

# Unit 4. Network Analysis

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya  
Study Abroad. Operations Research

*jordi.villa@uvic.cat*

16-23/05, 2023

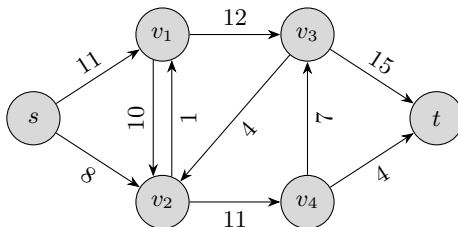
This course is strongly based on the monography on Operations Research by Carter, Price and Rabadi [Carter, 2019], and in material obtained from different sources (quoted when needed through the slides).

# Learning outcomes

- Getting familiar with the use of network analysis in OR
- Understanding maximum flow in networks
- Understanding minimum cost network flow problems
- Applying network connectivity to LP problems
- Solving shortest path problems
- Understanding and applying dynamic programming

# The concept

- Network analysis provides a framework for the study of a special class of linear programming problems that can be modeled as network programs.
- Some of these problems correspond to a physical or geographical network of elements within a system, while others correspond more abstractly to a graphical approach to planning or grouping or arranging the elements of a system.



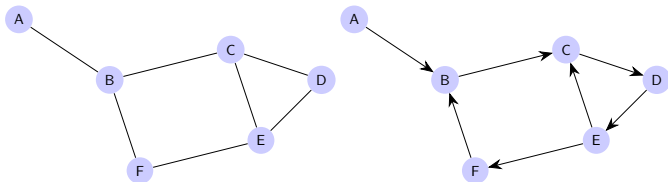
# Examples

- Systems of highways, railroads, shipping lanes, or aviation patterns, where some supply of a commodity is transported or distributed to satisfy a demand;
- pipeline systems or utility grids can be viewed as fluid flow or power flow networks;
- computer communication networks represent the flow of information;
- an economic system may represent the flow of wealth;
- routing a vehicle or a commodity between certain specified points in the network;
- assigning jobs to machines, or matching workers with jobs for maximum efficiency;
- project planning and project management, where various activities must be scheduled in order to minimize the duration of a project or to meet specified completion dates, subject to the availability of resources;
- and many, many others.

# Summary

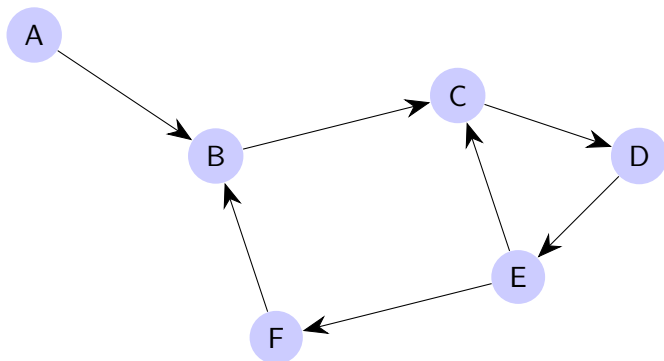
- 1 Definitions
- 2 Maximum flow
- 3 Minimum Cost Network Flow
- 4 References

# Graphs and networks: definitions



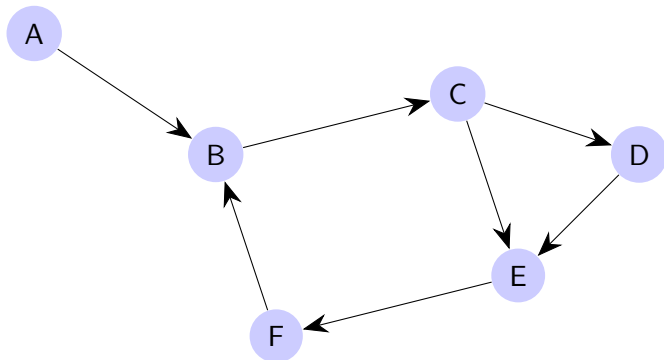
- A graph is a structure consisting of a set of nodes (vertices, points, or junctions) and a set of connections called arcs (edges, links, or branches).
- Each connection is associated with a pair of nodes and is usually drawn as a line joining two points. The graph can be defined as directed or undirected.
- The degree of a node is the number of arcs attached to it. An isolated node is of degree zero.
- In a directed graph, the arc is often designated by the ordered pair  $(A, B)$ .

# Paths



$A, (A, B), B, (B, C), \underbrace{C, (C, D), D, (D, E), E, (E, C)}_{\text{cyclic}}, C$

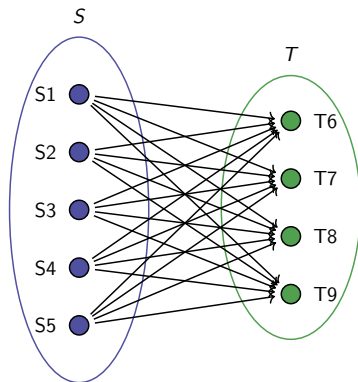
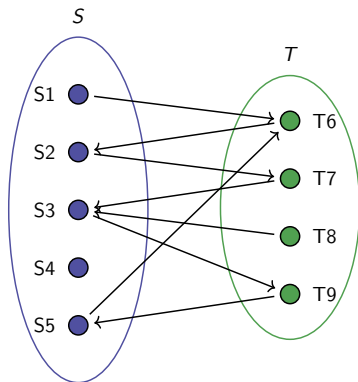




$A, (A, B), B, (B, C), \underbrace{C, (C, D), D, (D, E), E, (C, E), C}_{\text{noncyclic}}$

- If all the arcs in a path are forward arcs, the path is called **directed chain** or simply **chain**.
- **path** and **chain** are synonymous if the graph is undirected.
- In the second example above we saw a cyclic path but not a cyclic chain, as it included the backward arc  $(C, E)$ .
- A **connected graph** has at least one path connecting every pair of nodes.
- In a **bipartite graph** the nodes can be partitioned into two subsets  $S$  and  $T$ , such that each node is in exactly one of the subsets, and every arc in the graph connects a node in set  $S$  with a node in set  $T$ .
- Such a graph is **complete bipartite** if each node in  $S$  is connected to every node in  $T$ .

# Bipartite vs complete bipartite graphs



- A **tree** is a directed connected graph in which each node has at most one predecessor, and one node (the root node) has none. In an undirected graph, we have a tree if the graph is connected and contains no cycles.
- A **network** is a directed connected graph that is used to model/represent a system/process. The arcs are typically assigned weights representing cost, value or capacity corresponding to each link.
- Nodes in networks can be designated as **sources** or **sinks**. A **cut set** is any set of arcs which, if removed from the network, would disconnect the source(s) from the sink(s).
- **Flow** can be thought of as the total amount of an entity that originates at the source, makes it through the different nodes and reaches the sink.

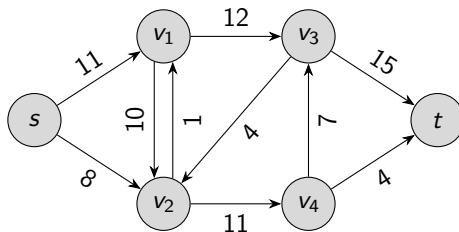
# Summary

- 1 Definitions
- 2 Maximum flow
- 3 Minimum Cost Network Flow
- 4 References

## Maximum flow in networks

Determine the maximum possible flow that can be routed through the various network links, from source to sink, without violating the capacity constraints.

Important!: the commodity is only generated at the source and consumed at the sink.



The **maximum flow problem** can be stated as a LP formulation.

$$\text{maximize } z = f$$

$$\begin{aligned} & \sum_{i=2}^n x_{1i} = f \\ & \sum_{i=1}^{n-1} x_{in} = f \\ \text{subject to } & \sum_{i=1}^n x_{ij} = \sum_{k=1}^n x_{jk}, \quad \text{for } j = 2, 3, \dots, n-1 \\ & x_{ij} \leq u_{ij}, \quad \text{for all } i, j = 1, 2, \dots, n \end{aligned}$$

# Maximum flow algorithm

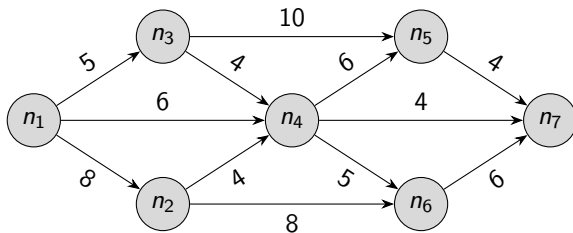
All network problems here can be solved using the Simplex method, but the network structure can help us solving it more efficiently. In the **Ford-Fulkerson labelling algorithm**:

- 1 Use a labelling procedure to look for a flow augmenting path. If none can be found, stop; the current flow is optimal;
- 2 Increase the current flow as much as possible in the flow augmenting path, until reaching capacity of some arc. Come back to step 1.



## Exercise 1

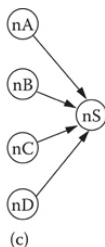
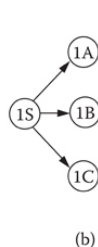
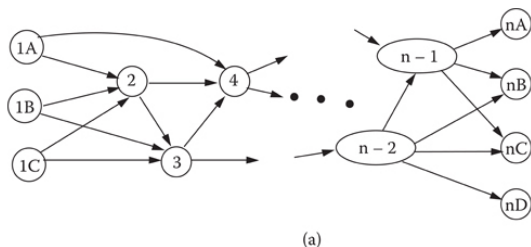
Find the maximum flow in this network using the Ford-Fulkerson labelling algorithm:



- In any network, there is always a bottleneck that in some sense impedes the flow through the network.
- The total capacity of the bottleneck is an upper bound on the total flow in the network.
- Cut sets are, by definition, essential in order for there to be a flow from source to sink, since removal of the cut set links would render the sink unreachable from the source.
- The capacities on the links in any cut set potentially limit the total flow.
- The minimum cut (i.e., the cut set with minimum total capacity) is in fact the bottleneck that precisely determines the maximum possible flow in the network (Max-Flow Min-Cut Theorem): the capacity of the cut is precisely equal to the current flow and this flow is optimal. In other words, a saturated cut defines the maximum flow.

# Multiple sinks and sources

We can generate a supersource or a supersink node with unlimited capacity and repeat the process of optimization as above:



# Summary

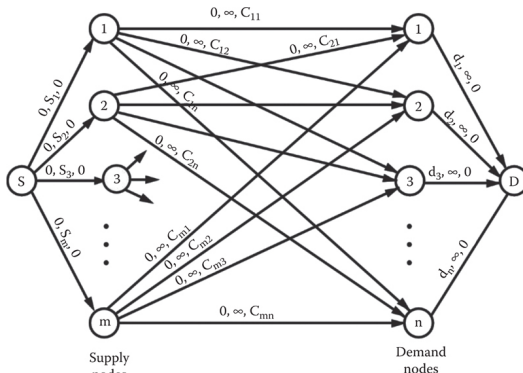
- 1 Definitions
- 2 Maximum flow
- 3 Minimum Cost Network Flow**
- 4 References

# Transportation problem

- Useful when there are costs associated with the flow, given a link capacity.
- Let us assume that every node is a source (supply) and a sink (demand). Imagine a distributor with several warehouses and a group of costumers. Serving each customer from a given warehouse has an associated cost.
- For  $m$  supply nodes, each providing  $s_i$  supply, and  $n$  demand nodes, each demanding  $d_j$ . Assuming that the total demand equals the total supply:  $\sum_{i=1}^m s_i = \sum_{j=1}^n d_j$  we aim at satisfying the demand using the available supply minimizing cost routes.

$$\text{minimize } z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

$$\begin{aligned} \text{subject to } \quad & \sum_{j=1}^n x_{ij} = s_i \quad \text{for } i = 1, \dots, m \\ & \sum_{i=1}^m x_{ij} = d_j \quad \text{for } j = 1, \dots, n \\ & x_{ij} \geq 0 \quad \text{for all } i, j \end{aligned}$$



## Exercise 2

Find the minimum cost in this transportation problem:

Sources	Sinks (Customers)					
(Warehouses)	1	2	3	4	5	Supply
1	28	7	16	2	30	20
2	18	8	14	4	20	20
3	10	12	13	5	28	25
Demand	12	14	12	18	9	65

NOTE: The Simplex method says that we should first find any basic feasible solution and then look for a simple pivot to improve the solution. repeat until the optimal solution is found.

# Optimizing

- ① Finding initial solution.
  - Northwest corner rule
  - Minimum cost method
  - Minimum "row" cost method
  - Vogel's method
- ② Transportation simplex



# Transportation simplex

Once we have any feasible solution, we aim at finding the optimal one.  
Consider:

## Minimum Row Cost Final Solution

Sources		Sinks (Customers)									
(Warehouses)	1		2		3		4		5		Supply
1		28	2	7		16	18	2		30	20
2	8	18	12	8		14		4		20	20
3	4	10		12	12	13		5	9	28	25
Demand	12		14		12		18		9		65

# Transportation simplex

We can reduce the total cost by reducing the individual costs in every row  $i$  by  $u_i$  and in every column  $j$  by  $v_j$ :

$$c'_{ij} = c_{ij} - u_i - v_j$$

Check that, now:

- $\sum_i \sum_j x_{ij} c'_{ij} = 0$
- Check how some costs are now negative in non-basic cells.
- If we increase the number of units in those non-basic cells from 0 to some value, reducing at the same time the number of units in the basic cells, we can reduce the overall cost  $\sum_i \sum_j x_{ij} c_{ij}$

# Transportation simplex

In practice:

- ① We find first an initial feasible solution as explained above
- ② Calculate the  $u_i$  and  $v_j$ , taking into account that  $c_{ij} = u_i + v_j$ , for all basic variables (used squares in the table). We start by assigning  $u_1 = 0$ .
- ③ We calculate the *improvement index* by  $l_{ij} = c_{ij} - u_i - v_j$  for all non-used squares in the table.
- ④ If all  $l_{ij}$  are positive, the solution is already optimal and we are done.
- ⑤ If some  $l_{ij} < 0$ , then build a loop with such value in the corner and alternative  $\pm$  signs in all vertex.
- ⑥ Use the above  $\pm$  to increase the number of units in the position that had  $l_{ij} < 0$
- ⑦ We return to step 2.

# Summary

- 1 Definitions
- 2 Maximum flow
- 3 Minimum Cost Network Flow
- 4 References



Michael W. Carter, Camille C. Price,  
and Ghaith Rabadi (2019)  
Operations Research, 2nd Edition  
*CRC Press.*



David Harel, with Yishai Feldman  
(2004)  
Algorithmics: the spirit of computing,  
3rd Edition  
*Addison-Wesley.*



K.F. Riley, M.P. Hobson, S.J. Bence  
(2002)  
Mathematical Methods for Physics and  
Engineering (2nd Ed)  
*McGraw Hill.*



J. Nocedal, S. J. Wright (2006)  
Numerical Optimization (2nd Ed)  
*Springer.*