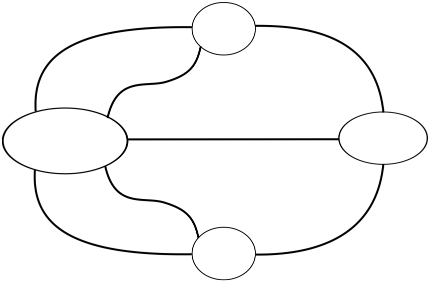
Graph Theory

Probabilistic Graphical Models

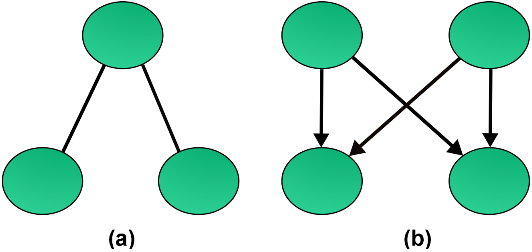
L. Enrique Sucar, INAOE



Outline

Graphs

- A *graph* provides a compact way to represent binary relations between a set of objects
- Objects are represented as circles or ovals, and relations as lines or arrows
- There are two basic types of graphs: *undirected graphs* and *directed graphs*



Directed Graphs

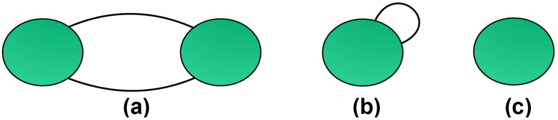
- A *directed graph* or *digraph* is an ordered pair, $G = (V, E)$, where V is a set of vertices or nodes and E is a set of arcs that represent a binary relation on V
- Directed graphs represent anti-symmetric relations between objects, for instance the “parent” relation

Undirected Graphs

- An *undirected graph* is an ordered pair, $G = (V, E)$, where V is a set of vertices or nodes and E is a set of edges that represent symmetric binary relations: $(V_j, V_k) \in E \rightarrow (V_k, V_j) \in E$
- Undirected graphs represent symmetric relations between objects, for example, the “brother” relation

More Definitions

- If there is an edge $E_i(V_j, V_k)$ between nodes j and k, then V_j is adjacent to V_k
- The *degree* of a node is the number of edges that are incident in that node
- Two edges associated to the same pair of vertices are said to be *parallel edges*



More Definitions

- A vertex that is not an endpoint to any edge is an *isolated vertex* –it has degree 0
- In a directed graph, the number of arcs pointing to a node is its *in degree*
- The number of edges pointing away from a node is its *out degree*

Types of Graphs (I)

Chain graph: a hybrid graph that has directed and undirected edges.

Simple graph: a graph that does not include cycles and parallel arcs.

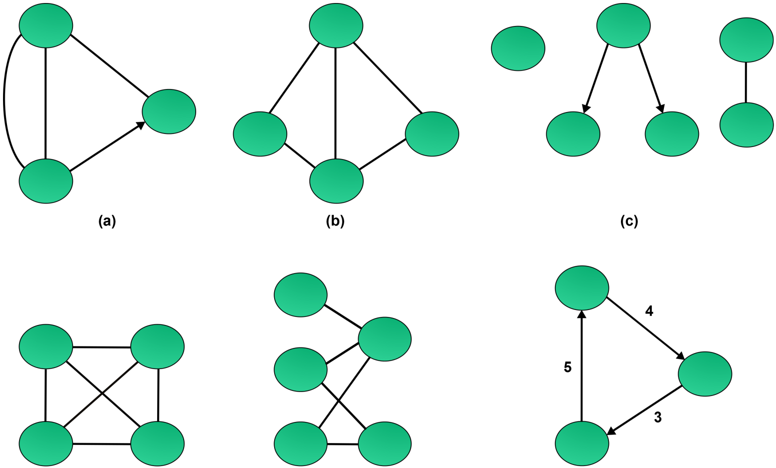
Multigraph: a graph with several components (subgraphs), such that each component has no edges to the other components, i.e., they are disconnected.

Complete graph: a graph that has an edge between each pair of vertices.

Bipartite graph: a graph in which the vertices are divided in two subsets, G_1, G_2, such that all edges connect a vertex in G_1 with a vertex in G_2; that is, there are no edges between nodes in each subset.

Weighted graph: a graph that has weights associated to its edges and/or vertices.

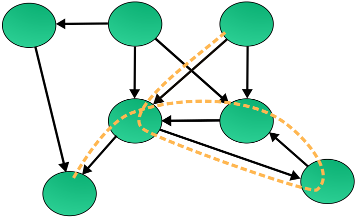
Types of Graphs (II)



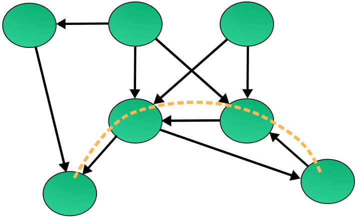
(a) (b) (c) (d) (e) (f)

Trajectories

- A *trajectory* is a sequence of edges, $E_1, E_2, ..., E_n$ such that the final vertex of each edge coincides with the initial vertex of the next edge in the sequence
- A *simple* trajectory does not include the same edge two o more times; an *elemental* trajectory is not incident on the same vertex more than once



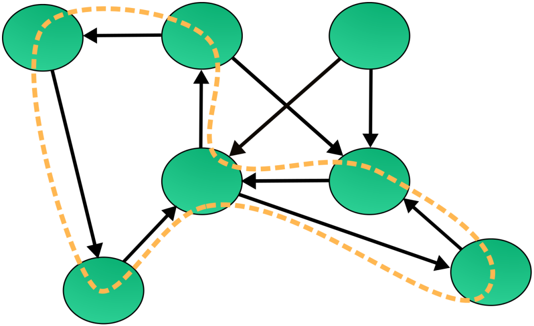
(a)



(b)

Circuits

- A *circuit* is a trajectory such that the final vertex coincides with the initial one
- A *simple* circuit does not include the same edge two or more times; an *elemental* circuit is not incident on the same vertex more than once (except the initial/final vertex)



Directed Acyclic Graphs

- A DAG is a directed graph that has no directed circuits (a directed circuit is a circuit in which all edges in the sequence follow the directions of the arrows)

Problems

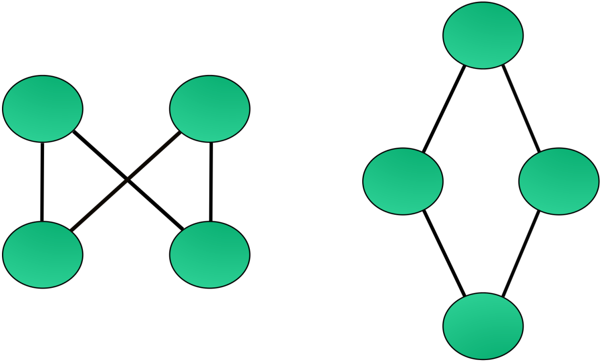
- Finding a trajectory that includes all edges in a graph only once (Euler trajectory).
- Finding a circuit that includes all edges in a graph only once (Euler circuit).
- Finding a trajectory that includes all vertices in a graph only once (Hamiltonian trajectory).
- Finding a circuit that includes all vertices in a graph only once (Hamiltonian circuit).
- Finding a Hamiltonian circuit in a weighted graph with minimum cost (Traveling salesman problem)[1].

[1] In this case the nodes represent cities and the edges roads with an associated distance or time, so the solution will provide a traveling salesman with the "best" (minimum distance or time) route to cover all the cities.

Isomorphism (I)

- Two graphs are isomorphic if there is a one to one correspondence between their vertices and edges, so that the incidences are maintained
- Types:
  1. *Graph isomorphism*. Graphs G_1 and G_2 are isomorphic.
  2. *Subgraph isomorphism*. Graph G_1 is isomorphic to a subgraph of G_2 (or vice versa).
  3. *Double subgraph isomorphism*. A subgraph of G_1 is isomorphic to a subgraph of G_2.

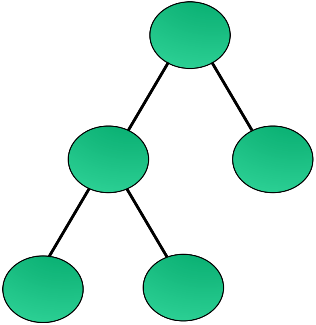
Isomorphism (II)



- Determining if two graphs are isomorphic (type 1) is an NP problem; while the subgraph and double subgraph isomorphism problems (type 2 and 3) are NP-complete

Undirected trees

- An undirected tree is a connected graph that does not have simple circuits
- There are two classes of vertices or nodes in an undirected tree: (i) leaf or terminal nodes, with degree one; (ii) internal nodes, with degree greater than one

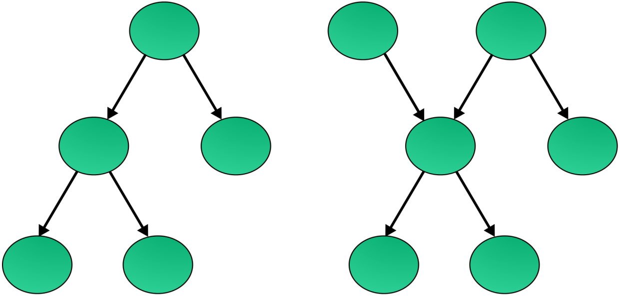


Properties

- There is a simple trajectory between each pair of vertices.
- The number of vertices, $\mid V \mid$, is equal to the number of edges, $\mid E \mid$ plus one: $\mid V \mid = \mid E \mid + 1$.
- A tree with two or more vertices has at least two leaf nodes.

Directed trees

- A directed tree is a connected directed graph such that there is only a single directed trajectory between each pair of nodes
- A rooted tree has a single node with an in degree of zero (the root node) and the rest have in degree of one
- A polytree might have more than one node with in degree zero (roots), and certain nodes (zero or more) with in degree greater than one



Terminology (I)

Root: a node with in degree equal to zero.

Leaf: a node with out degree equal to zero.

Internal node: a node with out degree greater than zero.

Parent / Child: if there is a directed arc from A to B, A is parent of B and B is a child of A.

Brothers: two or more nodes that have the same parent.

Ascendants /Descendants: if there is a directed trajectory from A to B, A is an ascendant of B and B is a descendant of A.

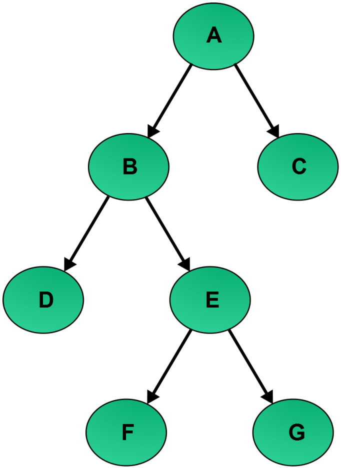
Subtree with root A: a subtree with A as its root.

Subtree of A: a subtree with a child of A as its root.

K-ary Tree: a tree in which each internal node has at most K children. It is a regular tree if each internal node has K children.

Binary Tree: a tree in which each internal node has at most two children.

Terminology (II)

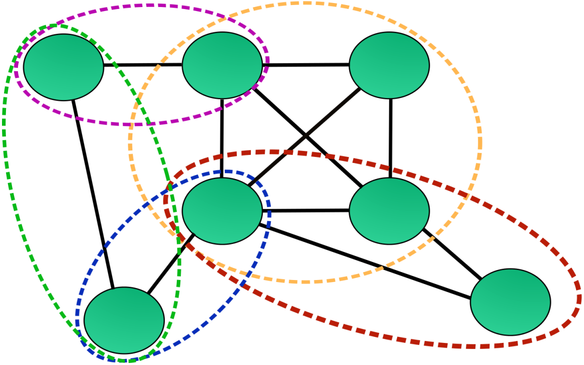


Complete set and subsets

- A *complete graph* is a graph, G_c, in which each pair of nodes is adjacent; that is, there is an edge between each pair of nodes
- A *complete set*, W_c is a subset of G that induces a complete subgraph of G. It is a subset of vertices of G so that each pair of nodes in this subgraph is adjacent

Cliques

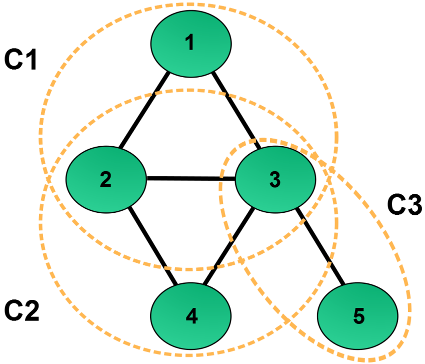
- A *clique*, C, is a subset of graph G such that it is a complete set that is maximal; that is, there is no other complete set in G that contains C



Ordering

- An ordering of the nodes in a graph consists in assigning an integer to each vertex
- Given a graph $G = (V, E)$, with n vertices, then $\alpha = [V_1, V_2, ..., V_n]$ is an ordering of the graph; V_i is *before* V_j according to this ordering, if $i < j$
- An ordering α of a graph $G = (V, E)$ is a *perfect ordering* if all the adjacent vertices of each vertex V_i that are before V_i, according to this ordering, are completely connected

Perfect Ordering

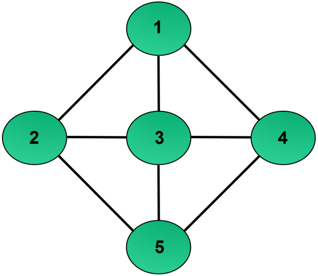


Clique Ordering

- In an analogous way as an ordering of the nodes, we can define an ordering of the cliques, $\beta = [C_1, C_2, ..., C_p]$
- An ordering β of the cliques has the *running intersection property*, if all the common nodes of each clique C_i with previous cliques according to this order are contained in a clique C_j; C_j is the *parent* of C_i
- It is possible that a clique has more than one parent

Triangulated graphs

- A graph G is *triangulated* if every simple circuit of length greater than three in G has a chord
- A chord is an edge that connects two of the vertices in the circuit and that is not part of that circuit
- A condition for achieving a perfect ordering of the vertices, and having an ordering of the cliques that satisfies the running intersection property, is that the graph is triangulated

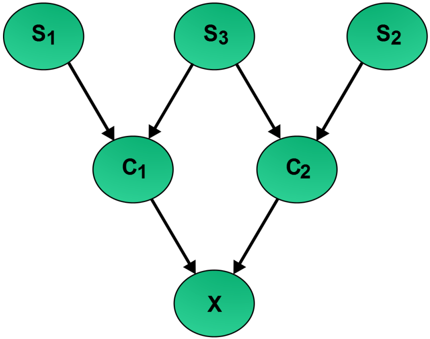


Maximum Cardinality Search

- Given that a graph is triangulated, the following algorithm guarantees a perfect ordering:

1. Select any vertex from V and assign it number 1.
2. WHILE Not all vertices in G have been numbered:
   1. From all the non-labeled vertices, select the one with higher number of adjacent labeled vertices and assign it the next number.
   2. Break ties arbitrarily.

Example

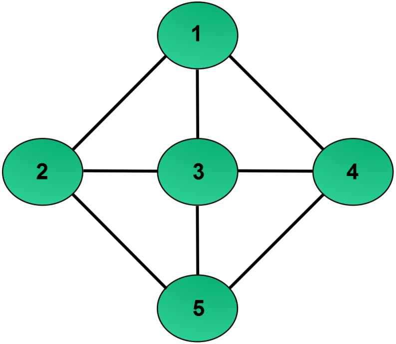


Graph filling

- The filling of a graph consists of adding arcs to an original graph G to make it triangulated
- The following algorithm makes the graph triangulated:

1. Order the vertices V with maximum cardinality search: $V_1, V_2, ..., V_n$.
2. FOR $i = n$ TO $i = 1$
    1. For node V_i, select all its adjacent nodes V_j such that $j > i$. Call this set of nodes A_i.
    2. Add an arc from V_i to V_k if $k > i$ and $V_k \notin A_i$.

Example



A_5: $\emptyset$
A_4: 5
A_3: $4, 5$
A_2: $3, 5$. An arc is added from 2 to 4.
A_1: $2, 3, 4$. An arc is added from 1 to 5.

The resulting graph has two additional arcs $2 - 4$ and $1 - 5$

Book

Sucar, L. E, *Probabilistic Graphical Models*, Springer 2015 – Chapter 3

Additional Reading

- Aho, A.V., Hopcroft, J.E, Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Boston (1974)
- Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Elsevier, Netherlands (1994)
- Gould, R.: Graph Theory. Benjamin/Cummings, Menlo Park (1988)
- Gross, J.L, Yellen, J.: Graph Theory and its Applications. CRC Press, Boca Raton (2005)
- Neapolitan, R.: Probabilistic Reasoning in Expert Systems: Theory and Algorithms. Wiley, New York (1990)