Network models and graph theory

G. Ferrari Trecate

Dipartimento di Ingegneria Industriale e dell'Informazione (DIII)
Università degli Studi di Pavia

Industrial Automation

Outline

Outline

Optimization on networks

Methods for solving decision problems where the unknowns can only take finitely many values

Focus on problems that can be represented as a graph or a network such as

- logistic or transport problems
- network design problems
- project management problems

- Several optimization problems on networks are computationally *very demanding*
- However, there are interesting industrial problems that can be solved in an efficient fashion

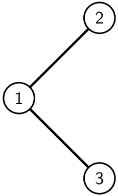
Outline

Basic definitions

Undirected graph

A graph is *undirected* if $G = (V, E)$ is a pair comprising a finite set of *vertices (or nodes)* $V = \{1, 2, \ldots, n\}$ and a set of $E \subset V \times V$ *unordered* pairs called *edges (or arcs)*.

Example: $V = \{1, 2, 3\}$, $E = \{(1, 2), (1, 3)\}$



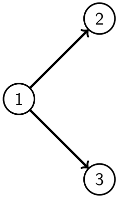
Undirected edges : $(1, 2) = (2, 1)$, $(1, 3) = (3, 1)$

Basic definitions

Directed graph

A graph G is *directed* (or digraph) if all pairs in E are *ordered* .

Example:: $V = \{1, 2, 3\}$, $E = \{(1, 2), (1, 3)\}$



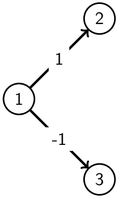
Ordered edges: $(1, 2) \neq (2, 1)$, $(1, 3) \neq (3, 1)$

Basic definitions

Network

If a function $c : E \to \mathbb{R}$ is specified, a graph is called *weighted* or *network*. If the graph is directed, one has a directed network.

Example of directed network: $V = \{1, 2, 3\}$, $E = \{(1, 2), (1, 3)\}$
$c(1, 2) = 1$ e $c(1, 3) = -1$

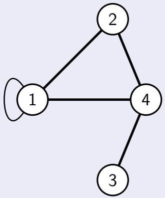


Basic definitions

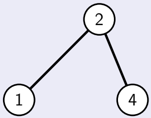
Subgraph

The graph $H = (U, F)$ is a *subgraph* of $G = (V, E)$ if $U \subseteq V$, $F \subseteq E$ and edges in F connect only vertices in U.

Graph $G = (V, E)$



Graph $H = (U, F)$



$H = (U, F)$ is a subgraph of G because $U = \{1, 2, 4\} \subseteq V$,
$F = \{(1, 2), (2, 4)\} \subseteq E$ and edges in F connects only vertices in U.

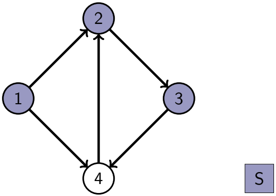
Cuts

Directed cuts

Let $G = (V, E)$ be a digraph and $S \subseteq V$. The *directed cuts* associated to S are the sets of edges

$$\delta^+(S) = \{(i,j) \in E : i \in S,\ j \notin S\} \qquad \text{(edges leaving } S\text{)}$$
$$\delta^-(S) = \{(i,j) \in E : i \notin S,\ j \in S\} \qquad \text{(edges entering in } S\text{)}$$

For an undirected graph $\delta^+(S) = \delta^-(S)$.



$$\delta^+(\{1,2,3\}) = \{(1,4),(3,4)\}$$
$$\delta^-(\{1,2,3\}) = \{(4,2)\}$$

$$\delta^-(\{3,4\}) = \{(1,4),(2,3)\}$$

Graph connectivity

Path

A sequence of arcs $e_1 e_2 \cdots e_k$ such that

$$e_1 = (v_1, v_2), e_2 = (v_2, v_3), \ldots, e_k = (v_k, v_{k+1})$$

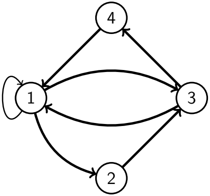
is a *path* from v_1 to v_{k+1}.

Notation: $v_1 v_2 \cdots v_{k+1}$

Path classification

- A path from a vertex to itself is a *cycle*.
- A path is *elementary* if it does not contain the same *edge* twice.
- A path is *simple* if it does not not pass through the same *vertex* twice (with the exception of the starting vertex for a cycle).

Graph connectivity



- The path 12313 is elementary but not simple.
- The path 1234 is simple and elementary.
- The paths 1231 and 11 are cycles.

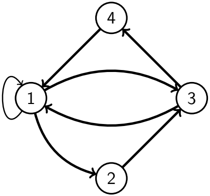
Remarks

- All simple paths are also elementary (if the same vertex is not crossed twice, the path cannot contain the same edge twice).

Graph connectivity

Hamiltonian paths

A path is *Hamiltonian* if it simple and contains all vertices (except the starting vertex for a cycle).



The path 1234 is Hamiltonian. The path 11234 is not Hamiltonian. The cycle 12341 is Hamiltonian.

Remark

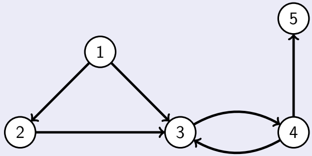
There is no simple algorithm for checking if a graph contains a Hamiltonian cycle ...

Graph connectivity

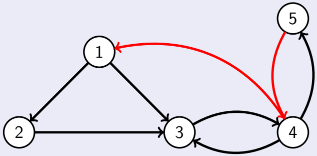
Connectivity and completeness

A vertex v_2 is *connected* to v_1 if there is a path from v_1 to v_2.
- A graph is *connected* if all pairs of vertices are connected.
- A graph $G = (V, E)$ is *complete* if $E = V \times V$

Disconnected graph



Connected graph

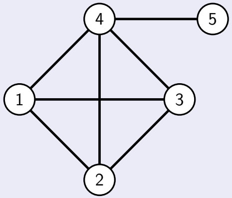


Graph connectivity

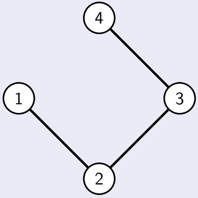
Tree

If $G = (V, E)$ is undirected, a connected subgraph with k vertices and $k - 1$ edges is a *tree*. A tree is *spanning* if $k = n$, with $n = |V|$.

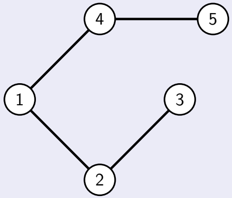
Graph



Tree



Spanning tree

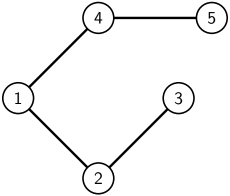


Graph connectivity

Tree theorem

Let T be an undirected graph. The following conditions are equivalent:

- T is a tree
- T is a connected acyclic graph
- T is acyclic and the addition of any arc produces a simple cycle
- T is connected and the deletion of any arc makes T disconnected
- Every pair of vertices of T is connected by a unique simple path



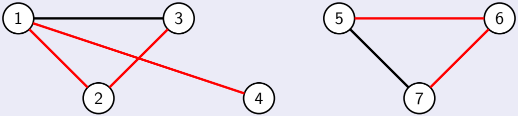
Graph connectivity

Forest

A *forest* is an undirected acyclic graph. A subgraph of G is a *maximal* forest if the addition of any edge produces a cycle.

Example

Red edges define a maximal forest



Remarks

A forest is always the union of disjoint trees. A spanning tree is a maximal forest.

Optimization on networks

Some interesting problems

Problem A Is an undirected graph connected ?

Problem B Given a digraph and two nodes v_1 and v_2, check if v_2 is connected to v_1.

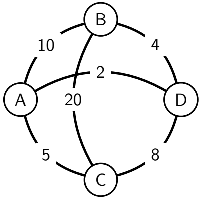
Problem C Does an undirected graph contain a Hamiltonian cycle ?

Problem TSP (Travelling Salesman Problem) Given an undirected complete network and a number $r \in \mathbb{R}$ check if it contains a Hamiltonian cycle of cost less than r.

Problem TSP

- Vertices = towns
- Weights = distances (in Km)

The traveling salesman must visit all town and be back to the first one covering less than r Km



Optimization on networks

Enumeration algorithm

All problems can be solved by computing all paths of length less than (or equal to) $|V|$ contained in the graph
Highly inefficient method: the number of paths explodes with the number of vertices

Which is the computational time needed for solving problems A, B, C and TSP ? It depends upon the adopted algorithm ...
- A rigorous answer is provided by the computational complexity theory

Computational complexity

- Quantify the efficiency of a given algorithms
- Quantify the intrinsic difficulty of a problem