Graphs, network flows, graph mining

lecturer: András London

University of Szeged
Institute of Informatics
Department of Computational Optimization

Graphs

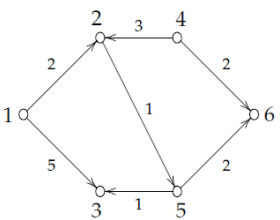
G = **graph** or **network** consists of

- a set V of **vertices** (nodes, points) and
- a set E of **edges** (arcs, lines) which are connections between vertices.

write $G = (V, E)$; write $V(G)$ for vertices of G, and $E(G)$ for edges of G.

(vertices are usually denoted u or v with subscripts; edges we usually denote e)

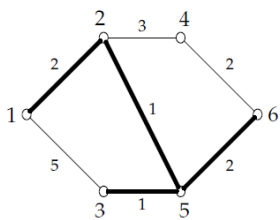
edges may have **direction**: an edge e between u and v may go from u to v, we write $e = (u, v)$,



vertices $V = \{1, 2, 3, 4, 5, 6\}$

edges $E = \{(1,2), (1,3), (2,5), (4,2), (4,6), (5,3), (5,6)\}$

weights $c(1,2) = 2 \quad c(1,3) = 5$
$c(2,5) = 1 \quad c(4,2) = 3 \quad c(4,6) = 2$
$c(5,3) = 1 \quad c(5,6) = 2$



Graphs

if all edges do not have a direction (are undirected), we say that the network is **undirected**

edges may have **weight**: a weight of edge $e = (u, v)$ is a real number denoted $c(e)$ or $c(u, v)$, c_e, c_{uv}

a sequence of nodes and edges $v_1, e_1, v_2, e_2, \ldots v_{k-1}, e_k, v_k$ is

- a **path** (directed path) if each e_i goes from v_i to v_{i+1}
- a **chain** (undirected path) if each e_i connects v_i and v_{i+1} (in some direction)

(often we write: $e_1, e_2, \ldots, e_k$ is a path (we omit vertices) or write: $v_1, v_2, \ldots, v_k$ is a path (we omit edges))

a network is **connected** if for every two nodes there is a path connecting them; otherwise it is **disconnected**

a **cycle** (loop, circuit) is a path starting and ending in the same node, never repeating any node or edge

a **forest** (acyclic graph) is an undirected graph that contains no cycles

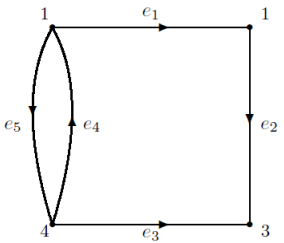
a **tree** is a connected forest

Claim: A tree with n nodes contains exactly $n - 1$ edges. Adding any edge to a tree creates a cycle. Removing any edge from a tree creates a disconnected forest.

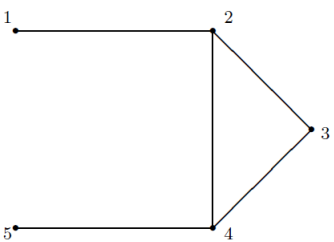
Graph representations

- Graphs can be represented by matrices. For us the most important ones are the
  - incidence matrix
  - adjacency matrix

	e_1	e_2	e_3	e_4	e_5
1	-1	0	0	1	-1
2	1	-1	0	0	0
3	0	1	1	0	0
4	0	0	-1	-1	1



	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	0	0	0	1	0



Brief history and motivation

The Shortest Path Problem is one of the most important efficient computational tools at this disposal

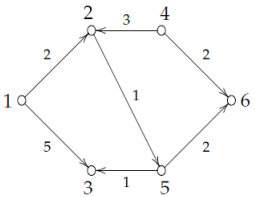
- It is used everywhere, from GPS navigation and network communication to project management, layout design, robotics, computer graphics, and the list goes on
- First algorithms for the Shortest Path problem were designed by Ford (1956), Bellman (1958), and Moore (1959). For non-negative weights, a more efficient algorithm was first suggested by Dijkstra (1959).
- All-pairs Shortest Path problem the first algorithms were founds by Shimbel (1953), and by Roy (1959), Floyd (1962), and Warshall (1962)

Brief history and motivation

The Minimum Spanning Tree problem also has a rich history

- The first known algorithm was developed by Boruvka (1926) for efficient distribution of electricity
- Later independently discovered by many researchers over the years: Jarnik (1930), Kruskal (1956), Prim(1957), and Dijkstra (1959)

Shortest path problem



vertices $V = \{1, 2, 3, 4, 5, 6\}$

edges $E = \Big\{(1,2), (1,3), (2,5), (4,2), (4,6), (5,3), (5,6)\Big\}$

weights $c(1,2) = 2 \quad c(1,3) = 5$
$c(2,5) = 1 \quad c(4,2) = 3 \quad c(4,6) = 2$
$c(5,3) = 1 \quad c(5,6) = 2$

Given a network $G = (V, E)$ with two distinguished vertices $s, t \in V$, find a shortest path from s to t

Example: In Figure 1 (left), a shortest path from $s = 1$ to $t = 6$ is $1, 2, 5, 6$ of total length 5, while for $t = 3$ a shortest path is $1, 2, 5, 3$ of length 4. We say that **distance** from node 1 to node 6 is 5. Note that there is no path from s to $t = 4$; we indicate this by defining the distance to 4 as ∞.

LP formulation: decision variables x_{ij} for each $(i, j) \in E$

$$\text{Min} \quad \sum_{(i,j)\in E} w_{ij} x_{ij}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{otherwise} \end{cases} \qquad \text{for each } i \in V$$

$$x_{ij} \in \{0, 1\} \qquad \text{for each } (i, j) \in E$$

Dijksra's algorithm

Algorithm finds the length of a shortest path from s to every vertex of G (not only t)

Weights of edges are assumed to be **non-negative**, else the algorithm may output incorrect answer.

variables: d_u for each $u \in V$, an **estimate** on the distance from s to u

initialize: $d_u = \begin{cases} 0 & \text{if } u = s \\ \infty & \text{otherwise} \end{cases}$ all vertices are initially *unprocessed*

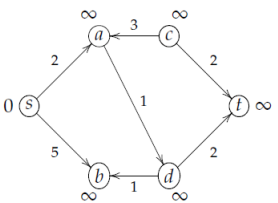
1. Find an *unprocessed* vertex u with smallest d_u
2. For each $(u, v) \in E$, update $d_v = \min\{d_v, d_u + c_{uv}\}$
3. Mark u as processed; repeat until all vertices are processed.
4. Report d_t as distance from s to t

Dijksra's algorithm

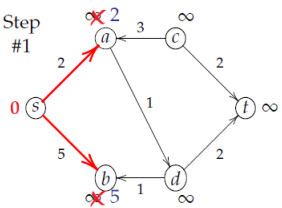
1. Find an *unprocessed* vertex u with smallest d_u
2. For each $(u, v) \in E$, update $d_v = \min\{d_v,\ d_u + c_{uv}\}$
3. Mark u as processed; repeat until all vertices are processed.
4. Report d_t as distance from s to t

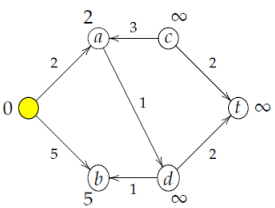
Step#	s	a	b	c	d	t
1.	$\underline{0}$	∞	∞	∞	∞	∞
		2	5			
2.	0^*	$\underline{2}$	5	∞	∞	∞
					3	
3.	0^*	2^*	5	∞	$\underline{3}$	∞
			4			5
4.	0^*	2^*	$\underline{4}$	∞	3^*	5
5.	0^*	2^*	4^*	∞	3^*	$\underline{5}$
6.	0^*	2^*	4^*	$\underline{\infty}$	3^*	5^*
final	0^*	2^*	4^*	∞^*	3^*	5^*

Example:

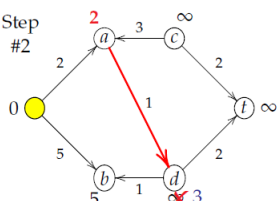


Step #1

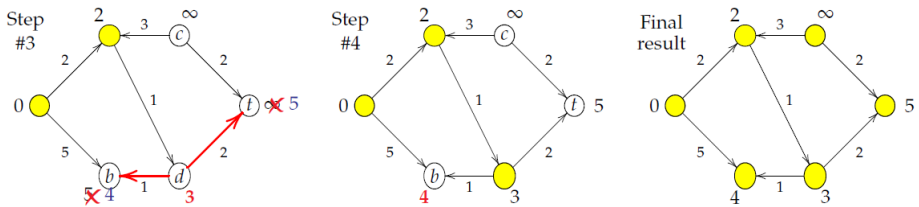




Step #2



Dijksra's algorithm



We can read the shortest path from 1 to 6: that is path 1,2,5,6.

Minimum spanning tree

A power company delivers electricity from its power plant to neighbouring cities. The cities are interconnected by power lines operated by various operators. The power company wants to rent power lines in the grid of least total cost that will allow it to send electricity from its power plant to all cities.

Given an undirected network $G = (V, E)$ find a collection $F \subseteq E$ of minimum weight so that (V, F) is a tree.

(we say that (V, F) is a **spanning** tree because it spans all vertices)

Kruskal's (Prim's) algorithm

initialize: F to be empty; all edges are initially *unprocessed*

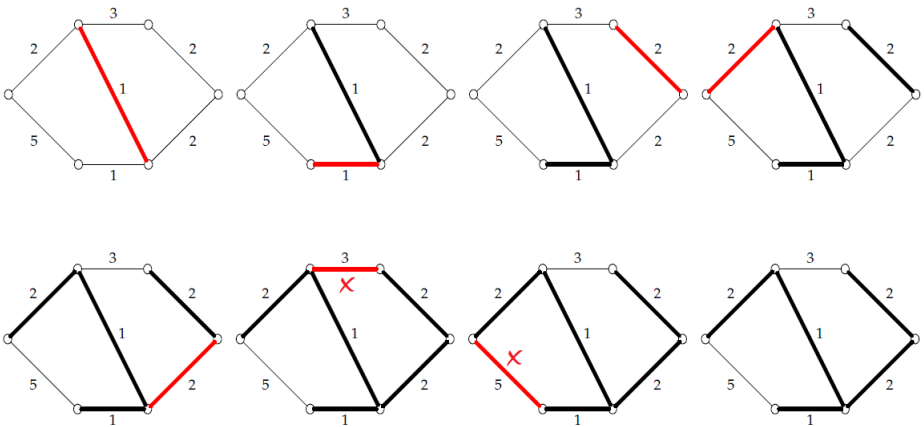
Kruskal's algorithm:

1. Find an unprocessed edge e of smallest weight w_e.
2. If $(V, F \cup \{e\})$ is a forest, then add e to F.
3. Mark e as processed and repeat until all edges have been processed.
4. Report (V, F) as a minimum-weight spanning tree.

Prim's algorithm: replace 1 by $1'$

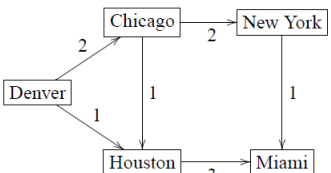
$1'$ Find an unprocessed edge e of smallest weight that shares an endpoint with some edge in F

Minimum spanning tree - Kruskal's algorithm



Maximum flow problem

A delivery company runs a delivery network between major US cities. Selected cities are connected by routes as shown below. On each route a number of delivery trucks is dispatched daily (indicated by labels on the corresponding edges). A customer is interested in hiring the company to deliver his products daily from Denver to Miami, and needs to know how much product can be delivered on a daily basis.



maximize z

$$
\begin{array}{rcl}
-x_{DC}-x_{DH} & = & -z \\
x_{DC}-x_{CH}-x_{CN} & = & 0 \\
x_{DH}+x_{CH}-x_{HM} & = & 0 \\
x_{CN}-x_{NM} & = & 0 \\
x_{NM}+x_{HM} & = & z
\end{array}
$$

conservation of flow

$$
\begin{array}{l}
0 \le x_{DC} \le 2 \\
0 \le x_{DH} \le 1 \\
0 \le x_{CH} \le 1 \\
0 \le x_{CN} \le 2 \\
0 \le x_{NM} \le 1 \\
0 \le x_{HM} \le 3
\end{array}
$$

Maximum flow problem

In general, network $G = (V, E)$:

$s =$ **source** (Denver) — $u_{ij} =$ *capacity* of an edge ij (# trucks dispatched daily between i and j)

$t =$ **sink** (Miami) — $x_{ij} =$ *flow* on an edge ij (# trucks delivering the customer's products)

$$\max\ z$$

$$\underbrace{\sum_{\substack{j \in V \\ ji \in E}} x_{ji}}_{\text{flow into } i} - \underbrace{\sum_{\substack{j \in V \\ ij \in E}} x_{ij}}_{\text{flow out of } i} = \begin{cases} -z & i = s \\ z & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{for all } ij \in E$$

Maximum flow problem - The Ford-Fulkerson algorithm

In general, network $G = (V, E)$:

$s =$ **source** (Denver) $\qquad u_{ij} =$ *capacity* of an edge ij (# trucks dispatched daily between i and j)

$t =$ **sink** (Miami) $\qquad x_{ij} =$ *flow* on an edge ij (# trucks delivering the customer's products)

$$\max\ z$$

$$\underbrace{\sum_{\substack{j \in V \\ ji \in E}} x_{ji}}_{\text{flow into } i} - \underbrace{\sum_{\substack{j \in V \\ ij \in E}} x_{ij}}_{\text{flow out of } i} = \begin{cases} -z & i = s \\ z & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{for all } ij \in E$$

Maximum flow problem - The Ford-Fulkerson algorithm

Initial feasible flow $x_{ij} = 0$ for all $ij \in E$.

A sequence of nodes $v_1, v_2, \ldots, v_n$ is a *chain* if $v_i v_{i+1} \in E$ (*forward* edge) or $v_{i+1} v_i \in E$ (*backward* edge) for all $i = 1, \ldots, n-1$. If $v_1 = s$ and $v_n = t$, then we call it an (s,t)-chain. Consider an (s,t)-chain P.

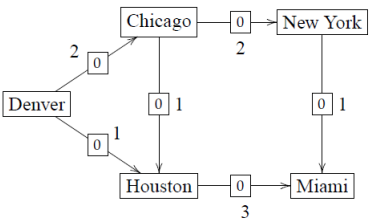
The *residual capacity* of a forward edge ij on P is defined as $u_{ij} - x_{ij}$ (the remaining capacity on the edge ij). The *residual capacity* of a backward edge ij on P is defined as x_{ij} (the used capacity of the edge ij).
The *residual capacity* of P is the **minimum** taken over residual capacities of edges on P.

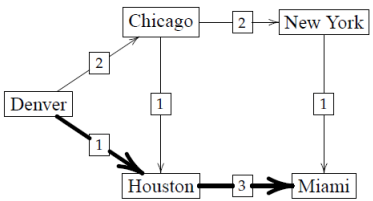
If the *residual capacity* of P is positive $\varepsilon > 0$, then P is an **augumenting chain**. If this happens, we can increase the flow by increasing the flow on all forward edges by ε, and decreasing the flow on all backward edges by ε. This yields a feasible flow of larger value $z + \varepsilon$. (Notice the similarity with the Transportation Problem and the ratio test in the Simplex Method – same thing in disguise.)

Optimality criterion: The flow x_{ij} is optimal if and only if there is no augmenting chain.

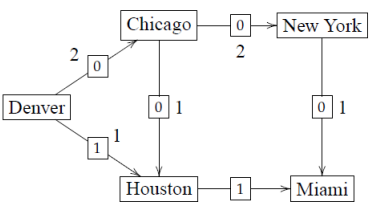
Maximum flow problem - The Ford-Fulkerson algorithm

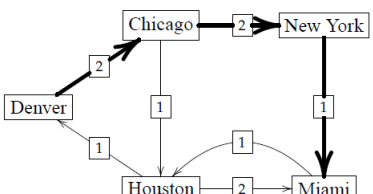
Starting feasible flow $x_{ij} = 0$ (indicated in boxes) $\rightarrow$ residual network (residual capacity shown on edges)





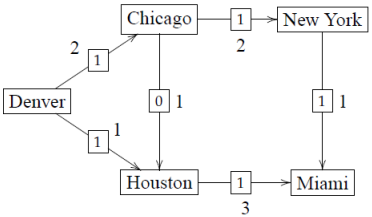
augmenting chain of residual capacity 1 $\rightarrow$ increase flow by 1

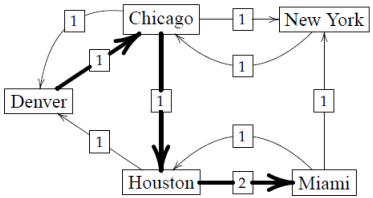




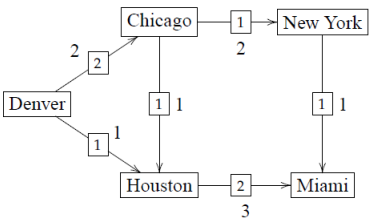
Maximum flow problem - The Ford-Fulkerson algorithm

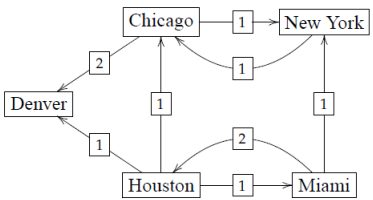
augmenting chain of residual capacity 1 → increase flow by 1





augmenting chain of residual capacity 1 → increase flow by 1





no path from Denver to Miami in the residual network → no augmenting chain → **optimal solution** found

→ maximum flow has value 3

Maximum flow problem - Minimum cut

Minimum Cut

For a subset of vertices $A \subseteq V$, the edges going between the nodes in A and the rest of the graph is called a **cut**. We write $(A, \overline{A})$ to denote this cut. The edges going out of A are called **forward edges**, the edges coming into A are **backward edges**. If A contains s but not t, then it is an (s,t)-cut.

The **capacity** of a cut $(A, \overline{A})$ is the sum of the capacities of its forward edges.

For example, let $A = \{\text{Denver,Chicago}\}$. Then $(A, \overline{A})$ is an (s,t)-cut of capacity 4. Similarly, let $A_* = \{\text{Denver, Chicago, New York}\}$. Then $(A_*, \overline{A_*})$ is an (s,t)-cut of capacity 3.

Theorem. (Max. flow - Min. cut) *The maximum value of an (s,t)-flow is equal to the minimum capacity of an (s,t)-cut.*

Maximum flow problem - Minimum cut

This is known as the Max-Flow-Min-Cut theorem – a consequence of strong duality of linear programming.

maximize z

$$
\begin{array}{rrrrrrr}
-x_{DC} & -x_{DH} & & & & = & -z \\
x_{DC} & & -x_{CH} & -x_{CN} & & = & 0 \\
& x_{DH} & +x_{CH} & & -x_{HM} & = & 0 \\
& & & x_{CN} & -x_{NM} & = & 0 \\
& & & & x_{NM} & +x_{HM} = & z
\end{array}
$$

$$
\begin{array}{l}
0 \le x_{DC} \le 2 \\
0 \le x_{DH} \le 1 \\
0 \le x_{CH} \le 1 \\
0 \le x_{CN} \le 2 \\
0 \le x_{NM} \le 1 \\
0 \le x_{HM} \le 3
\end{array}
$$

Dual:

minimize $2v_{DC} + v_{DH} + v_{CH} + 2v_{CN} + v_{NM} + 3v_{HM}$

$$
\begin{array}{rrrrrrr}
y_D & - y_C & & & & \le & v_{DC} \\
y_D & - & y_H & & & \le & v_{DH} \\
& y_C & - y_H & & & \le & v_{CH} \\
& y_C & - & y_N & & \le & v_{CN} \\
& & & y_N & - y_M \le & v_{NM} \\
& & y_H & & - y_M \le & v_{HM} \\
y_D & & & & - y_M \ge & 1
\end{array}
$$

$$v_{DC}, v_{DH}, v_{CH}, v_{CN}, v_{NM}, v_{HM} \ge 0$$

y_D, y_C, y_H, y_N, y_M unrestricted

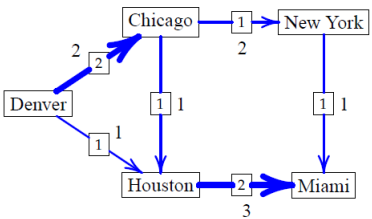
Optimal solution (of value 3)

$y_D = y_C = y_N = 1 \quad \rightarrow \quad A = \{D, C, N\}$ min-cut
$y_H = y_M = 0$

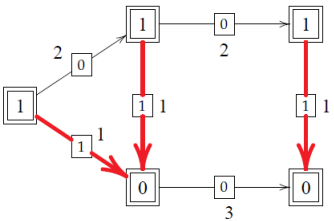
$v_{DH} = v_{CH} = v_{NM} = 1$
$v_{DC} = v_{CN} = v_{HM} = 0$

$\rightarrow$ given an optimal solution, let A be the nodes whose y value is the same as that of source
$\rightarrow (A, \overline{A})$ **minimum cut**

Maximum flow problem - Minimum cut



maximum flow



minimum cut

$$\max\ z$$

$$\sum_{\substack{j\in V\\ ji\in E}} x_{ji} - \sum_{\substack{j\in V\\ ij\in E}} x_{ij} = \begin{cases} -z & i = s \\ z & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } ij \in E$$

$$\min \sum_{ij\in E} u_{ij} v_{ij}$$

$$y_i - y_j \le v_{ij} \qquad \text{for all } ij \in E$$

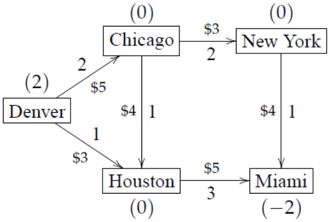
$$y_s - y_t \ge 1$$

$$v_{ij} \ge 0 \qquad \text{for all } ij \in E$$

$$y_i \text{ unrestricted} \qquad \text{for all } i \in V$$

Minimum cost flow problem

A delivery company runs a delivery network between major US cities. Selected cities are connected by routes as shown below. On each route a number of delivery trucks is dispatched daily (indicated by labels on the corresponding edges). Delivering along each route incurs a certain cost (indicated by the $ figure (in thousands) on each edge). A customer hired the company to deliver two trucks worth of products from Denver to Miami. What is the least cost of delivering the products?



minimize

$$5x_{DC} + 3x_{DH} + 4x_{CH} + 3x_{CN} + 4x_{NM} + 5x_{HM}$$

$$\underbrace{\begin{array}{rcl} x_{DC}+x_{DH} & = & 2 \\ -x_{DC} + x_{CH} + x_{CN} & = & 0 \\ -x_{DH} - x_{CH} + x_{HM} & = & 0 \\ -x_{CN} + x_{NM} & = & 0 \\ -x_{NM} - x_{HM} & = & -2 \end{array}}_{\text{conservation of flow}}$$

$$\begin{array}{l} 0 \le x_{DC} \le 2 \\ 0 \le x_{DH} \le 1 \\ 0 \le x_{CH} \le 1 \\ 0 \le x_{CN} \le 2 \\ 0 \le x_{NM} \le 1 \\ 0 \le x_{HM} \le 3 \end{array}$$

Minimum cost flow problem

Network $G = (V, E)$:

u_{ij} =*capacity* of an edge $(i, j) \in E$ (# trucks dispatched daily between i and j)

x_{ij} =*flow* on an edge $(i, j) \in E$ (# trucks delivering the customer's products)

c_{ij} =*cost* on an edge $(i, j) \in E$ (cost of transportation per each truck)

b_i =*net supply* of a vertex $i \in V$ (amount of products produced/consumed at node i)

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\underbrace{\sum_{\substack{j\in V\\ ij\in E}} x_{ij}}_{\text{flow out of } i} - \underbrace{\sum_{\substack{j\in V\\ ji\in E}} x_{ji}}_{\text{flow into } i} = \underbrace{b_i}_{\text{net supply}}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } ij \in E$$

Necessary condition: $\sum_i b_i = 0$.

If there are no capacity constraints, the problem is called the **Transshipment problem**.

Summary

Network $G = (V, E)$ has **nodes** V and **edges** E.

- Each edge $(i, j) \in E$ has a **capacity** u_{ij} and **cost** c_{ij}.
- Each vertex $i \in V$ provides **net supply** b_i.

For a set $S \subseteq V$, write $\overline{S}$ for $V \setminus S$ and write $E(S, \overline{S})$ for the set of edges $(i, j) \in E$ with $i \in S$ and $j \in \overline{S}$. The pair $(S, \overline{S})$ is called a **cut**. (Where applicable) there are two distinguished nodes: s =**source** and t =**sink**.

Minimum spanning tree

Primal

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\underbrace{\sum_{(i,j)\in E(S,\overline{S})} x_{ij}}_{\text{edges from } S \text{ to } \overline{S}} > 0 \qquad \text{for all } S \subseteq V \text{ where } \emptyset \neq S \neq V$$

$$x_{ij} \geq 0 \qquad \text{for all } (i, j) \in E$$

Obstruction (to feasibility):
set $S \subseteq V$ with $\emptyset \neq S \neq V$ such that $E(S, \overline{S}) = \emptyset$

Summary

Shortest path problem

Primal

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\underbrace{\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij}}_{\text{flow out of } i} - \underbrace{\sum_{\substack{j\in V\\(j,i)\in E}} x_{ji}}_{\text{flow into } i} = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & \text{else} \end{cases} \quad \text{for all } i \in V$$

$$x_{ij} \geq 0 \qquad \text{for all } (i,j) \in E$$

Dual

$$\max y_s - y_t$$

$$y_i - y_j \leq c_{ij} \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ with $s \in S$ and $t \in \overline{S}$ such that $E(S, \overline{S}) = \emptyset$

Summary

Maximum-flow problem

Primal

$$\max\ z$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = \begin{cases} z & i = s \\ -z & i = t \\ 0 & \text{else} \end{cases} \quad \text{for all } i \in V$$

$$0 \le x_{ij} \le u_{ij} \quad \text{for all } (i,j) \in E$$

$$z \text{ unrestricted}$$

Dual

$$\min \sum_{(i,j)\in E} u_{ij} v_{ij}$$

$$y_i - y_j + v_{ij} \ge 0 \quad \text{for all } (i,j) \in E$$

$$y_t - y_s = 1$$

$$v_{ij} \ge 0 \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ with $s \in S$ and $t \in \overline{S}$ such that $z > \underbrace{\sum_{(i,j)\in E(S,\overline{S})} u_{ij}}_{\text{capacity of the cut } (S,\overline{S})}$

(no flow bigger than the capacity of a cut)

Summary

Minimum-cost (s,t)-flow problem

Primal

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = \begin{cases} f & i=s \\ -f & i=t \\ 0 & \text{else} \end{cases} \quad \text{for all } i \in V$$

$$0 \le x_{ij} \le u_{ij} \quad \text{for all } (i,j) \in E$$

Dual

$$\max fy_s - fy_t - \sum_{(i,j)\in E} u_{ij}v_{ij}$$

$$y_i - y_j - v_{ij} \le c_{ij} \quad \text{for all } (i,j) \in E$$

$$v_{ij} \ge 0 \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ with $s \in S$ and $t \in \overline{S}$ such that $f > \sum_{(i,j)\in E(S,\overline{S})} u_{ij}$

Summary

Transshipment problem

Primal

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = \underbrace{b_i}_{\text{net supply}} \quad \text{for all } i \in V$$

$$x_{ij} \geq 0 \quad \text{for all } (i,j) \in E$$

Dual

$$\max \sum_{i\in V} b_i y_i$$

$$y_i - y_j \leq c_{ij} \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ such that $\sum_{i\in S} b_i > 0$ and $E(S,\overline{S}) = \emptyset$

Minimum-cost network flow problem

Primal

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = b_i \quad \text{for all } i \in V$$

$$0 \leq x_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in E$$

Dual

$$\max \sum_{i\in V} b_i y_i - \sum_{(i,j)\in E} u_{ij} v_{ij}$$

$$y_i - y_j - v_{ij} \leq c_{ij} \quad \text{for all } (i,j) \in E$$

$$v_{ij} \geq 0 \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ such that $\sum_{i\in S} b_i > \sum_{(i,j)\in E(S,\overline{S})} u_{ij}$

Example #1

A new car costs \$12,000. Annual maintenance costs are as follows: $m_1 = \$2,000$ first year, $m_2 = \$4,000$ second year, $m_3 = \$5,000$ third year, $m_4 = \$9,000$ fourth year, and $m_5 = \$12,000$ fifth year and on. The car can be sold for $s_1 = \$7,000$ in the first year, for $s_2 = \$6,000$ in the second year, for $s_3 = \$2,000$ in the third year, and for $s_4 = \$1,000$ in the fourth year of ownership.

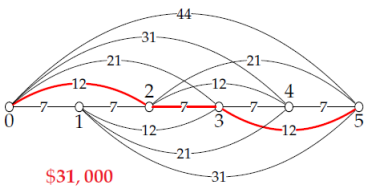
An existing car can be sold at any time and another new car purchased at \$12,000. What buying/selling strategy for the next 5 years minimizes the total cost of ownership?

Nodes = $\{0, 1, 2, 3, 4, 5\}$

Edge (i, j) represents the act of buying a car in year i and selling in year j. The weight is the price difference plus the maintanence cost, i.e., the weight is

$$c(i, j) = \$12,000 - s_{(i-j)} + m_1 + m_2 + \ldots + m_{(i-j)}$$

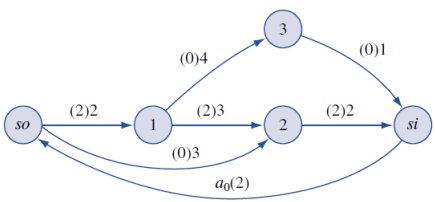
Answer: the length of a shortest path from node 0 to node 5.



\$31,000

Example #2

Sunco Oil wants to ship the maximum possible amount of oil (per hour) via pipeline from node so to node si in Figure. On its way from node so to node si, oil must pass through some or all of stations 1, 2, and 3. The various arcs represent pipelines of different diameters. The maximum number of barrels of oil (millions of barrels per hour) that can be pumped through each arc is shown in the Table. Each number is called an arc capacity. Formulate an LP that can be used to determine the maximum number of barrels of oil per hour that can be sent from so to si.



Arc	Capacity
(so, 1)	2
(so, 2)	3
(1, 2)	3
(1, 3)	4
(3, si)	1
(2, si)	2

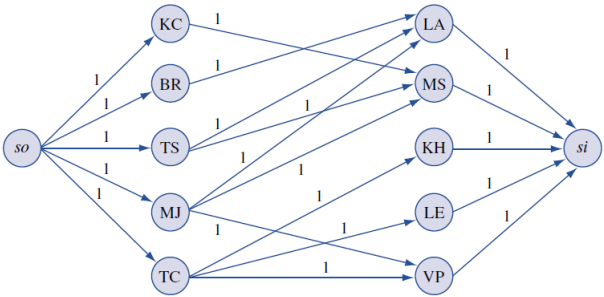
Example #3

Five male and five female entertainers are at a dance. The goal of the matchmaker is to match each woman with a man in a way that maximizes the number of people who are matched with compatible mates. Table describes the compatibility of the entertainers. Draw a network that makes it possible to represent the problem of maximizing the number of compatible pairings as a maximum-flow problem.

	Loni Anderson	Meryl Streep	Katharine Hepburn	Linda Evans	Victoria Principal
Kevin Costner	—	C	—	—	—
Burt Reynolds	C	—	—	—	—
Tom Selleck	C	C	—	—	—
Michael Jackson	C	C	—	—	C
Tom Cruise	—	—	C	C	C

Note: C indicates compatibility.

Example #3



the arc joining each woman to the sink has a capacity of 1, conservation of flow ensures that each woman will be matched with at most one man. Similarly, because each arc from the source to a man has a capacity of 1, each man can be paired with at most one woman. Because arcs do not exist between noncompatible mates, we can be sure that a flow of k units from source to sink represents an assignment of men to women in which k compatible couples are created.

- We discuss three special types of linear programming problems:
  1. **transportation**
  2. **assignment**
  3. **transshipment**
- Each of these can be solved by the simplex algorithm, but **specialized algorithms** for each type of problem are **much more efficient.**

Example 1

PowerCo has three electric power plants that supply the needs of four cities. Each power plant can supply the following numbers of kilowatt-hours (kwh) of electricity: plant 1 - 35 million; plant 2 - 50 million; plant 3 - 40 million. The peak power demands in these cities, which occur at the same time (2 P.M.), are as follows (in kwh): city 1 - 45 million; city 2 - 20 million; city 3 - 30 million; city 4 - 30 million. The costs of sending 1 million kwh of electricity from plant to city depend on the distance the electricity must travel. Formulate an LP to minimize the cost of meeting each city's peak power demand.

Shipping Costs, Supply, and Demand for Powerco

	To				
From	City 1	City 2	City 3	City 4	Supply (million kwh)
Plant 1	$8	$6	$10	$9	35
Plant 2	$9	$12	$13	$7	50
Plant 3	$14	$9	$16	$5	40
Demand (million kwh)	45	20	30	30	

Example 1 - solution

PowerCo must determine how much power is sent from each plant to each city, we define (for $i = 1, 2, 3$ and $j = 1, 2, 3, 4$)

x_{ij} = number of (million) kwh produced at plant i and sent to city j

In terms of these variables, the total cost of supplying the peak power demands to cities 1–4 may be written as

$$\begin{aligned} &8x_{11} + 6x_{12} + 10x_{13} + 9x_{14} \text{ (Cost of shipping power from plant 1)} \\ &+9x_{21} + 12x_{22} + 13x_{23} + 7x_{24} \text{ (Cost of shipping power from plant 2)} \\ &+14x_{31} + 9x_{32} + 16x_{33} + 5x_{34} \text{ (Cost of shipping power from plant 3)} \end{aligned}$$

Example 1 - solution

The LP formulation of PowerCo's problem contains the following three supply constraints:

$x_{11} + x_{12} + x_{13} + x_{14} \leq 35$ (Plant 1 supply constraint)

$x_{21} + x_{22} + x_{23} + x_{24} \leq 50$ (Plant 2 supply constraint)

$x_{31} + x_{32} + x_{33} + x_{34} \leq 40$ (Plant 3 supply constraint)

PowerCo must satisfy the following four demand constraints:

$x_{11} + x_{21} + x_{31} \geq 45$ (City 1 demand constraint)

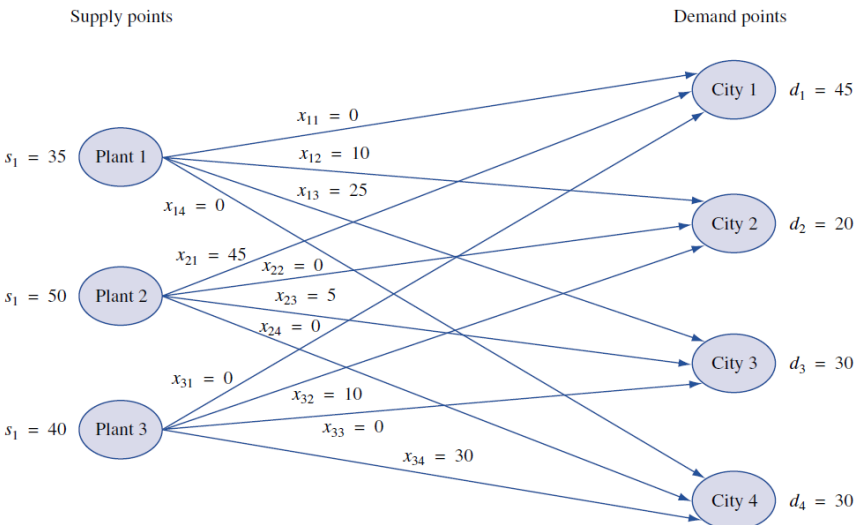
$x_{12} + x_{22} + x_{32} \geq 20$ (City 2 demand constraint)

$x_{13} + x_{23} + x_{33} \geq 30$ (City 3 demand constraint)

$x_{14} + x_{24} + x_{34} \geq 30$ (City 4 demand constraint)

Because all the x_{ij} must be nonnegative, we add the sign restrictions $x_{ij} \geq 0$ $(i = 1, 2, 3; j = 1, 2, 3, 4)$.

Example 1 - graphical representation



General description

In general, a transportation problem is specified by the following information:

1 A set of *m supply points* from which a good is shipped. Supply point i can supply at most s_i units. In the Powerco example, $m = 3$, $s_1 = 35$, $s_2 = 50$, and $s_3 = 40$.

2 A set of *n demand points* to which the good is shipped. Demand point j must receive at least d_j units of the shipped good. In the Powerco example, $n = 4$, $d_1 = 45$, $d_2 = 20$, $d_3 = 30$, and $d_4 = 30$.

3 Each unit produced at supply point i and shipped to demand point j incurs a *variable cost* of c_{ij}. In the Powerco example, $c_{12} = 6$.

Let

$$x_{ij} = \text{number of units shipped from supply point } i \text{ to demand point } j$$

then the general formulation of a transportation problem is

$$\min \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{j=n} x_{ij} \le s_i \quad (i = 1, 2, \ldots, m) \qquad \text{(Supply constraints)}$$

$$\sum_{i=1}^{i=m} x_{ij} \ge d_j \quad (j = 1, 2, \ldots, n) \qquad \text{(Demand constraints)}$$

$$x_{ij} \ge 0 \quad (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$$

Although the transportation simplex appears to be very efficient, there is a certain class of transportation problems, called **assignment problems**, for which the transportation simplex is often very inefficient.

Example 1

MachineCo has four machines and four jobs to be completed. Each machine must be assigned to complete one job. The time required to set up each machine for completing each job is shown in the table.

Setup Times for Machineco

	Time (Hours)			
Machine	Job 1	Job 2	Job 3	Job 4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

MachineCo wants to minimize the total setup time needed to complete the four jobs. Use linear programming to solve this problem.

Example 1 - solution

Machineco must determine which machine should be assigned to each job. We define (for $i, j = 1, 2, 3, 4$)

$x_{ij} = 1$ if machine i is assigned to meet the demands of job j

$x_{ij} = 0$ if machine i is not assigned to meet the demands of job j

Then Machineco's problem may be formulated as

$$\begin{aligned}
\min z = {} & 14x_{11} + 5x_{12} + 8x_{13} + 7x_{14} + 2x_{21} + 12x_{22} + 6x_{23} + 5x_{24} \\
& + 7x_{31} + 8x_{32} + 3x_{33} + 9x_{34} + 2x_{41} + 4x_{42} + 6x_{43} + 10x_{44} \\
\text{s.t.} \quad & x_{11} + x_{12} + x_{13} + x_{14} = 1 \qquad \text{(Machine constraints)} \\
& x_{21} + x_{22} + x_{23} + x_{24} = 1 \\
& x_{31} + x_{32} + x_{33} + x_{34} = 1 \\
& x_{41} + x_{42} + x_{43} + x_{44} = 1 \\
& x_{11} + x_{21} + x_{31} + x_{41} = 1 \qquad \text{(Job constraints)} \\
& x_{12} + x_{22} + x_{32} + x_{42} = 1 \\
& x_{13} + x_{23} + x_{33} + x_{43} = 1 \\
& x_{14} + x_{24} + x_{34} + x_{44} = 1 \\
& x_{ij} = 0 \quad \text{or} \quad x_{ij} = 1
\end{aligned}$$

Example 1 - solution

The first four constraints ensure that each machine is assigned to a job, and the last four ensure that each job is completed. If $x_{ij} = 1$, then the objective function will pick up the time required to set up machine i for job j; if $x_{ij} = 0$, then the objective function will not pick up the time required.

Ignoring for the moment the $x_{ij} = 0$ or $x_{ij} = 1$ restrictions, we see that MachineCo faces a **balanced transportation problem** in which each supply point has a supply of 1 and each demand point has a demand of 1.

In general, an assignment problem is a balanced transportation problem in which all supplies and demands are equal to 1.

A transportation problem allows only shipments that go directly from a supply point to a demand point.

In many situations, shipments are allowed between supply points or between demand points. Sometimes there may also be points (called transshipment points) through which goods can be transshipped on their journey from a supply point to a demand point. Shipping problems with any or all of these characteristics are **transshipment problems**.

Fortunately, the optimal solution to a transshipment problem can be found by solving a transportation problem.

- We define a **supply point** to be a point that can send goods to another point but cannot receive goods from any other point
- Similarly, a **demand point** is a point that can receive goods from other points but cannot send goods to any other point.
- A **transshipment point** is a point that can both receive goods from other points and send goods to other points.

Example 1

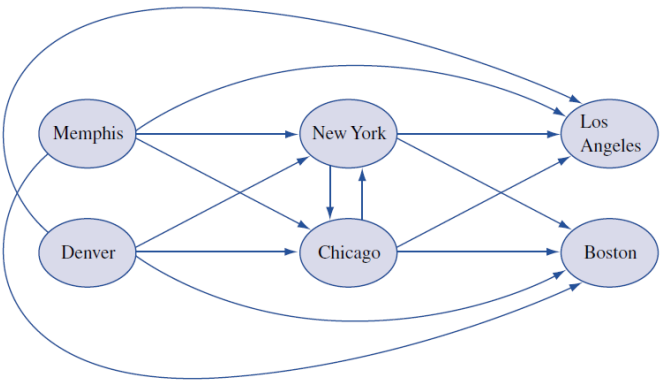
WidgetCo manufactures widgets at two factories, one in Memphis and one in Denver. The Memphis factory can produce as many as 150 widgets per day, and the Denver factory can produce as many as 200 widgets per day. Widgets are shipped by air to customers in Los Angeles and Boston. The customers in each city require 130 widgets per day. Because of the deregulation of airfares, WidgetCo believes that it may be cheaper to first fly some widgets to New York or Chicago and then fly them to their final destinations. The costs of flying a widget are shown in Table. WidgetCo wants to minimize the total cost of shipping the required widgets to its customers.

Shipping Costs for Transshipments

	To ($)					
From	Memphis	Denver	N.Y.	Chicago	L.A.	Boston
Memphis	0	—	8	13	25	28
Denver	—	0	15	12	26	25
N.Y.	—	—	0	6	16	17
Chicago	—	—	6	0	14	16
L.A.	—	—	—	—	0	—
Boston	—	—	—	—	—	0

Example 1 - solution

- Memphis and Denver are supply points, with supplies of 150 and 200 widgets per day, respectively
- New York and Chicago are transshipment points
- Los Angeles and Boston are demand points, each with a demand of 130 widgets per day



Example 1 - solution

- The optimal solution to a transshipment problem can be found by solving a transportation problem
- Given a transshipment problem, we create a balanced transportation problem by the following procedure (assume that total supply exceeds total demand):
  1. If necessary, add a dummy demand point to balance the problem
  2. Construct a transportation tableau: Because $s = (\text{total supply}) = 150 + 200 = 350$ and $d = (\text{total demand}) = 130 + 130 = 260$, the dummy demand point has a demand of $350 - 260 = 90$. The other supplies and demands in the transportation tableau are obtained by adding $s = 350$ to each transshipment point's supply and demand.

Exercise 1

OilCo has oil fields in San Diego and Los Angeles. The San Diego field can produce 500,000 barrels per day, and the Los Angeles field can produce 400,000 barrels per day. Oil is sent from the fields to a refinery, either in Dallas or in Houston (assume that each refinery has unlimited capacity). It costs \$700 to refine 100,000 barrels of oil at Dallas and \$900 at Houston. Refined oil is shipped to customers in Chicago and New York. Chicago customers require 400,000 barrels per day of refined oil; New York customers require 300,000. The costs of shipping 100,000 barrels of oil (refined or unrefined) between cities are given in the table. Formulate a balanced transportation model of this situation.

	To ($)			
From	Dallas	Houston	N.Y.	Chicago
L.A.	300	110	—	—
San Diego	420	100	—	—
Dallas	—	—	450	550
Houston	—	—	470	530

Exercise 2

Three professors must be assigned to teach six sections of finance. Each professor must teach two sections of finance, and each has ranked the six time periods during which finance is taught, as shown in the table. A ranking of 10 means that the professor wants to teach that time, and a ranking of 1 means that he or she does not want to teach at that time. Determine an assignment of professors to sections that will maximize the total satisfaction of the professors.

Professor	9 A.M.	10 A.M.	11 A.M.	1 P.M.	2 P.M.	3 P.M.
1	8	7	6	5	7	6
2	9	9	8	8	4	4
3	7	6	9	6	9	9

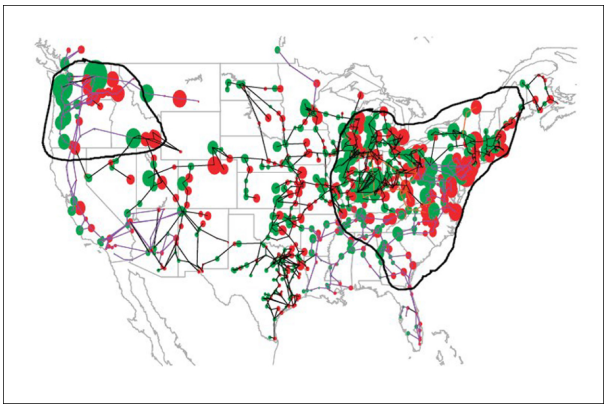
Networks are everywhere

ábra. Facebook social network

Networks are everywhere

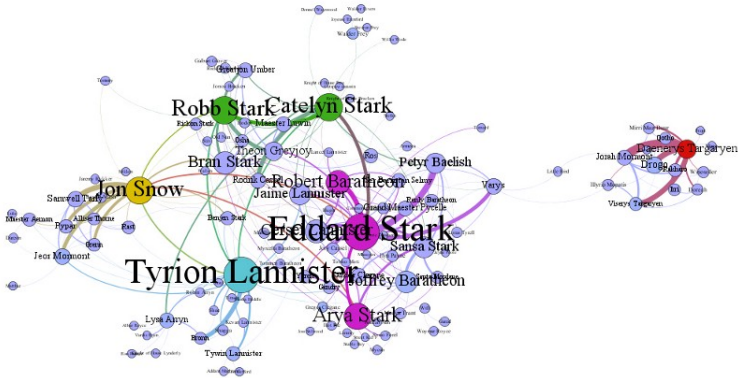
ábra. European airline network.

Networks are everywhere



ábra. Electric power supply network of the USA. On 14.08.2013 the electric supply of North-East stopped due to a heavy storm.

Networks are everywhere



ábra. interaction network of the actors of Game of Thrones. (1st and 2nd season). The thickness of the links is proportional to the number of interactions.

Why model networks?

For instance...

- They have central role in information/„infection" spreading
- in what we purchase, what language we speak, how we vote, what education we receive, how successful career we will have, etc...

Key to understand:

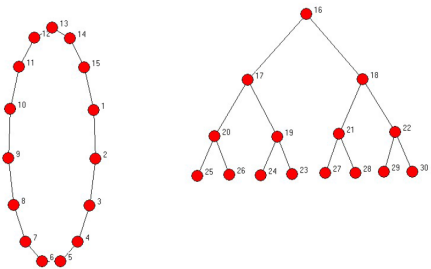
1. How network structure act on the behavior of the actors (agents, players) $\longrightarrow$ functioning
2. What patterns appears in social and economic networks?
3. In general: how can we use network information?

Why they are complex?

- Many interacting actors, act on each other
- Adaptivity: feedback, cooperation
- Evolution, changing in structure over time
- No „linearity": The whole is more than the „sum" of the parts

Diameter

- ℓ_{ij} – shortest path between i and j
- $\Delta = \max_{i,j} \ell_{ij}$ – diameter: the longest one among all-pair shortest paths



ábra. What is the diameter of a circle and a binary tree of n nodes?

Average path length

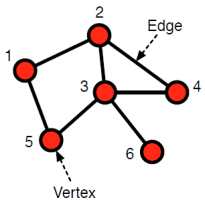
- The average of all-pair shortest path lengths

$$\langle \ell \rangle = \frac{1}{\binom{n}{2}} \sum_{i,j} \ell_{ij}$$

- Why is it interesting?
- What this value is for real networks (social, engineered, etc)?

Degree distribution

- $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ – G adjacency matrix:
- Degree of node i: $k_i = \sum_{j=1}^{n} a_{ij}$
- Degree distribution: $\mathbb{P}$(a randomly chosen node has degree k)



k	$\Pr(k)$
1	1/6
2	3/6
3	1/6
4	1/6

- Can real networks characterized by their degree distribution?
- How to determine degree distribution?

$\longrightarrow$ It is a key concept, we discuss it in bit more detail.

What are the „central" actors of a network?

- From structural point of view, a node is central if
  - it has high degree (many connections)
  - it is in the center (in some sense)
  - it has important role in dynamic processes on the network (e.g. information spreading)

$$\Longrightarrow \text{Centrality}$$

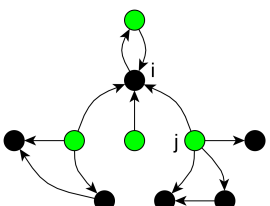
- The more central a node the more important it is.
- How to measure centrality?

Example: PageRank

- „Random surfer" model by Google [1]
- Recursion formula:

$$PR(i) = \frac{1-\lambda}{n} + \lambda \sum_{j \in N^+(i)} \frac{PR(j)}{k^{out}(j)},$$

where $\lambda \in [0,1]$ is a parameter (it is called „damping factor"), $N^+(i)$ the „in-neighborhood" of i



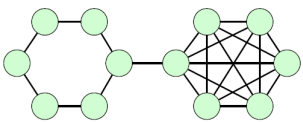
[1] Brin & Page, *Computer networks and ISDN systems* ,1998

Communities in networks

ábra. Facebook social network

Communities in networks

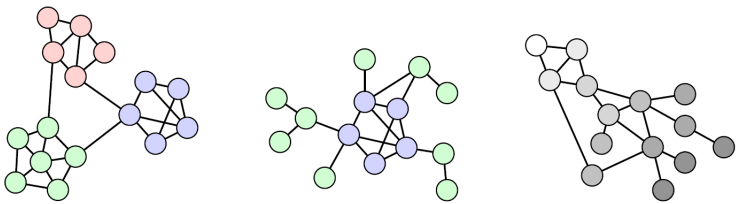
- Simple metrics: average degree, degree distribution, clustering coefficient (def!), average path length $\longrightarrow$ a lot of information **BUT**
- the large-scale heterogeneity remains undiscovered



ábra. Many triangles, but concentrated at a certain part of the network

- What are the large-scale patterns?
- **Small networks**: we can simply observe using an appropriate visualization technique.
- **Large networks** (a few millions of nodes and edges): **we need quantitative measures**

Large-scale organization patterns



ábra. Community structure, core-periphery structure, ordered (hierarchical) structure (source: Aaron Clauset, Network analysis and modelling course)

Network dynamics

- Network flow theory can be applied efficiently
- Several other dynamical processes have been developed
- Here just a brief overview is provieded

Some refernces:

1. Juraj Stacho's lecture notes on Operations Research
2. Aaron Clauset's lecture notes on networks
3. A-L Barabási's books on networks
4. Ahuja et al.: Network flows: theory, algorithms and applications ...