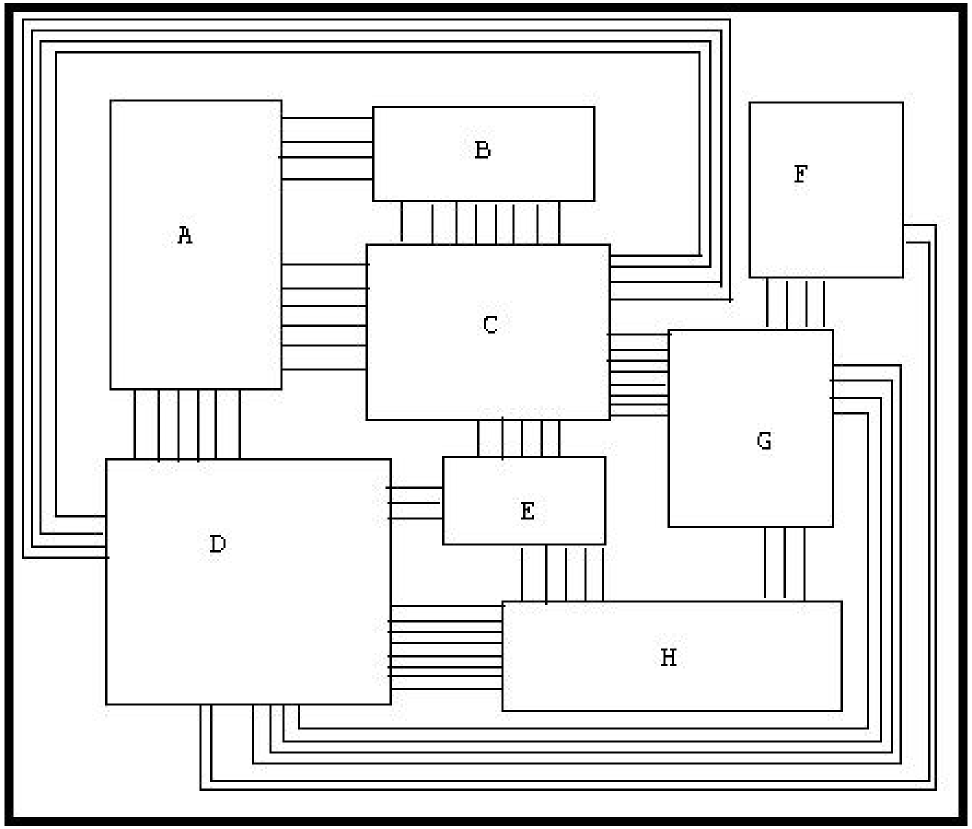
Graph Algorithms in VLSI CAD

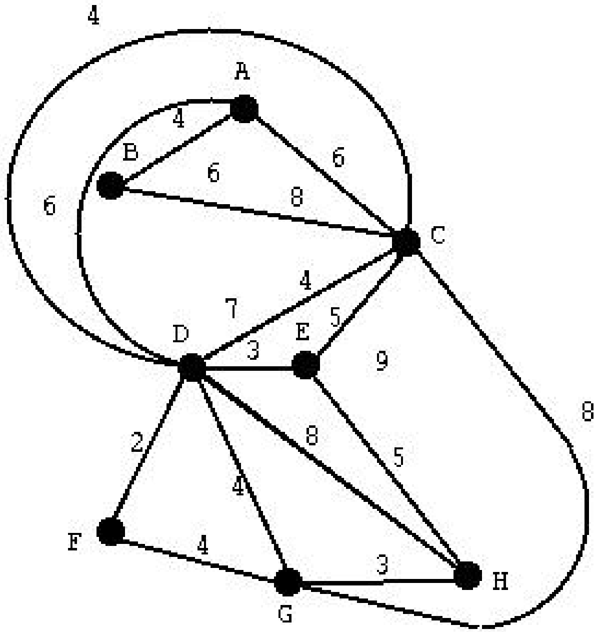
Susmita Sur-Kolay
I. S. I. Kolkata

Motivation

- A VLSI layout

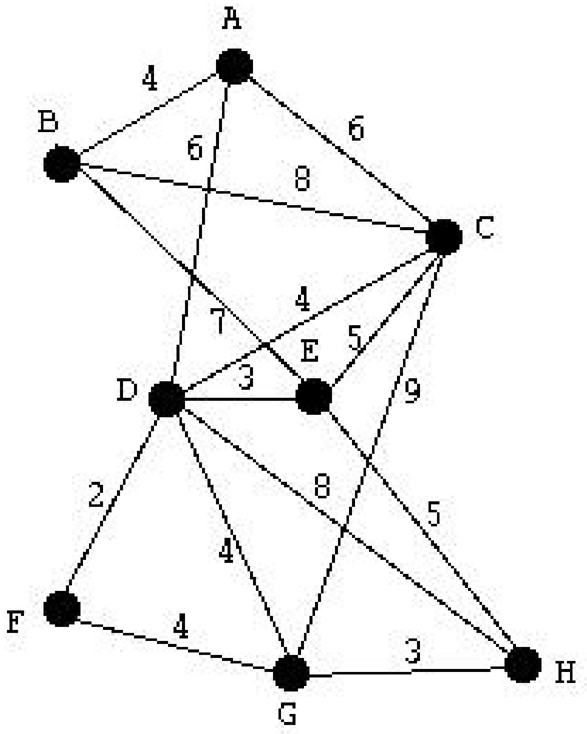


Its graph representation

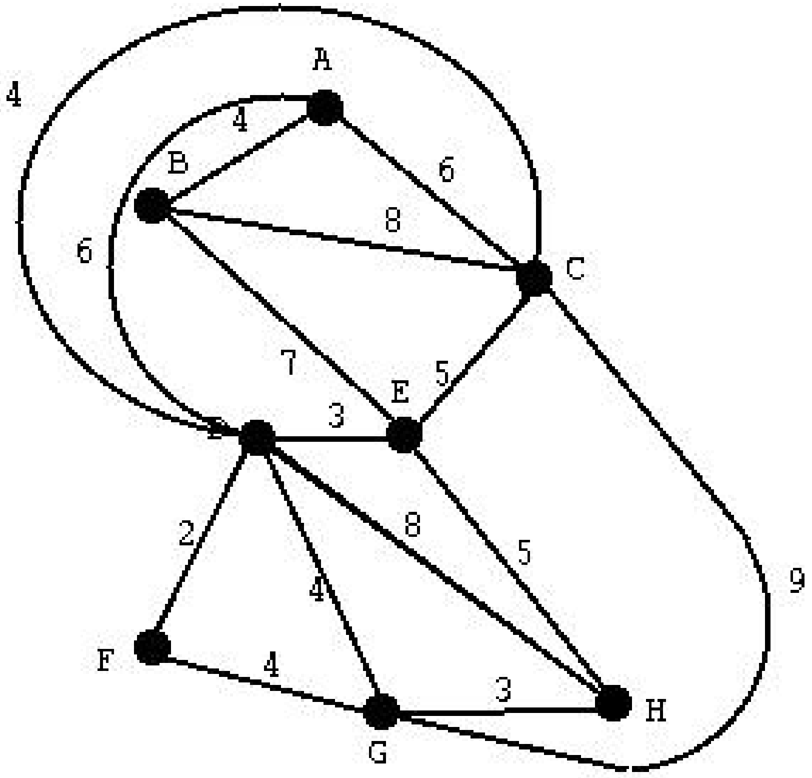


Layout Problem

Interconnection information among circuit modules

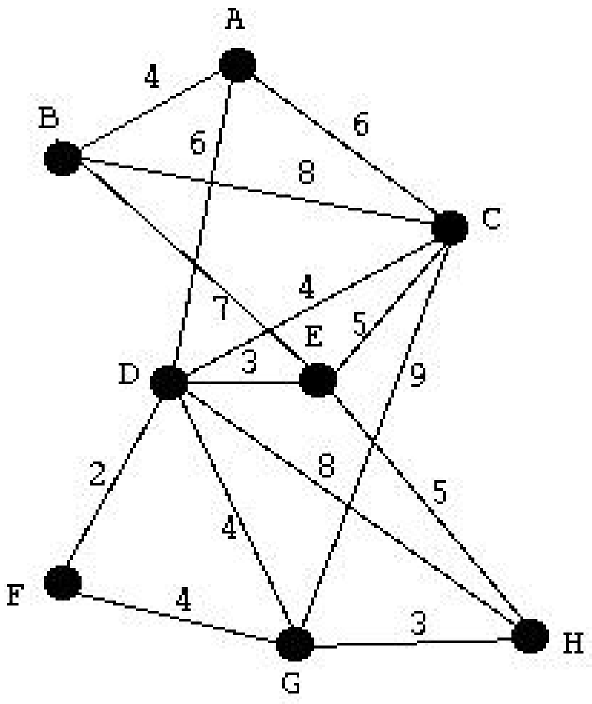


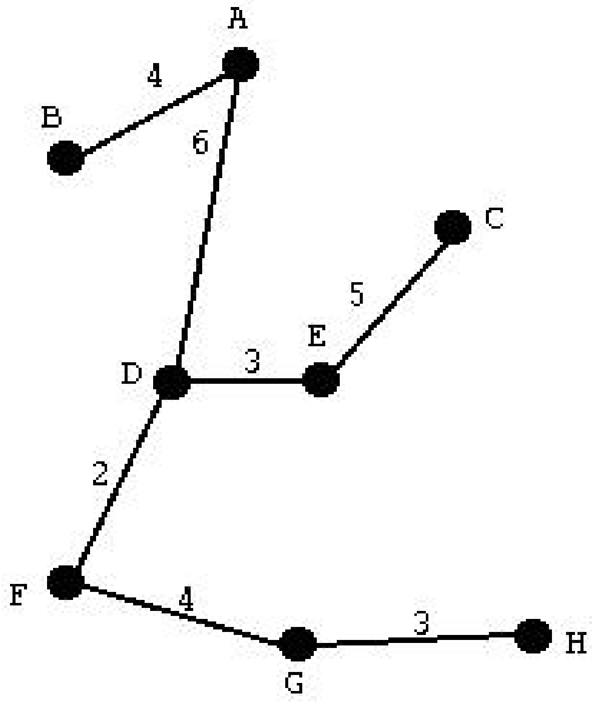
maximal planar subgraph extraction



Routing Problem

- An electrical signal net connected with a set of modules
- Edge costs indicate the cost of connections
- Objective : Connect all the modules with minimum cost





Data Structures and Algorithms for VLSI Design

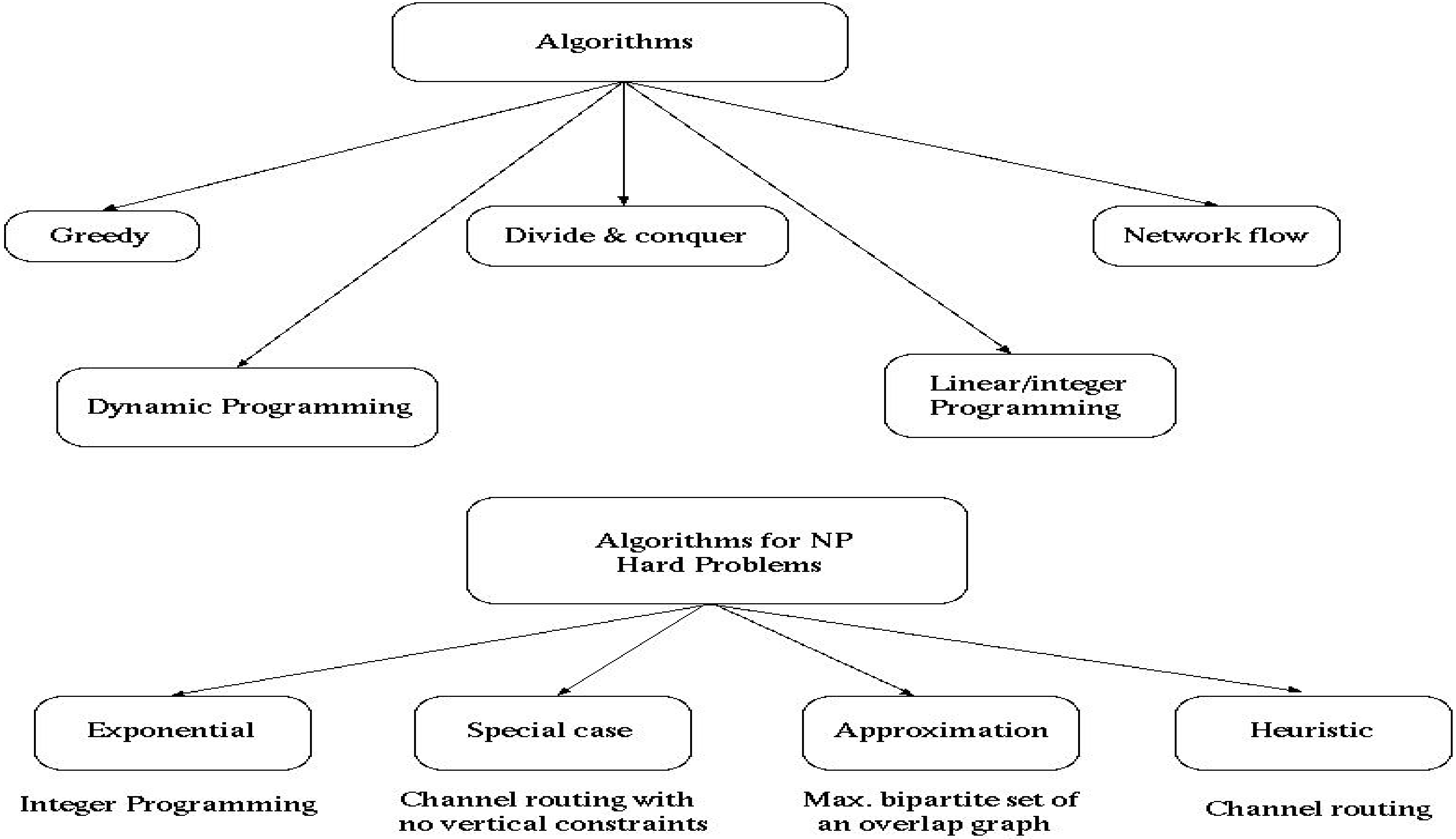
Objectives:

- To understand how a layout is represented and manipulated
- To review basic graph algorithms
- Abstraction of VLSI design problems as an optimization/search problems in appropriate graphs

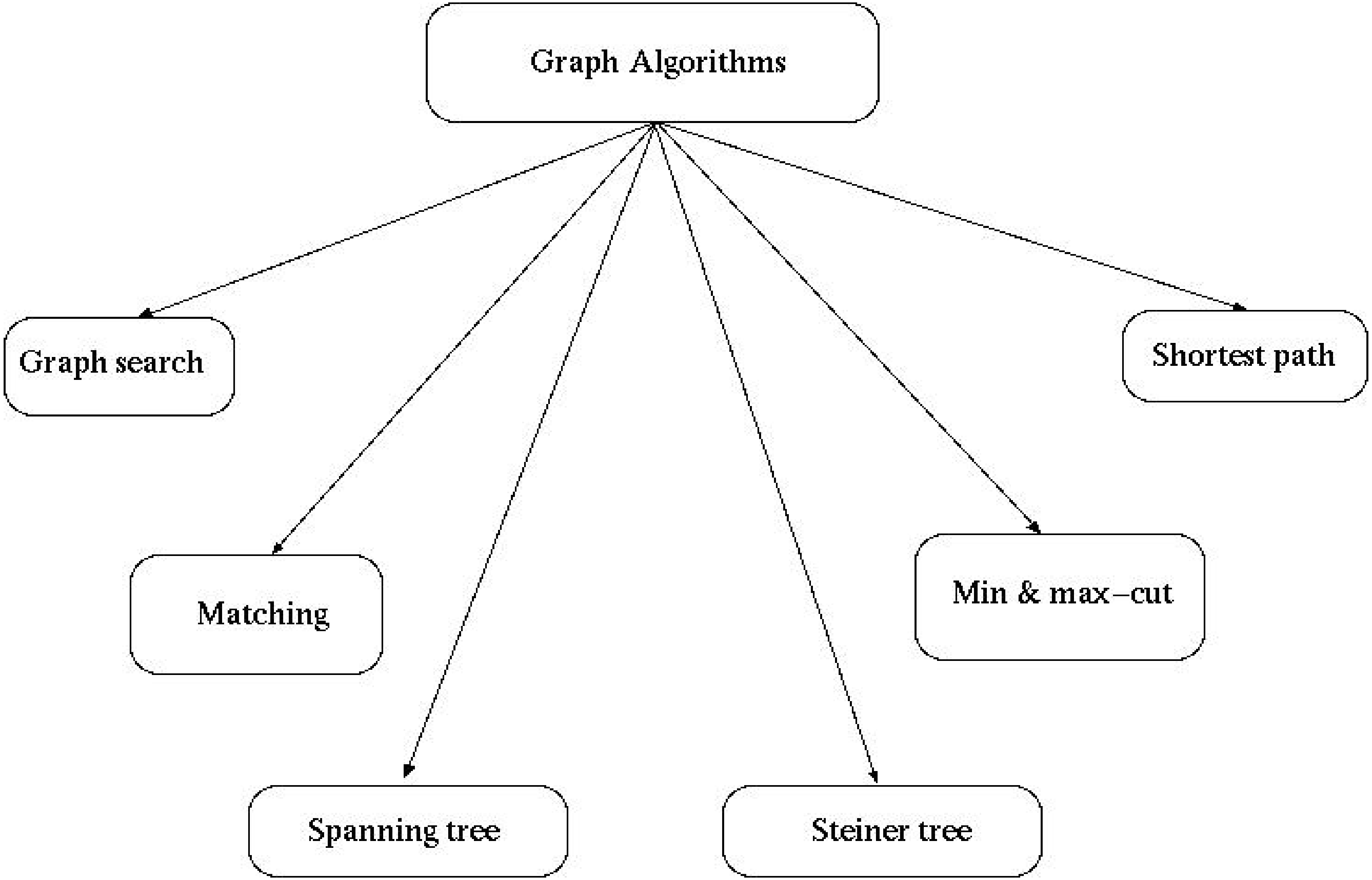
Standard graphs used in studying VLSI design problems

Basic algorithms on those graphs

Classification of Algorithms

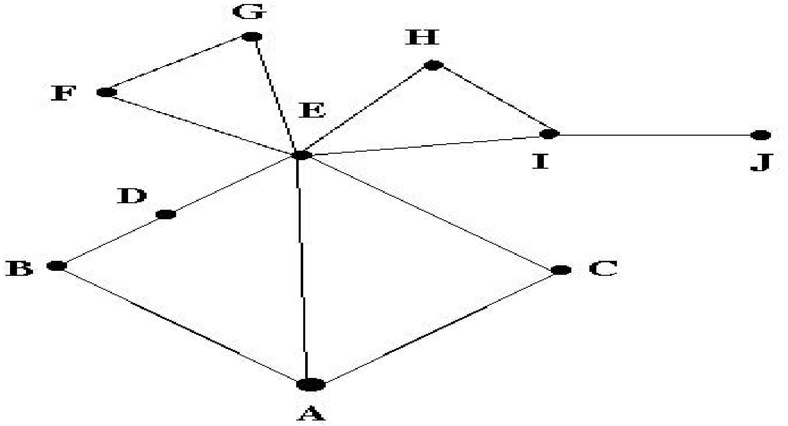


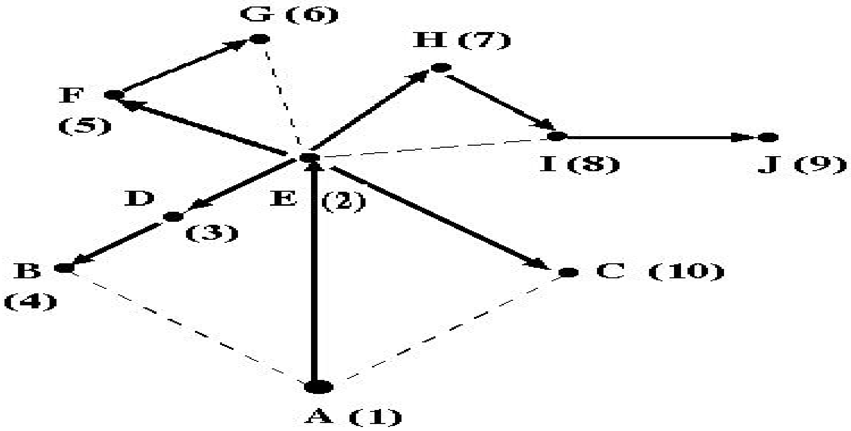
Graph Algorithms



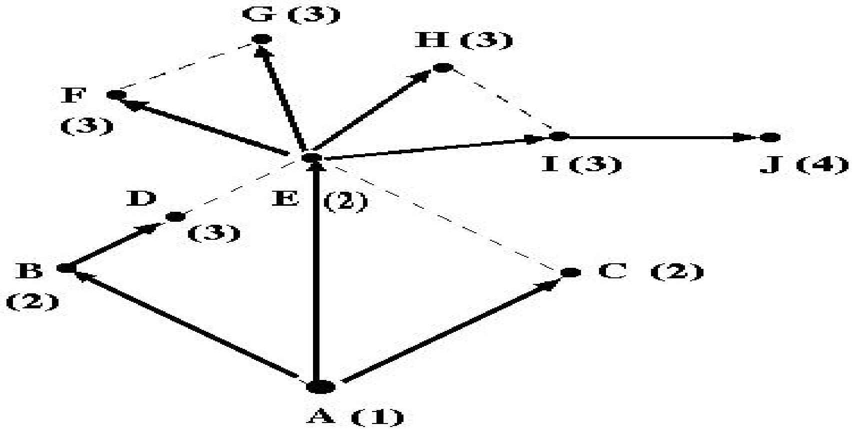
Basic Graph search algorithms

A graph





Depth first search tree



Breadth first search tree

Basic Graph search algorithms

```
Algorithm DEPTH-FIRST SEARCH(u)

begin
  MARK(u) = 1;
  for each vertex v, such that (u,v) ∈ E
  if MARK(v) = 0 then
    DEPTH-FIRST-SEARCH(v);
end.
```

```
Algorithm BREADTH-FIRST-SEARCH(u)

begin
    put the start vertex in Q;
    while Q not empty do
       u = first element of Q
       for each vertex v, such that (u,v) ∈ E
             process v;
             put v in Q;
    endwhile
end.
```

Algorithm for MST

- **Instance: A connected weighted undirected graph G(V,E)**
- **Solution space: All trees that span nodes of G**
- **Objective: Minimize l(T) = $S_{e \in T}$ l(e)**
- **Algorithm:**

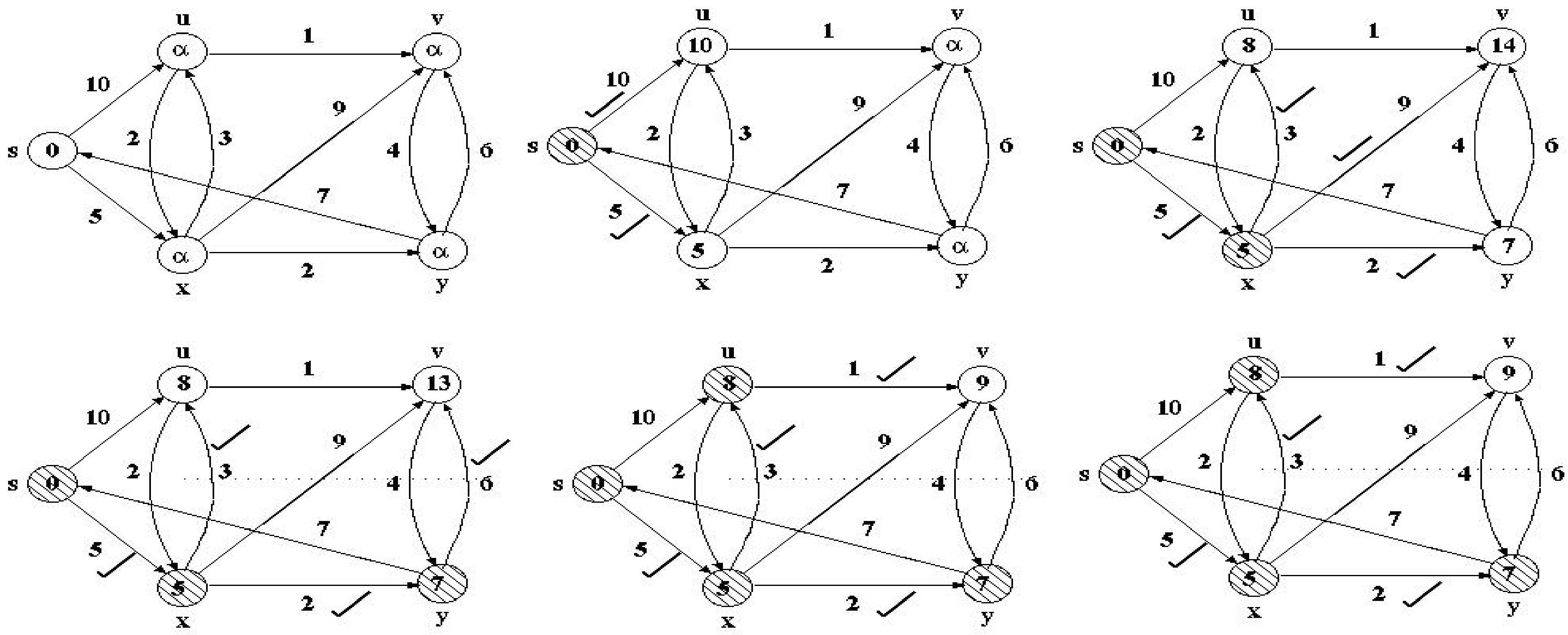
```
A = f;
for each vertex $v \in$ V
        do MAKE-SET ($v$)
sort the edges of E by non-decreasing weights
for each edge ($u$,$v$) $\in$ E (* in order by non-decreasing weight *)
        do if FIND-SET($u$) $\neq$ FIND-SET($v$)
                then  A $\leftarrow$ A $\cup$ {(u,v)}
                        UNION ($u$,$v$)
Return A
```

Single Source Shortest Path

Applications : Routing multi-terminal nets

Input: A directed graph *G*(*V*,*E*) with non-negative edge weights

Output: A set *S* containing the weight of the shortest path of each node from the source node *s*.



DIJKSTRA’s Algorithm

Data Structure: Each node is attached with a pointer π to point its predecessor on the Shortest path

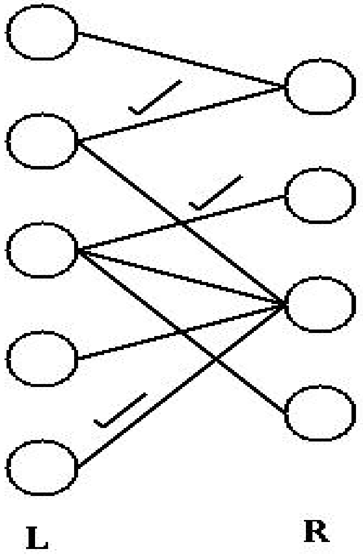
```
Algorithm DIJKSTRA($G$, $w$, $s$)
begin
  S = $\phi$;   $Q = V(G)$; (* $Q \rightarrow$ Priority Queue *)
  while $Q \neq \phi$ do
    $u$ = EXTRACT-MIN($Q$);   $S = S \cup \{u\}$;
    for each vertex $v \in$ Adj($u$) do
      RELAX ($u,v,w$);
  endwhile
end.
```

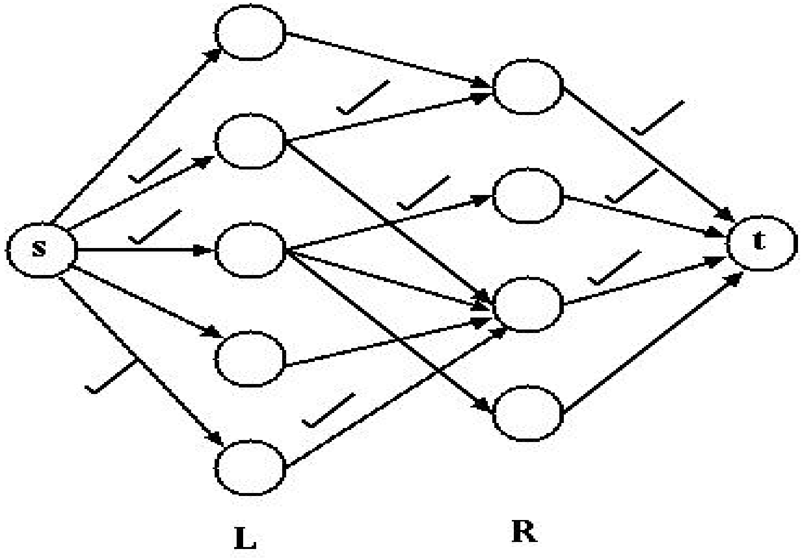
```
Procedure RELAX ($u,v,w$)
begin
  if $d(v) > d(u) + w(u,v)$ then
    $d(v) = d(u) + w(u,v)$;
    $\pi$(v) = $u$;
  endif
end.
```

Bipartite Matching

Definition: Given an undirected graph *G*(*V*, *E*),

- A **matching** is a subset of edges E′ ⊆ E such that for all vertices v ∈ V, at most one edge of E′ is incident on v.
- A **maximum matching** is a matching with maximum cardinality.
- A matching is called **bipartite matching** if the graph G is bipartite.





Transformation
Matching problem → Max-flow problem

Given a bipartite graph $G(V, E)$, where $V = L \cup R$,

construct a weighted digraph $G'(V', E')$ as follows:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R, (u, v) \in E\} \cup \{(v, t) : v \in R\}$
- Assign unit capacity to each edge.

Result: If M is a matching in G, then there is a integer valued flow f in G' with $|f| = |M|$.

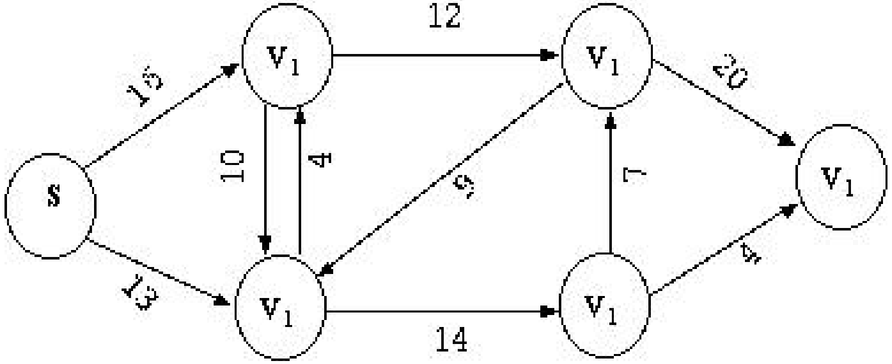
Conversely, if f is an integer valued flow in G', then there is a matching M with cardinality $|M| = |f|$.

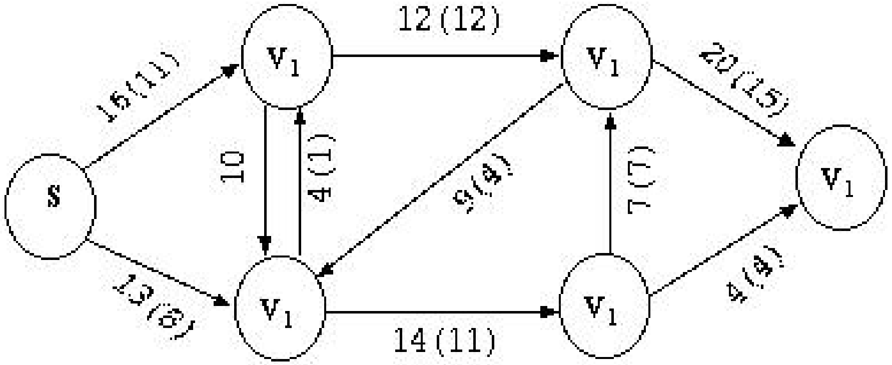
Problem: Find the maximum flow in the network G'.

Matching edges are those edges of G through which flow pass.

Max-flow Min-cut Problem

Given a digraph $G(V, E)$ with each edge having capacity $c(u, v) \geq 0$, and two designated nodes s (source) and t (sink),





Flow $f(u,v)$: a real-valued function (may be “+” / “-”/ 0), and satisfies

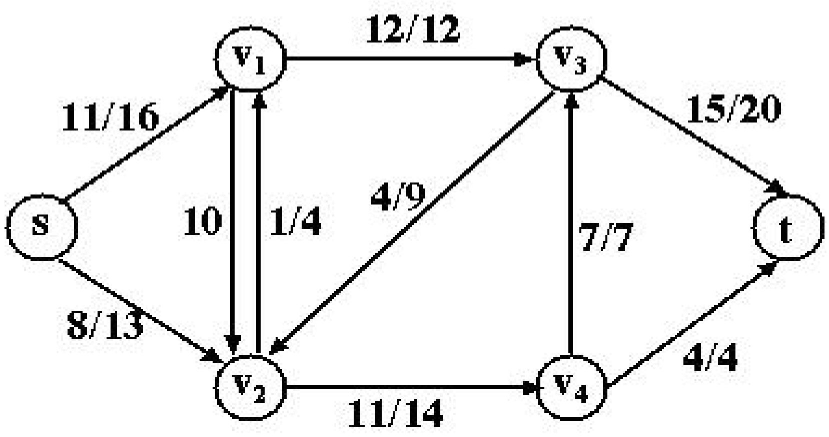
Capacity constraint: For all $u, v \in V$, we require $f(u, v) \leq c(u, v)$.

Skew symmetry: For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.

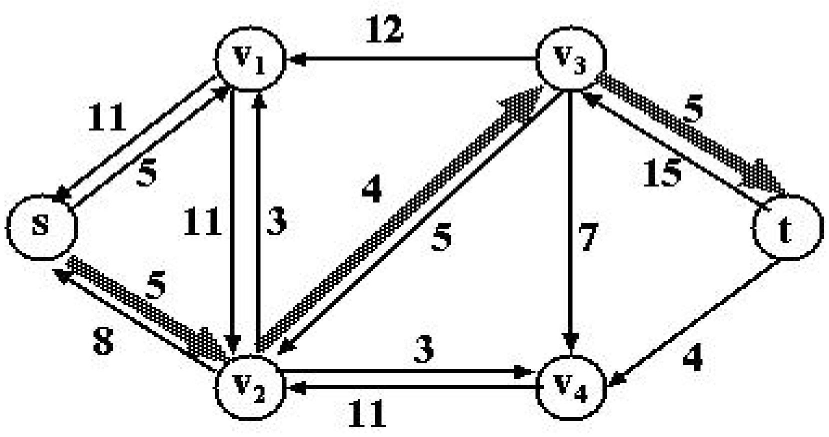
Flow conservation: For all $u \in V - \{s, t\}$, we require $S_{v \in V}\ f(u, v) = 0$.

Objective: ***maximize*** value of the flow $|f| = S_{v \in V}\ f(s, v)$

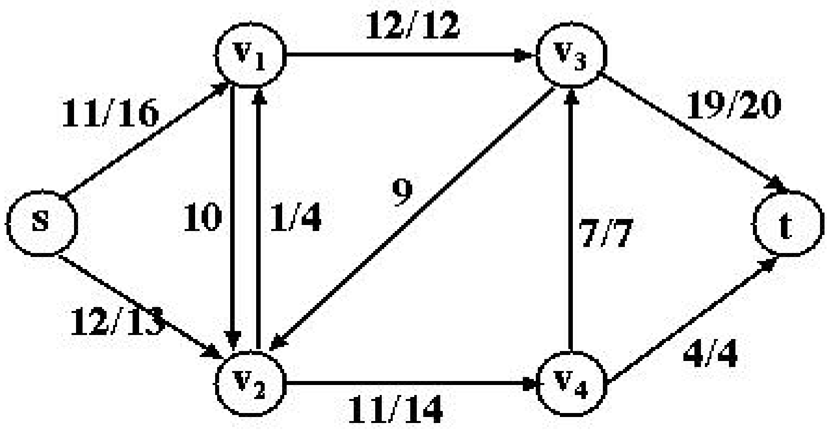
A flow network



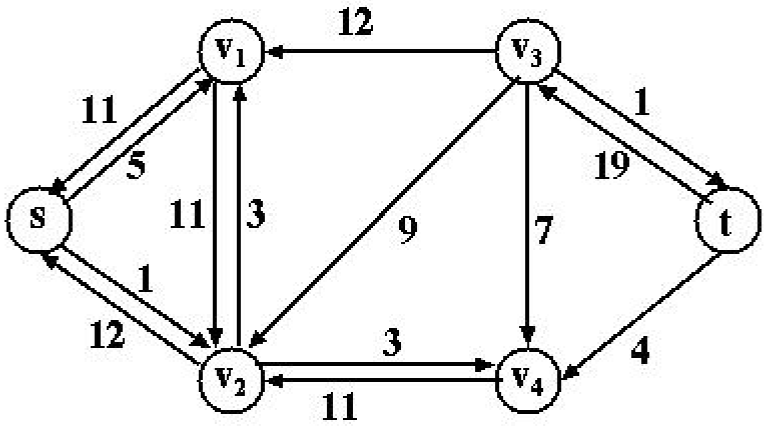
A flow augmenting path



Augmented flow



The residual network
$(c(u,v) = c(u,v) - f(u,v))$



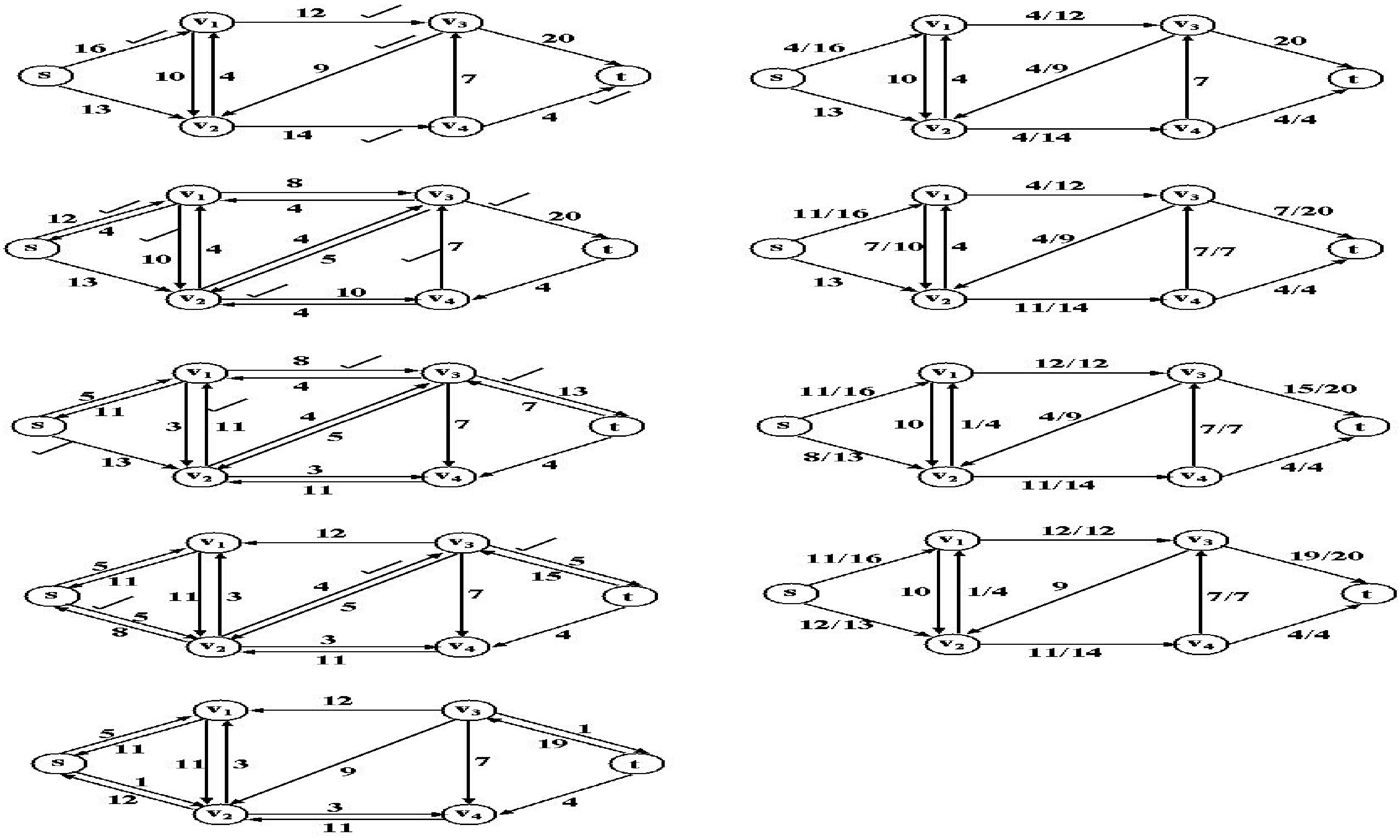
Ford-Fulkerson Method (G, s, t)

```
(* Initialize flow f to 0 *)
        for each edge (u,v) $\in$ E
                do f[u,v] $\leftarrow$ 0;    f[v,u] $\leftarrow$ 0;
    while an augmenting path $p$ from $s$ to $t$ exists in the residual network
            (*    use breadth-first search to find a shortest path
                  from $s$ to $t$ in residual network *)
        do  (* augment flow $f$ along $p$ *)
                $c_f(p) \leftarrow \min\{c_f(u,v) : (u,v) \text{ is in } p\}$
                for each edge $(u,v)$ in $p$
                        do    $f[u,v] \leftarrow f[u,v] + c_f(p)$;
                              $c[u,v] \leftarrow c[u,v] - c_f(p)$;
                              $f[u,v] \leftarrow - f[v,u]$;
    return $f$
```

Time complexity : $O(E\ |f^*|)$

where f* is the maximum flow.

Execution Trace



Significant NP-complete graph problems

Independent set problem

Instance: Graph $G = (V, E)$, and a positive integer $k \leq |V|$.

Question: Does G contain an independent set of size k or more, i.e., a subset

$V'' \subset V$ such that no two vertices of V'' are adjacent, and $|V''| \geq k$

Clique problem

Instance: Graph $G = (V, E)$, and a positive integer $k \leq |V|$.

Question: Does G contain a clique of size k or more, i.e., a subset $V'' \subset V$ such that every pair vertices of V‘ are adjacent, and $|V''| \geq k$.

Graph k-colorability

Instance: Graph $G = (V, E)$, and a positive integer $k \leq |V|$.

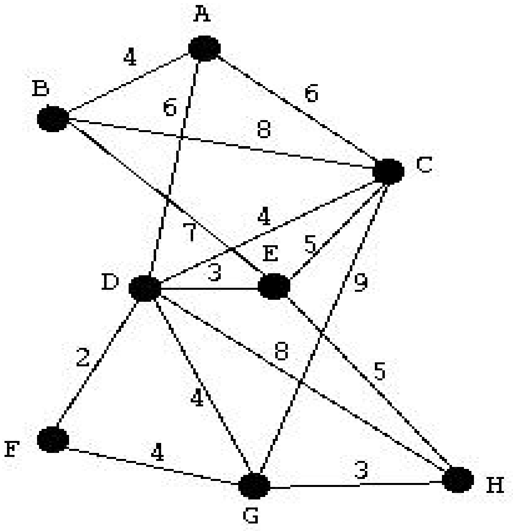
Question: Is G k-colorable, i.e., does there exist a function

$$f : V \rightarrow \{1, 2, \ldots k\} \text{ such that } f(u) \neq f(v) \text{ whenever } \{u,v\} \in E \text{ ?}$$

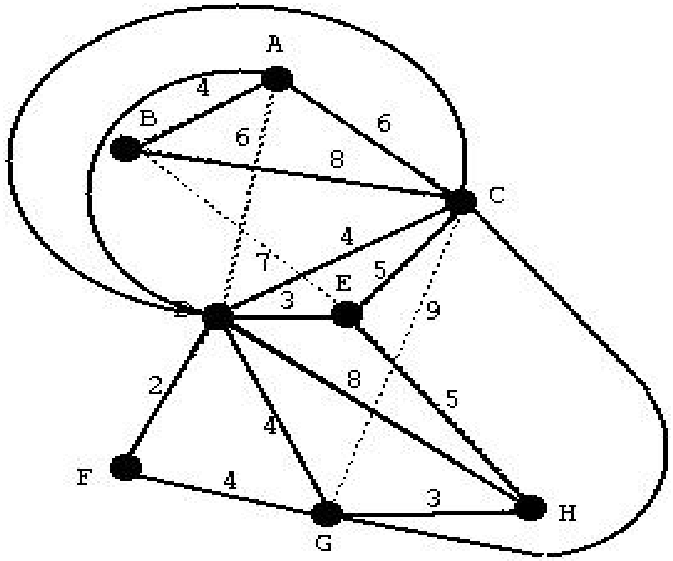
Special Classes of graphs in VLSI CAD

- Planar graphs
- Interval graphs
- Circle graphs
- Permutation graphs

Planar Graphs



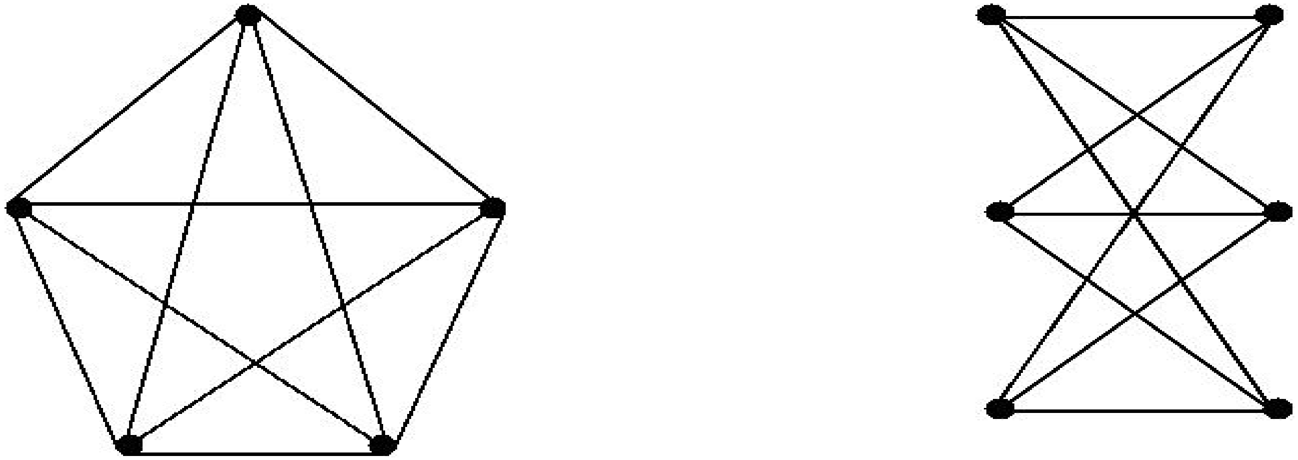
A planar graph



Its planar embedding

- A graph G is planar if it can be drawn in the plane without edges crossing.
- The importance of this type of graphs stems from the fact that several graph problems which are hard in general, are easy if G is known to be planar.
- This class of graphs is useful in circuit layout design

Two smallest non-planar graphs



Kuratowski's Theorem: *G* is planar if and only if *G* is not subgraph-homeomorphic to K_5 or $K_{3,3}$.

Characterization of Planar graphs

Euler's Theorem: If *P* is an arbitrary planar embedding of a connected planar graph *G* with *n* vertices and *m* edges, and if *P* has *f* faces (including outer face), then

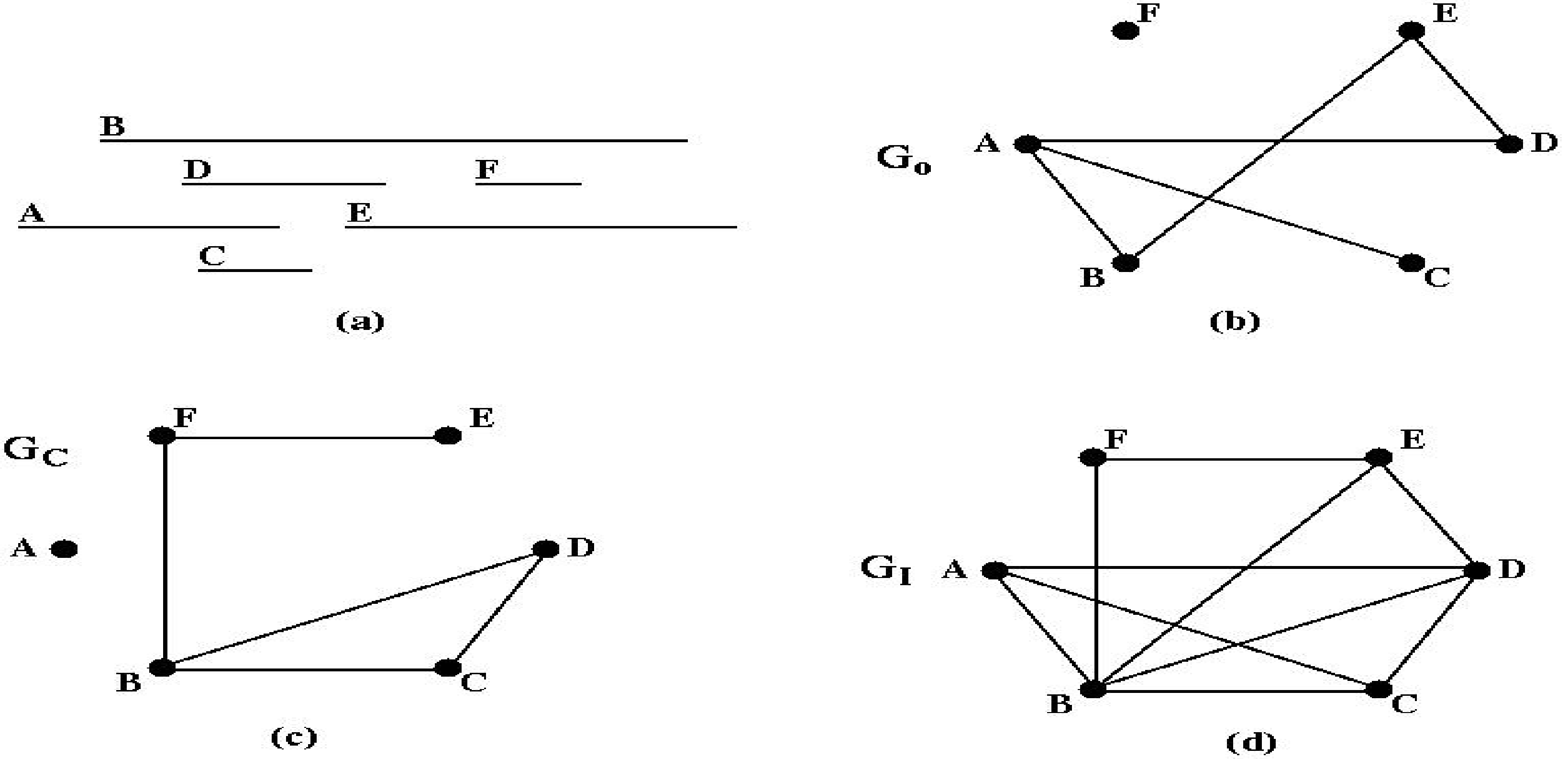
$$n - m + f = 2$$

Results on planar graphs

- If *G* is planar with *n* vertices and *m* edges, then $m \leq 3n$-6. If each face of *G* consists of 3 edges (excepting the outer face), then *m* = 3*n*-6. If each face in *G* has four edges then *m* = 2*n*-4.
- If *G* is a planar graph with *n* > 4 vertices, then *G* has at least four vertices whose degrees are at most 5.
- The time complexity for testing whether a graph is planar or not is *O*(*n*), where *n* is the number of nodes in the graph.
- A planar embedding of a planar graph can be obtained in *O*(*n*) time.
- The problem of computing maximum clique, maximum independent set, minimum coloring, etc. are all polynomial time computable.
- The most important problem of finding maximal planar subgraph of an arbitrary undirected graph is NP-hard.

Interval Graphs

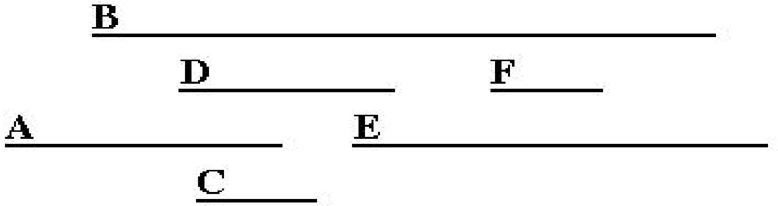
Graphs associated with a set of intervals

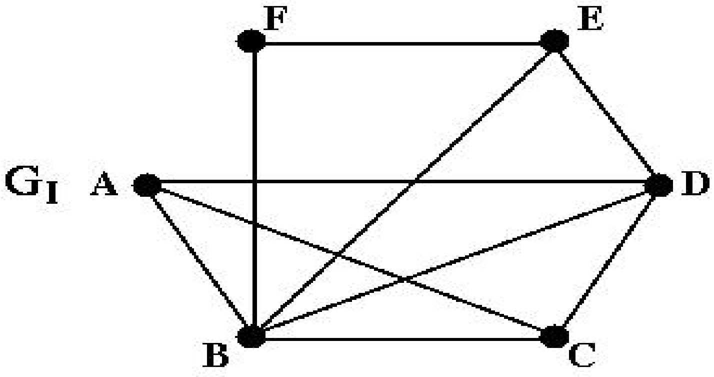


(a) Intervals, (b) overlap graph,

(c) containment graph, (d) interval graph

Definition: A graph G(V,E) is an interval graph if its vertices can be represented by a set of non-empty intervals on the real axis such that edges exist between pairs of vertices if their corresponding intervals overlap.

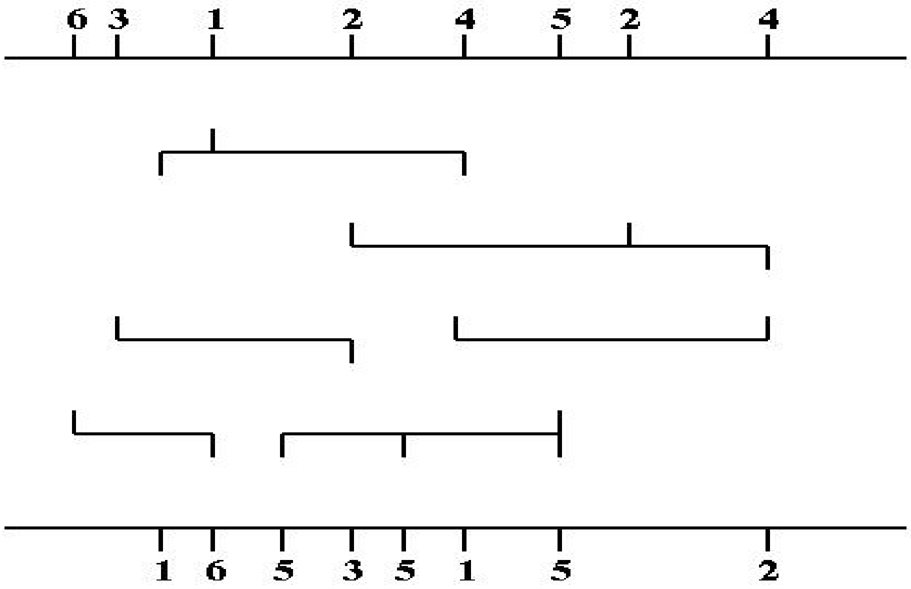




An important application of Interval Graph

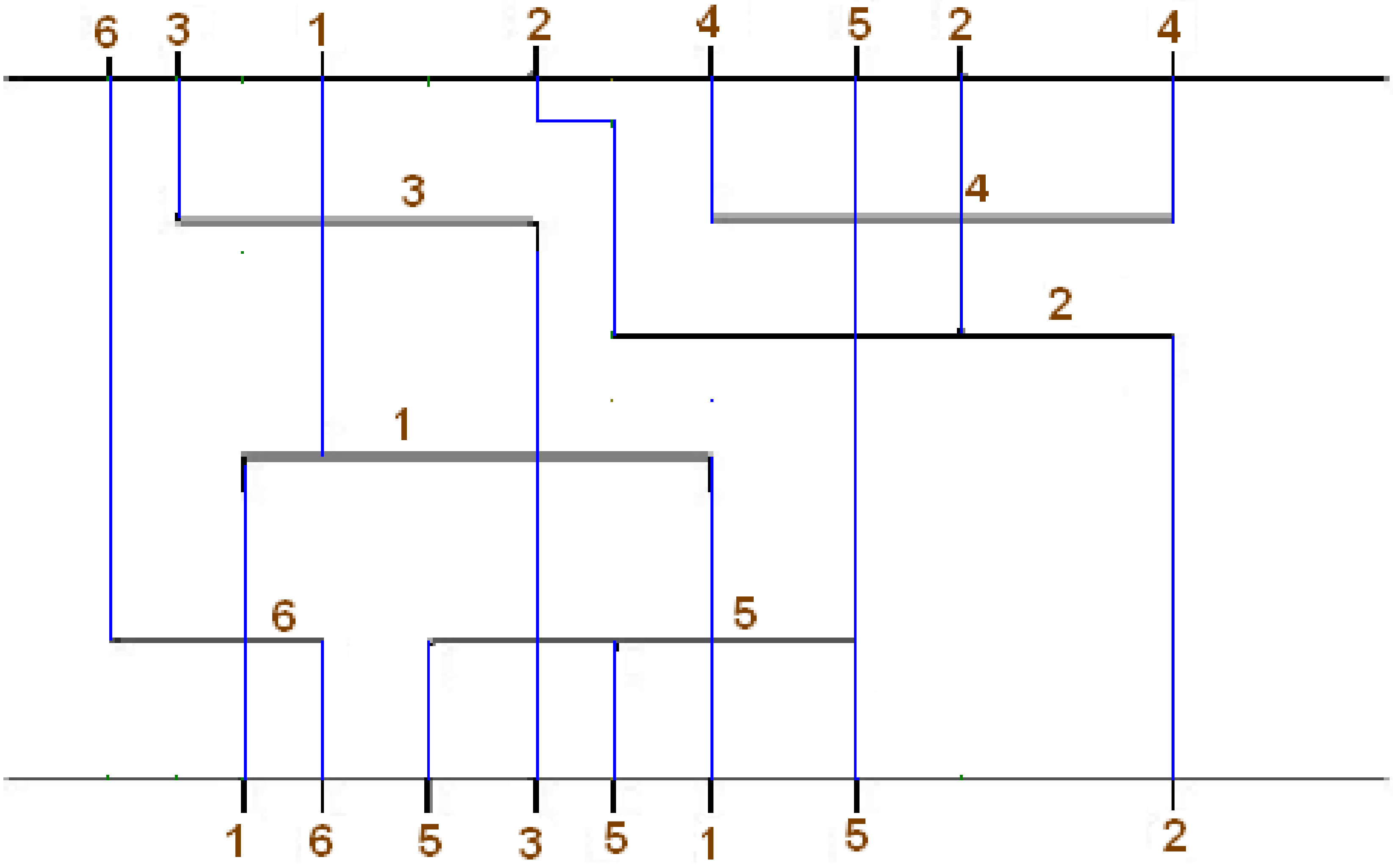
Channel-Width-Minimization problem in the jog free manhattan channel routing model.

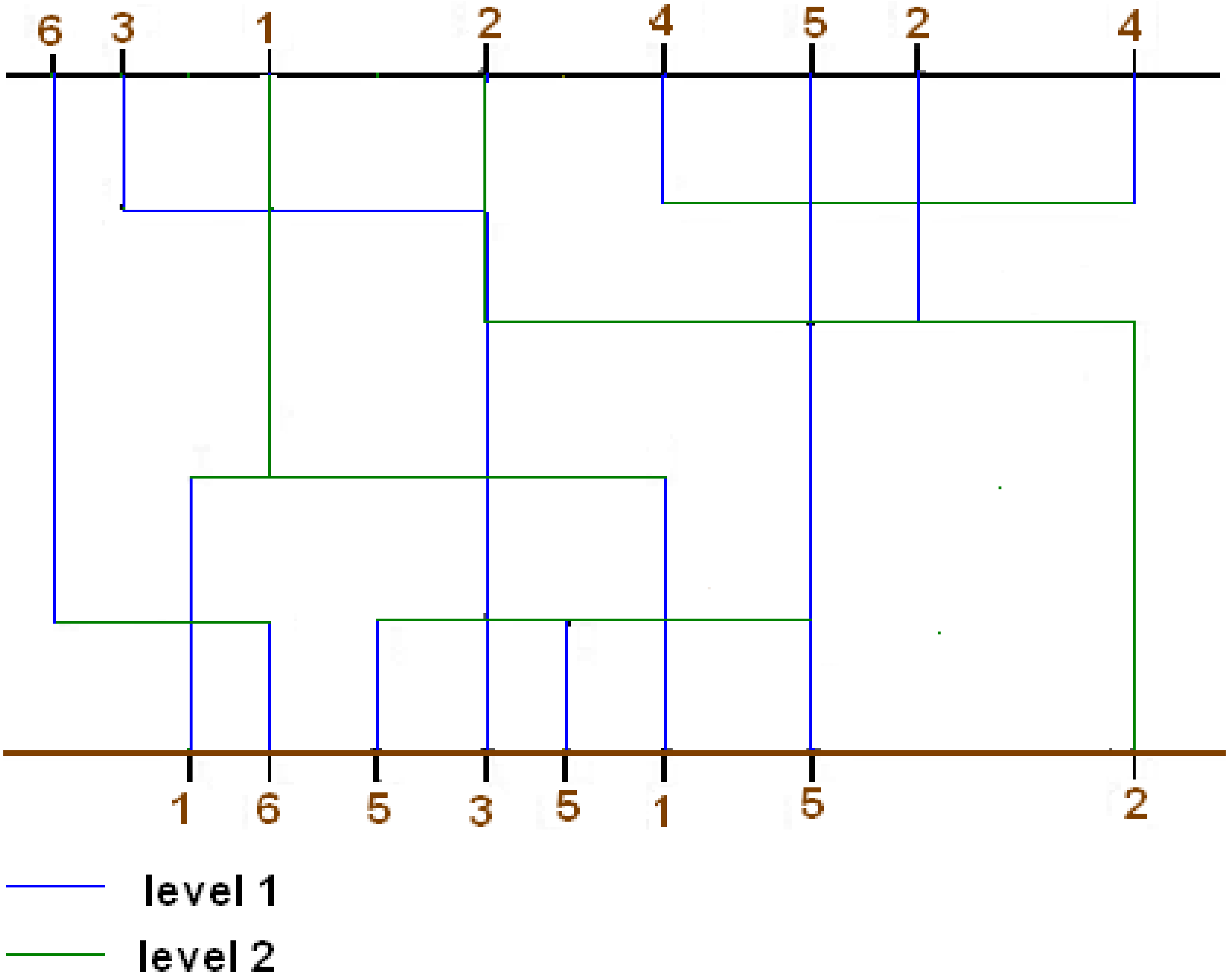
A routing instance:



$[l_i, r_i]$ leftmost and rightmost terminals of net n_i.

- Horizontal constraint between n_i and $n_j \Rightarrow$
  $[l_i, r_i] \cap [l_j, r_j] \neq \phi$
- Vertical constraint from net n_i to $n_i \Rightarrow$ one of the two outside terminals of both the nets share a common column.
- In this case, we say that n_i is below n_j or n_j is above n_i.
- $t_i \Rightarrow$ the track assigned to net n_i


6
3
1
2
4
5
2
4
3
4
2
1
6
5
1
6
5
3
5
1
5
2


6
3
1
2
4
5
2
4
1
6
5
3
5
1
5
2
level 1
level 2

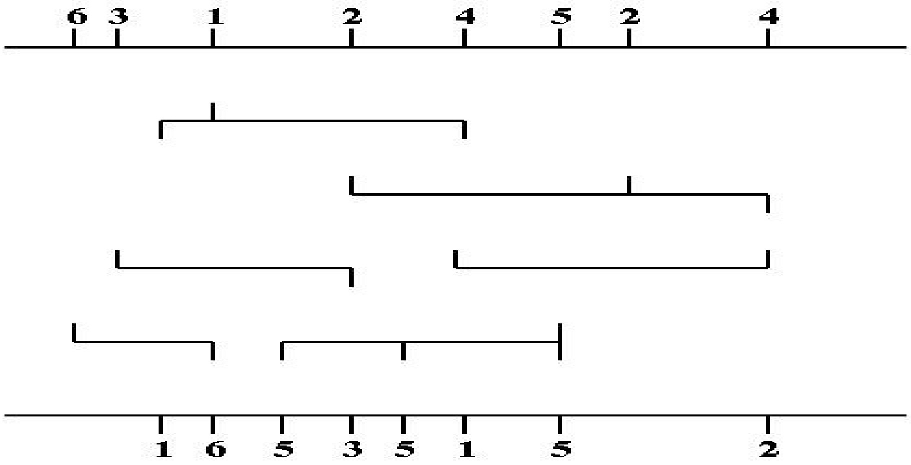
Application of Interval Graph (contd.)

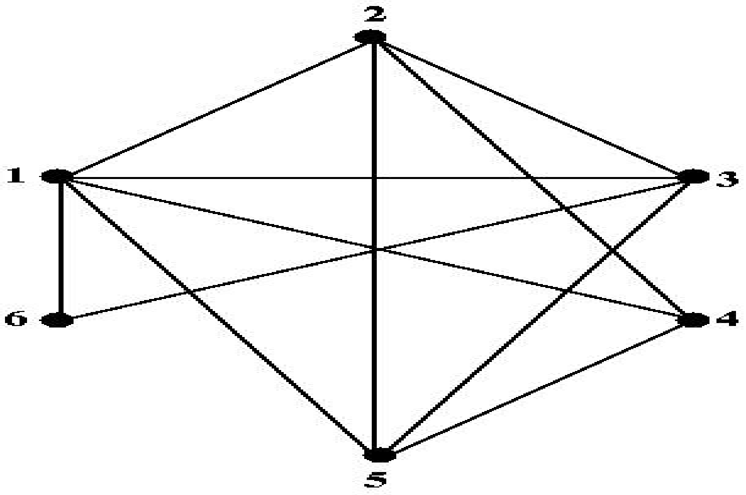
A Jog-free manhattan channel routing model $\Rightarrow$ A routing instance

$I = (t_1, t_2, \ldots, t_n)$ such that

(i) if n_i and n_j overlap then $t_i \neq t_j$ and

(ii) If n_i is below n_j, then $t_i < t_j$





Horizontal constraint graph $G_h(I) = (V_h, E_h) \Rightarrow$ an undirected graph where V_h = Set of nets of I;

$E_h = \{(n_i, n_j)$, if n_i and n_j overlaps$\}$

Optimization Problem: Find the track assignment for the nets with minimum number of tracks

Solution: Find all maximal cliques of the Horizontal constraint graph.

Data structures :

S ⇒ an array containing 2n elements corresponding to $\{(l_i, r_i), i = 1, 2, \dots n\}$.

Each element *S*[*j*] is attached with two additional fields, called *interval_id* and *tag*

S[*j*].*interval_id* contains = *i* if the j-th element of *S* is equal to l_i or r_i

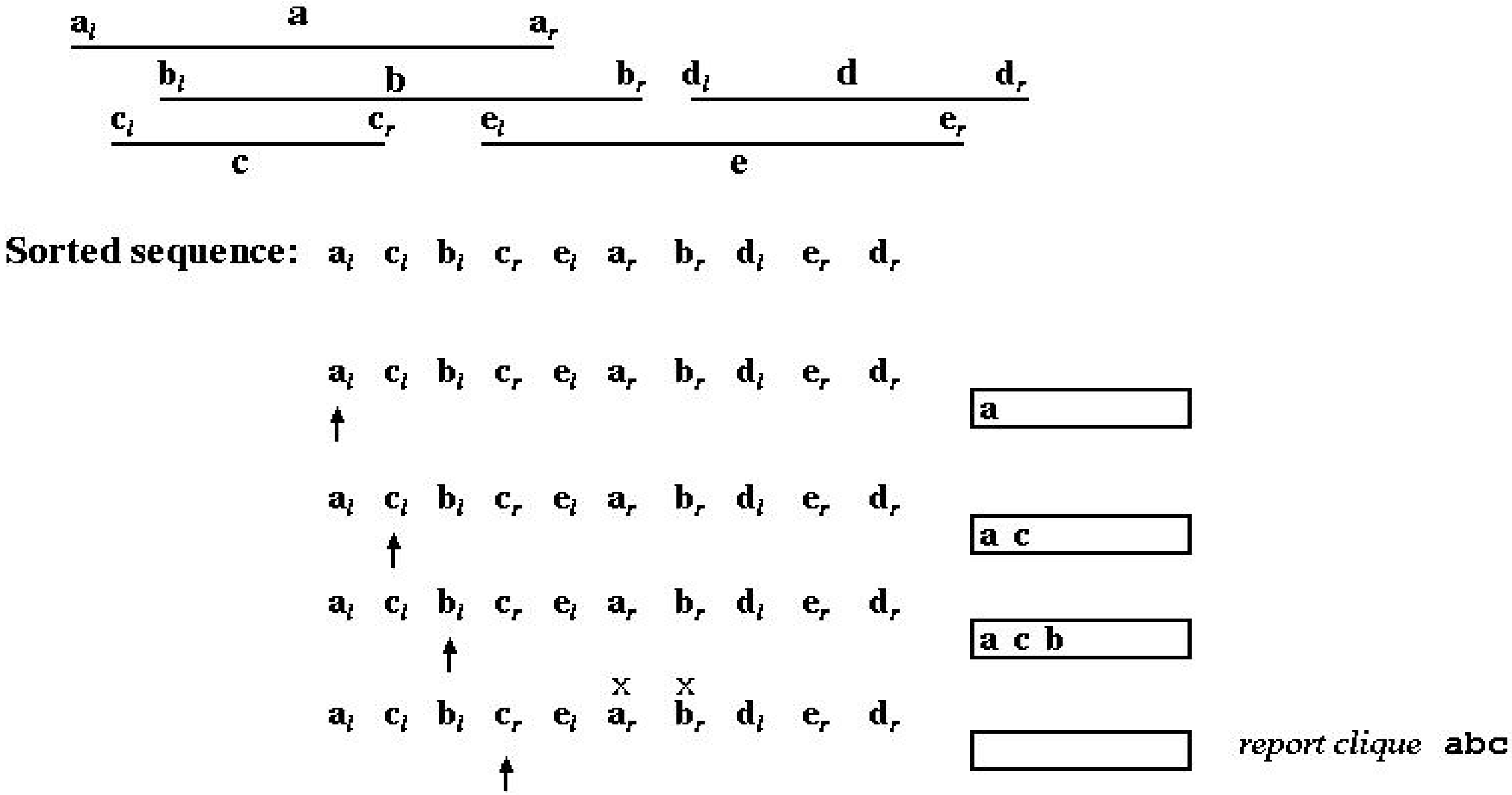
S[*j*].*tag* = L/R depending on whether S[*j*] corresponds to a left/right end point.

Algorithm:

```
Sort elements of S in increasing order of their values;
for i = 1 to 2n do
    if S[i].tag = L then
        insert S[i].interval_id in list L;        new_clique_flag ← 1;
    if S[i].tag = R then
        if new_clique_flag = 1 then
            return all the elements in L as a clique;
        remove S[i].interval_id from list L;      set new_clique_flag ← 0;
```

Size of the largest clique is the minimum track requirement for this routing problem.

Demonstration and Complexity Results



Time complexity results of different problems on Interval Graph

Largest clique : $O(n\log n)$.

Maximum independent set $O(n\log n)$

Permutation Graphs

Π = a permutation $[\pi_1, \pi_2, \quad , \pi_\nu]$ of n integers.

(e.g. [4,3,6,1,5,2], here $\pi_1 = 4$, $\pi_2 = 3$ etc.)

$\pi_i^{\ -1}$ = position in the sequence where number *i* is found.

(e.g. $\pi_4^{-1} = 1$, $\pi_3^{-1} = 2$, etc.)

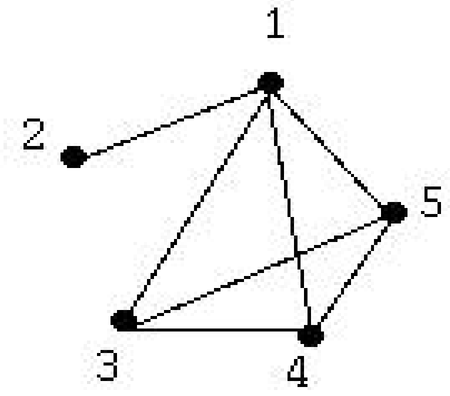
Permutation graph: An undirected graph $G_\pi(V,E)$ such that

V = {1, 2, …, n}

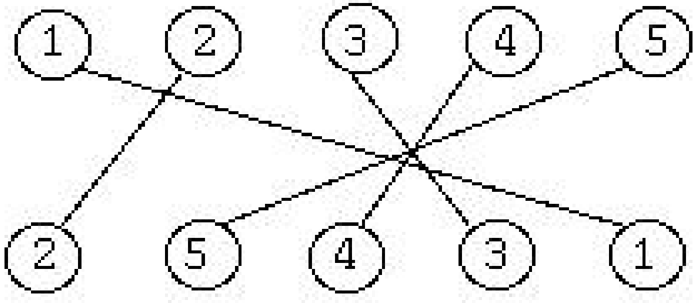
E = $\{(i,j) \mid$ if $i < j$ and $\pi_j^{-1} < \pi_i^{-1}\}$ (* i.e., the larger of the integers appear to the left of the smaller one in π $*$).

(in other words, $(i,j) \in$ E if $(i\text{-}j)\times(\pi_i^{-1} - \pi_j^{-1}) < 0$.)

A graph and its permutation labeling



The graph $G_{[2,5,4,3,1]}$



Important uses:

Recognizing monotone channels (for routing) among a set of rectangular circuit modules on a VLSI floorplan.

Properties of permutation graphs

- **Number of edges : O(n^2).**
- **Transitively orientable**
- **Complement graph is also permutation graph.**

- **Running time Complexities:**
- Chromatic number: O(*n* log*n*)
- Maximum independent set: O(*n* log*n*)
- Largest clique: O(*n* log*n*)

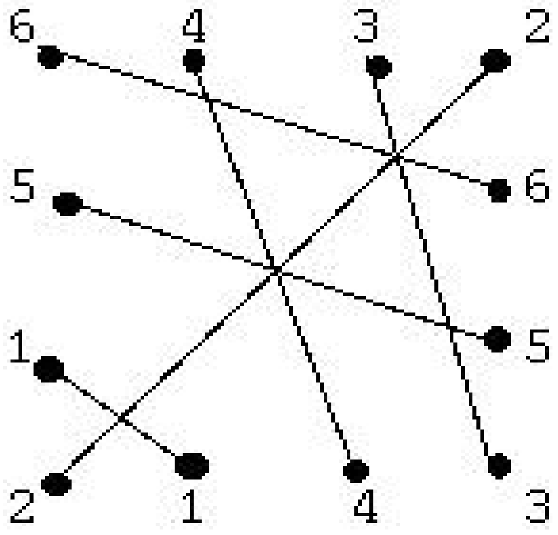
Note: In the permutation labeling,

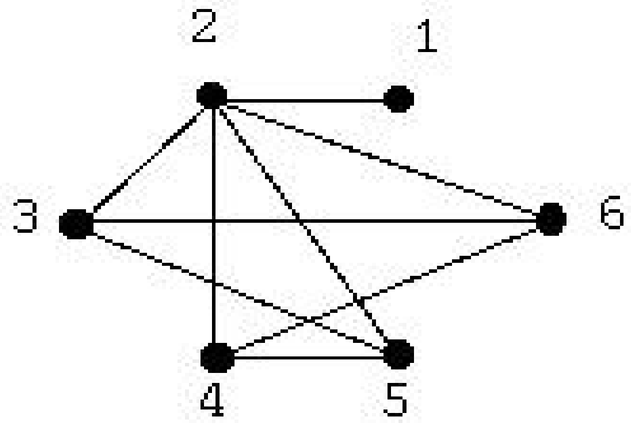
an increasing subsequence represents an independent set,

an decreasing subsequence represents a clique.

Circle Graphs

- An undirected graph G is a circle graph if it is isomorphic to the intersection graph of a finite collection of chords of a circle.

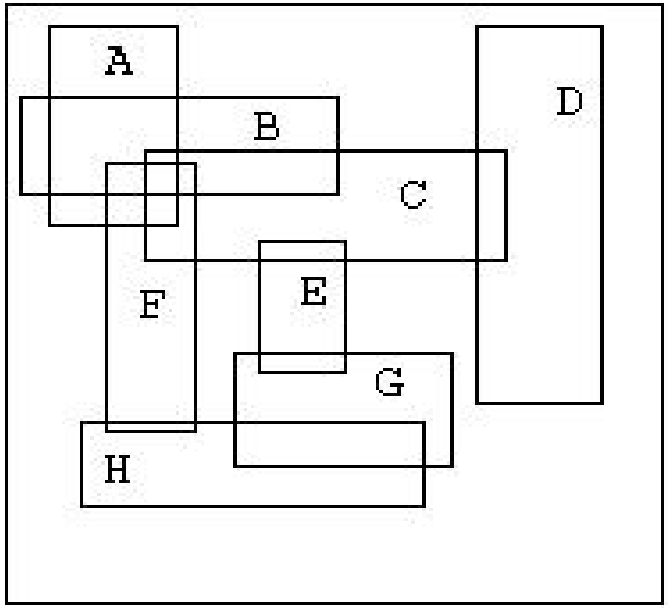


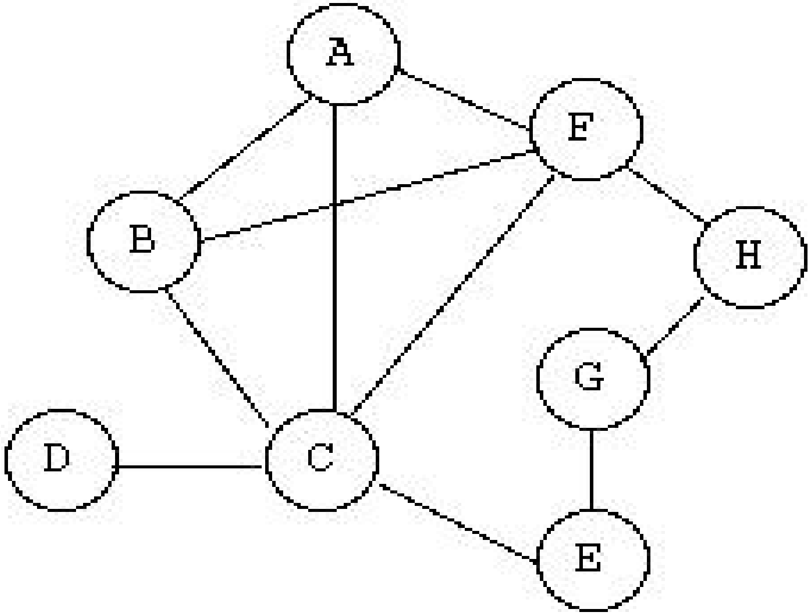


- The graph obtained by considering the intersection of lines in a switchbox is equivalent to a circle graph.

Graphs related to a set of rectangles

- Rectangle intersection graphs





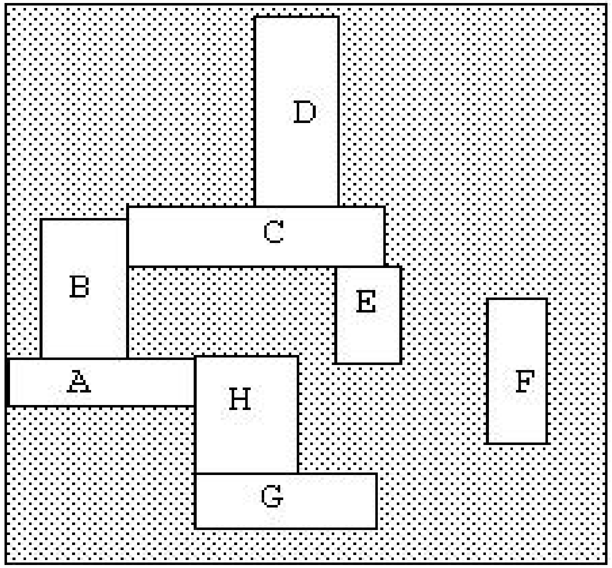
Results on rectangle intersection graphs

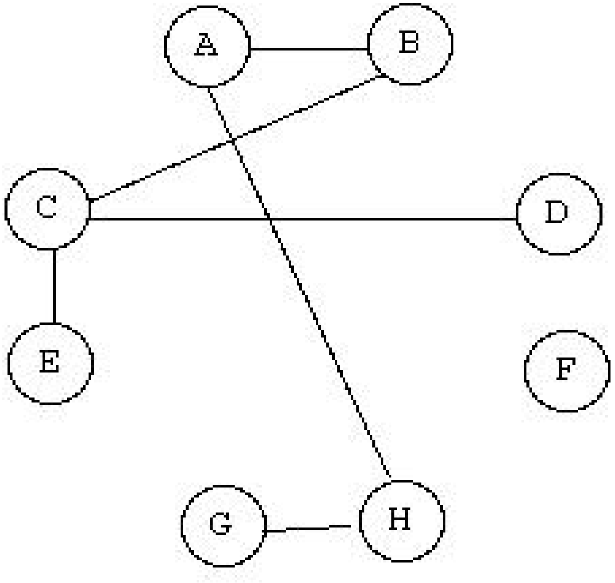
- Satisfies *Helly's Property.*
- All maximal cliques are computable in polynomial time.
- Maximal independent set problem is NP-hard.

- Applications:

- Compaction
- Finding free area for placing a block
- Other geometric optimization problems.

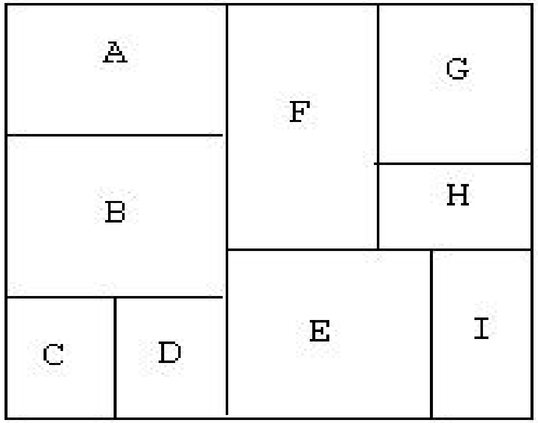
Rectangle Neighborhood Graphs

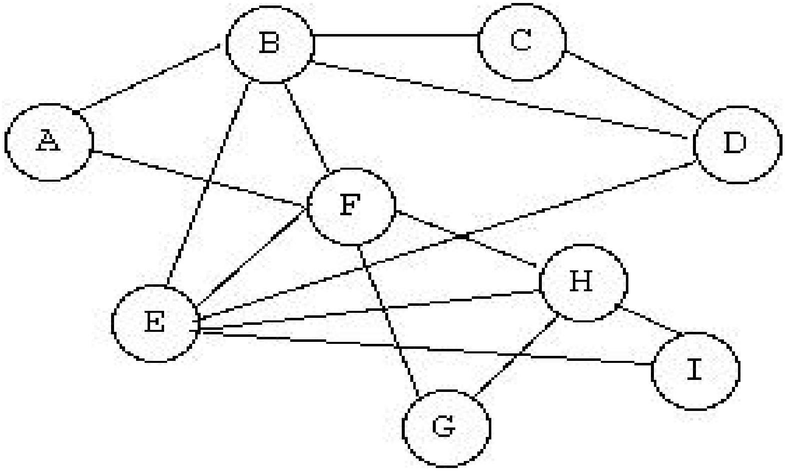




- Useful in global routing for describing physical adjacency relationship among the circuit modules.

Rectangular duals



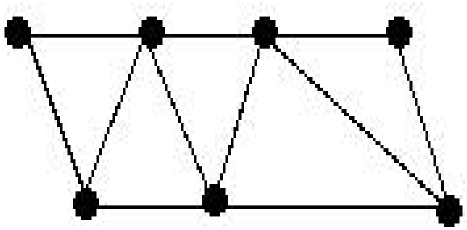


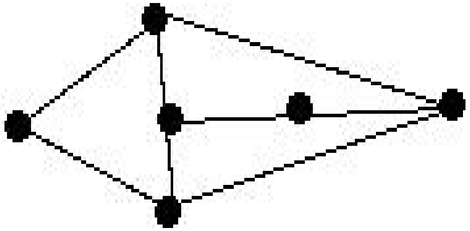
Given a graph G(V,E), its *rectangular dual* is a set of rectangles $R = \{R_1, R_2, \ldots, R_n\}$ where each vertex v_i corresponds to the rectangle $R_i \in R$ and two rectangles are adjacent if the corresponding vertices are adjacent.

- Note : Not all graphs are rectangularly dualizable.
- Use: In floorplanning phase of physical design.

Triangulated Graphs

An undirected graph is called triangulated if every cycle of length strictly greater than 3 possesses a chord.





Characterizing a triangulated graph

A vertex x of G is called *simplicial* if its adjacency set Adj(x) induces a complete subgraph (clique) of G.

A vertex elimination scheme: Repeatedly locate a simplicial vertex and eliminate it from the graph until no other vertex remains, or at some stage no simplicial vertex exists.

Theorem: In the former case, the graph G is triangulated.

In the latter case, the graph G is not triangulated.

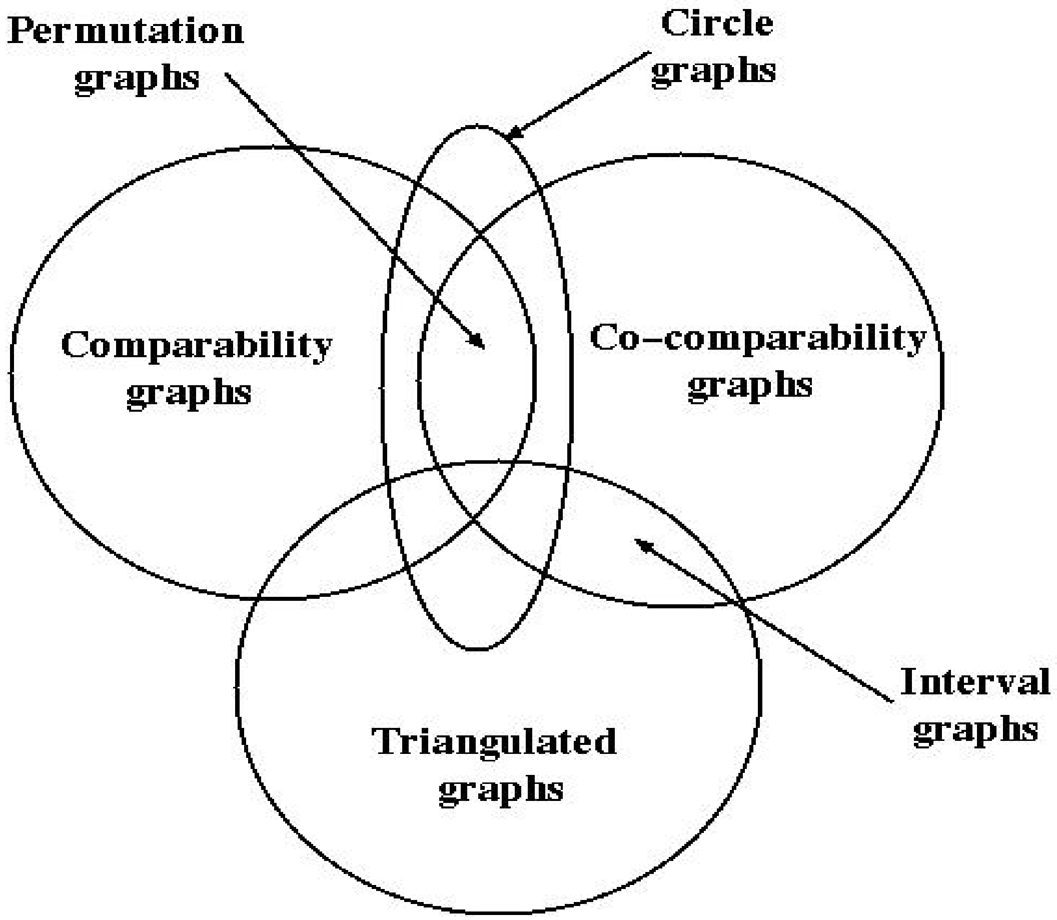
Let G = (V,E) be an undirected graph and let s = [v_1, v_2,…, v_n] be an ordering of the vertices. We say that s is a *perfect vertex elimination scheme* if each v_i is a simplicial vertex of the induced subgraph $G_{\{vi,\ v2,\ldots,\ vn\}}$. In other words, each set

$$X_i = \{v_j \in \mathrm{Adj}(v_i) \mid j > i\}$$

is complete.

Theorem: An undirected graph G is triangulated if it has a perfect vertex elimination scheme. Moreover, any simplicial vertex can start a perfect vertex elimination scheme.

Relationship between different graph classes



Perfect Graphs

Consider the following parameters of an undirected graph:

♦ w(G) ⇒ the *clique number* of the graph *G*: the size of the largest complete subgraph of G.

♦ χ(G) ⇒ the *chromatic number* of the graph *G*: the fewest number of colors needed to properly color the vertices of G.

♦ α(G) ⇒ the *size of the maximum independent set* of the graph *G*: the size of the largest subset of vertices such that there exists no arc among any pair of vertices among the members in this set.

♦ *k*(G) ⇒ the *clique cover number* of the graph *G*: the fewest number of complete subgraphs needed to cover the vertices of G.

Relation among these parameters

- Intersection of a clique and a maximal independent set may be at most one vertex.
- For any graph G, $\omega(G) \leq \chi(G)$.
- For any graph G, $\alpha(G) \leq k(G)$.
- If G^c is the complement of graph G

  then $\alpha(G) = \omega(G^c)$, and $k(G) = \chi(G^c)$.

Perfect graph theorem:

For an undirected graph G=(V,E), the following statements are equivalent:

1. $\omega(G_A) = \chi(G_A)$, for all $A \subseteq V$.
2. $\alpha(G_A) = k(G_A)$, for all $A \subseteq V$.
3. $\omega(G_A) \times \alpha(G_A) \geq |A|$, for all $A \subseteq V$.

Important results on perfect graphs

The following problems are polynomial time solvable for perfect graphs:

- **Minimum COLORING**
- **Maximum CLIQUE**
- **Largest INDEPENDENT SET**
- **Minimum CLIQUE COVER**

1. Introduction to Algorithms
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest
Prentice Hall of India Pvt. Ltd.

2. Algorithms for VLSI Physical Design Automation
Naveed Sherwani
Kluwer Academic Publishers

3. Combinatorial Algorithms for Integrated Circuit Layout
Thomas Lengauer
John Wiley & Sons.

4. Algorithmic Graph Theory and Perfect Graphs
Martin Charles Golumbic
Academic Press

5. Introduction to Graph Theory
Douglas B. West
Prentice Hall of India Pvt. Ltd.