad, ae, bc, bd, ba, be\}, E_2 = \{cd\}.$$

After the third iteration

$$x_3y_3 = dc, B_3 = \{dc\}, F = \{ac, ab, ad, ae, bc, bd, ba, be, dc\}, E_2 = \emptyset.$$

Thus the graph is TRO.

The execution of the algorithm on the second graph will be the following :

Initialize : $E_1 = \{ab, ba, bc, cb, cd, dc, de, ed, ea, ae\}, F = \emptyset$. After the first iteration we have $x_1y_1 = ab, B_1 = \{ab, ae, cb, cd, ed, ea, ba, bc, dc, de\}, B_1 \cap B_1^{-1} \neq \emptyset$, so the graph is not TRO.

The edges we choose in step (1) in the algorithm determine which transitive orientation of the set of transitive orientations of the graph will be the output of the algorithm. If we run the algorithm several times, and in every time we choose different edges in step (1) we may get different transitive orientations, but one can prove that the number of iterations of the algorithm in every run will be the same.

9.2 The Complexity of Comparability Graph Recognition

9.2.1 Implementation

We shall show that the complexity of the above algorithm is $O(\delta|E|)$ time and $O(|E| + |V|)$ space, where δ is the maximum degree of a vertex, and we'll give the implementation of this algorithm that gives such complexity. Let $G(V, E)$ be an undirected graph with $V = \{v_1, v_2, \ldots, v_n\}$. The algorithm uses a function called $CLASS(i, j)$, which is defined as the following : $CLASS(i, j) = 0$, if $v_iv_j \notin E$; $CLASS(i, j) = k$, if v_iv_j has been assigned to B_k; $CLASS(i, j) = -k$, if v_jv_i has been assigned to B_k; or undefined, if v_iv_j has not been assigned yet.

Let d_i be the out-degree of v_i after directing the graph G.

The algorithm gets as an input graph $G(V, E)$ such that $V = \{v_1, v_2, \ldots, v_n\}$ and its adjacency sets such that $j \in Adj(i)$ iff $v_iv_j \in E$. The output of the algorithm :

1. a variable $FLAG$ which tells if G is a comparability graph. $FLAG = 0$ iff G is comparability graph.

2. if $FLAG = 0$ then the transitive orientation that was found by the algorithm is the union of all edges having positive $CLASS$.

The algorithm consists of two subroutines, the EXPLORE procedure which is shown in figure 9.5, and the main procedure which is shown in figure 9.6.

The set of edges explored in iteration k is denoted B_k. In iteration k, the algorithm chooses an edge e that has not been assigned to any $B_i, i < k$, and start exploring from e. If $B_k \cap B_k^{-1} \neq \emptyset$ then the value of the variable $FLAG$ will be changed from 0 to 1.

9.2.2 Complexity Analysis

The input graph G is represented by adjacency lists. The adjacency sets are stored as linked lists sorted in increasing order and every element of the list represents a vertex. Each element in the linked list $Adj(i)$ is described by four fields. For every edge v_iv_j in G there are two elements as shown in figure 9.7.

1. j
2. $CLASS(i, j)$
3. pointer to $CLASS(j, i)$
4. pointer to next element on $Adj(i)$

This data structure requires $O(|V| + |E|)$ space. Let us compute the time complexity of the algorithm: In executing the 'for' command in $EXPLORE(i, j)$ we use two pointers: one points to $Adj(i)$ and the other points to $Adj(j)$. These two pointers will scan these two sets in parallel looking for a vertex m which satisfies the condition of the 'for' command and then the value of $CLASS$ will be updated. The loop will take $O(d_i + d_j)$. The complexity of the second loop is similar because it is executed in the same way. The algorithm calls the procedure $EXPLORE$ twice for every edge, hence the time complexity of the algorithm is

$$C \sum_{v_iv_j \in E} (d_i + d_j) \in O(\delta|E|)$$

In summary :

Theorem 9.1 *Comparability graph recognition and finding a transitive orientation can be done in $O(\delta|E|)$ time and $O(|V| + |E|)$ space, where δ is the maximum degree of a vertex.*

```
procedure EXPLORE(i,j):
for each m ∈ Adj(i) such that m ∉ Adj(j) or |CLASS(j,m)| < k do
      begin
         if CLASS(i,m) is undefined then
            begin
               CLASS(i,m) ⟵ k; CLASS(m,i) ⟵ −k;
               EXPLORE(i,m);
            end
         else
            if CLASS(i,m) = −k then
               begin
                  CLASS(i,m) ⟵ k; FLAG ⟵ 1;
                  EXPLORE(i,m);
               end
      end
for each m ∈ Adj(j) such that m ∉ Adj(i) or |CLASS(i,m)| < k do
      begin
         if CLASS(m,j) is undefined then
            begin
               CLASS(m,j) ⟵ k; CLASS(j,m) ⟵ −k;
               EXPLORE(m,j);
            end
         else
            if CLASS(m,j) = −k then
               begin
                  CLASS(m,j) ⟵ k; FLAG ⟵ 1;
                  EXPLORE(m,j);
               end
      end
return
```

Figure 9.5: Procedure EXPLORE

```
begin
    initialize : k ⟵ 0; FLAG ⟵ 0;
    for each edge v_i v_j in E do
        if CLASS(i, j) is undefined then
          begin
            k ⟵ k + 1;
            CLASS(i, j) ⟵ k; CLASS(j, i) ⟵ −k;
            EXPLORE(i, j);
          end
end
```

Figure 9.6: The main procedure of the algorithm

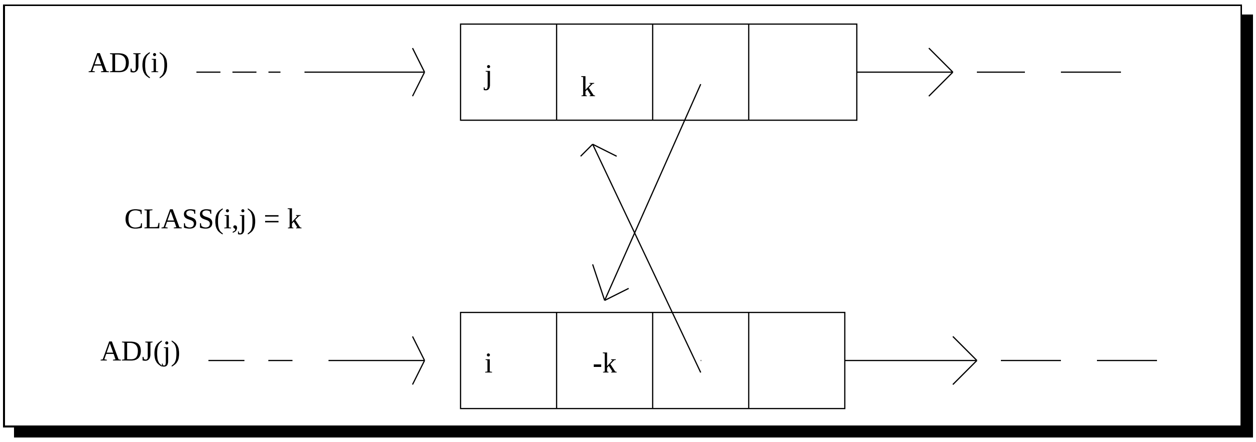


Figure 9.7: Two elements for each Edge-

As we define the procedure $EXPLORE$ it explores the edges in a depth-first search, we can change the algorithm to breadth-first search by storing the edges in queue (instead of stack), then it will run in the same time and space complexity.

There are several different algorithms that solve the TRO recognition problem. Until recently, the time complexity of the best known algorithm was due to Spinrad and has complexity $O(n^{2.5})$, In 1994 McConnel and Spinrad found an algorithm that in $O(m+n\log n)$ constructs an orientation F in the graph G such that if G is a comparability graph then F is transitive. There is an algorithm that decide if F is transitive in $O(n^{2.371\cdots})$.

9.3 Coloring and Other Problems on Comparability Graphs

9.3.1 Comparability Graphs Are Perfect

On a directed and acyclic graph $G(V, F)$, define a strict partial ordering of vertices, namely, $x > y$ iff there is nontrivial path in F from x to y. A height function h can be defined on such graphs : $h(v) = 0$ if out-degree of v is 0, $h(v) = 1 + max\{h(w)|vw \in F\}$. An example of $h(v)$ is shown in figure 9.8. Here in subsequent figures we use *Hasse diagram* of the partially ordered set, i.e. all edges are pointing upwards and transitive edges are omitted.

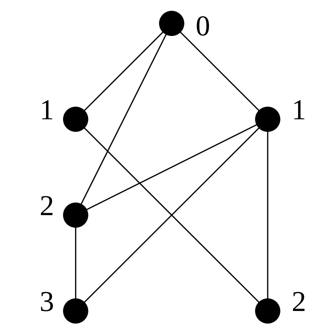


Figure 9.8: Example 4

$h(v)$ is equal to the number of vertices in the longest path from one of G roots to v. $h(v)$ can be computed in linear time using BFS. The function h gives a proper vertex coloring of G, because for every v and u if $vu \in F$ then $h(v) < h(u)$, but it is not necessarily a minimum coloring. For a graph G which is undirected, for any acyclic orientation F, the corresponding h will be a proper vertex coloring of G. We now prove that if G is transitive then h is an optimal coloring.

Theorem 9.2 *Every comparability graph is a perfect graph.*

Proof: Let F be a transitive orientation of a comparability graph $G(V,E)$. Every path in F corresponds to a clique of G, because of transitivity, and every clique corresponds to a path, hence the function h is a proper vertex coloring of G that uses exactly $\omega(G)$ colors, this is an optimal vertex coloring because $\chi \geq \omega$. Because of the hereditary property of comparability graphs we find that $\omega(G_A) = \chi(G_A)$ for all induced subgraph G_A of G, so G is a perfect graph. ∎

Let $(X, \leq)$ be a partially ordered set. A subset of X that is completely ordered is called *chain*. A subset in X that doesn't contain two members comparable by $\geq$ is called *antichain*.

The sets $\{c,e,a\}, \{d,g,b\}, \{f,b\}$ are chains in the partial order shown in figure 9.9, the sets $\{c,d\}, \{e,f,g\}$ are antichains in the same partial order.

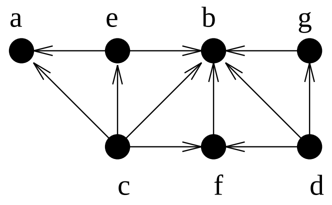


Figure 9.9: Example 5

A *Hasse diagram* is a compact representation of a transitive order. For a partially ordered set (V,F), the Hasse diagram is (V,H) where $H \subseteq F$ and $uv \in H$ iff $uv \in F$ and there is no $m \in V$ s.t. $um \in V, mv \in V$. The edges of Hasse diagram is drawn without the direction arrow, with the convention that all edges are pointing up. For example the Hasse diagram of the graph in figure 9.9 is in figure 9.10.

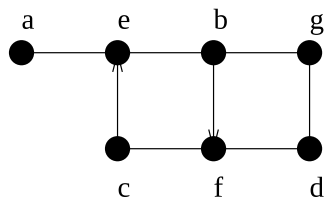


Figure 9.10: Example 6

Theorem 9.3 *(Dilworth 1950) Let $(X, \leq)$ be a partially ordered set, the minimum number of chains needed to partition (cover) X is equal to the maximum cardinality of a antichain in X.*

Proof: $(X, \leq)$ corresponds to comparability graph G which is a perfect graph, hence $k(G) = \alpha(G)$. k is equal to the minimum number of chains needed to partition X, because a vertex set S induces a clique in G iff there is a path in X passing through all vertices in S. α is equal to the maximum cardinality of an antichain. So, the minimum number of chains needed to partition X is equal to the maximum cardinality of antichain. ∎

For example consider the partial order that is shown in figure 9.9. The number of chains which partition the partial order is 3, also, the maximum cardinality of antichains is 3.

Corollary 9.4 *In a comparability graph with transitive orientation F, optimal proper coloring and maximum clique can be calculated in $O(|E| + |V|)$ time.*

We shall illustrate this by solving a slightly more general problem.

9.3.2 Maximum Weighted Clique

For a given undirected graph G and an assignment of a weights $w(v)$ to each vertex, the problem is to find a clique in G for which the sum of the weights of its vertices is largest. The algorithm calculates for every vertex v the value $W(v)$ which is the sum of weights of every vertex in the heaviest path from v to one of the graph sinks. A pointer is assigned to v designating its successor on that heaviest path.

Input : a transitive orientation F of a comparability graph $G(V, E)$ and a weight function w defined on V.

Output : a clique K of G whose weight is maximum.

The algorithm uses two subroutine, the EXPLORE procedure which is shown in figure 9.11, and the main procedure which is shown in figure 9.12.

Proving the correctness of the algorithm and displaying an implementation whose complexity is linear are left as exercises.

After running the algorithm on the graph that its Hasse diagram is shown in figure 9.13 we will have $W(a) = 5, W(b) = 6, W(c) = 12, W(d) = 2, W(e) = 11, W(f) = 4, W(g) = 14, POINTER(a) = null, POINTER(b) = a, POINTER(C) = b, POINTER(d) = null$, $POINTER(e) = f, POINTER(F) = null, POINTER(g) = e$ and the maximum weight clique is f, e, g.

9.3.3 Calculating $\alpha(G)$

We shall show a polynomial-time method for finding $\alpha(G)$ for a comparability graph G. The method gets as an input a graph $G(V, E)$ and a transitive orientation F. We transform F into a transportation network by adding two new vertices s and t and edges sx and yt for each source x and sink y. Then we assign a lower bound 1 on the flow through each vertex. We compute a minimum-flow, which is a polynomial problem. The value of such flow will

```
procedure EXPLORE(v):
    if Adj(v) = ∅ then
        W(v) = w(v);
        POINTER(v) ⟵ ∧;
        return;
    else for all x ∈ Adj(v) do
        if x is unexplored then
            EXPLORE(x);
    end for all;
    select y ∈ Adj(v) such that W(y) = max{W(x)|x ∈ Adj(v)};
    W(v) ⟵ w(v) + W(y);
    POINTER(v) ⟵ y;
    return;
end.
```

Figure 9.11: Procedure EXPLORE

```
procedure MAXWEIGHT CLIQUE(V, F):
    for all v ∈ V do
        if v is unexplored then
            EXPLORE(v);
    end for all;
    select y ∈ V such that W(y) = max{W(v)|v ∈ V};
    K ⟵ y;
    y ⟵ POINTER(y);
    while y ≠ ∧ do
        K ⟵ K ∪ {y};
        y ⟵ POINTER(y);
    return;
end.
```

Figure 9.12: The main procedure of the algorithm

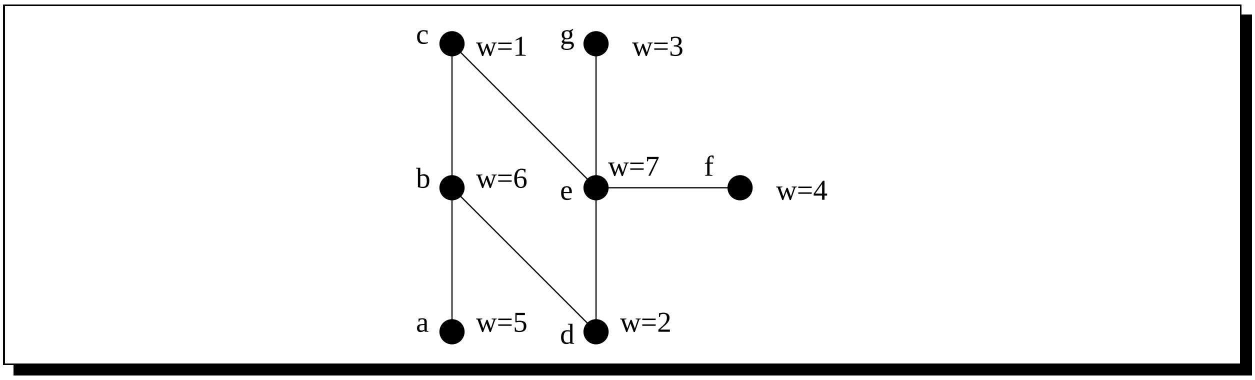


Figure 9.13: Example 7

equal to $k(G)$, the size of the smallest covering of the vertices by cliques. This value is equal to $\alpha(G)$, the maximum cardinality independent set since every comparability graph is perfect.

9.4 The Dimension of Partial Orders

Every partial order (X, P) can be extended to a linear order (also called its topological order). Such an order is called a *linear extension* of P. Let $L(P)$ be the set of linear extensions of P. A subset $L \subseteq L(P)$ that satisfies $\bigcap_{l \in L} l = P$ is called a *realizer* of P, and its size is $|L|$.

Example: The orders $L_1 = (a, b, c, d, e); L_2 = (a, c, b, e, d); L_3 = (a, b, c, e, d)$; are linear extensions of the partial order in figure 9.14. $\{L_1, L_2\}$ is realizer of this partial order.

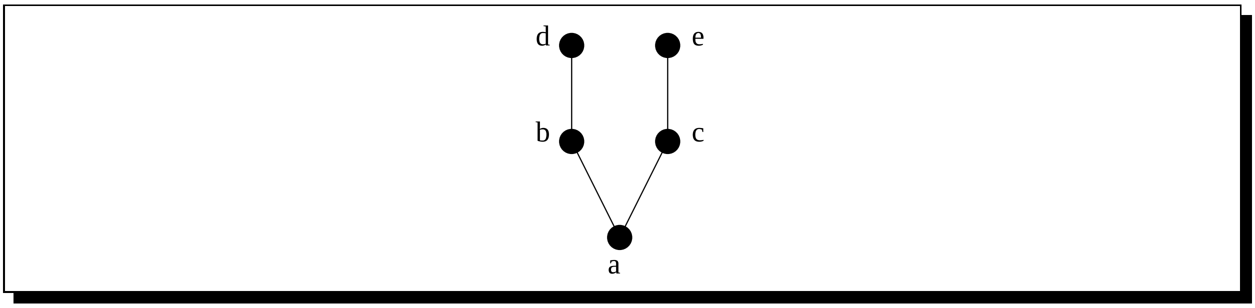


Figure 9.14: Example 8

A realizer of partial order P with the smallest size is called a *minimum realizer.* The *dimension* of partial order P, $dim(P)$ is the size of a minimum realizer of P.

The dimension of the partial order in figure 9.14 is 2.

Remark 9.5 *P is a complete order iff $dim(P) = 1$.*

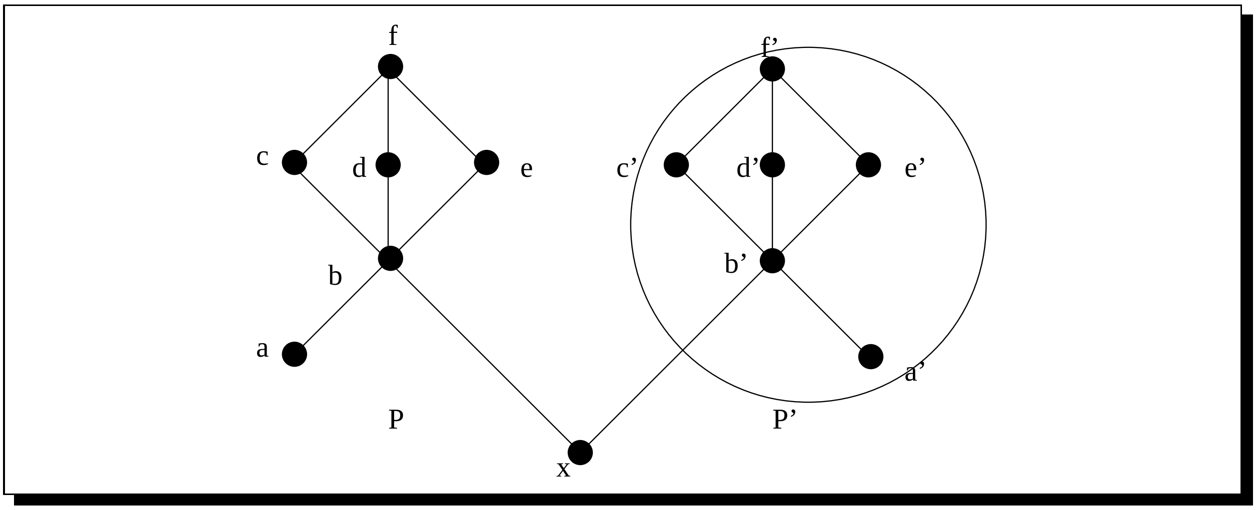


Figure 9.15: Example 9

Example: The orders

$$L_1 = (a, x, b, c, d, e, f, a', b', e', d', c', f'), L_2 = (a', x, b', c', d', e', f', a, b, e, d, c, f)$$

are linear extensions of the partial order P in figure 9.15, $\{L_1, L_2\}$ is the realizer of P, $dim(P) = 2$. And

$$L_1' = (b', e', d', c', f'), L_2' = (b', c', d', e', f')$$

are linear extensions of P' in figure 9.15, $\{L_1', L_2'\}$ is the realizer of P', $dim(P') = 2$.

Example: The orders $L_1 = (a, b, c', c, b', a'), L_2 = (b, c, a', a, c', b'), L_3 = (c, a, b', b, a', c')$ are linear extensions of the partial order P in figure 9.16, $\{L_1, L_2, L_3\}$ is realizer of P, and $dim(P) = 3$. To Show this, let us assume that $dim(P) = 2$, then there are two linear extensions l_1, l_2, that in one of them, say l_1, the elements (a, b, c) will be ordered: a, b, c. Then in l_2 these three elements must be ordered: c, b, a . l_1 must contain the subsequent a, b, c, b', a' and l_2 must contain the subsequent c, b, a, b' so we see that both extensions have b, b', a contradiction.

Lemma 9.6 *Let (X, P) be partial order. For every subset Y of X, $Y \subseteq X$ we have $dim(P_Y) \leq dim(P_X)$*

Proof: Let L be a realizer of P. Restricting L to the elements of Y yields a realizer of P_y. L is a minimum realizer for P so $dim(P_Y) \leq dim(P)$ ■

Theorem 9.7 *(Hiraguchi 1951) Let $P = P_0[P_1, P_2, \ldots, P_k]$ be a composition of a disjoint partial orders (X_i, P_i) $i = 0, 1, \ldots, k$ then $dim(P) = max\{dim(P_i)|0 \leq i \leq k\}$*

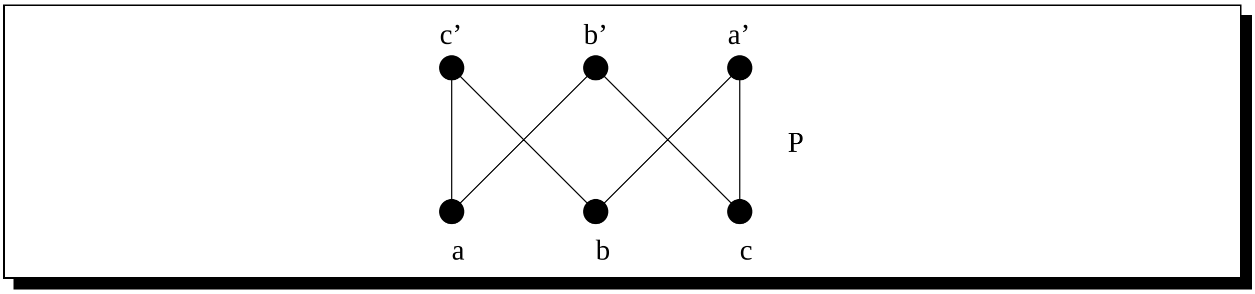


Figure 9.16: Example 10

Proof: Let $m = max\{dim(P_i)|0 \leq i \leq k\}$ and $\{L_{i_1}, L_{i_2}, \ldots, L_{i_m}\}$ be a realizer of P_i. Define $\wedge_i = L_{oj}[L_{1j}, L_{2j}, \ldots, L_{kj}]$, then $\{\wedge_j | j = 1, 2, \ldots, m\}$ is a realizer of P, so $dim(P) \leq m$, every P_i is a subset of P, so $dim(P) \geq max\{dim(P_i), 0 \leq i \leq k\} \longrightarrow dim(P) = max\{dim(P_i), 0 \leq i \leq k\}$ ∎

For an example see figure 9.17.

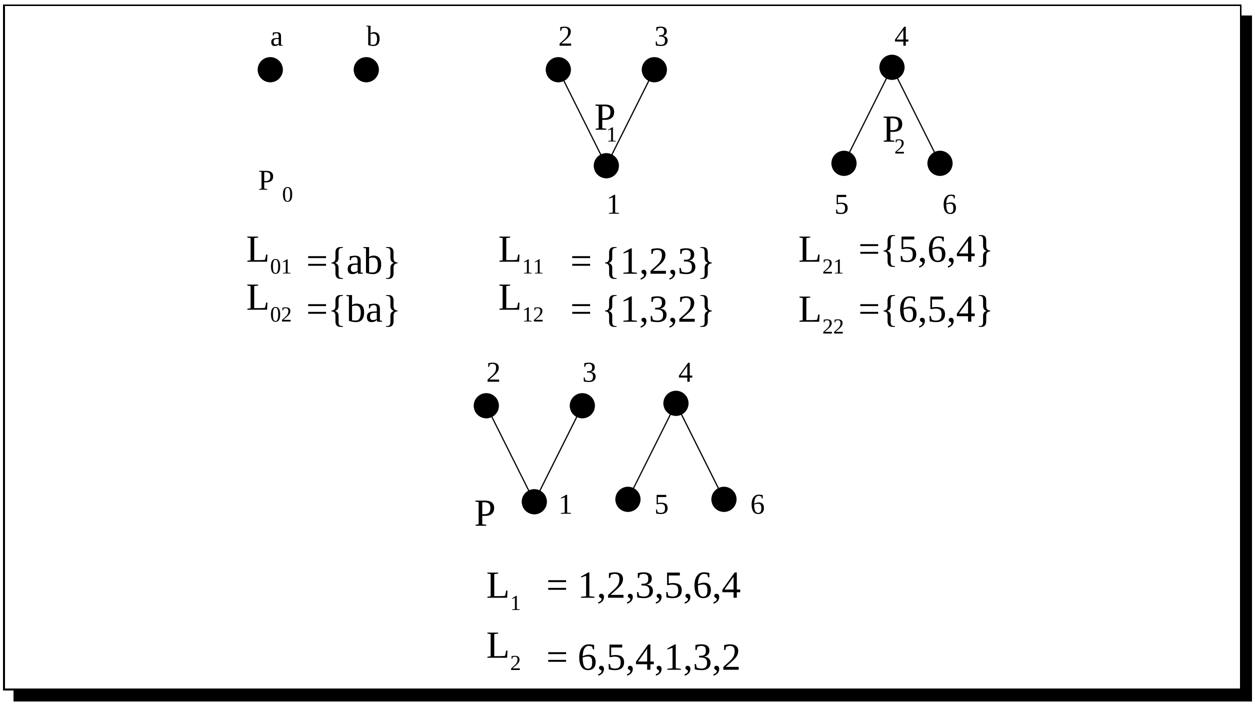


Figure 9.17: Example 11

Theorem 9.8 *(Dushnik-Miller, 1941) Let G be a comparability graph of poset $P, dim(P) \leq 2$ iff $\overline{G}$ the complement graph of G, is a comparability graph.*

Proof:

$\Rightarrow$: Let F be a transitive orientation of $\overline{G}$. Consider the graph $G'(V, E')$ when $E' = P + F$. This graph satisfies :

1. for every $a, b \in V$ either $ab \in E'$ or $ba \in E'$(but not both)
2. G' is transitive

To prove 2, note that if $ab, bc, ca \in E'$, it is obvious that ab, bc, ca cannot be in the same set (all in F or all in P), because F, P are transitive. So, two of these three edges are in the same set (F or P), say $bc, ab, ab \in F$ and by transitivity of F, we see that $ac \in F$ so $ca \notin P$. (1) and (2) imply that G' is a complete order on V. Because of the same reasons we know that $G''(V, E'')$ when $E'' = P + F^{-1}$ is a complete order on V. $L = \{P + F, P + F^{-1}\}$ is a realizer of P because if $ab \in P$ then $ab \in P + F, P + F^{-1}$, and if $ab \notin P$ then either $ab \in F$ and $ab \notin F^{-1}$, or $ab \in F^{-1}$ and $ab \notin F$. Hence $dim(P) <= 2$.

$\Leftarrow$: Assume that $dim(P) \leq 2$. Let $L = \{l_1, l_2\}$ be a realizer of P. Define $F = l_1 - P = (l_2 - P^{-1})$ (the equality holds because L is a realizer. To calculate $l_1 - P$ we look at l_1 as a complete directed graph and remove all the edges from P, see figure 9.18)

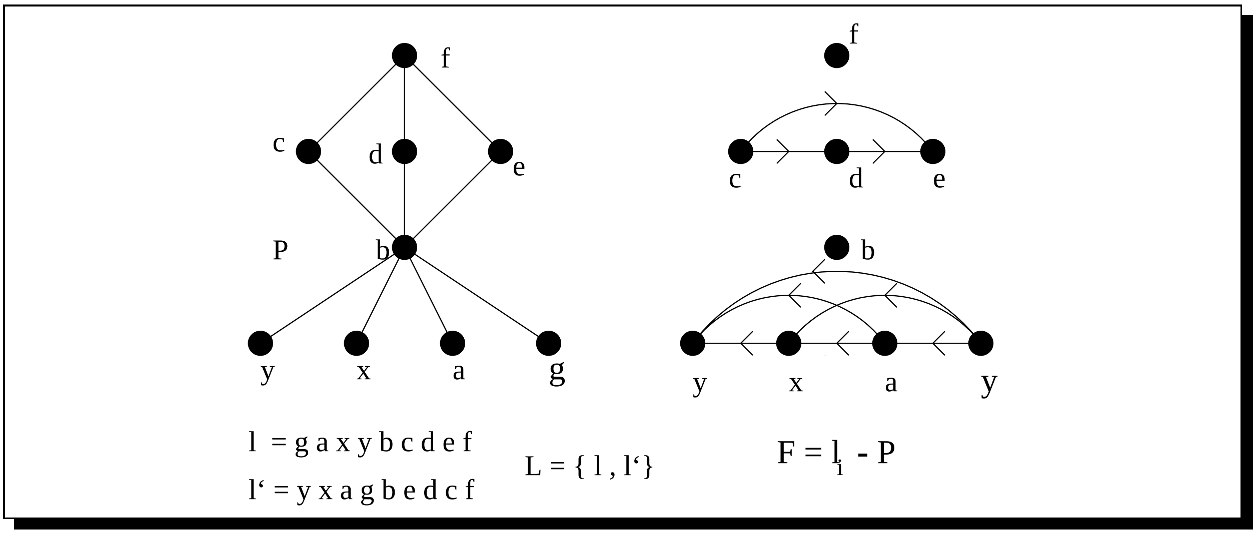


Figure 9.18: Example 12

We will show that F is a transitive orientation of $\overline{G}$, assume that $ab, bc \in F$ and $ac \notin F$. Then $ac \in P$. $ab, bc \in F \longrightarrow ba, cb \in l_2 \longrightarrow ca \in l_2 \longrightarrow ca \in P$ contradiction , hence F is transitive. ∎

Corollary 9.9 *Let G be a comparability graph of poset P. $dim(P) \leq 2$ iff G is a permutation graph.*

Proof: We will give two proofs :

1. $dim(P) \leq 2$ from the previous theorem $\longleftrightarrow \overline{G}$ is a comparability graph. From the theorem of Pnuely-Even-Lempel (1971), $G, \overline{G}$ are comparability graphs $\longleftrightarrow G$ is a permutation graph.

2. let us look at the matching diagram of permutation graph(see figure 9.19).

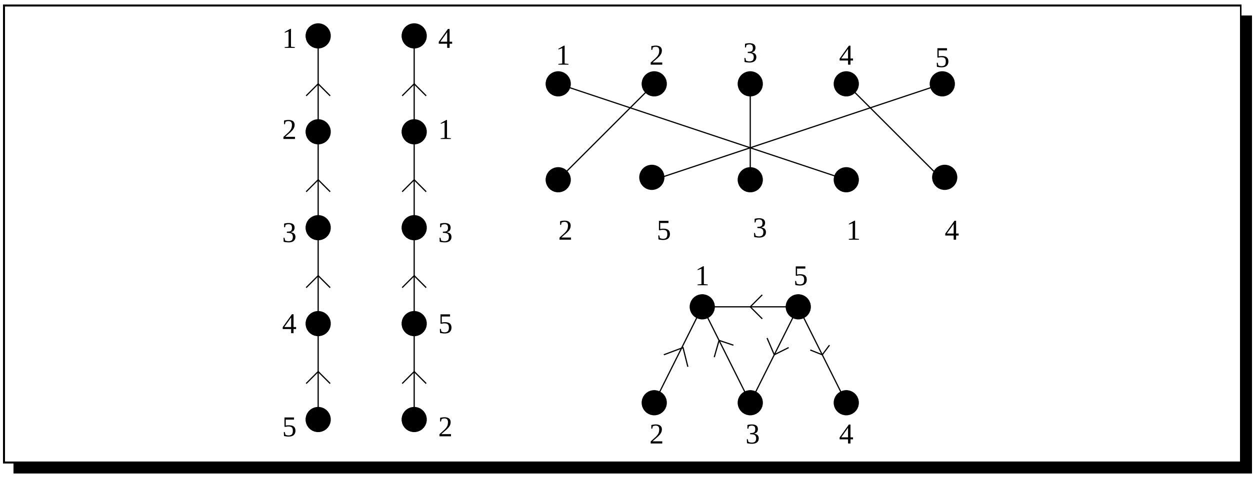


Figure 9.19: Example 13

$(i, j) \in E \longleftrightarrow [\pi^{-1}(i) > \pi^{-1}(j), i < j]$.Hence, the complete orders $l_1 = (n, n-1, \ldots, 1)$ and $l_2 = (\pi^{-1}(1), \pi^{-1}(2), \ldots, \pi^{-1}(n))$ is a realizer of P, so $dim(P) \leq 2$. Conversely, from the realizer $\{l_1, l_2\}$ we can construct the matching diagram in the same way. ■

The theorem of Dushnik-Miller tells that for two partial orders which have the same comparability graph, the dimension of both ≤ 2 or the dimension of both ≤ 2.

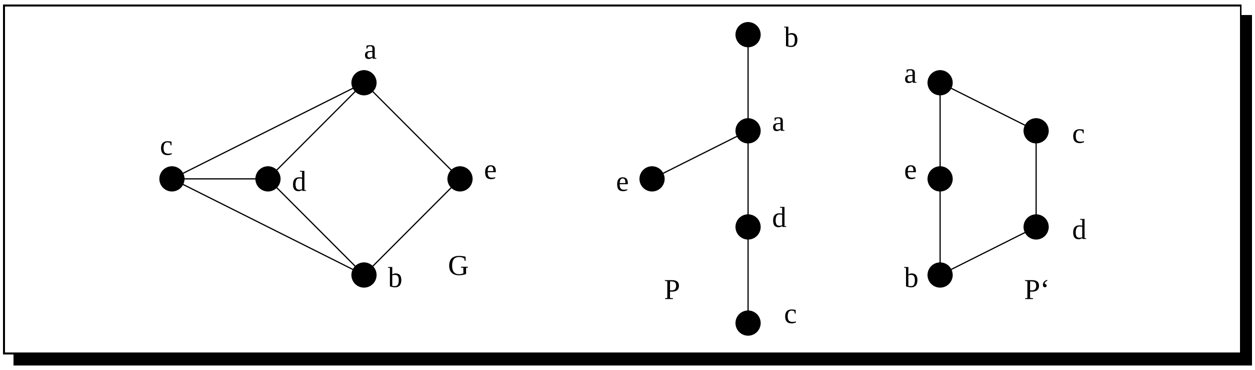


Figure 9.20: Example 14

Theorem 9.10 (Trotter, Moore, Sumner 1976, Gysin 1977) *Two partial orders which have the same comparability graph have the same dimension.*

Figure 9.20 gives an example of two partial orders that have the same comparability graph.

The problem of calculating the dimension of partial order mentioned as a major open problem in the Garey and Johnson book. In 1982, Yannakakis proved that the problem is NPC. In addition he proved deciding if $dim(P) \leq k$, is NPC, for every $k \geq 3$.

¡

Chapter 10

Lecture 10

Advanced Topics in Graph Algorithms
©Tel-Aviv University
Spring Semester, 1994

Lecturer : Ron Shamir
Scribe : Michael Seltser
Lecture 10 : June 16

10.1 Comparability invariants

In lecture # 9 we quoted the following theorem:

Theorem 10.1 (Trotter, Moore, Sumner [1976], Gysin [1977]) *If two partial orders have the same comparability graph then their dimensions are same.*

Hence, the dimensions of all posets which have the same comparability graph is the same. This leads to the following definition:

Definition Let ψ be a property of partially ordered sets (posets). Then for poset P ψ (P) := 1 if ψ holds on P and $\psi(P) := 0$ otherwise. Property ψ is called *comparability invariant* (CI) if for every two posets P, P' $G(P) = G(P')$ implies $\psi(P) = \psi(P')$.

Some of the known comparability invariants are: the dimension, length of maximum chain (because it is $\omega(G)$), minimum number of chains (or anti-chains) needed for covering of P (because they are $k(G)$, $\alpha(G)$ in correspondence), and the number of linear extensions of P.

In 1991 *Brightwell* and *Winkler* showed that the problem of counting the number of linear extensions of a poset is $\#P$ - complete.

Definition Let $P = (V, E)$ be poset, $L = (V, E')$ is linear extension. Without loss of generality order for E' is $v_1 < v_2 < \ldots < v_n$. The *jump number* (JP) of L is the number of pairs (v_i, v_{i+1}) such that $\{v_i, v_{i+1}\} \notin E$.

More generally:

$$J(P, L) = \mid \{i \mid (L^{-1}(i), L^{-1}(i+1)) \notin E\} \mid$$

The jump number of P is the least jump number over all linear extensions of P i.e.

$$J(P) = min\{J(P, L) \mid L \text{ is a linear extension of } P\}.$$

The jump number is known to be a comparability invariant. In 1991 *Syslo* proved that calculating $J(P)$ is NP - complete.

Theorem 10.2 (Habib, [1982]) *Property ψ is comparability invariant iff the following is true:*

(1) $\psi(P) = \psi(P^{-1})$ $\forall$ poset P.

(2) Let $P = P_0[P_1, \ldots, P_m]$, $Q = Q_0[Q_1, \ldots, Q_m]$ be proper decompositions and $G(P_i) = G(Q_i)$, $0 \leq i \leq m$. If $\psi(P_i) = \psi(Q_i)$ $\forall i$, then $\psi(P) = \psi(Q)$.

Definition Let $P_1 = (\ V_1,\ <_1\ )$, $P_2 = (\ V_2, <_2\ )$ be posets where V_1, V_2 are disjoint. Then we'll define a *parallel composition of* P_1, P_2 by $P_1 + P_2 = (\ V_1 + V_2,\ <_+\ )$ where

$$x <_+ y \Leftrightarrow ((x, y \in V_1,\ x <_1 y) \vee (x, y \in V_2,\ x <_2 y)).$$

The *serial composition of* P_1, P_2 is $P_1 * P_2 = (V_1 + V_2, <_*)$ where

$$x <_* y \Leftrightarrow ((x, y \in V_1,\ x <_1 y) \vee (x, y \in V_2,\ x <_2 y) \vee (x \in v_1,\ y \in v_2)).$$

Thus, $P_1 + P_2 \equiv 2K_1[P_1, P_2]$, $P_1 * P_2 \equiv P_1[P_1, P_2]$.

Definition A poset $P = (V, <)$ is called *series parallel* (SP) if either $\mid V \mid = 1$ or one can derive it from any series parallel posets P_1, P_2 by parallel or series compositions.

For examples see Figure 10.1.

Definition A graph is called *cograph* if it is P_4 - free.

Theorem 10.3 *G is a cograph iff for every induced subgraph H of G either H or $\bar{H}$ is disconnected.*

Proof: Exercise. ■

Theorem 10.4 *Every cograph is a permutation graph.*

Theorem 10.5 *A is a cograph iff A is the comparability graph of a series-parallel poset.*

Corollary 10.6 *The property "being a series-parallel poset" is a comparability invariant.*

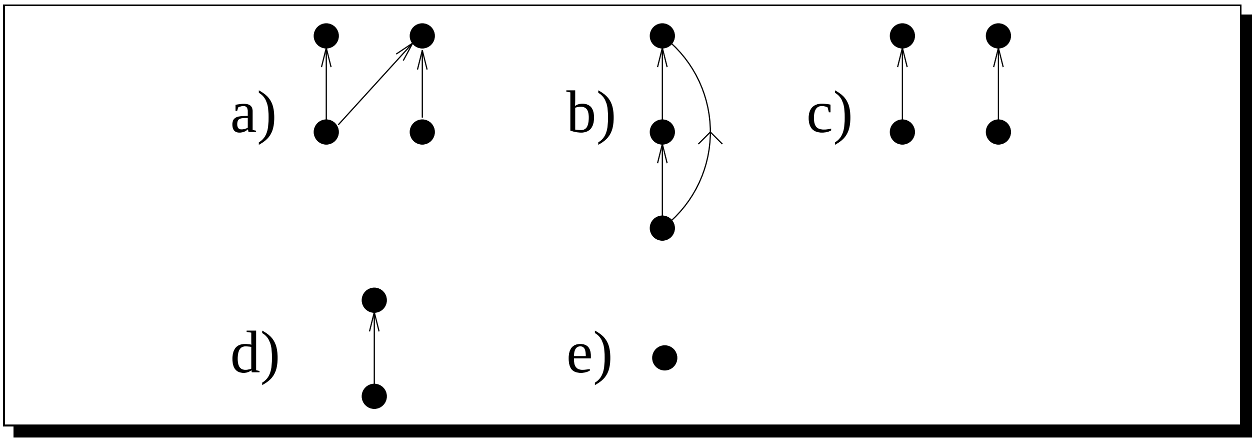


Figure 10.1: a) not SP poset; b) - e) SP posets

10.2 Interval graphs

Definition graph $G = (V, E)$ is called an *interval graph* if one can assign to each vertex $v \in V$ an interval on the real line I_v such that $(v_1, v_2) \in E \Leftrightarrow I_{v_1} \bigcap I_{v_2} \neq \emptyset$.

Interval graph were introduced by *Hajös(1957)* and*Benzer(1958)*.

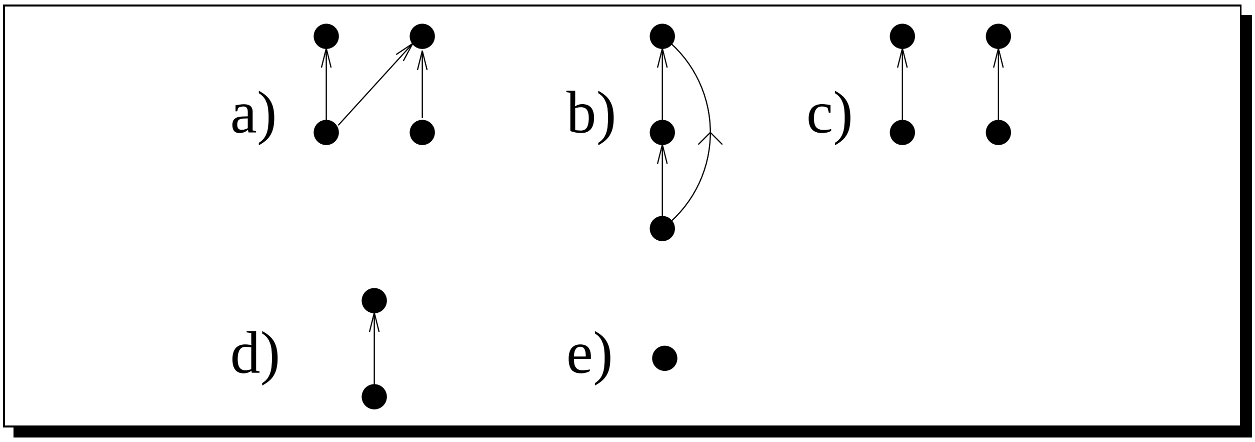


Figure 10.2: a) interval graph; b) it's interval representation; c) chordal, but not interval graph

For examples see Figure 10.2.

Theorem 10.7 (Gilmore - Hoffman [1964]) *The following are equivalent:*

(1) G is an interval graph.

(2) G contains no induced C_4 and $\bar{G}$ is a comparability graph.

(3) It is possible to order the maximal cliques in G so that for every vertex $v \in V$ the cliques that contain v appear consecutively in this order.

Proof:

(1) $\Rightarrow$(2) Immediate, because it is easy to see that an interval graph G contains no induced C_4 and a transitive orientation for $\bar{G}$ is obtained by taking an interval realization of G, $\{I_v\}_{v \in V}$ and setting $(u, v) \in E(\bar{G}) \Leftrightarrow I_u$ is completely to the left of I_v.

(2) $\Rightarrow$(3) Let F be a transitive orientation for $\bar{G}$.

Lemma: Let A_1, A_2 be distinct non-empty maximal cliques of G, and let (V, F) be a transitive orientation of $\bar{G}$. Then 1) there exists an arc from F between A_1 and A_2;

2) all such arcs between A_1 and A_2 have the same direction.

Proof:

(1) It is trivial because otherwise $B := A_1 \bigcup A_2$ is a clique in G and $\mid B \mid > \mid A_1 \mid, \mid A_2 \mid$, in contradiction to the definition of A_1, A_2.

(2) Assume that $a, c \in A_1, b, d \in A_2, (a, b), (d, c) \in F$. For example see Figure 10.3. If $a \equiv c$ (or $b \equiv d$) then we get (d,b) $\in F$, a contradiction to A_2 being a clique in G. Therefore the vertices are different. The situation $\{a, d\} \in E$ and $\{b, c\} \in E$ is impossible, because otherwise G contains an induced C_4. Without loss of generality assume that $\{ad\} \notin E$. Then we have two possible cases now:

(*i*) $(a, d) \in F$.Then the transitivity of F implies $(a, c) \in F$.

(*ii*) $(d, a) \in F$. Transitivity of F implies $(d, b) \in F$.

In both cases we get a contradiction.

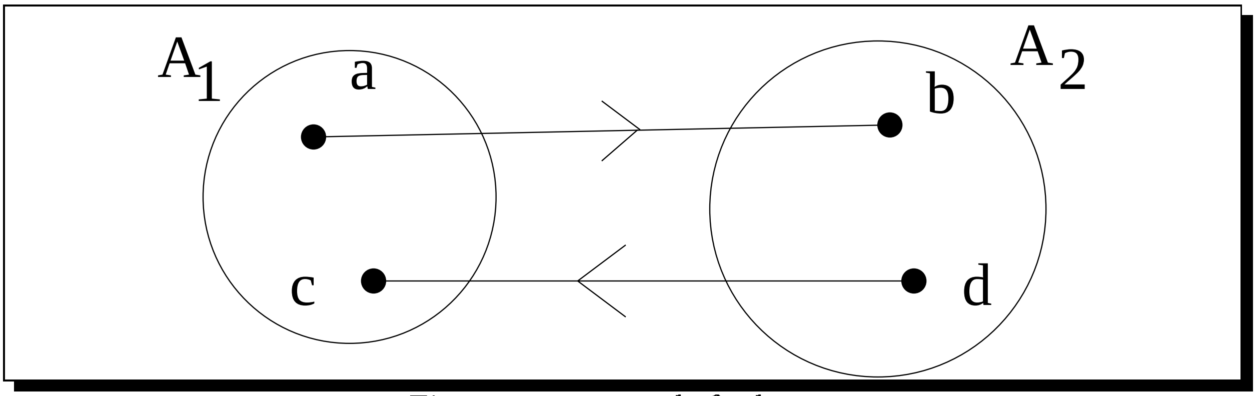


Figure 10.3: example for lemma

$\blacksquare$

We now return to the theorem. Define a relation among maximal cliques in G as follows: for max.cliques A_1 and A_2, $A_1 < A_2 \Leftrightarrow$ there is exists arc from F whose tail is in A_1 and it's head is in A_2. According to lemma, this relation defines a tournament on the set of maximal cliques of G. We'll prove that it is transitive and therefore defines a complete order on G. Assume $A_1 < A_2 < A_3$. Then $\exists\ (w, x) \in F : w \in A_1, x \in A_2$ and $\exists\ (y, z) \in F : y \in A_2, z \in A_3$. For example see Figure 10.4.

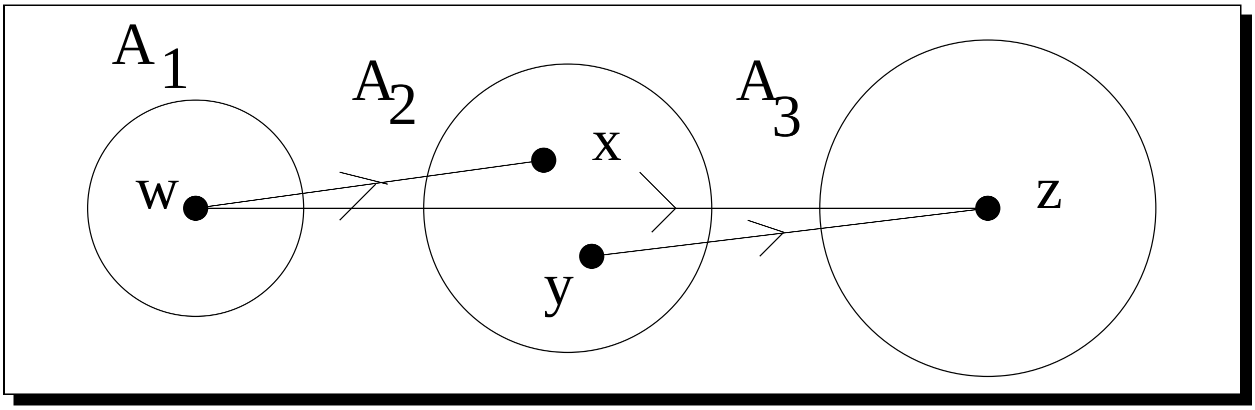


Figure 10.4: example for theorem of Gilmore-Hoffman

(α) if $\{x, z\} \notin E$ then the lemma implies that $(x, z) \in F$ and because F is a transitive orientation for G we get $(w, z) \in F$ and, hence, $A_1 < A_2$.

(β) if $\{w, y\} \notin E$ then analogously we get $(w, y) \in F$ and $A_1 < A_2$.

(γ) $\{x, z\}, \{w, y\} \in E$. G contains no C_4 and therefore $\{w, z\} \notin E$. It follows that, since F is transitive, $(w, z) \in F$. Hence, $A_1 < A_3$.

We have shown that $<$ is complete order (linear) on the maximal cliques of G. Let $x \in V$. Suppose that there exist $A_i < A_j < A_k$ such that $x \in A_i, A_k$ and $x \notin A_j$.

$$(x \notin A_j) \Rightarrow \exists y \in A_j : \{x, y\} \notin E$$
$$(A_i < A_j) \Rightarrow (x, y) \in F$$
$$(A_j < A_k) \Rightarrow (y, x) \in F$$

We have a contradiction. Therefore the claim is true.

(3) $\Rightarrow$(1) Let $I(x)$ be the set of maximal cliques of G that contain x. By (3) each such set is an interval in complete order $(C, <)$ of maximal cliques. $\forall x, y \in V\ \{x, y\} \in E \Leftrightarrow I(x) \bigcap I(y) \neq \emptyset$, because two vertices are neighbors iff they are contained together in some maximal clique.

■

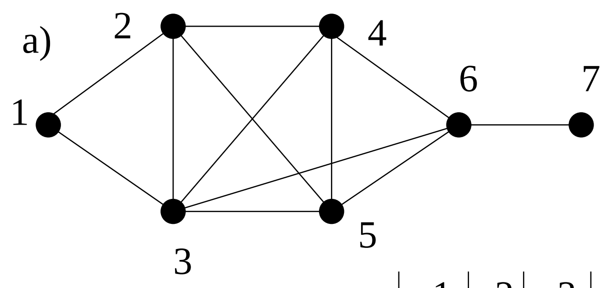
Corollary 10.8 *An undirected G is an interval graph iff G is chordal and $\bar{G}$ is comparability.*

Proof: If G is interval then it is chordal by the theorem of *Hajös* (1958) (lecture # 1).The rest holds according to the last theorem. Conversely, if G is chordal then it contains no C_4 and therefore if $\bar{G}$ is a comparability graph then G is interval. ■

Reminder: The *clique matrix of graph* is a (0,1) matrix M such that its rows correspond to the maximal cliques of graph, its columns to the vertices and $M_{ij} = 1$ iff v_j belongs to the i-th maximal clique. Theorem 10.4 implies that if rows are ordered in the order $<$ then the ones in every column will appear contiguously. This property is called the *consecutive ones property for columns.* For example see Figure 10.5.

Theorem 10.9 (Fulkerson-Gross [1965]) *An undirected graph G is interval iff its clique matrix has the consecutive ones property.*

Proof: Immediately follows from the third characterization of Theorem 10.4. ■



b)

	1	2	3	4	5	6	7
{1, 2, 3}	1	1	1	0	0	0	0
{2, 3, 4, 5}	0	1	1	1	1	0	0
{3, 4, 5, 6}	0	0	1	1	1	1	0
{6, 7}	0	0	0	0	0	1	1

Figure 10.5: a) interval graph; b) it's clique matrix.

Definition In an undirected graph G a triplet of vertices $< x, y, z >$ is called *an asteroidal triplet* if $G_{<x,y,z>}$ is stable and for each pair from $\{x, y, z\}$ there exists path between these two vertices which does not pass through neighbors of the third vertex.

Theorem 10.10 (Lekkerkerker-Boland [1962]) *G is an interval graph iff it is chordal and asteroidal triplet-free.*

Proof: If G interval then it is chordal. Consider an interval realization of G. Assume that $< z, y, x >$ is asteroidal triplet. Without loss of generalization suppose that the left endpoint of I_z is smallest among left endpoints of the intervals of the triplet in this realization and right endpoint of I_x is biggest among right endpoints of this intervals. Then there exists sequence of intervals $z = v_1, \ldots, v_k = x$ such that $\forall\ 1 \leq i \leq k-1\ I_{v_i} \bigcap I_{v_{i+1}} \neq \emptyset$. On other hand, $\forall\ 2 \leq j \leq k-1\ I_y \bigcap I_{v_i} = \emptyset$. It is contradiction. Therefore G is asteroidal triplet-free. For example see Figure 10.6. Proof of other direction is omitted here. ∎

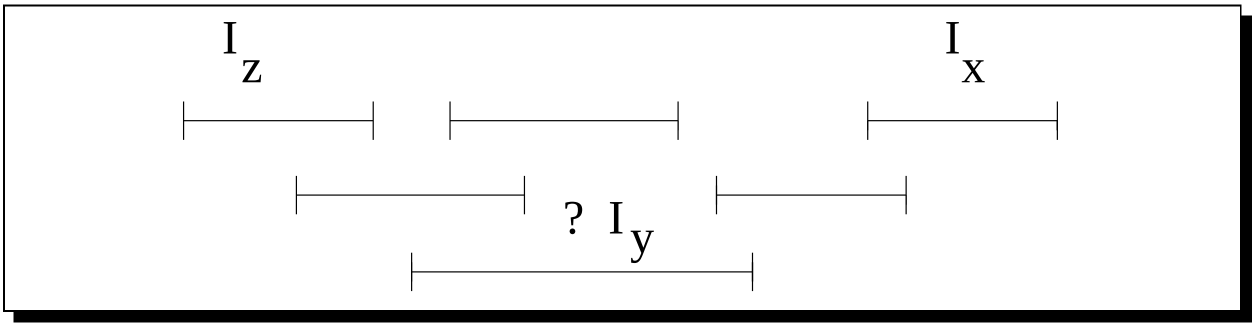


Figure 10.6: Example for Lekkerkerker - Boland theorem

Chapter 11

Lecture 11

Advanced Topics in Graph Algorithms

Spring Semester, 1994

Lecturer	:	Ron Shamir
Scribe	:	Vered Zilkha
Lecture 11	:	June 23

11.1 Preference and Indifference

Let V be a set of possibilities. A decision maker defines a preference relation on this set: for every pair of distinct members of V, he either prefers one over the other, or he feels indifferent about them.

We construct an oriented graph $H = (V, P)$, and an undirected graph $G = (V, E)$ as follows:

$$\forall x, y \in V$$

$xy \in P \Leftrightarrow x$ is preferred over y

$xy \in E \Leftrightarrow$ indifference is felt between x and y

By definition, $(V, P + P^{-1} + E)$ is complete.

We assume that the preference relation is acyclic. In fact we would want it to be transitive. Thus we require that P be a partial order.

One way of formalizing preference relations (in utility theory) is the notion of *Semiorder*.

Definition Given $\delta > 0$, a real-valued function $u : V \to R$ is called *a utility function of semiorder* if $\forall x, y \in V$, $xy \in P \Leftrightarrow u(x) \geq u(y) + \delta$

Remark: If P has a utility function of semiorder then P is a partial order (irreflexive,transitive).

Let us now characterize the partial orders that have a utility function of semiorder. We first state without proof a fundamental result from utility theory:

Theorem 11.1 (Scott - Suppes [1958]) *A binary relation* (V, P) *has a utility function of semiorder iff* $\forall x, y, z, w \in V$

(S1) P *is irreflexive*

(S2) If $xy \in P$, $zw \in P$ *then either* $xw \in P$ *or* $zy \in P$

(S3) If $xu \in P$, $yz \in P$ *then either* $xw \in P$ *or* $wz \in P$

Definition An *interval order* is a binary relation (V, P) such that $\forall i \in V$,we can match an interval I_i on the real line, such that $ij \in P \Leftrightarrow I_i < I_j$ (i.e $I_i \cap I_j = \emptyset$ and I_i is on the left side of I_j)

Claim 11.2 *A partial order* (V, P) *is an interval order iff it satisfies (S1),(S2).*

(The proof of this claim is left as an exercise)

The claim implies: If (V, P) is an oriented graph, (V, E) is an undirected graph, and $P + P^{-1} + E$ is the edge set of the complete graph on V then if (V, P) is a semiorder then (V, E) is an interval graph.

(The converse is not true. For example see figure 11.1).

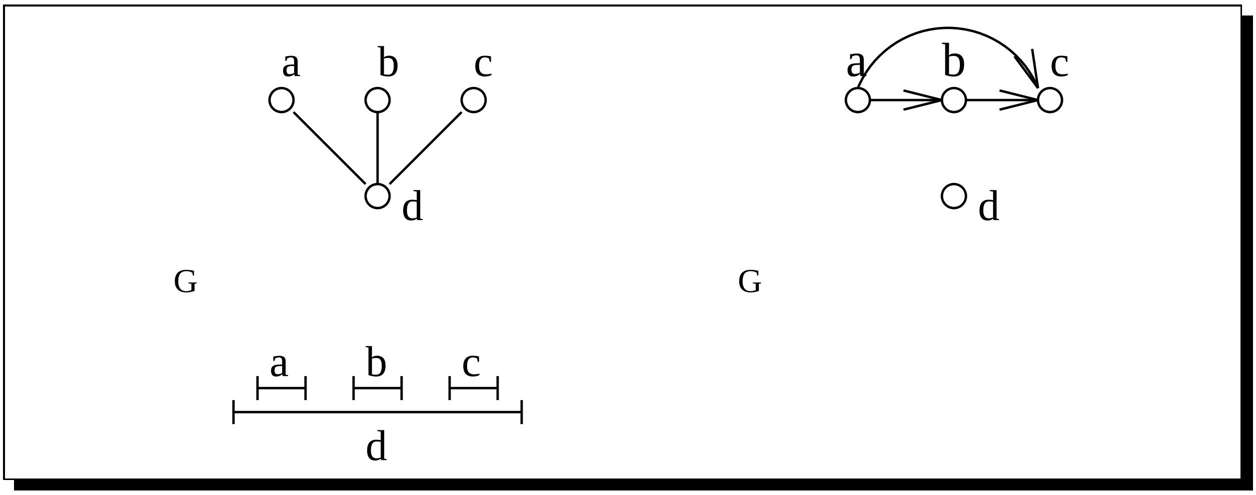


Figure 11.1: Example

Definition An interval graph $G = (V, E)$ is called a *proper* interval graph if it has a realization $\{I_v\}_{v \in V}$ such that no interval properly contains another.

Theorem 11.3 (Roberts [1969]) *Let $G = (V, E)$ be an undirected graph. The following statements are equivalent:*

(1) There exists a real valued function $u : V \to R$ such that $\forall xy \in V$, $x \neq y$, $xy \in E \Leftrightarrow | u(x) - u(y) | < 1$

(2) There exists a semiorder (V, P) such that $\bar{E} = P + P^{-1}$

(3) $\bar{G}$ is a comparability graph and every transitive orientation of $\bar{G} = (V, \bar{E})$ is a semiorder

(4) G is an interval graph that doesn't contain an induced $K_{1,3}$

(5) G is a proper interval graph

(6) G is a unit *interval graph*

Proof:

$(1) \Rightarrow (6)$ Let u be a function that satisfies (1). We will match to each vertex $x \in V$ an open interval $I_x = (u(x) - \frac{1}{2}, u(x) + \frac{1}{2})$. Then $\forall x, y, x \neq y$, $I_x \cap I_y \neq \emptyset \Leftrightarrow | u(x) - u(y) | < 1 \Leftrightarrow xy \in E$. Therefore $\{I_x\}_{x \in V}$ is a unit-interval realization of G.

$(6) \Rightarrow (5)$ In a unit-interval realization of G, no interval can properly contain another, because they all have length 1. Therefore this realization will be proper.

$(5) \Rightarrow (4)$ Let $\{I_x\}_{x \in V}$ be a proper interval realization of G. Suppose G contains an induced subgraph $G_{\{y,z_1,z_2,z_3\}}$ isomorphic to $K_{1,3}$ where $\{z_1, z_2, z_3\}$ is a stable set and y is adjacent to each z_i. Without lost of generality, suppose that $I_{z_1} < I_{z_2} < I_{z_3}$. Then I_y must properly contain I_{z_2}, a contradiction. Thus, G can have no induced copy of $K_{1,3}$.

$(4) \Rightarrow (3)$ Since G is an interval graph, $\bar{G} = (V, \bar{E})$ is a comparability graph. Let F be a transitive orientation of $\bar{G}$. It is easy to show that F satisfies $(S1), (S2)$. Now we will see that it also satisfies $(S3)$. By Scott-Suppes Theorem this will imply F is a semiorder.

Assume to the contrary that $xy \in F$, $yz \in F$, $xw \notin F$, $wz \notin F$. By the transitivity of F it follows that: $wx \notin F$, $zw \notin F$, $wy \notin F$, $yw \notin F$. But $xz \in F$. Thus $G_{x,y,z,w}$ induces a $K_{1,3}$, a contradiction.

$(3) \Rightarrow (2)$ Immediate.

$(2) \Rightarrow (1)$ If (V, P) is a semiorder, then by definition there exist a real valued function $\tilde{u} : V \to R$ and a constant $\delta > 0$ such that $xy \in P \Leftrightarrow \tilde{u}(x) - \tilde{u}(y) \geq \delta$. Denoting $u(x) = \frac{\tilde{u}(x)}{\delta}$ we get $xy \in P \Leftrightarrow u(x) - u(y) \geq 1$, and because $P + P^{-1} = \bar{E}$ we get that $xy \in E \Leftrightarrow xy \notin P$ and $xy \notin P^{-1} \Leftrightarrow | u(x) - u(y) | < 1$.

■

Additional results on unit interval graphs:

Definition A (0,1)-valued matrix satisfies the *consecutive 1's property* for the columns if its rows can be permuted such that on each column, the 1's will appear consecutively.

Definition A **layout** of a graph $G = (V, E)$ is a one-to-one function (a permutation) $L : V \rightarrow \{1, \ldots, |V|\}$. We will denote $\underline{\alpha_L}$ - the linear order $L^{-1}(1) < L^{-1}(2) < \ldots < L^{-1}(n)$.

Let $G = (V, E)$ be an interval graph. Then other equivalent conditions to G being unit interval are:

- The clique matrix of G satisfies the consecutive 1's property for rows and for columns.
- Let $A^*(G)$ be the adjacency matrix of G when we add 1's on the diagonal. Then $A^*(G)$ satisfies the consecutive 1's property for columns. (Roberts 1968)
- G is a triangulated graph and it doesn't contain as an induced subgraph any of the graphs appearing in figure 11.2.

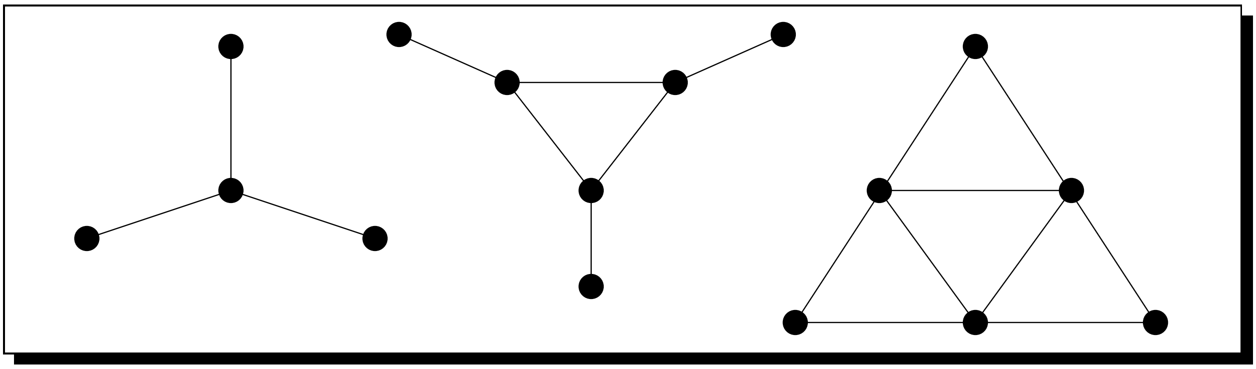


Figure 11.2: Wegner 1967

- There exists a layout L of G such that for every vertex v, $L(Adj(v)) = \{L(v)-k_v, L(v)-k_v+1, \ldots, L(v)-1, L(v)+1, \ldots, L(v)+l_v\}$ (Deng, Hell, Huang 1992)

Some hard optimization problems involving unit interval graphs are:

- Adding a minimum number of edges to a graph to get a unit interval graph is NP-complete. (Golumbic, Kaplan, Shamir 1993)
- Removing a minimum number of edges to get a unit interval graph is NP-complete. (Golumbic, Kaplan, Shamir 1993)

11.2 Recognizing interval graphs

Theorem 11.4 *$G = (V, E)$ is an interval graph iff it has a layout L that satisfies:*

$$[L(u) < L(v) < L(w), uw \in E] \Rightarrow vw \in E \tag{11.1}$$

(The proof of this theorem is left as an exercise).

Theorem 11.5 *Let $G = (V, E)$ be a co-comparability graph, and let F be a transitive orientation of $\bar{G}$. We denote $d(u)$ the out-degree of u in (V, F). If L is a layout of G such that*

$$d(u) > d(v) \Rightarrow L(u) < L(v) \tag{11.2}$$

then G is an interval graph iff L satisfies condition 11.1

Proof:

$\Rightarrow$ If condition 11.1 holds then according to Theorem 11.4 G is an interval graph.

$\Leftarrow$ Assuming, to the contrary that L satisfies condition 11.2 but not condition 11.1, we will prove that G contains an induced C_4:

(a) if $uv \in F$ then by transitivity of F, $d(u) > d(v)$, and thus $L(u) < L(v)$.

(b) if $L(u) < L(v)$ and $uv \notin E$ then $uv \in F$ (else $vu \in F$ and from (a) we get $L(v) < L(u)$).

From (b) and the transitivity of F it follows that:

$$[L(u) < L(v) < L(w), uv \notin E, vw \notin E] \Rightarrow uw \notin E \tag{11.3}$$

$$[L(u) < L(v) < L(w), uw \in E, uv \notin E] \Rightarrow vw \in E \tag{11.4}$$

Now, suppose L does not satisfy 11.1. Thus there exist u, v, w such that

$$L(u) < L(v) < L(w), uw \in E, vw \notin E \tag{11.5}$$

From (b) it follows that $vw \in F$, and from condition 11.3 it follows that $uv \in E$.

Since $L(u) < L(v)$, it follows that $d(u) \geq d(v)$. $vw \in F, uw \notin F$, thus there exists y such that $uy \in F$, $vy \notin F$. Moreover, $yv \notin F$ (else $uy, yv \in F \Rightarrow uv \in F$, a contraction), thus $vy \in E$. consider that

If $yw \in F$ then $uy, yw \in F \Rightarrow uw \in F$, a contradiction.

If $wy \in F$ then $vw, wy \in F \Rightarrow vy \in F$, a contradiction.

Thus $yw \in E$, and $G_{\{u,v,w,y\}}$ induces a C_4.

■

Theorem 11.6 *Let $G = (V, E)$ be a graph with a layout L, and let us define functions $l, h : V \to \{0, \ldots, \mid V \mid\}$:*

$$l(u) = \min_{v \in N[u]} L(v)$$

$$h(u) = \begin{cases} \max\limits_{\substack{v \notin N[u] \\ L(v) < L(u)}} L(v) & \{v \mid L(v) < L(u), v \notin N(u)\} \neq \emptyset \\ \\ 0 & \text{otherwise} \end{cases}$$

then L satisfies condition 11.1 iff $\forall w \in V, h(w) < l(w)$

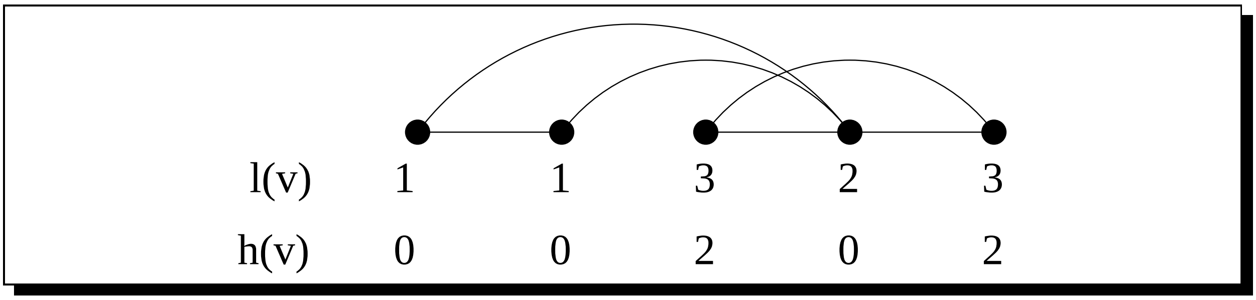


Figure 11.3: Example of function l, h

Proof:

- $\Rightarrow$ If L does not satisfy condition 11.1 then there exist u, v, w such that $L(u) < L(v) < L(w)$, and $uw \in E$, but $vw \notin E$. Thus $l(w) \leq L(u) < L(v) \leq h(w)$.

- $\Leftarrow$ If there exists $w \in V$ such that $l(w) < h(w)$ then taking $u = L^{-1}(l(w))$, $v = L^{-1}(h(w))$ gives us vertices u, v, w such that $L(u) < L(v) < L(w)$, and $uw \in E$, but $vw \notin E$, thus contradicting condition 11.1.

■

11.2.1 A quadratic algorithm 1

We first describe an elementary $O(n^2)$ algorithm for recognizing interval graph. This algorithm is due to Ramalingam and Rangan [1990].

Let $G = (V, E)$ be an undirected graph.

1. Find a transitive orientation of $\bar{G}$, and call it F.

2. Compute a layout of G according to a decreasing order of out-degrees of vertices in (V, F).

3. For each vertex, compute $l(u), h(u)$.

G is an interval graph iff $\forall w \in V,\ h(w) < l(w)$.

correctness:

1. If G is an interval graph then $\bar{G}$ has a transitive orientation. From theorem 11.5, the layout L computed in step 2 of the algorithm satisfies condition 11.1. From theorem 11.6 the condition $h < l$ holds.

2. If G is not an interval graph then no layout L satisfies condition 11.1, and from theorem 11.6 the condition $h < l$ fails.

complexity:

1. Finding a transitive orientation of $\bar{G}$: $O(n^2)$ (Spinrad [1985]), $O(n + \bar{m} \log n)$ (McConnell - Spinrad 1994). (We don't need to check that the orientation is transitive)

2. Sorting by out-degrees of vertices: $O(\bar{m} + n \log n)$. [Where $\bar{m} =\mid E(\bar{G}) \mid$].

3. Computing $l(u)$ for all vertices: $O(\bar{m})$.

   Computing $h(u)$ for all vertices: $O(m)$.

We can see that the complexity does not exceed $O(n^2)$.

Remarks:

* We can produce a realization of an interval graph, using the layout L.(this follows from the proof of theorem 11.4, which is constructive.

* The proof implies that every transitive orientation of $\bar{G}$ has a linear extension L (one or more) that satisfies condition 11.1.

11.2.2 A Linear Algorithm

To check whether a graph $G = (V, E)$ is an interval graph we can use the following algorithm:

1. Check whether G is triangulated

2. Produce all the maximal cliques in G, and then produce the clique matrix M of G

3. Check whether the matrix M satisfies the consecutive 1's property

The first two stages can be implemented in a linear time: $O(|V|+|E|)$. We will see now that the third stage can be implemented in a linear time, by an algorithm of Booth and Leuker [1975].

The data structure being used is called PQ - trees: a PQ-tree is a tree with a root, that has vertices of two kinds: P and Q. The sons of a P-vertex are an unordered set. The sons of a Q-vertex are an ordered set. The leaves of the tree are labeled by 1-1 correspondence with the members of the set X. The *frontier* of the tree is the permutation of X, obtained by reading the labels on the leaves from left to right.

Two PQ-trees, T, T' are *equivalent*, denoted $T \equiv T'$, if one can be obtained from the other by applying a sequence of the following transformation rules:

- Arbitrarily permute the children of a P-vertex.
- Reverse the order of children of a Q-vertex.

Any frontier obtainable from a PQ-tree equivalent with T is said to be *consistent* with T. the set of permutations on X consistent with T is denoted by:

$$CONSISTENT(T) = \{FRONTIER(T') \mid T \equiv T'\}$$

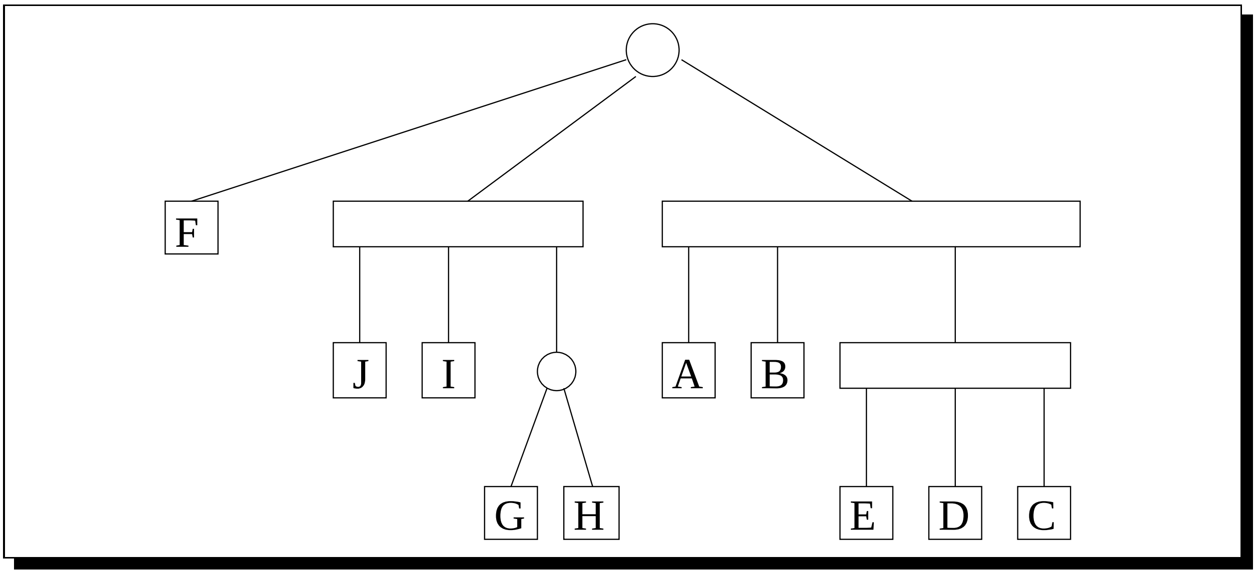


Figure 11.4: a PQ-tree. The FRONTIER here is [F J I G H A B E D C]

The *null tree* T_0 has no nodes and $CONSISTENT(T_0) = \emptyset$. The *universal tree* has one internal P-vertex, the root, and a leaf for every member of X.

The relation between the consecutive arrangement problem and PQ-trees is as follows:

Let φ be a collection of subsets of a set X, and let $\Pi(\varphi)$ be the collection of all permutations π of X such that the members of each $I \in \varphi$ occur consecutively in π.

Theorem 11.7 (Booth - Leuker [1976])

1. *For every collection of subsets* φ *of* X *there exists a PQ-tree* T *such that* $\Pi(\varphi) = CONSISTENT(T)$.
2. *For every PQ-tree there exists a collection of subsets* φ *such that* $\Pi(\varphi) = CONSISTENT(T)$.

The role of the Q-vertices is to restrict the number of permutations by making some of the brother relationships rigid.

The main approach of the algorithm is as follows:

Begin with the collection of all possible permutations. Introduce the subjects one at a time, reducing the tree to those permutations that also satisfy the constraints of the next subset. Below is an implementation of the algorithm:

Procedure CONSECUTIVE(X, φ):
begin
  $T \leftarrow$ universal tree;
  for each $I \in \varphi$ **do**
  $T \leftarrow$ REDUCE(T,I);
return T;
end

The routine REDUCE attempts to apply on the existing tree, one of 11 templates, according to the tree and the set I, and replaces a part of the tree accordingly. The templates are applied from the bottom to the top of the tree. We omit the details here. figure 11.5 gives an example of the execution of the algorithm.

Theorem 11.8 (Booth - Leuker [1976]) *The collection of permutations* $\Pi(\varphi)$ *can be calculated (when implemented with a PQ-tree) in time* $O(\mid \varphi \mid + \mid X \mid + \sum_{I \in \varphi} \mid I \mid)$.

Corollary 11.9 *A* $(0,1)$*-valued matrix* $A_{m \times n}$ *with* f *1's can be checked for the consecutive 1's property in* $O(m+n+f)$ *steps.*

Corollary 11.10 *Interval graphs can be recognized in linear time.*

Definition A (0,1)-valued matrix on satisfies the *circular 1's property* in the columns if its rows can be rearranged such that on each column, the 1's will appear cyclicly consecutive.

Remarks:

- A (0,1)-valued matrix satisfies the circular 1's property iff it satisfies the circular 0's property.

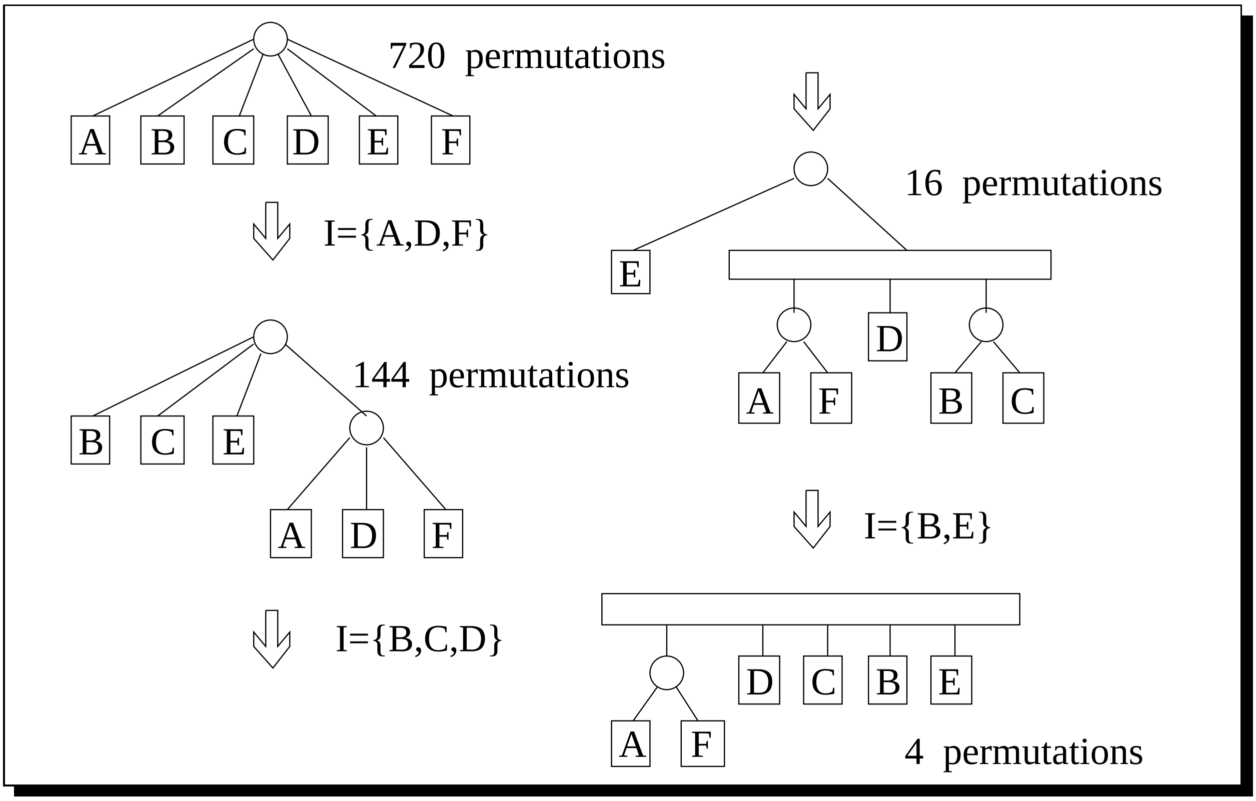


Figure 11.5: Example of execution of CONSECUTIVE

- If a matrix satisfies the circular 1's property for the columns, then after switching 0's with 1's in a column, the property still holds.

Theorem 11.11 (Tucker [1971) *] Given a (0,1)-valued matrix M, switch 0's with 1's in every column that has 1 in the k-th row. Denote the new matrix M'.*
M satisfies the circular 1's property iff M' satisfies the consecutive 1's property.

Proof:

$\Rightarrow$ If M satisfies the circular 1's property then M' satisfies the circular 1's property. Move the k-th row to the first row by rotating the rows cyclicly. In this row there will be 0's only in M'. Hence, M' satisfies the consecutive 1's property.

$\Leftarrow$ If M' satisfies the consecutive 1's property then it satisfies the circular 1's property, and therefore M satisfies the circular 1's property.

■

Theorem 11.12 (Booth [1975]) *A (0,1)-valued matrix $A_{m \times n}$, with f 1's can be checked for the circular 1's property in time $O(m+n+f)$, assuming that the matrix is given as a list of the 1's for each row.*

Proof: Let k be the row with the smallest number of 1's, and let M' be the matrix obtained from M by switching 0's with 1's in every column that has 1 in the k-th row.

M' has at most $2f$ 1's, because in every row the number of 1's is at most doubled (the number of 0's that become 1's is not larger than the original number of 1's in the row). Then apply the algorithm for consecutive 1's on M'. According to the previous theorem, M' satisfies the consecutive 1's property iff M satisfies the circular 1's property. ■

11.3 More about the consecutive 1's property

Theorem 11.13 (Booth [1975]) *Given a (0,1)-valued matrix $M_{r \times c}$ and an integer k, deciding whether or not there exists an $r \times k$ submatrix of M satisfying the consecutive 1's property (or circular 1's property) is NP-complete.*

Proof: It is easy to show that the problem is NP. To prove that the problem is NP complete we can show a reduction from the Hamiltonian-path problem: Given an undirected graph $G = (V, E)$, we produce the adjacency-matrix A of size $|V| \times |E|$ (The rows represent vertices and the columns represent edges). Then it is easy to see that A has a submatrix of size $|V| \times |V|$ that satisfies the consecutive 1's property $\Leftrightarrow$ There exists a Hamiltonian path in G. For circular 1's property, reduce from Hamiltonian cycle instead. ■

Theorem 11.14 (Kou [1977]) *Given a (0,1)-valued matrix, finding a permutation of the rows that minimizes the total number of consecutive blocks of 1's appearing in the columns is NP-complete.*

Theorem 11.15 (Kou [1977]) *Given a (0,1)-valued matrix, finding a permutation of the rows that minimizes the number of times a row must be split into two pieces to obtain consecutive 1's in the columns is NP-complete.*

Proof: First, we will introduce an equivalent problem: Given a finite alphabet Γ and a collection of subsets $s_1, \dots s_k$ such that $s_i \subseteq \Gamma$, we want to find a chain $\sigma \in \Gamma^*$ of length $\leq B$, such that for each s_i there exists a consecutive interval of length $\mid s_i \mid$ in G which contains all letters in s_i.

It is obvious that the problem is NP. to show that it is NP-complete we will show a reduction from the Hamiltonian-path problem: given a graph $G = (V, E)$ we will define $\Gamma = V + E$, $B = 2 \mid E \mid +1$, $s_i = \{< v_i > \cup \bigcup_{x \in Adj(v_i)} (v_i, x)\}$ $\forall i = 1 \dots n$.

- $\Rightarrow$ If there exists a Hamiltonian path, without lost of generality $v_1 \dots v_n$, then we can produce the chain:

  $$v_1, \underbrace{(v_1, \cdot), (v_1, \cdot), \dots, (v_1, v_2)}_{\text{all edges of } v_1}, v_2, \underbrace{(v_2, \cdot), \dots, (v_2, v_3)}_{\text{the other edges of } v_2}, v_3, \dots, v_n, \underbrace{(v_n, \cdot), \dots, (v_n, \cdot)}_{\text{the other edges of } v_n}.$$

  The length of the chain: $\mid V \mid +2 \mid E \mid -(\mid V \mid -1) = B$.

- $\Leftarrow$ If there exists a chain with length $\leq B$, since for every two sets s_i, s_j, $s_i \not\subseteq s_j$, $\mid s_i \cap s_j \mid \leq 1$, it follows that the length of the chain is at least:

  $$\sum_{i=1}^{n} \mid s_i \mid -(n-1) = n + 2 \mid E \mid -(n-1) = B$$

  Therefore: if there exists a chain of length B, it matches a Hamiltonian cycle (there is an edge between each pair of adjacent s_i, s_j).

■

Remarks

- The problem is NP-complete even if $\forall i$, $\mid s_i \mid = 4$ (the same reduction from a 3-regular graph).
- the problem is polynomial if $\forall i$, $\mid s_i \mid = 2$. (exercise)
- Finding a polynomial approximation is an open problem.

Theorem 11.16 (Goldberg, Golumbic, Kaplan, Shamir [1993)] *Given a $(0,1)$-valued matrix, deciding there exists a permutation of the rows that gives at most k blocks of 1's in every column is NP-complete, for every $k \geq 2$.*

Chapter 12

Lecture 12

Advanced Topics in Graph Algorithms		
©Tel-Aviv University		
Spring Semester, 1994		
Lecturer	:	`Ron Shamir`
Scribe	:	`Michael Seltser`
Lecture 12	:	`June 30`

12.1 Temporal Reasoning and Interval algebras

In this section we deal with problems in temporal reasoning. Let us start with an example.

Example 12.1 *A paleonthologist wants to determine the relative span of existence of the species Brontosaurus, Parasaurulophus, Stegosaurus, and Tyrannosaurus. His assumptions:*

- *B and P did not exist simultaneously*
- *P perished before S emerged*
- *S emerged before T and perished not after it*
- *Either P perished before T emerged or P emerged when T perished*
- *B emerged and perished during T's existence*

Is there a scenario satisfying all his assumptions ?

See figure 12.1 for a possible scenario consistent with the assumptions.

Allen in 1983 has defined a model for temporal logic whose elements are time intervals (corresponding to events that happen during a certain period of time) and used the following 13 primitive relations:

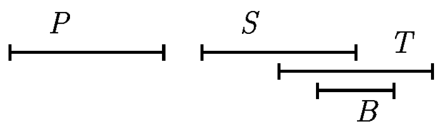


Figure 12.1: A consistent scenario

$$\{\prec, \succ, m, m^{-1}, o, o^{-1}, s, s^{-1}, f, f^{-1}, d, d^{-1}, \equiv\}$$

to express the relative position of two intervals (See Figure 12.2). We will call this *the 13-valued interval algebra* $\mathcal{A}_{13}$.

RELATION	NOTATION	INTERPRETATION
x before y	$\prec$	
y after x	$\succ$	
x meets y	m	
y met-by x	m^{-1}	
x overlaps y	o	
y overlapped-by x	o^{-1}	
x starts y	s	
y started-by x	s^{-1}	
x during y	d	
y includes x	d^{-1}	
x finishes y	f	
y finished-by x	f^{-1}	
x equals y	$\equiv$	

Figure 12.2: The 13-valued interval algebra $\mathcal{A}_{13}$. Single line: x interval. Double line: y interval.

From these we define the 3-valued interval algebra $\mathcal{A}_3$ whose elements are called its *atomic relations*

$$\mathcal{A}_3 \;:\; \{\prec, \succ, \cap\}$$

Here $X \cap Y$ denotes that events (intervals) x and y have a non-empty intersection, i.e., any one of the eleven relations excluding $\prec$ and $\succ$ in Allen's algebra may hold between x and y. The choice of which algebra to use depends on the nature of the application.

12.2 Interval satisfiability problems

We now define our temporal reasoning problems in the context of constraint satisfiability. For each pair of events x and y, let $D(x,y)$ be a set of atomic relations in the algebra $\mathcal{A}_i$. The semantics here is that we do not know precisely the relationship between x and y, but it must be one of those in the set $D(x,y)$. (In the language of constraint satisfiability, there is a variable $v(x,y)$ representing the relation between x and y, and its value must be taken from those in the set $D(x,y)$ corresponding to its domain.)

For example, we read $D(x,y) = \{\prec, o\}$ as x is either before or overlaps y. We also call a set of atomic relations a *relation set.* Occasionally we shall use the notation $xD(x,y)y$ for short. For example, $D(x,y) = \{\prec, d^{-1}, \equiv\}$ and $x\{\prec, d^{-1}, \equiv\}y$ both mean "either x ends before y starts, or y is during x, or the two are equal". We omit braces and commas when there is no ambiguity, e.g., $x{\prec}y$ or $x{\cap}y$.

In particular, relation sets in $\mathcal{A}_3$ will be presented by concatenation of the atomic relations. Hence, $x{\prec\succ}y$ denotes x and y are disjoint, $x{\prec\cap}y$ denotes $x\{\prec, \cap\}y$, etc.

An *instance* (or *input*) for all the problems studied in this paper consists of n events, and a relation set $D(x,y)$ for every pair of distinct events x,y. Without loss of generality, we assume that for each pair of events x and y, the relation sets $D(x,y)$ and $D(y,x)$ given as input are consistent, i.e., for each atomic relation R, $R \in D(x,y) \Leftrightarrow R^{-1} \in D(y,x)$. Otherwise, it is a simple matter to restrict the relation sets further so that they satisfy these properties or are shown to be unsatisfiable. Hence, we can assume that the input contains for each pair of events x,y only one relation set, say $D(x,y)$, and that the other relation set $D(y,x)$ is given implicitly by the above rule.

Remark 12.2 *The* size *of an instance is the number s of non-trivial relation sets in it, and is linearly equivalent to the binary input length for each $\mathcal{A}_i$. We will assume that relation sets which are trivial (i.e., contain all the atomic relations) are not listed explicitly as part of a problem input, and that each event is involved in at least one non-trivial relation set.*

Hence, the input size s of a problem with n events satisfies $s = O(n^2)$ and $s = \Omega(n)$.

An *interval realization* for the instance $\{D(x,y)\}$ is an assignment of intervals on the real line to events, so that for each pair of events, one of the atomic relations in their relation set holds. Since the algebras discussed here are concerned with topological properties only, a realization can be viewed also as a complete weak order of the interval endpoints, thereby identifying all realizations which differ in metric only. Two realizations are *distinct* if the orders of the endpoints in them differ. The input data $\{D(x,y)\}$ is *consistent* (or *satisfiable*) if it admits at least one realization.

The *interval satisfiability problem* (ISAT) is determining the existence of (and finding) one instantiation that is consistent with the input data $\{D(x,y)\}$. Occasionally, we shall make an additional distinction between the *decision version* of ISAT, whose output is only a yes/no answer, and the *constructive version*, whose output is the answer 'no' if the input is inconsistent and a solution if it is consistent.

The *minimal labeling problem* (MLP) is to determine the minimal sets $D'(x,y) \subseteq D(x,y)$ such that the set of realizations is unchanged, and every remaining atomic relation participates in some solution.

Example 12.3 $x\{\prec, m, o\}y$, $y\{\prec, \equiv, \succ\}z$, $z\{f, s\}x$.
Here xoy, $y{\succ}z$, zsx, *and* $x{\prec}y$, $y{\succ}z$, zfx *are both consistent with the input, as shown in Figure 12.3. On the other hand,* $y{\prec}z$ *and* $z \equiv y$ *are impossible. The minimal labeling for this problem is* $x\{\prec, m, o\}y$, $y{\succ}z$, $z\{f, s\}x$.

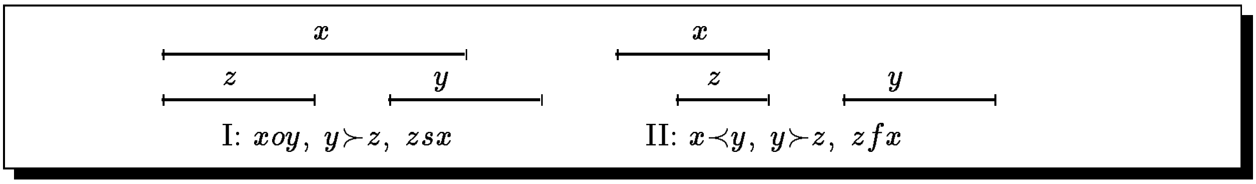


Figure 12.3: Two interval realizations for Example 12.2

The *all-consistent-realizations problem* (ACRP) is finding of all the consistent realizations. Here there may be exponentially many solutions, but we would like to generate them in time polynomial per each new realization.

A *fragment* of an algebra ($\mathcal{A}_3$ or $\mathcal{A}_{13}$) is a collection of relation sets in that algebra. A problem is restricted to a fragment F if all relation sets participating (implicitly or explicitly) in the instance belong to F. We will denote the problem A restricted to fragment F by $A(F)$.

Example 12.4 *In the problem ISAT* $(\prec, \succ)$, *for every pair of events* x, y *either* $x{\prec}y$ *or* $x{\succ}y$. *Clearly the input is consistent iff the relation* $\prec$ *is a linear order, as no intersections are allowed. This can be checked in linear time.*

Example 12.5 *ISAT* $(\prec, \succ, \prec\succ)$ *is polynomial by a similar argument. Here the instance is consistent iff* $\prec$ *is a partial order.*

Example 12.6 *ISAT* $(\prec\succ, \cap)$ *is equivalent to interval graph recognition: Form a graph* G *with a vertex for each event with an edge between* x *and* y *iff* $x{\cap}y$. *The instance has a realization iff* G *is an interval graph. Hence the problem is solvable in linear time by the algorithm of Booth and Leuker (1976).*

Theorem 12.7 (Golumbic - Shamir, 93) *For each of* $\mathcal{A}_{13}, \mathcal{A}_3$, *MLP and ACRP are polynomially equivalent to ISAT.*

J.Allen defined a constraint propagation polynomial heuristic for solving ISAT on $\mathcal{A}_{13}$. If the algorithm returns a negative answer then it is correct, but a positive answer may be incorrect. He conjectured that ISAT is NP-complete on $\mathcal{A}_{13}$. This was proved by Vilain and Kautz (1986). Vilain, Kautz and van Beek (1989) have shown that Allen's algorithm always gives the correct answer if the input data is restricted to a fragment of $\mathcal{A}_{13}$. Nökel (1988) has given another polynomial algorithm on a different fragment of $\mathcal{A}_{13}$.

Applications of interval satisfiability problems are numerous. They include planning, molecular biology (DNA), knowledge representation, natural language processing, information systems (databases), artificial intelligence, behavioral psychology, archaeology (seriation), graph theory and others.

12.3 Interval sandwich problems and ISAT

Let E^1 and E^2 be two disjoint sets of edges defined on the same vertex-set V. A graph $G(V, E)$ with $E^1 \subseteq E \subseteq E^1 \cup E^2$ is called a *sandwich graph* for (E^1, E^2). The interval graph sandwich (IGS) problem is the following:

Interval Graph Sandwich problem:

INPUT: Two graphs $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$ with the same set of vertices, and disjoint edge-sets E^1 and E^2.

QUESTION: Is there a sandwich graph for (E^1, E^2) which is an interval graph?

Define $F = \overline{\{E^1 \cup E^2\}}$ (i. e., F is the set of non-edges in the graph $(V, E^1 \cup E^2)$.) When $E^1 = \emptyset$ or $F = \emptyset$, the answer is trivially yes. When $E^2 = \emptyset$, the problem is equivalent to recognizing if $G(V, E^1)$ is interval and thus is polynomial by the algorithm of Booth and Lueker. We shall show that in the general case, the problem is NP-complete.

Given a set of intervals on the real line, one can define a partial order on the intervals by $a \prec b$ if and only if the interval a is completely to the left of interval b. We use this *interval order* to give a reduction from the following problem:

BETWEENNESS:

INPUT: A set of elements $S = \{a_1, \ldots, a_n\}$, and a set $T = \{T_1, \ldots, T_m\}$ of ordered triplets of elements from S, where $T_i = (a_{i_1}, a_{i_2}, a_{i_3})$ $i = 1, \ldots, m$.

QUESTION: Does there exist a one-to-one function $f : S \to \{1, 2, \ldots, n\}$ such that either $f(a_{i_1}) < f(a_{i_2}) < f(a_{i_3})$ or $f(a_{i_1}) > f(a_{i_2}) > f(a_{i_3})$ for $i = 1, \ldots, m$?

The problem was shown to be NP-complete by Opatrny (1980).

Theorem 12.8 *The interval graph sandwich problem is NP-complete.*

Proof: Membership in NP follows since a given interval realization can be checked to fit the data in polynomial time. Given an instance of BETWEENNESS, we transform it to an interval sandwich instance as follows: Define a vertex v_i for each element a_i, and two vertices x_j^1 and x_j^2 for triplet T_j. The vertex set is $V = \{v_1, \ldots, v_n\} \cup \{x_j^1, x_j^2 \mid j = 1, \ldots, m\}$. The edges sets are

$$E^1 = \{(v_{i_1}, x_i^1), (x_i^1, v_{i_2}), (v_{i_2}, x_i^2), (x_i^2, v_{i_3}) \mid i = 1, \ldots, m\}$$

$$F = \{(v_i v_j) \mid i \neq j\} \cup \{(v_{i_1}, x_i^2), (v_{i_3}, x_i^1), (x_i^1, x_i^2) \mid i = 1, \ldots, m\}$$

In other words, to each triplet corresponds a 5-chain (see figure 12.4(a)) which should also appear as a 5-chain in the sandwich graph. In addition, all the v_i-s should form an independent set in that graph. The reduction is clearly polynomial.

The key to the reduction is the following simple observation: in any interval realization of a 5-chain (a, x, b, y, c), either $a \prec b \prec c$ or $c \prec b \prec a$ (see figure 12.4(b)).

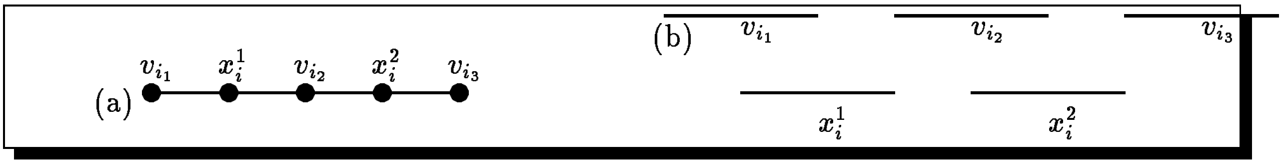


Figure 12.4: (a) The 5-chain $(v_{i_1}, x_i^1, v_{i_2}, x_i^2, v_{i_3})$. (b) An interval realization of the 5-chain.

Suppose there is an ordering f of the set S which satisfies the betweenness conditions. Create an interval realization for an interval sandwich as follows: Draw n disjoint intervals corresponding to the v_i-s in the same order as the a_i-s appear in f. That is, $I(v_i) \prec I(v_j)$ if and only if $f(a_i) < f(a_j)$. Since the order conditions on triplets are satisfied, additional intervals corresponding to x_i^1 and x_i^2 can be added for each triplet T_i such that the subgraph induced by the five vertices $\{v_{i_1}, v_{i_2}, v_{i_3}, x_i^1, x_i^2\}$ is a 5-chain. The resulting interval graph on V is a sandwich graph.

For the converse, suppose there exists an interval sandwich solution with realization I. Since in that graph the set $\{v_1, \ldots, v_n\}$ induces an independent set, a complete ordering on S is generated: Number the elements in the order of their intervals from left to right, namely, $f(a_i) < f(a_j)$ if and only if $I(v_i) \prec I(v_j)$. Since for each triplet T_i the five vertices $\{v_{i_1}, v_{i_2}, v_{i_3}, x_i^1, x_i^2\}$ must induce a 5-chain in every sandwich graph, by the above observation, every triplet satisfies the betweenness condition. ∎

Interestingly, we can also show that a restricted version of the above problem is NP-complete: A function $c : V \to Z$ is called a *coloring* of the graph $G = (V, E)$ if for every $(u, v) \in E$, $c(u) \neq c(v)$. The problem is defined as follows:

COLORED INTERVAL SANDWICH:
INPUT: A graph $G = (V, E)$ and a coloring c of G.
QUESTION: Does there exist a set E' so that $E \subseteq E'$, $G' = (V, E')$ is an interval graph and c is also a coloring for G'?

Notice that this is a special case of the sandwich problem where $E^1 = E$ and $F = \{(u,v) \mid c(u) = c(v)\}$ is given implicitly by the coloring.

Theorem 12.9 *The colored interval sandwich problem is NP-complete.*

Proof: Reduction from BETWEENNESS, where the vertex set and $E = E^1$ is defined as in Theorem 12.8 and the coloring is $c(x_i^1) = c(x_i^2) = i$ $(i = 1, \ldots, m)$ and $c(v_1) = c(v_2) = \cdots = c(v_n) = m + 1$. A slight modification of the arguments in Theorem 12.8 proves the result. ■

Remark 12.10 *Golumbic, Kaplan, and Shamir (1993) have shown that the analog problems to Theorem 12.8 and 12.9, with the additional condition that all intervals have the same length, are still NP-complete.*

Corollary 12.11 *ISAT($\prec\succ, \cap, \prec\cap\succ$) is NP-complete.*
Proof: *Just consider $V = \{$ events $\}$, $(i,j) \in E^1 \Leftrightarrow i \cap j$, $(i,j) \in F \Leftrightarrow i \prec\succ j$, $(i,j) \in E^2 \Leftrightarrow i \prec\cap\succ j$.* ■

Corollary 12.12 *ISAT($\mathcal{A}_3$) is NP-complete.*
Proof: *Follows from Corollary 12.11 since ISAT($\prec\succ, \cap, \prec\cap\succ$) is a restriction of ISAT($\mathcal{A}_3$).* ■

Corollary 12.13 *ISAT, MLP, ACRP are NP-hard for $\mathcal{A}_3$ and $\mathcal{A}_{13}$.*

Theorem 12.14 *ISAT($\prec$, $\succ$, $\prec\succ$, $\cap$) is polynomial.*

Proof: exercise. ■

12.4 A linear time algorithm for ISAT(Δ_1)

In this section we deal with ISAT problems on the fragment:

$$\Delta_1 = \{\prec, \succ, \cap, \prec\cap, \cap\succ, \prec\cap\succ\}.$$

That is, $\mathcal{A}_3 - \prec\succ$. We shall give efficient algorithms for ISAT, MLP and ARP on this fragment, and they will apply immediately to any subfragment of Δ_1. Hence, by excluding just a *single* relation set from $\mathcal{A}_3$ the problems become tractable.

The polynomial algorithm for ISAT(Δ_1) utilizes a directed graph, which will be described later, with vertices corresponding to the endpoints of event intervals and arcs representing the relative order of endpoints. The key observation here is that every relation in Δ_1 is equivalent to a certain *order requirement* between a pair (or two pairs) of such endpoints. This observation is proved below:

Lemma 12.15 *Let i and j be the event intervals $[l_i, r_i]$ and $[l_j, r_j]$, respectively. In each of the following cases, the intervals satisfy the set of relations if and only if their endpoints satisfy the corresponding inequalities:*

$$i \prec j \;\Leftrightarrow\; r_i < l_j \qquad\qquad i \succ j \;\Leftrightarrow\; r_j < l_i \tag{12.1}$$

$$i \prec\cap j \;\Leftrightarrow\; l_i \leq r_j \qquad\qquad i \cap\succ j \;\Leftrightarrow\; l_j \leq r_i \tag{12.2}$$

$$i \cap j \;\Leftrightarrow\; l_i \leq r_j \text{ and } l_j \leq r_i \tag{12.3}$$

If $i \prec\cap\succ j$, no constraint is imposed.

Proof: (1) is immediate. The conditions in (2) are the negation of those in (1). (3) is equivalent to the intersection of the two conditions in (2). ∎

The graph is now constructed as follows: For an instance J of ISAT(Δ_1) with n events, form a directed graph $G(J) = G(V, E)$ with vertex set $V = \{r_1, \ldots r_n, l_1, \ldots l_n\}$. The arc set E consists of two disjoint subsets, E_0 and E_1.

The former will represent weak inequality and the latter will represent strict inequality between pairs of endpoints.

The arcs are defined as follows:

$$(l_i, r_i) \in E_0 \qquad\qquad i = 1, \ldots, n \tag{12.4}$$

$$(r_i, l_j) \in E_1 \qquad\qquad \forall i, j \text{ s.t. } i \prec j \tag{12.5}$$

$$(l_i, r_j) \in E_0 \qquad\qquad \forall i, j \text{ s.t. } i \prec\cap j \tag{12.6}$$

$$(l_i, r_j) \in E_0 \text{ and } (l_j, r_i) \in E_0 \qquad\qquad \forall i, j \text{ s.t. } i \cap j \tag{12.7}$$

For pairs i, j with the relation $i \prec\cap\succ j$, no arc is introduced. Define now $E = E_0 \cup E_1$. We call the arcs in E_0 (resp., E_1) the *weak arcs* (resp., *strict arcs*). Note that the graph G is bipartite. Denote the two parts of the vertex set by $R = \{r_1, \ldots, r_n\}$ and $L = \{l_1, \ldots, l_n\}$, and call an arc an *RL-arcs* (resp., *LR-arc*) if it is directed from R to L (resp., from L to R).

Remark 12.16 *In the graph G all the RL-arcs are strict and all the LR-arcs are weak. Hence, we need not record explicitly the type of each arc since it is implied by its direction.*

Lemma 12.17 *Every cycle in G contains a strict arc.*

Algorithm: Δ_1-*CONSISTENCY*
begin
1. construct $G(J)$ according to rules (12.4) - (12.7).
2. if $G(J)$ contains a cycle - J is not satisfiable. Otherwise, it is.

end *construct G-decomposition*

Figure 12.5: Algorithm to construct $Delta_1$-CONSISTENCY

Proof: Since G is bipartite, a cycle must contain vertices both from R and from L. In particular, it must contain an RL-arc, so the claim follows by Remark 12.16. ∎

An algorithm for solving ISAT(Δ_1) is described in figure 12.5

Lemma 12.18 *Suppose $G(J) = (V, E)$ is acyclic. Then a linear order on V is consistent with the partial order $G(J)$ if and only if it is a realization of J.*

Proof: Take any linear order P which extends the partial order G. P is an ordering of all the endpoints, which by Lemma 12.15 satisfies all the relations in the input, so P gives a realization for J. On the other hand, every realization of J gives a linear order of the endpoints, in which each of the input relations must be satisfied, by Lemma 12.15. Hence, the linear order must be consistent with the partial order $G(J)$. ∎

Theorem 12.19 *Algorithm Δ_1-$CONSISTENCY$ correctly recognizes if an instance of ISAT(Δ_1) is satisfiable in linear time.*

Proof: Validity: By Lemma 12.15, each arc reflects the order relation of a pair of interval endpoints as prescribed by the input relations.

If G contains a cycle, then by Lemma 12.17 that cycle contains a strict arc. Hence, that cycle must satisfy $r_i < l_j \leq \ldots \leq r_i$, which implies that the input relations cannot be satisfied. In case G is acyclic,

Lemma 12.18 implies that J is satisfiable.

Complexity: Constructing $G(J)$ requires $O(s)$ steps, where s is the number of input relation sets, since the effort is constant per relation. Checking if G is acyclic can be done, for example, by depth first search, in time linear in $|E|$, the number of arcs.

Since $|E| \leq 2s + n$, using remark 12.2 we conclude that the algorithm is linear. ∎

In a similar way one can construct algorithms for MLP(Δ_1) and ACRP(Δ_1) in the following manner: An algorithm for MLP is obtained by calculation of transitive closure of G. The algorithm for ACRP(Δ_1) is based on finding of all linear extensions of G.

Remark 12.20 *ISAT is polynomial if all events are points ($\mathcal{A}_3 = \mathcal{A}_{13}$ in that case).*

Proof: Exercise. ■

Remark 12.21 *One can handle additional metric constraints in $\mathcal{A}_3 - \prec\succ$, by solving the linear programming problem defined by (12.1) - (12.3) with additional metric constraints on the endpoints.*

		$\cap$	$\prec\succ$	$\prec\succ, \cap$
	–	T	T	Poly interval graph
$\prec$	Poly linear order	Poly interval order	Poly acyclic graph	Poly
$\prec\cap$	T	T	NPC*	NPC*
$\prec\cap\succ$	T	T	T	NPC interval sandwich
$\prec, \prec\cap$	Poly	Poly	NPC*	NPC*
$\prec, \prec\cap\succ$	Poly	Poly	Poly	NPC
$\prec\cap, \prec\cap\succ$	T	T	NPC	NPC
$\prec, \prec\cap, \prec\cap\succ$	Poly	Poly	NPC	NPC

Figure 12.6: Complexity of ISAT on all fragments of $\mathcal{A}_3$. T: satisfied by setting all intervals disjoint or identical. The four asterisk cases were shown to be NPC by Webber (1994).

Figure 12.6 summarizes results from Golumbic - Shamir (1993) and Webber (1994) on the complexity of ISAT on all fragments of $\mathcal{A}_3$.

12.5 Bandwidth, Pathwidth and Proper Pathwidth

We now switch to different problems related to interval and proper interval graphs. The motivation studying to these problems arises in physical mapping of DNA, a major problem in the Human Genome Project.

Lemma 12.22 *Every interval graph (and every proper interval graph) has an interval (resp., proper interval) representation on the real line in which all endpoints are distinct, and the left endpoints are assigned to the integers* $1, 2, \ldots, |V|$.

We shall call such a representation *canonical.*

Definition Pathwidth and Proper Pathwidth A *path decomposition* of a given graph $G = (V, E)$, is a sequence of subsets of V, $X = (X_1, \ldots, X_l)$ such that

(1) $V = \cup_i X_i$

(2) For each edge $(u, v) \in E$, there exists some $i \in \{1, \ldots, l\}$ so that both u and v belong to X_i.

(3) For each $v \in V$ there exist some $s(v), e(v) \in \{1, \ldots, l\}$ so that $s(v) \leq e(v)$, and $v \in X_j$ if and only if $j \in \{s(v), s(v)+1, \ldots, e(v)\}$.

The *width* of X is defined by $pw_X(G) = \max\{|X_i| \mid i = 1, \ldots, l\} - 1$. The *pathwidth* of G, denoted $pw(G)$, is the minimum value of $pw_X(G)$ over all path decompositions, i.e., $pw(G) = \min\{pw_X(G) \mid X \text{ is a path decomposition of } G\}$. The notion of pathwidth was originally introduced by Robertson and Seymour (1983).

Lemma 12.23 (e.g., Mohring, 91) *For any path decomposition $X = (X_1, \ldots, X_l)$ of a graph G, every clique in G must be contained in some X_i.*

The *PATHWIDTH* problem is to decide for a given graph G and a given integer k if $pw(G) \leq k$. Equivalent problems arise in various areas, including VLSI layout, processor management, node searching and vertex separation.

Lemma 12.24 *For every graph G, the pathwidth of G is one less than the least clique size of any interval supergraph of G.*

We introduce here the notion of a *proper path decomposition* of G; It is as a path decomposition X which satisfies (1)-(3) and also:

(4) For every $u, v \in V$, $\{s(u), s(u)+1, \ldots, e(u)\} \not\subset \{s(v), s(v)+1, \ldots, e(v)\}$.

(As usual, $\subset$ denotes strict containment.) The *proper pathwidth* of G, denoted $ppw(G)$, is the minimum value of $pw_X(G)$ over all proper path decompositions of G, namely $ppw(G) = \min\{pw_X(G) \mid X \text{ is a proper path decomposition of } G\}$.

Clearly, $pw(G) \leq ppw(G)$, but note that $pw(G)$ and $ppw(G)$ can differ dramatically: For the star graph with $2n+1$ vertices, $G = K_{1,2n}$, $pw(G) = 1$ but $ppw(G) = n$. An easy way to see the last equality is using the characterizations we shall provide in the next section.

Bandwidth: Another measure of graph width which we shall use is bandwidth: For $G = (V, E)$ with $|V| = n$, a linear *layout* of the graph is a 1-1 function $L : V \to \{1, \ldots, n\}$. The *bandwidth of L* is $bw_L(G) = \max_{(u,v) \in E}\{|L(u) - L(v)|\}$. The *bandwidth* of G is the minimum bandwidth over all layouts of G, namely, $bw(G) = \min\{bw_L(G) \mid L \text{ is a layout of } G\}$. The *BANDWIDTH* problem is to decide for a given graph G and integer k, if $bw(G) \leq k$. This problem has been studied intensely because of its application to sparse matrix algebra. It is known to be NP-complete even for binary trees (Garey, Graham, Johnson, and Knuth, 1978) and for caterpillars with hair length at most three (Monien, 1986). On the other hand, it is solvable in $O(n^k)$ for arbitrary k and in linear time for $k = 2$ (Garey, Graham, Johnson, and Knuth, 1978).

Lemma 12.25 (Kaplan - Shamir, 94) *For every graph G, the proper pathwidth of G is one less than the least clique size of any proper interval supergraph of G.*

Proof: Suppose $X = (X_1, \ldots, X_l)$ is a proper path decomposition of G and $pw_X(G) = ppw(G) = k$. On the linear order $1 < 2 < \cdots < l$, let $I(v)$ be the interval $[s(v), e(v)]$. Then $\{I(v)\}_{v \in V}$ is a representation of an interval graph G'. G' is proper since no interval is strictly contained in the other, by property (4) in the definition of a proper path decomposition. G' is a supergraph of G, by property (2). Since X is also a path decomposition of G', Lemma 12.23 implies that every clique in G' must be contained in some X_i. Hence, $\omega(G') \leq k + 1$.

Conversely, suppose $k + 1$ is the least clique size of any proper interval supergraph of G. Let $\{I(v)\}_{v \in V}$ be a proper interval representation of such supergraph $G' = (V, E')$ with $\omega(G') = k + 1$, in which all endpoints are distinct, and $I(v) = [l(v), r(v)]$. Order the 2n endpoints from left to right $p_1, \ldots p_{2n}$. Define $X_i = \{v \in V \mid p_i \in I(v)\}$ for $i = 1, \ldots, 2n$. (Here and throughout $n = |V|$). We claim that $(X_1, \ldots, X_{2n})$ is a proper path decomposition of G. Checking the four requirements from a proper path decomposition one can observe that (1) is immediate. (2) follows since $E \subseteq E'$, and if $(u, v) \in E'$ and $l(u) < l(v)$ then $l(v) \in I(v) \cap I(u)$. In other words, every edge in E' is represented in some X_i. Since G' is a proper interval graph, it follows that $s(v) = l(v)$ and $e(v) = r(v)$ satisfy (3) and (4). Since the clique size of G' is $k + 1$, each point p_i can meet at most $k + 1$ intervals in the representation. Hence, $\max_i |X_i| \leq k + 1$ so $ppw(G) \leq k$. ∎

Theorem 12.26 (Kaplan - Shamir, 94) *The bandwidth of a graph equals its proper pathwidth.*

Proof: For the graph $G = (V, E)$ with $ppw(G) = k$, by Lemma 12.25 there exists a supergraph $G' = (V, E')$ of G which is proper interval with clique size $k + 1$. Let $\{I(v)\}_{v \in V}$ be a canonical representation for G', where $I(v) = [l(v), r(v)]$. Define a layout L on G by $L(u) = l(u)$ for all $u \in V$. Suppose $L(u) - L(v) > k$ for some $(u, v) \in E$, where $L(v) = i$. Then $\{L^{-1}(i), \ldots, L^{-1}(i + k + 1)\}$ form a clique of size $k + 2$ in G', a contradiction. Hence, $bw(G) \leq bw_L(G) \leq ppw(G)$.

Conversely, let L be a layout of G so that $bw_L(G) = bw(G) = k$. Form a set of equal-length intervals on the real line $\{I(v)\}_{v \in V}$ where $I(v) = [L(v), L(v) + k]$. Let $G' = (V, E')$ be the intersection graph of that set of intervals. Clearly $E' \supseteq E$ since if $(u, v) \in E$ then $|L(u) - L(v)| \leq k$, so $I(u) \cap I(v) \neq \emptyset$. Since the maximum clique size for G' is at most $k + 1$, using Lemma 12.25, $ppw(G) \leq k$. Hence, $bw(G) \geq ppw(G)$. ∎

This interesting - and somewhat surprising - equivalence between bandwidth and proper pathwidth allows us to draw several immediate conclusions from the results on bandwidth:

Corollary 12.27

A. *Finding the proper pathwidth is NP-hard, even for binary trees and for caterpillars with*

hair length at most three.
B. *Problem A can be solved in* $O(f(k)n^{k-1})$ *time, and in particular is polynomial when* k *is fixed.*
C. *Deciding whether the proper pathwidth of a graph is not greater than two can be done in linear time.*

These results follow immediately from known results on bandwidth, using Theorem 12.26.

We finish by mentioning two parametric results. The first deals with the unit interval sandwich problem mentioned above:

Theorem 12.28 (Kaplan-Shamir 94) *Deciding if a sandwich instance admits a proper interval sandwich graph with clique size at most* k *can be done in* $O(C_k n^k)$ *time, and in particular in polynomial time when* k *is fixed. Furthermore, it cannot be solved in* $O(C_k n^\alpha)$ *where* α *is independent of* k*, unless deciding if a given graph has an independent set of size* k *can be solved in* $O(C_k' n^{\alpha'})$.

In contrast to that negative result, the completion problem for proper interval graphs is tractable, in the parametric sense:

Theorem 12.29 (Kaplan, Shamir, Tarjan 94) *Given a graph* $G = (V, E)$*, one can decide if there exists a proper interval graph* $G' = (V, E')$ *with* $E \subseteq E'$ *and* $|E' \setminus E| \leq k$ *in time* $O(C_k(|V| + |E|))$.

Index