A Short Introduction to Graph Theory
Modélisation et Performance des Réseaux

Marceau Coupechoux*

* TELECOM ParisTech (INFRES/RMS) and CNRS LTCI
marceau.coupechoux@telecom-paristech.fr

Feb 2014

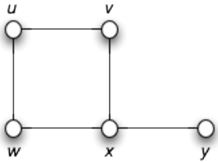
Introduction

- Graph theory is a powerful tool in communication networks
- Performance evaluation: the theory provides optimal solutions or near-optimal algorithms that can be used for benchmarking existing or future propositions
- Protocol design: several MAC, routing protocols or scheduling algorithms use the results of graph theory
- Examples of applications:
  - Ethernet: the spanning tree protocol
  - Routing: Link State algorithms
  - Scheduling: OFDMA (matching), TDMA (frame design)
  - Ad hoc networks: MAC design, routing
  - Cellular networks: frequency assignment

First Concepts: a Graph

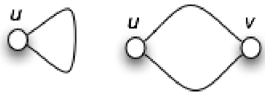
Definition

A **graph** G is a triple consisting in a vertex set $V(G)$, an edge set $E(G)$ and a relation that associates to each edge two vertices called its endpoints.



- Here: $V(G) = \{u, v, w, x, y\}$ and $E(G) = \{uv, uw, vx, wx, xy\}$
- A **subgraph** H of G is such that $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and the relation bw. vertices and edges in H is the same as in G

First Concepts: a Simple Graph



- A *loop* is an edge whose endpoints are equal
- *Multiple edges* are edges with the same pair of endpoints

Definition

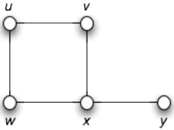
A **simple graph** is a graph with no loops nor multiple edges.

- Two endpoints of an edge are said to be *adjacent* or *neighbors*
- We write $u \leftrightarrow v$

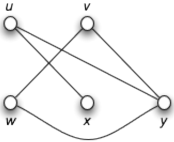
First Concepts: Complement Graph

Definition

The **complement** $\bar{G}$ of a simple graph such that $V(\bar{G}) = V(G)$ and $uv \in E(\bar{G}) \Leftrightarrow uv \notin E(G)$. A **clique** is a set of pairwise adjacent vertices. An **independant set** (or **stable set**) is a set of pairwise non adjacent vertices.



Graph G



Complement graph $\overline{G}$

- In G, $\{u, v\}$ is a clique of size 2, $\{v, w, y\}$ is a stable set of size 3
- In $\bar{G}$, cliques become independent sets and vice versa

First Concepts: Bipartite Graph

Definition

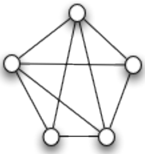
A graph G is **bipartite** if $V(G)$ is the union of two independent sets, called partite sets.

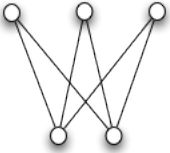
- Example: wedding
- There are 4 men and 3 women
- Not all couples are feasible
- Can we do a good matching bw. men and women ?

First Concepts: Complete Graph

Definition

A graph is **complete** if all its vertices are pairwise adjacent. It is denoted K_n when it has n vertices. In a **complete bipartite graph**, two vertices are adjacent if and only if they are in different partite sets.



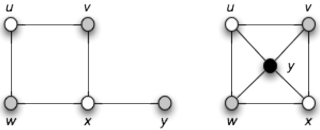


- A complete graph is characterized by its number of vertices n
- The number of edges of a complete graph is given by: $m = n(n-1)/2$

First Concepts: Chromatic Number

Definition

The **chromatic number** of a graph G, $\chi(G)$, is the minimum number of colors needed to label the vertices so that no adjacent vertices have the same color.

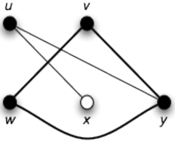


Graph G Graph G'

- Graph G has a chromatic number of $\chi(G) = 2$
- Graph G' has a chromatic number of $\chi(G') = 3$
- $\chi(G)$ is also the minimum number of independent sets

First Concepts: Paths, Cycles and Connectedness

- A **path** is a simple graph whose vertices can be ordered so that vertices are adjacent if and only if they are consecutive in the list
- A **cycle** is a simple graph whose vertices can be ordered in a cyclic sequence so that two vertices are adjacent if and only if they are consecutive in the list.
- $\{u, y, v, w\}$ is a path, $\{v, w, y\}$ is cycle



Definition

A graph G is **connected** if each pair of vertices belongs to a path.

First Concepts: Degree

Definition

The **degree** of a vertex v of a graph G, denoted $d_G(v)$, is the number of edges incident to v (loops count twice).

Theorem

If G is a graph, then: $\sum_{v \in V(G)} d_G(v) = 2m$.

- The minimum degree is denoted δ_G, the maximum degree Δ_G
- For a graph G, $n\delta_G \leq 2m \leq n\Delta_G$

First Concepts: Weighted Graph

Definition

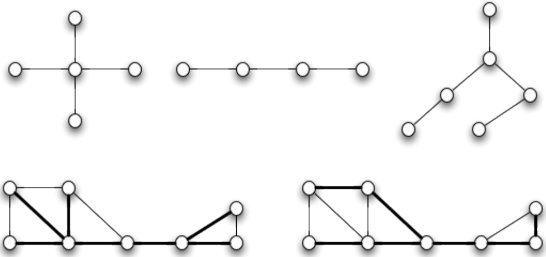
A **weighted graph** G consists in a vertex set $V(G)$, an edge set $E(G)$, a relation that associates to each edge two vertices and an weighting function $w : E \to \mathbb{R}$.

- A simple graph can be represented by an adjacency matrix $M = \{w(u,v)\}_{u,v \in V}$.
- If $(u,v) \notin E(G)$, $w(u,v) = \infty$.

Trees: Definitions

Definition

A **tree** is a connected acyclic graph. A **leaf** is a vertex of degree 1. A **spanning subgraph** of G is a subgraph with vertex set $V(G)$. A **spanning tree** is a spanning subgraph that is a tree.



Trees: Characterization

Theorem

For a n-vertex graph G ($n \geq 1$), the following are equivalent:

- *(1) G is a tree (connected and acyclic).*
- *(2) G is connected and has $m = n - 1$ edges.*
- *(3) G is acyclic and has $m = n - 1$ edges.*
- *(4) For any $u, v \in V(G)$, G has exactly one (u, v)-path.*

Trees: Kruskal's Algorithm

Kruskal's Algorithm

- **Input:** A simple connected weighted graph.
- **Output:** A minimum weight spanning tree.
- **Begin** $F \leftarrow E$; $A \leftarrow 0$
- **While:** $|A| < n - 1$ **do:**
- Find $e \in F$ such that $w(e)$ is minimum;
- $F \leftarrow F - \{e\}$;
- **If** $G(A \cup \{e\})$ is acyclic **then:** $A \leftarrow A \cup \{e\}$;
- **endif**
- **endwhile**

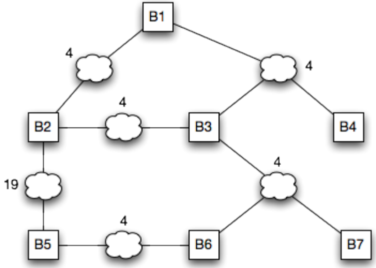
Trees: Kruskal's Algorithm

REMARKS:

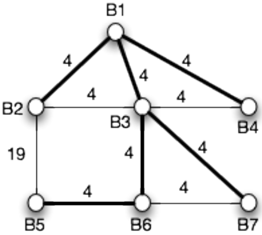
- F is the set of edges to be considered, A is the set of selected edges.
- An exhaustive search would be prohibitive (typical from optimization problems).
- We know that a solution exists.
- The algorithm is **greedy** : at each step, we consider the best solution.
- The algorithm stops when we have a tree ($|A| = n - 1$).

Trees: Example in Communication Networks

- Bridges $B1 \ldots B7$ interconnect six LAN
- Each LAN is associated to a path cost related to its data rate (defined in IEEE 802.1D)
- We look for a **loop-free topology** in this bridged LAN
- In practice, the problem is solved in a distributed way using the **spanning tree protocol**



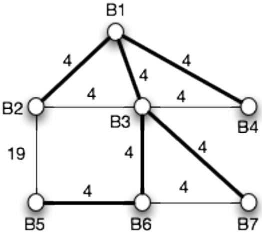
Trees: Example in Communication Networks



Trees: Example in Communication Networks

Algorithm steps:

1. $F = \{B1B2, B1B3, B1B4, B2B3, B3B4, B2B5, B3B6, B3B7, B5B6, B6B7\}$, $A = \emptyset$
2. $e = \{B1B2\}$, $F \leftarrow F - \{B1B2\}$, $A = \{B1B2\}$
3. $e = \{B1B3\}$, $F \leftarrow F - \{B1B3\}$, $A = \{B1B2, B1B3\}$
4. $e = \{B1B4\}$, $F \leftarrow F - \{B1B4\}$, $A = \{B1B2, B1B3, B1B4\}$
5. $e = \{B3B6\}$, $F \leftarrow F - \{B3B6\}$, $A = \{B1B2, B1B3, B1B4, B3B6\}$
6. $e = \{B3B7\}$, $F \leftarrow F - \{B3B7\}$, $A = \{B1B2, B1B3, B1B4, B3B6, B3B7\}$
7. $e = \{B5B6\}$, $F \leftarrow F - \{B5B6\}$, $A = \{B1B2, B1B3, B1B4, B3B6, B3B7, B5B6\}$, $|A| = 6$



Shortest Paths: Definitions

- Let $G = (E, V, w)$ be a simple directed weighted graph
- $w(e)$ is the length of edge e
- The **length** of a path is the sum of the lengths of its edges

Definition

The **distance** $d(u, v)$, $u, v \in V$ in a weighted graph is the minimum sum of the weights on the edges on a (u, v)-path.

Shortest Paths: Dijkstra's Algorithm

Dijkstra's Algorithm

- **Input:** A weighted graph with non-negative edge weights and a source u
- **Output:** The shortest path tree from u and all distances $d(u, v)$, $v \in V$
- **Begin** $S = \emptyset$, $t(u) = 0$,
  $\forall z \neq u, \; t(z) = \infty$ and $prev(z) = undefined$
- **Do:**
  - Select $v = \arg\min_{z \notin S} t(z)$
  - $S \leftarrow S \cup \{v\}$
  - **For each** edge vz, $z \notin S$,
    - **If** $t(v) + w(vz) < t(z)$
    - **then** $t(z) \leftarrow t(v) + w(vz)$ and $prev(z) \leftarrow v$
    - **endif**
- **Until** $S = V$ or $\forall z \notin S, \; t(z) = \infty$

Shortest Paths: Dijkstra's Algorithm

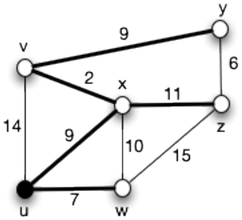
REMARKS:

- S is the set of vertices for which the shortest path to u is known
- For any vertex z, $t(z)$ is the shortest path length yet found
- $prev(z)$ is the previous vertex on the shortest (u, z)-path yet found
- The shortest paths form together a spanning tree generated from u
- Complexity: $O(n^2)$ (linear search of the min value in a linked list)

Shortest path from u to t

- **Begin** $P = \emptyset$, $v \leftarrow t$
- **While:** $prev(v)$ is defined
- **Do:**
  - Insert v at the beg. of P
  - $v \leftarrow prev(v)$
- **endwhile**

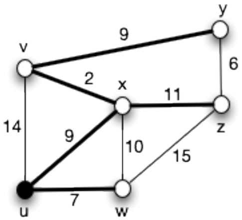
Shortest Paths: Example



Shortest Paths: Example

Algorithm steps:

1. $S = \emptyset$, $t(u) = 0$, $\forall z \neq u$, $t(z) = \infty$ and $prev(z) = undefined$
2. $S \leftarrow S \cup \{u\}$, $t(v) = 14$, $t(x) = 9$, $t(w) = 7$, $prev(v) = prev(x) = prev(w) = u$
3. $S \leftarrow S \cup \{w\}$, $t(z) = 22$, $prev(z) = w$
4. $S \leftarrow S \cup \{x\}$, $t(z) = 20$, $t(v) = 11$, $prev(z) = prev(v) = x$
5. $S \leftarrow S \cup \{v\}$, $t(y) = 20$, $prev(y) = v$
6. $S \leftarrow S \cup \{z\}$
7. $S \leftarrow S \cup \{y\}$



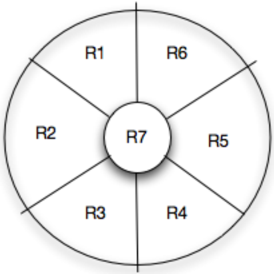
Shortest Paths: Example in Communication Networks

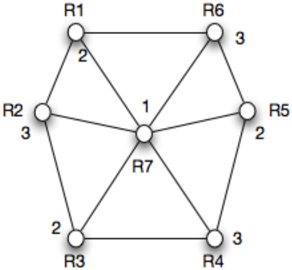
LINK STATE ROUTING (E.G. OSPF):

- Each router determines its neighbors
- *Link state advertisements* are flooded through the network, they include the node id and its neighbors
- Each router creates a graph (a map) of the network
- Each router independently runs Dijkstra's algorithm
- Routing table are built based on the best next hop for every destination

Edge Colouring: Introduction

- Vertex and edge colouring problems were at the origin of the graph theory
- Example: the "four colour" theorem (1976)





Edge Colouring: Introduction

- Frequency assignment with adjacent channel constraint in GSM is a T-coloring problem:
- In a region, there are n BS $V = \{x_1, ..., x_n\}$
- We define the *interference graph* $G = (V, E)$: $(x_i, x_j) \in E$ iff x_i and x_j interfere
- We wish to assign a frequency $f(x) \in N$ to each BS
- There is a set T of non negative integers of disallowed separations

$$(x, y) \in E \Rightarrow |f(x) - f(y)| \notin T$$

- Ex: 3 vertices complete graph and $T = \{0, 1, 4, 5\}$. Greedy algorithm provides 1, 3, 9 assignment (span is 8). Another T-coloring is 1, 4, 7 (span is 6).

Edge Colouring: Definitions

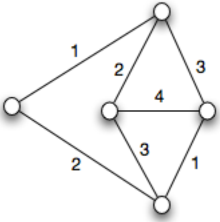
Definition

A k-**edge-colouring** is a labeling $f: E(G) \rightarrow S$ with $|S| = k$, the labels are **colors**. A k-edge-colouring is **proper** if incident edges have different labels. A graph is k-**edge-colorable** if it has a proper k-edge-colouring. The **edge chromatic index** of a loopless graph G, $\chi'(G)$, is the smallest k such that G is k-edge-colorable.

- A graph with loops has no proper edge-colouring
- Multiple edges are possible in this definition

Edge Colouring: Example

- The following graph is colored with 4 colors
- Its edge chromatic index is also $\chi'(G) = 4$



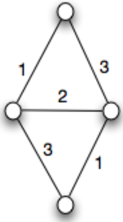
Edge Colouring: Some Results

First result

$$\Delta_G \leq \chi'(G) \leq m,$$

where Δ_G is the maximum degree and m is the number of edges.

- The upper bound is obvious (attribute one color per edge)
- The lower bound is often reached

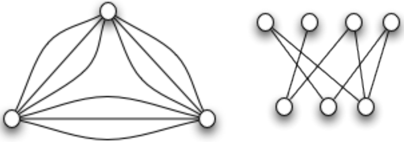


Edge Colouring: Some Results

Theorem

If G is a simple graph, $\chi'(G)$ is Δ_G or $\Delta_G + 1$

- This is not true any more in case of multiple edges
- Example of the flat triangle
- In this case, $\chi'(G) = 9$ and $\Delta_G = 6$



Theorem

If G is a bipartite graph (simple or not), $\chi'(G) = \Delta_G$

Edge Colouring: The Scheduling Problem

- p professors have to give lectures to c student classes
- Lectures are given in time slots along the week
- Lectures are characterized by the number of slots to be given
- We look for a schedule of the week
- Constraint 1: a professor hasn't two lectures at the same time
- Constraint 2: a student class has no lecture with two professors at the same time

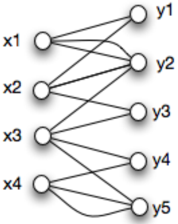
Edge Colouring: The Scheduling Problem

- Example: 4 professors and 5 classes

	y_1	y_2	y_3	y_4	y_5
x_1	1	2	0	0	0
x_2	1	1	1	0	0
x_3	0	1	1	1	1
x_4	0	0	0	1	2

Edge Colouring: The Scheduling Problem

- Modelization with a bipartite graph:
- $G = (X, Y, E)$, where
- X is the set of professors,
- Y the set of classes and
- $(x, y) \in E$ if x has to teach one slot to y

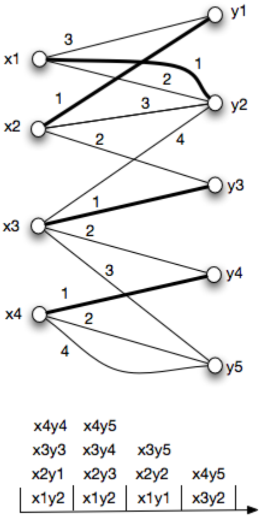


Edge Colouring: The Scheduling Problem

- G is a bipartite graph
- $m = 13$, $n = 9$ and $\Delta_G = 4$
- A possible schedule is a proper edge-colouring of G
- According to the previous theorem, the minimum number of needed slots is $\chi'(G) = \Delta_G = 4$

Edge Colouring: The Scheduling Problem

- **Geedy** algorithm:
- Consider vertices in descending order of degree
- Attribute to each edge an admissible color with the lowest index



Conclusion

Some classical topics not considered in this lecture:

- Vertex colouring
- Matching
- Flows
- Planar graphs
- Directed graphs
- Cycles

French/English Glossary

- Edge: arête
- Vertex: sommet
- Endpoint: extrémité
- Path: chaîne
- Connected: connexe
- Tree: arbre
- Spanning tree: arbre couvrant
- Weighted graph: graphe valué
- Directed graph: graphe orienté
- Greedy: glouton
- Component: composante connexe.
- Shortest paths: chemins optimaux
- Linked list: liste chaînée
- Set: ensemble
- Loop: boucle
- Bipartite: biparti
- Adjacency matrix: matrice d'adjacence
- Leaf: sommet pendant
- To sort: trier
- Neighbor: voisin
- Colouring: coloration

References

Results and examples are taken from the two references below:

- Jean-Claude Fournier, "Théorie des graphes et applications", Hermes, 2006
- Douglas B. West, "Introduction to Graph Theory", Pearson Education, 2001