ORF 522

Linear Programming and Convex Analysis

Network Flows

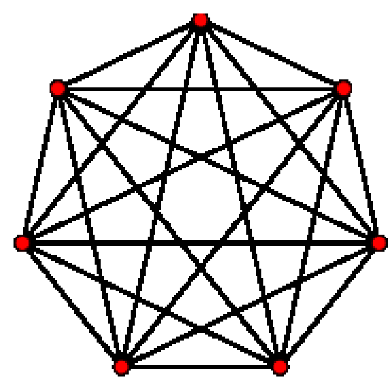
Marco Cuturi

Reminder

- **In previous lectures we have studied**
  - The ellipsoid method;
  - An interior point method: affine scaling;
  - Gave you slides about the potential reduction algorithm.
- Namely **different methods** to compute the optima of linear programs without using the fact that a solution is a BFS.
- Starting from **outside** or **inside** the polyhedron to converge iteratively to the solution.

Today : new family of linear problems, Network Flows

- **Network flows** are linear optimization problems with **particular constraints**.
- **Network flows** model interactions between linked locations , *i.e.* **graphs**
- **Optimization problem**: compute optimal **flows** between the points.
- **Practical problem**: when n nodes, up to $n(n-1)/2 \approx \frac{n^2}{2}$ edges.
- Example: K7, complete graph with 7 nodes and 21 edges.

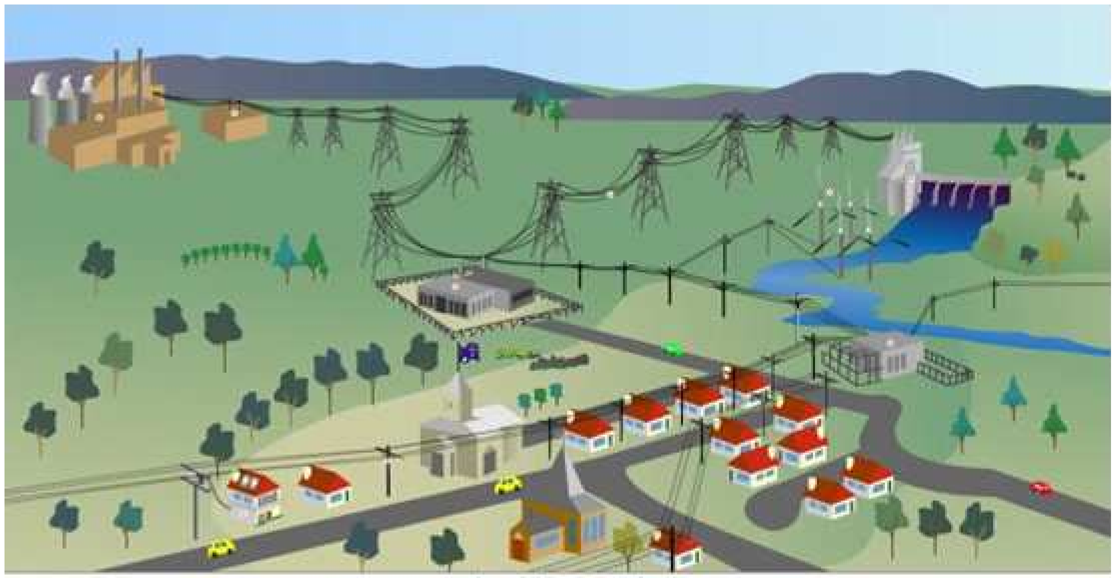


- If we hundreds of nodes $\Rightarrow$ very high dimensions...
- Fortunately, constraint matrix has special characteristics $\Rightarrow$ efficient algorithms.

Graph theory

Illustrations

- Let's start with a picture of the countryside

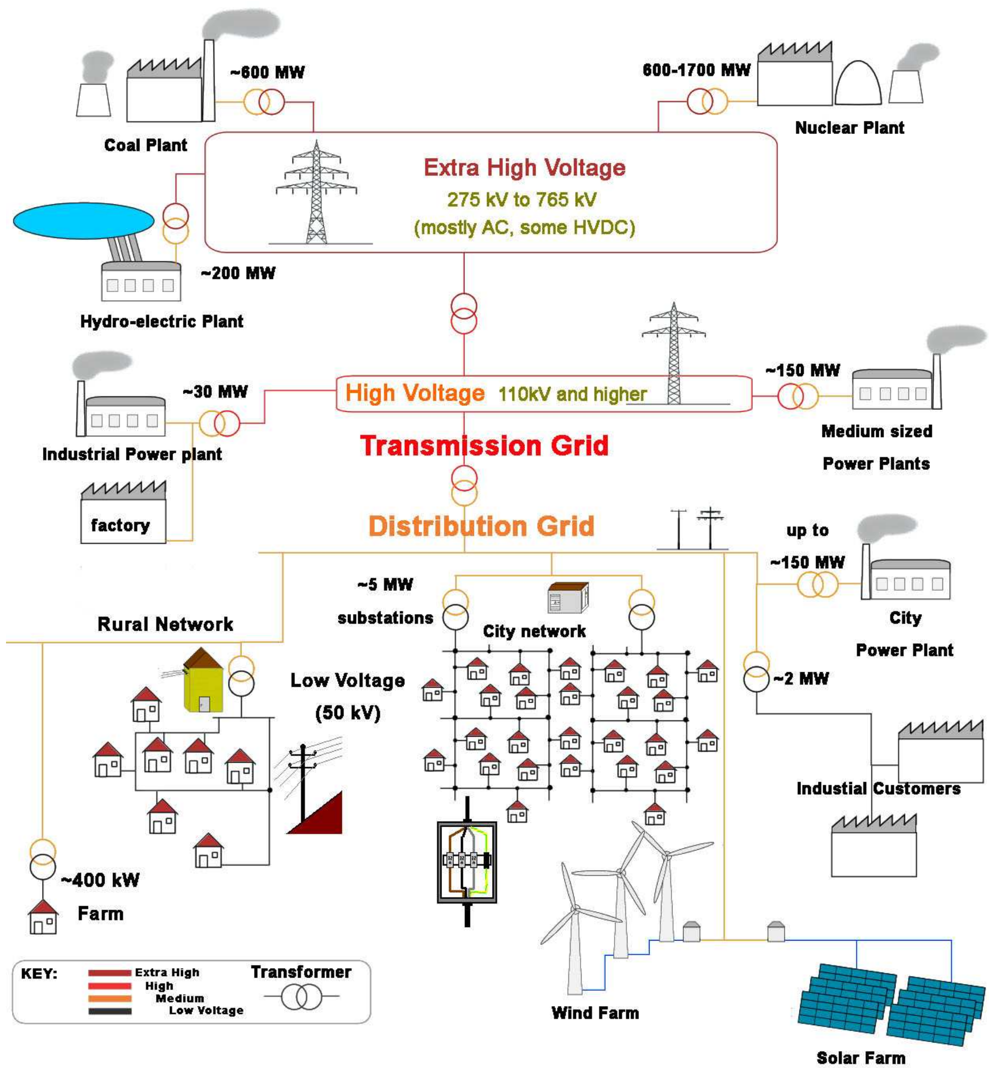


Electricity network

electricity network

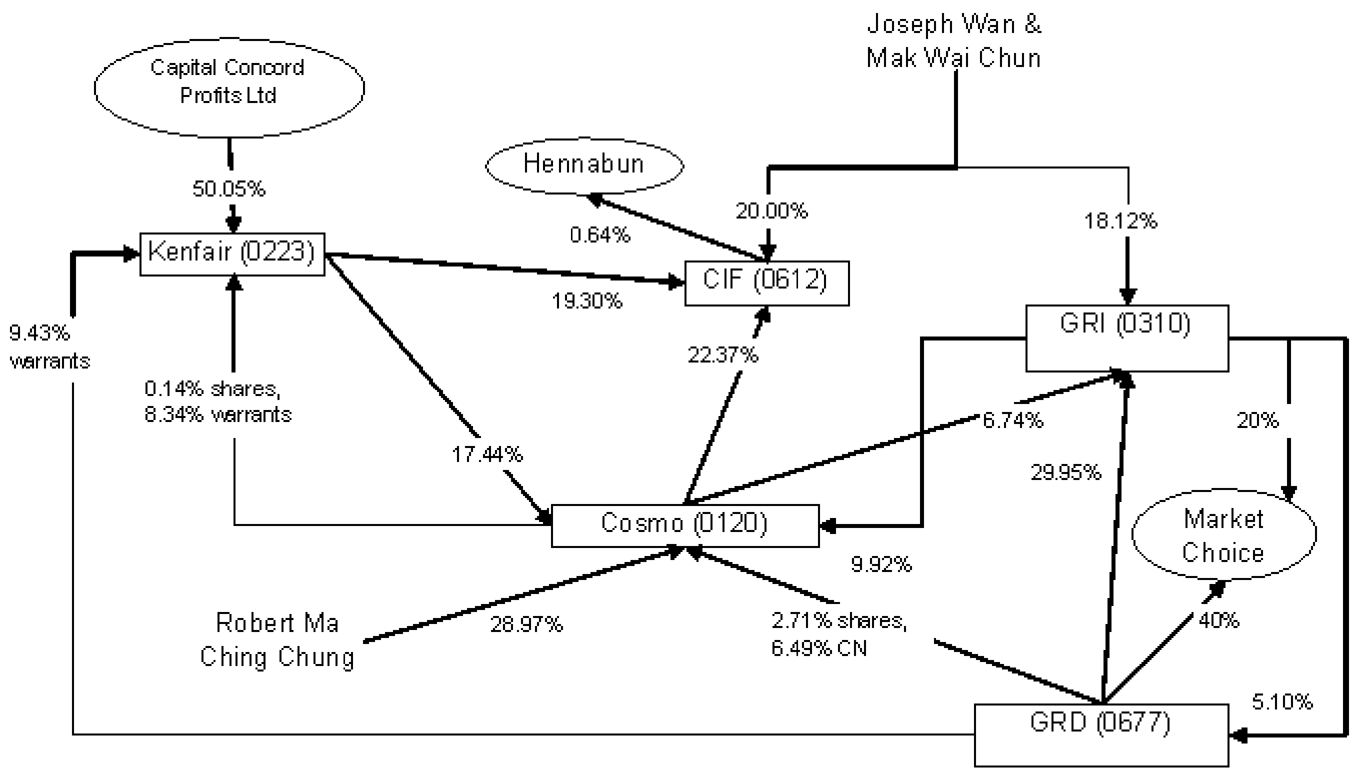
Illustrations

- More sophisticated



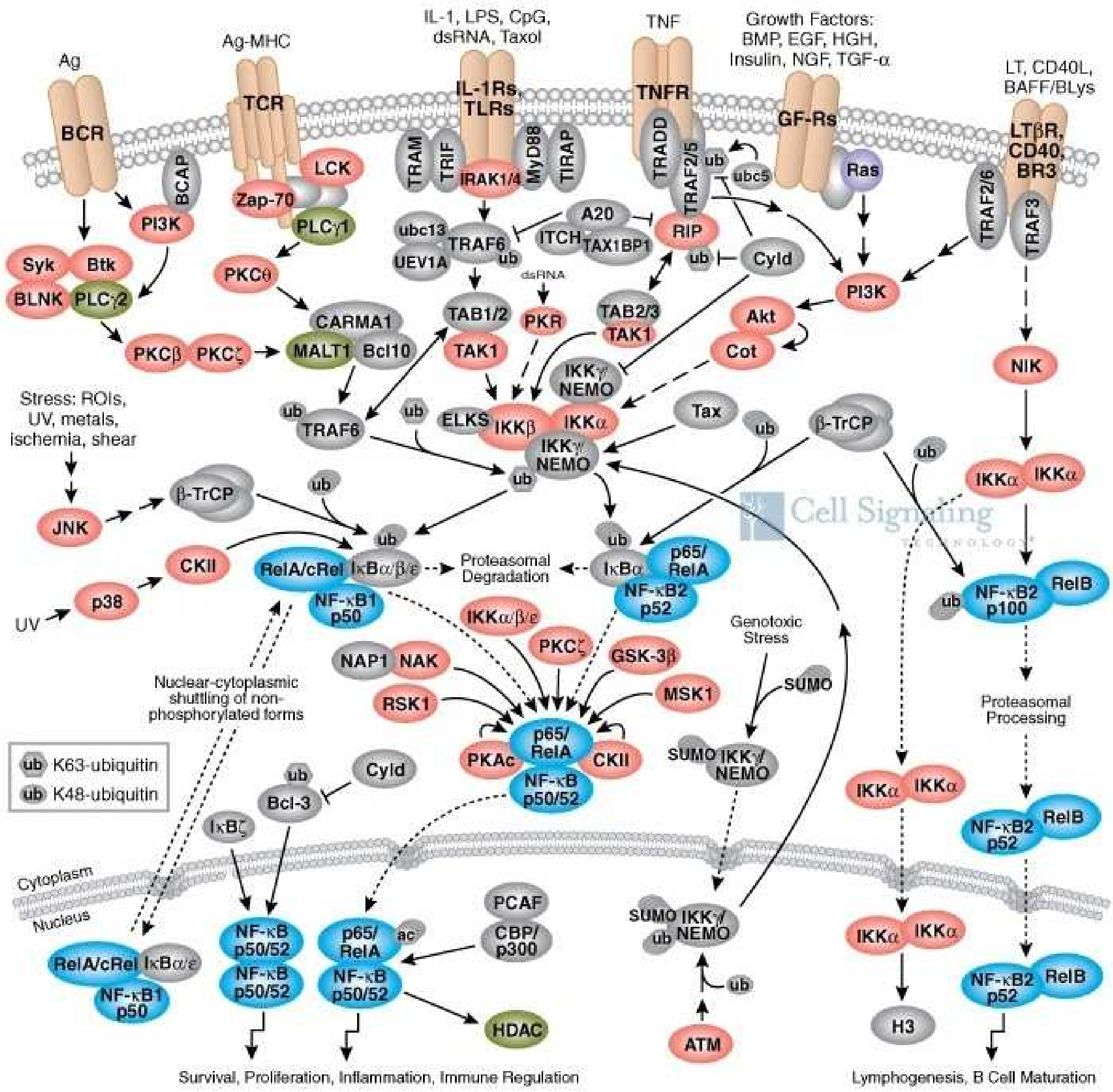
Illustrations

- Definition of **Cross-holding**: when listed corporations own securities issued by other listed corporations
- Not a good sign usually... favors manipulations and "poison-pill" schemes



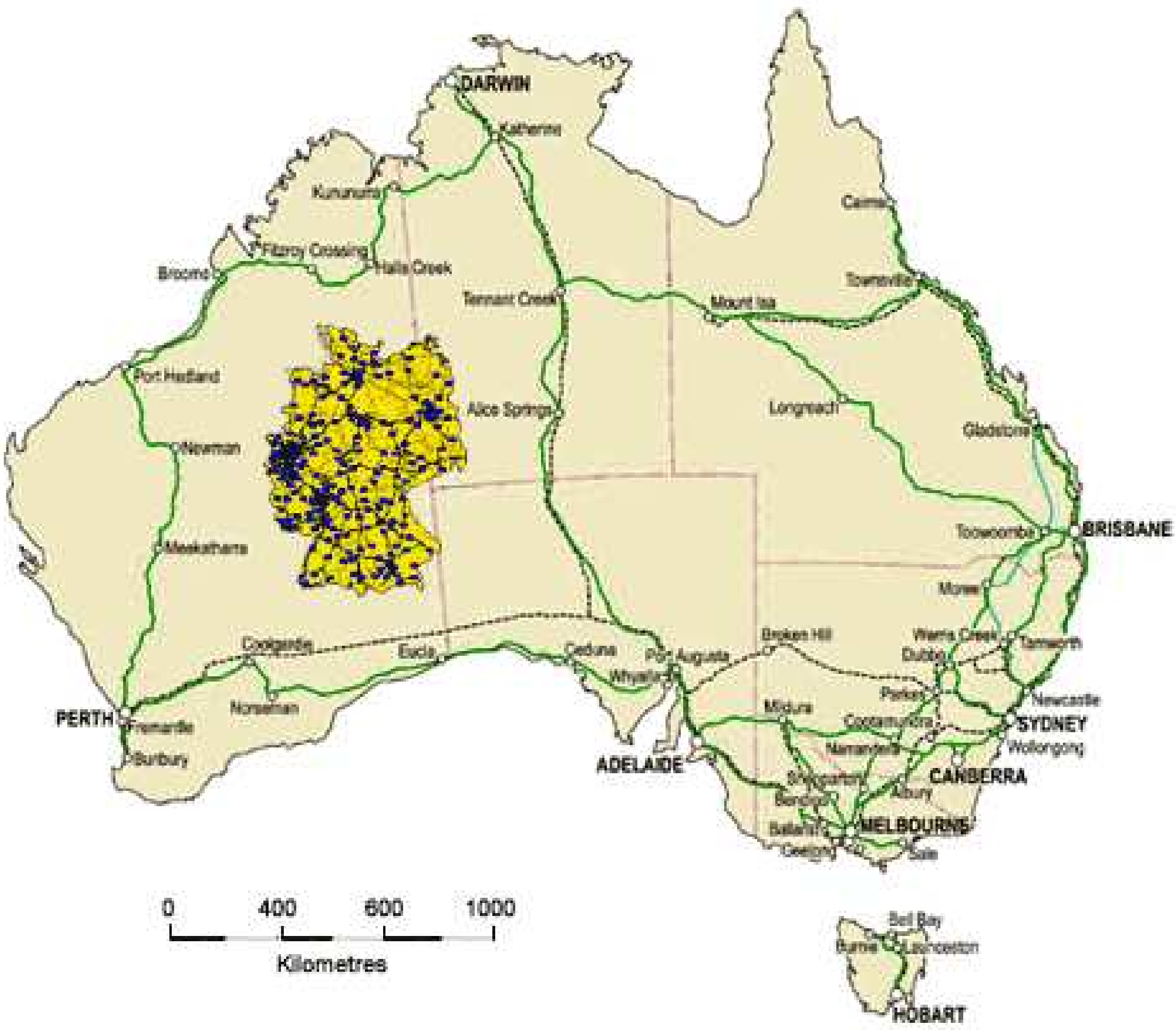
Illustrations

- Back to more noble causes: biological pathway



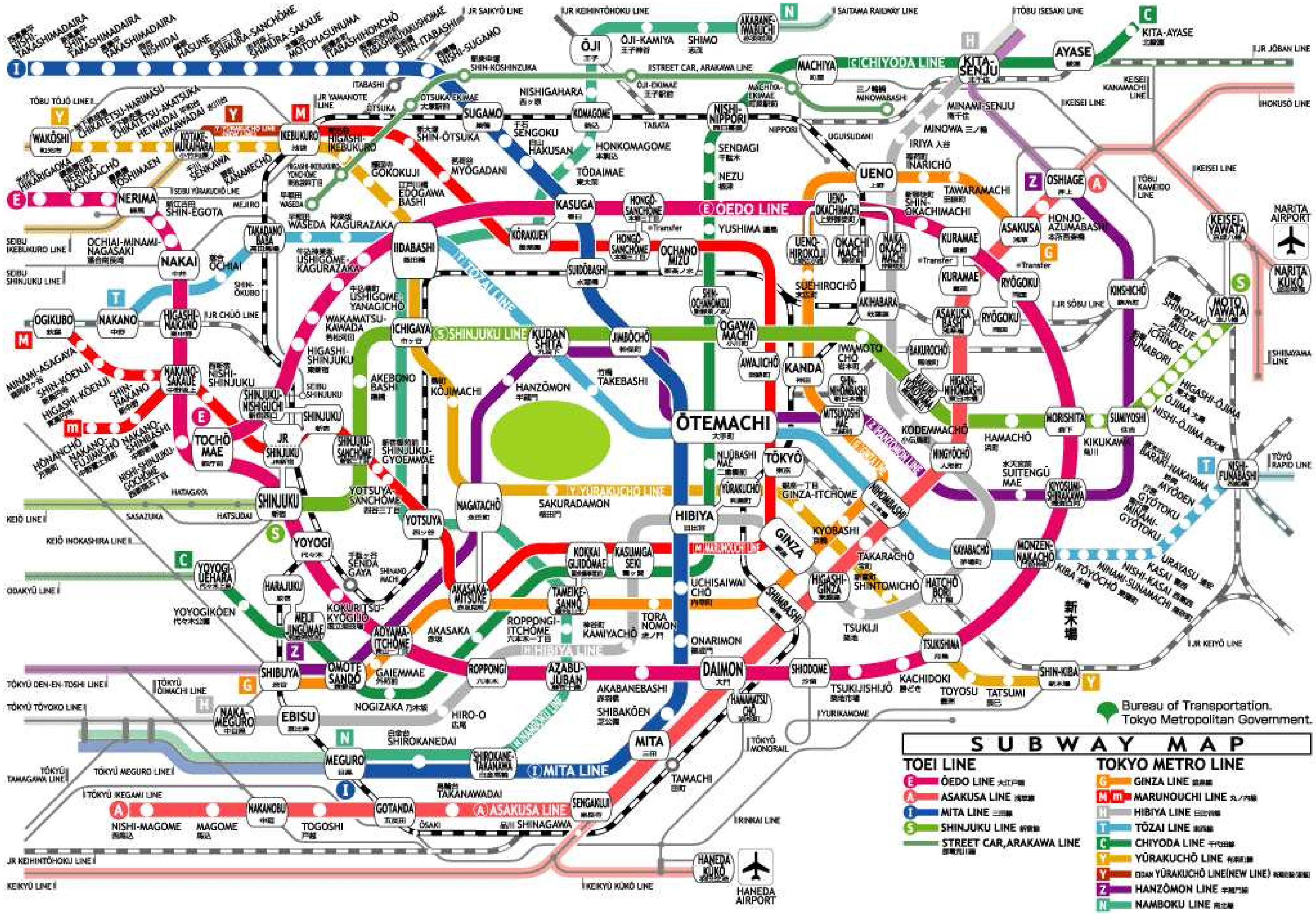
Illustrations

- An everyday graph: highways



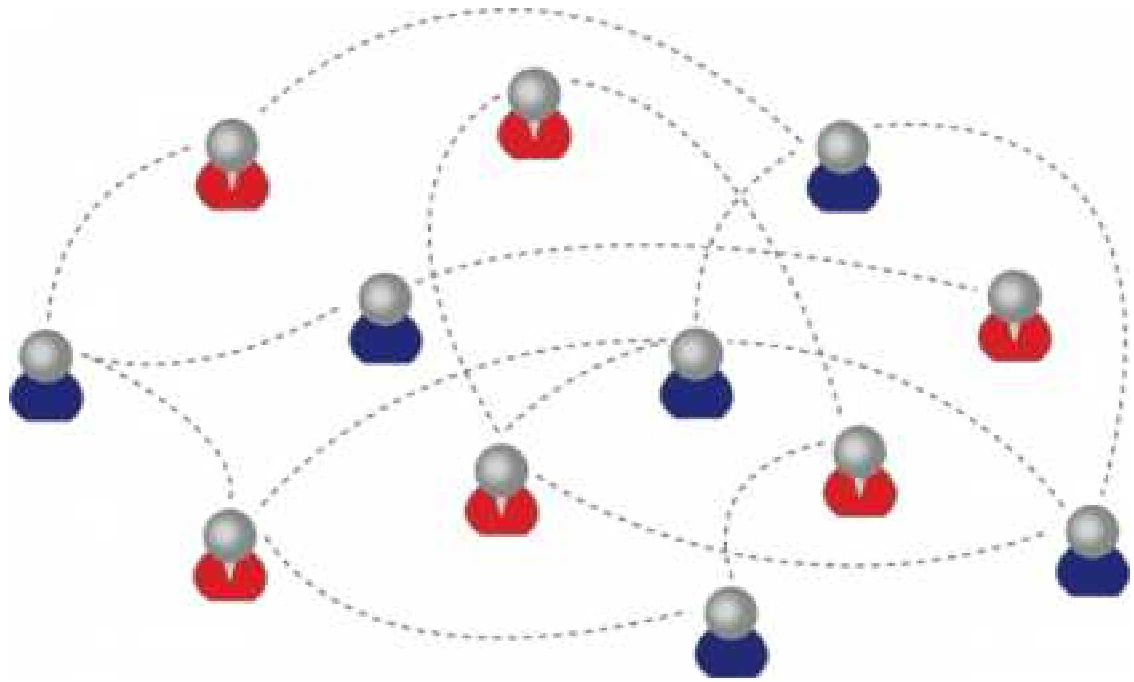
Illustrations

- Another one: trains.



Illustrations

- One that was *recently* fashionable to talk about, social networks.



Some intuitions for a model?

- Networks have different components of interest:
  - **Nodes**: cities, stations, houses/factories,... people.
  - **Connections**: highways, railways, electrical cables,... knowledge of someone.
  - **Flows**: cars, trains, electricity,... text/video/voice.

- Additionally: the connections can be:
  - unilateral (biological pathways).
  - bilateral (highways, railways).
  - undirected (electricity)

- Let's review a few **basic definitions**.
- Should be useful to you in many settings and not just network flows studies.
- Graph inference, graphical models (a.k.a bayesian networks), message passing algorithms, dynamic programming, probabilities/statistics etc...

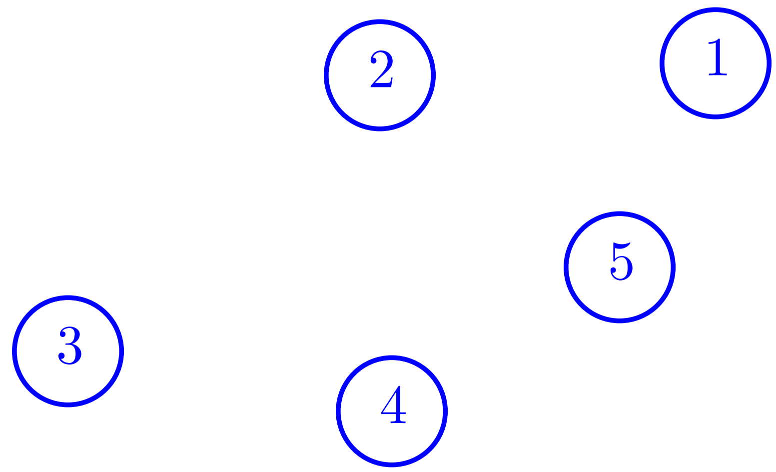
Reminders and Definitions

Building blocks

- Two key objects define a **graph**: $\mathcal{G} = (\mathcal{N}, \mathcal{E})$
  - set of **nodes** $\mathcal{N}$.
  - set of **edges** $\mathcal{E}$.
- if you add more information, then the graph becomes a **network**
  - set of **labels** $\mathcal{L}$ indexed by the edges.
  - Additional information about the nodes, costs etc..
- A graph is the topological description of a network.
- We will study **networks** later.

Nodes

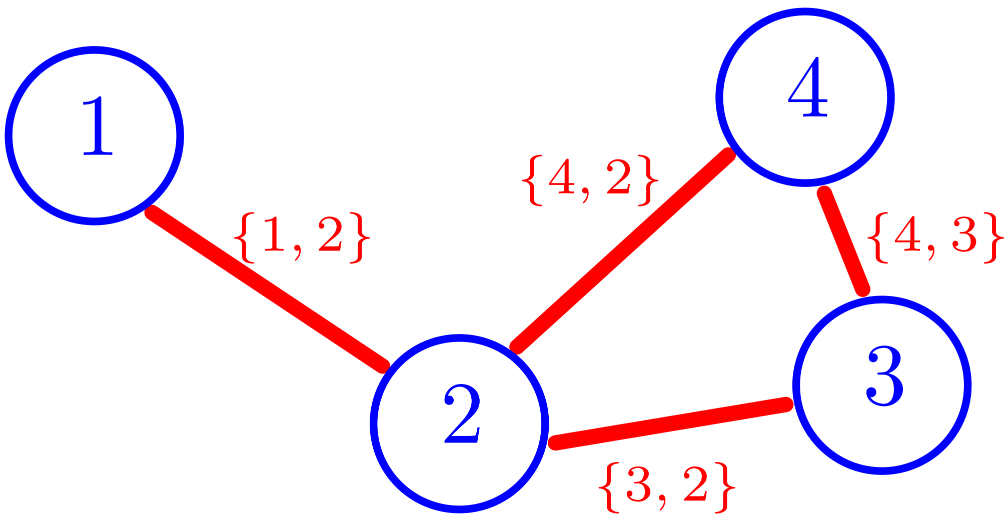
- $\mathcal{N}$ will be a finite set.
- We usually identify a node with its number $1 \le i \le N \stackrel{\text{def}}{=} \#\{\mathcal{N}\}$.
- Not much else to say...



Undirected Edges

The set $\mathcal{E}$ describes a connexion between two nodes $i, j \in \mathcal{N}$. **Two cases**:

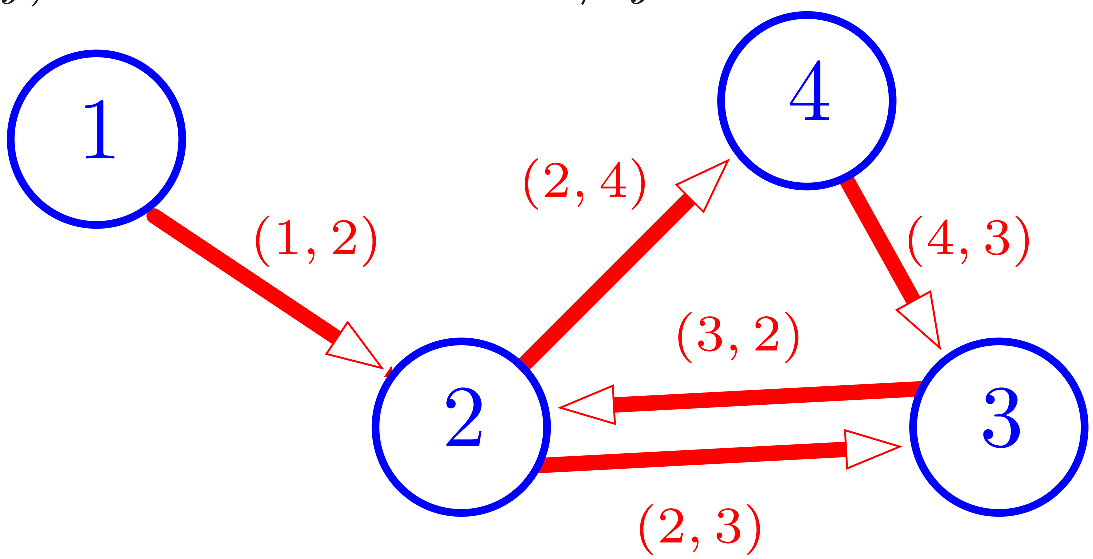
- **Undirected** graphs, nodes with edges or links): $\mathcal{E} \subset \mathcal{P}_2(\mathcal{N})$.
  - $\mathcal{E}$ is a set of subsets of $\mathcal{N}$ of cardinal 2.
  - If e is an edge, $e \in \mathcal{E} \Rightarrow \#\{e\} = 2$.
  - Any edge can be written $e = \{i, j\}, i \neq j$.



  - ▷ **Nodes** $\mathcal{N} = \{1, 2, 3, 4\}$
  - ▷ **Undirected Edges** $\mathcal{N} = \{\{1, 2\}, \{4, 2\}, \{4, 3\}, \{2, 3\}\}$

Directed Edges

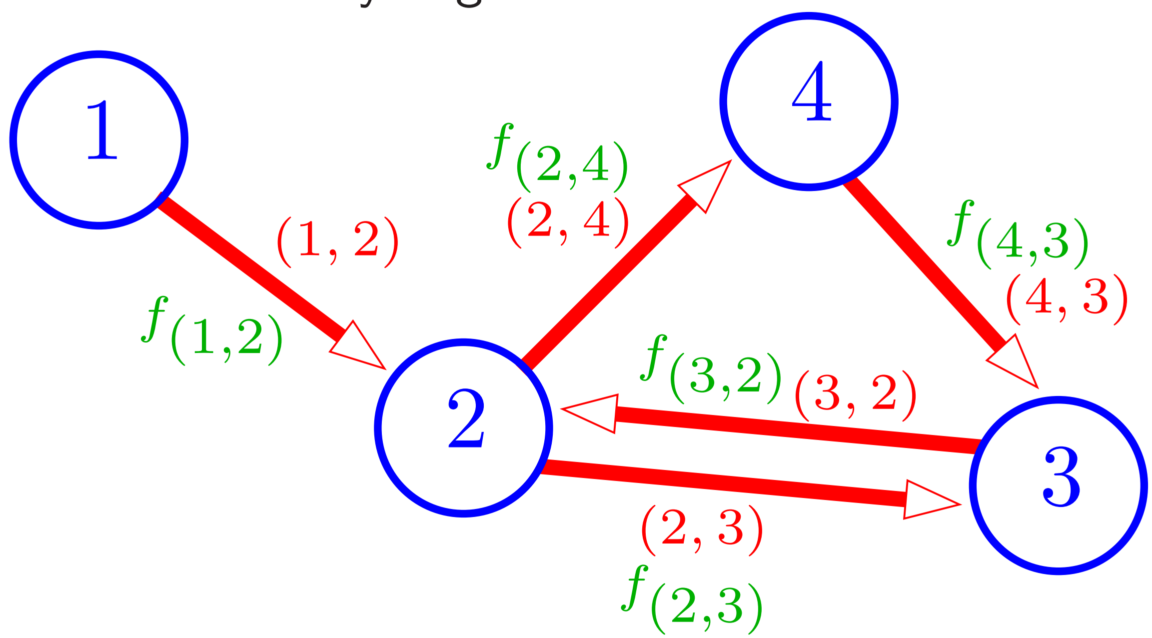
- **Directed** graphs, nodes with arrows, arcs: $\mathcal{E} \subset \mathcal{N} \times \mathcal{N} \setminus \Delta$.
  - $\Delta = \{(i,i), i \in \mathcal{N}\}$
  - An edge $e = (i,j)$ and we also assume $i \neq j$.



  - **Nodes** $\mathcal{N} = \{1, 2, 3, 4\}$
  - **Directed Edges** $\mathcal{E} = \{(1,2), (2,4), (4,3), (2,3), (3,2)\}$

Labels on Edges

- **Labels** $\mathcal{L}$ can be assigned to edges (to nodes as well, we do not consider this by now)
  - Label function $f : \mathcal{E} \mapsto \mathbb{R}$.
  - In practice, a vector labelled by edges in $\mathbb{R}^{\mathcal{E}}$



  - ▷ **Nodes** $\mathcal{N} = \{1, 2, 3, 4\}$
  - ▷ **Directed Edges** $\mathcal{E} = \{(1, 2), (2, 4), (4, 3), (2, 3), (3, 2)\}$
  - ▷ **Labelled Edges** $\mathcal{L} = \left\{f_{(1,2)}, f_{(2,4)}, f_{(4,3)}, f_{(2,3)}, f_{(3,2)}\right\}$

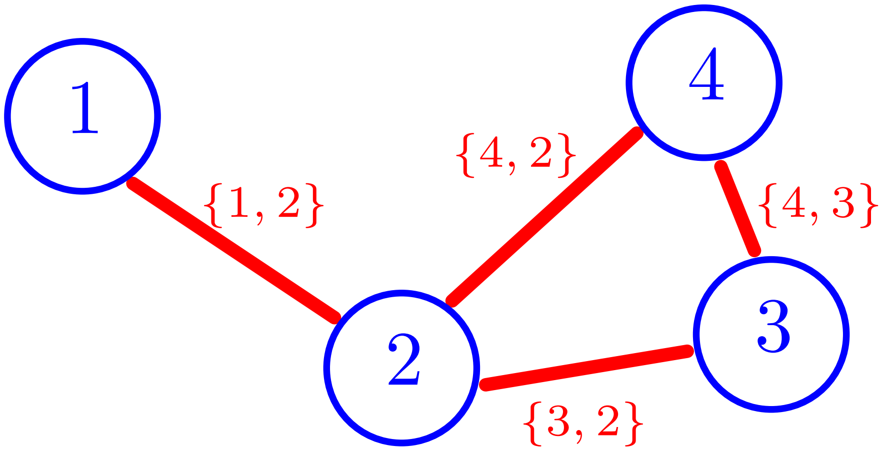
Degree, Walks, Paths, Cycles for *Undirected* Graphs

- The **degree** of a node is the number of edges incident to that node.
  - for $i \in \mathcal{N}$, $d(i) = \#\{e \in \mathcal{E} | i \in e\}$

- Given the graph structure, here are some important **sequences of nodes**:
  - A **walk** from node i_1 to node i_t is a finite sequence of nodes $i_1, i_2 \cdots, i_t$ such that $\{i_k, i_{k+1}\} \in \mathcal{E}$ for $1 \leq k \leq t-1$.
  - A **path** is a walk with no repetitions, $i.e.$with **pairwise distinct nodes**.
  - A **cycle** $i_1, \cdots, i_t$ is a walk such that $t \geq 3$, $i_1 = i_t$ and $(i_1, \cdots, i_{t-1})$ is a path.

- An undirected graph is **connected** if $\forall i, j \in \mathcal{N}$, there exists a path from i to j.

Degree, Walks, Paths, Cycles for Undirected Graphs



- Walk : $(1, 2, 3, 4, 2, 3, 4, 2, 3)$
- Path : $(3, 4, 2)$
- Cycle : $(2, 3, 4, 2)$

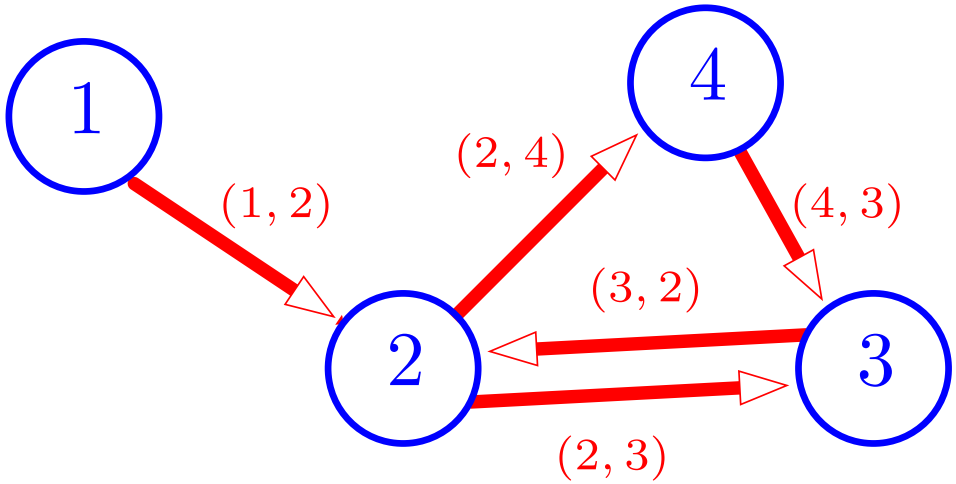
the graph is **connected**.

Directed Graphs

- To remove ambiguity, from now on, when considering **directed edges**,
  - we use the word **arc** for directed edges.
  - and write $\mathcal{A}$ instead of $\mathcal{E}$.
- For any arc $a = (i, j)$ in $\mathcal{A}$, i is its **start** node and j its **end** node.
- Given a node i, define the sets of nodes $I(i)$ and $O(i)$ of nodes which have resp.
  - an incoming arc towards i,
  - an outgoing arc from i.

$$I(i) = \{j \in \mathcal{N}, (j, i) \in \mathcal{A}\}$$
$$O(i) = \{j \in \mathcal{N}, (i, j) \in \mathcal{A}\}$$

Ingoing and Outgoing sets of a *Directed* Graph



- $I(4) = \{2\}, O(4) = \{3\}$
- $I(2) = \{1, 3\}, O(2) = \{4, 3\}$
- $I(1) = \emptyset, O(1) = \{2\}$
- $I(3) = \{4, 2\}, O(3) = \{2\}$

An undirected graph corresponding to a *directed* Graphs

- Build an undirected graph from directed:
  - Consider each arc (i, j) of $\mathcal{A}$ and add $\{i, j\}$ to a set of edges $\mathcal{E}$.
  - remove duplicates.

- Our directed graph example can be reduced to the undirected example.

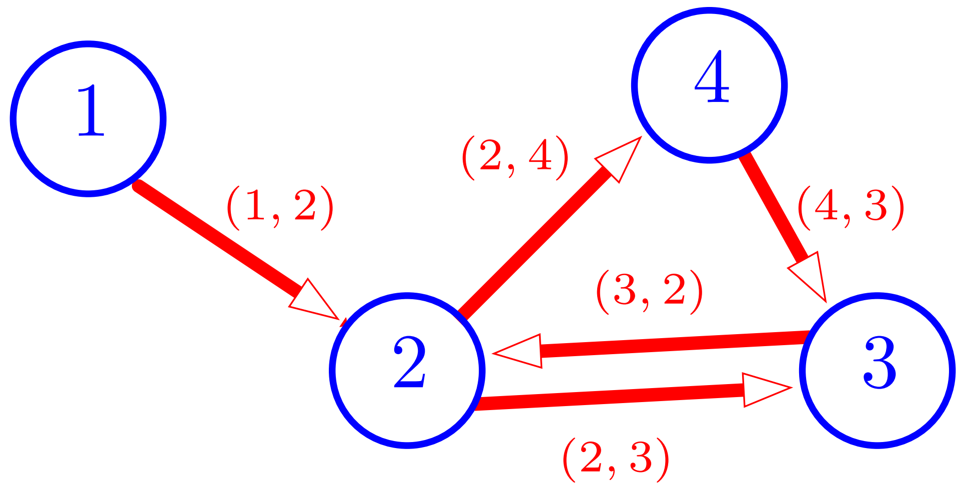
- A directed graph is **connected** if the corresponding undirected graph is.

Walks, Paths, Cycles for *Directed* Graphs

- Walks, paths, cycles: *similar* definitions than undirected case.
- Some ambiguity to take care of.
- A **walk** from node i_1 to node i_t is a finite sequence of nodes $i_1, i_2 \cdots, i_t$ paired with a sequence $a_1, \cdots, a_{t-1}$ of arcs of $\mathcal{A}$ such that a_k equals either (i_k, i_{k+1}) **or** (i_k, i_{k+1}).

- In a walk, for successive nodes i_k, i_{k+1} there are two possibilities for $a_k \in \mathcal{A}$,
  - if $a_k = (i_k, i_{k+1})$ then it is called a **forward** arc.
  - if $a_k = (i_{k+1}, i_k)$ then it is called a **backward** arc.
  - Sometimes both $(i_k, i_{k+1}), (i_{k+1}, i_k) \in \mathcal{A}$. need to choose.
- A walk is **directed** if it only has **forward arcs**.

Walks, Paths, Cycles for *Directed* Graphs



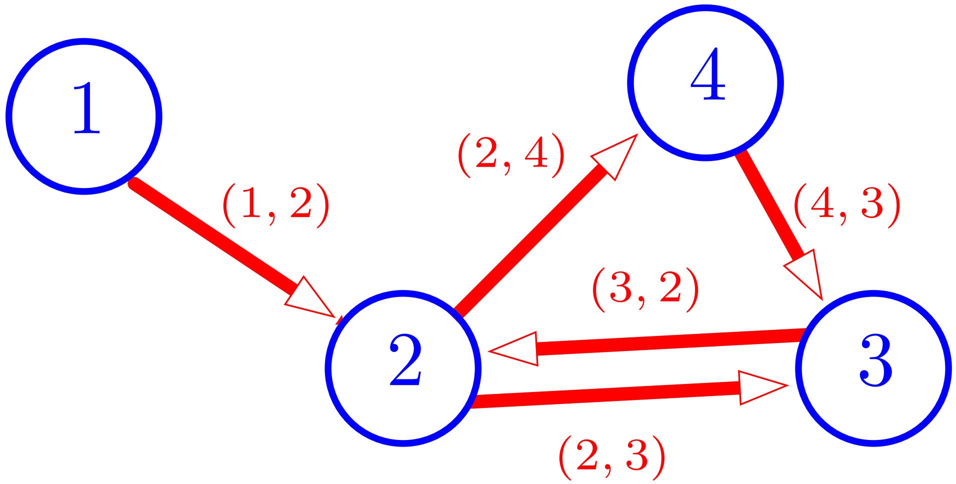
- walk: $1, (1, 2), 2, (3, 2), 3, (3, 2), 2, (4, 2), 4$
- **directed** walk $1, (1, 2), 2, (2, 3), 3, (3, 2), 2$

Degrees, Walks, Paths, Cycles for *Directed* Graphs

- A **path** is a walk with **distinct nodes**.
- A **cycle** $i_1, \cdots, i_t$ is a walk such that
  - $t \geq \mathbf{2}$,
  - $i_1 = i_t$ and $(i_1, \cdots, i_{t-1})$ is a path.
- Like walks, a path and a cycle are **directed** if they only have **forward arcs**.
- **Remark**: only need to keep track of nodes for a directed walk/path/cycle:

$$(i_1, (i_1, i_2), i_2, (i_2, i_3), \cdots, (i_{t-1}, i_t), i_t) \Leftrightarrow \text{ directed walk } (i_1, i_2, \cdots, i_t)$$

Degree, Walks, Paths, Cycles for Undirected Graphs

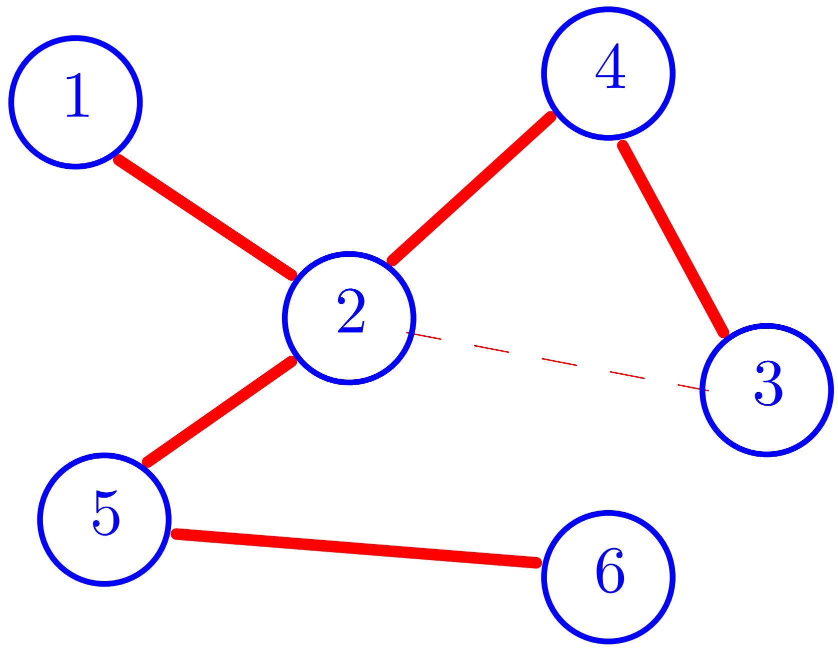


- Path: $1, (1,2), 2, (3,2), 3, (4,3), 4$
- **directed** Path : $1, (1,2), 2, (2,3), 3$
- Cycle : $3, (4,3), 4, (2,4), 2, (2,3), 3$
- **directed** Cycle: $3, (3,2), 2, (2,4), 4, (4,3), 3$

Trees and Spanning Trees

Trees

- An undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is called a tree if
  - it is **connected**.
  - it has **no cycles**.
- if a node in the tree has a degree equal to 1, it is called a **leaf**.



- Adding $\{2, 3\}$ would create a cycle with $(2, 3, 4)$.
- leaves: $\{1, 3, 6\}$.

Trees

Theorem 1. *Fundamental properties:*

(i) *Every tree with more than one node has* ***at least one leaf****.*

(ii) *An undirected graph is a tree iff it is connected and has* $\#(\mathcal{N}) - 1$ ***edges****.*

(iii) *For any* $i \neq j$ *two nodes in a tree there exists a* ***unique path*** *from* i *to* j*.*

(iv) *If you* ***add an edge to a tree****, the resulting graph contains* ***exactly one cycle*** *(up to shifting the order of the cycle)*

Fundamental properties: Proofs

(i) If all $\mathcal{N}$ nodes had a degree 2 or higher, then one can have paths of arbitrarily long size, hence create a cycle. So there must be at least one leaf.

(ii)
- $\Rightarrow$: prove recursively.
  - True if $\#(\mathcal{N}) = 1$. Suppose true for k nodes. Consider tree $\mathcal{T}$ with $k+1$ nodes.
  - There is one leaf in $\mathcal{T}$. Remove the edge that joins it to $\mathcal{T}$.
  - Resulting tree $\mathcal{T}'$ has $\#(\mathcal{N}) - 1$ nodes hence $\#(\mathcal{N}) - 2$ edges.
  - Hence $\mathcal{T}$ has $\#(\mathcal{N}) - 1$ edges.
- $\Leftarrow$: If not a tree, there is a cycle.
  - Notice that all nodes of a cycle have degree ≥ 2.
  - It is thus possible to remove an edge will keeping connectivity.
  - Repeat this until there is no cycle.
  - We get a tree out of the process, with $\#(\mathcal{N}) - 1$ edges thanks to (i).
- Since we have not added edges but only removed, the original graph was a tree.

Fundamental properties: Proofs

(iii) Tree is connected hence such path $p = (i_0 = i, i_1, \cdots, i_{m-1}, i_m = j)$ exists. Need to prove **unicity**.

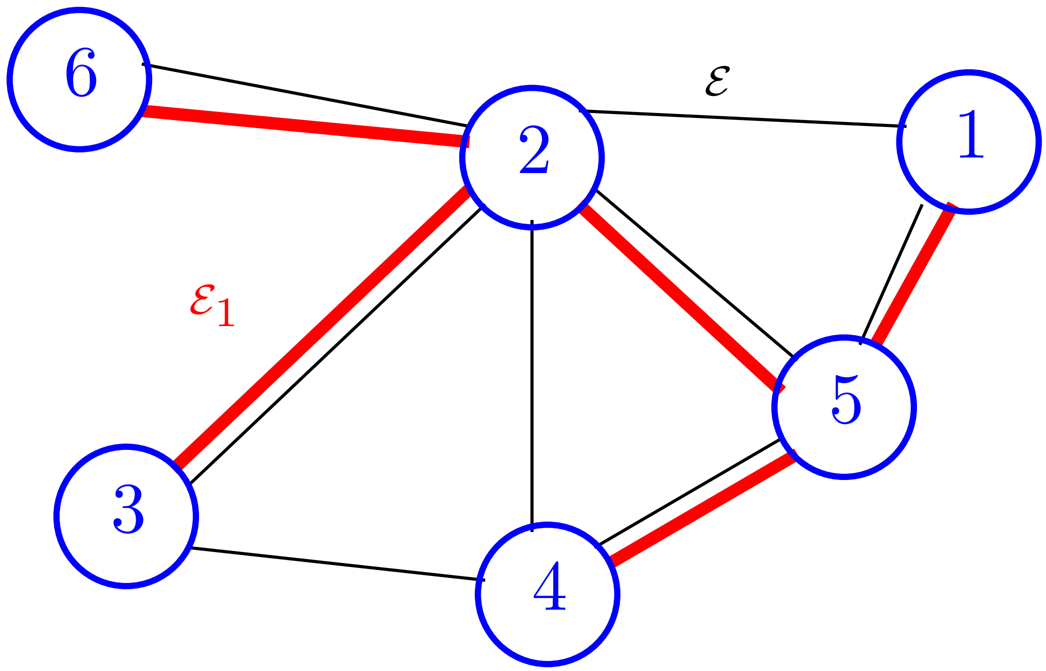
- Suppose $\exists p' = (i'_0 = i, i'_1, \cdots, i'_{m'-1}, i'_{m'} = j)$ another path. Write $n = \min(m, m')$
- Define $k = \min\{e \leq n |\ i_e \neq i'_e\}$ and $M = \max\{e |\ i_{m-e} \neq i_{m'-e}\}$.
- $0 \leq k \leq n - M \leq n$ are well defined, otherwise $p = p'$.
- Can show $i_{k-1} i_k \cdots i_{m-M} i_{m-M+1} i'_{m'-M} i'_{m'-M-1} \cdots i'_k i_{k-1}$ is a cycle.

(iv) Let $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ be a tree and add one edge $\{i, j\}$.

- With one edge more it cannot be a tree (i) and hence there is a cycle.
- The cycle necessarily includes the new edge $\{i, j\}$ and nodes i and j.
- The cycle links i and j through a path which is unique by (iii).
- The cycle is thus unique up to shifting the nodes order.

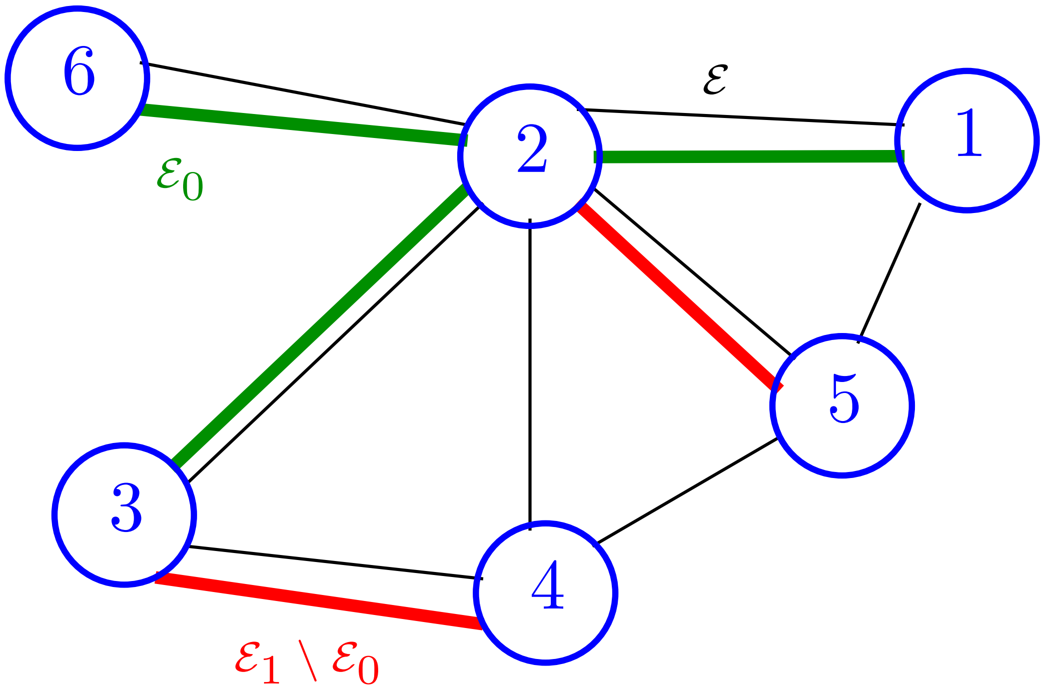
Spanning Trees

- Given a connected undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, let $\mathcal{E}_1$ be a subset of $\mathcal{E}$ such that $T = (\mathcal{N}, \mathcal{E}_1)$ is a tree.
- Such a tree $\mathcal{T}$ is called a spanning tree of $\mathcal{G}$.



Spanning Trees

Theorem 2. *Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a connected undirected graph and $\mathcal{E}_0$ a subset of $\mathcal{E}$. Suppose that the edges of $\mathcal{E}_0$ do not form cycles. Then $\mathcal{E}_0$ can be augmented to a set $\mathcal{E}_1$ such that $\mathcal{E}_0 \subset \mathcal{E}_1$ and $\mathcal{T} = (\mathcal{N}, \mathcal{E}_1)$ is a spanning tree.*



Spanning Trees : Proof

- **Proof**: Suppose $\mathcal{E}_0 \subset \mathcal{E}$ and that the edges of $\mathcal{E}_0$ do not form cycles.
- If $\mathcal{G}$ is a tree done, just set $\mathcal{E}_1 = \mathcal{E}$
- If not it contains one cycle. Start with $\mathcal{E}_1 \leftarrow \mathcal{E}$.
- Repeat the following until $\mathcal{E}_1$ has no cycle:
  - Consider that cycle $c = i_1 \cdots i_m$ and $i_m = i_1$.
  - $\exists e \in \mathcal{E}_1 \setminus \mathcal{E}_0$ such that $e = \{i_k i_{k+1}\}$.
  - Remove that edge from $\mathcal{E}_1 \leftarrow \mathcal{E}_1 \setminus \{e\}$.
- $\mathcal{T}(\mathcal{N}, \mathcal{E}_1)$ is now a tree and $\mathcal{E}_0 \subset \mathcal{E}_1$.

Network Flows

Mathematical Formulation

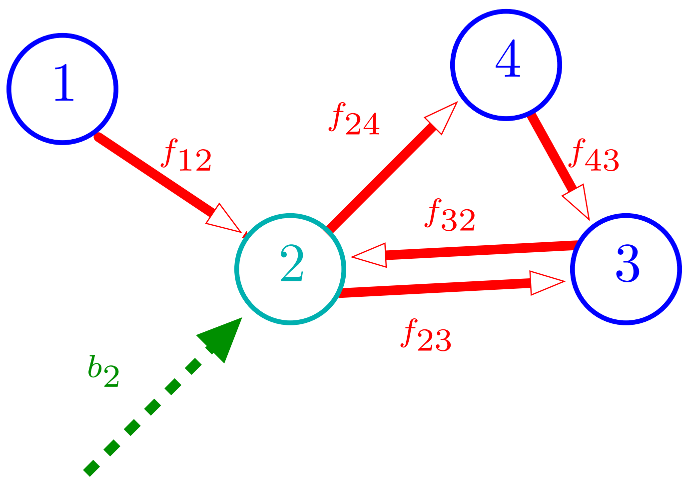
A **network** is a **directed graph** $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with side information, typically $\mathcal{L}$ and the following quantities:

- for a in $\mathcal{A}$, or equivalently $(i, j) \in \mathcal{A}$, a nonnegative f_a or $f_{(i,j)}$ and usually written f_{ij} quantifies a **flow** between nodes i and j.
- For each node $i \in \mathcal{N}$ b_i is a **supply** to that node from the exterior.
    - if $b_i > 0$ node i is usually called a **source**.
    - if $b_i < 0$ node i is usually called a **sink**.
- Each flow can be **capacitated** that is restricted to be less than $u_{(i,j)}$.
- When $u_{(i,j)} = \infty$ the flow is **uncapacitated**.
- Each arc might have a **cost** per unit of flow associated, c_{ij}.

Flow Equations *constraints*

Natural flow equations imply that

$$b_i + \sum_{j \in I(i)} f_{ji} = \sum_{j \in O(i)} f_{ij}$$
$$0 \leq f_{ij} \leq u_{ij} \tag{1}$$



in this case,

$$\boldsymbol{b_2} + f_{1\mathbf{2}} + f_{1\mathbf{2}} = f_{\mathbf{2}4} + f_{\mathbf{2}3}$$
$$0 \leq f_{12}, f_{24}, f_{32}, f_{23} \leq \ldots$$

Flow Equations *constraints*

- More terminology: any vector f with indexed by $\mathcal{E}$ is a flow.
- A flow is **feasible** if it satisfies the **linear** equations (**??**)
- Note that we also have

$$\sum_{i \in \mathcal{N}} b_i = 0$$

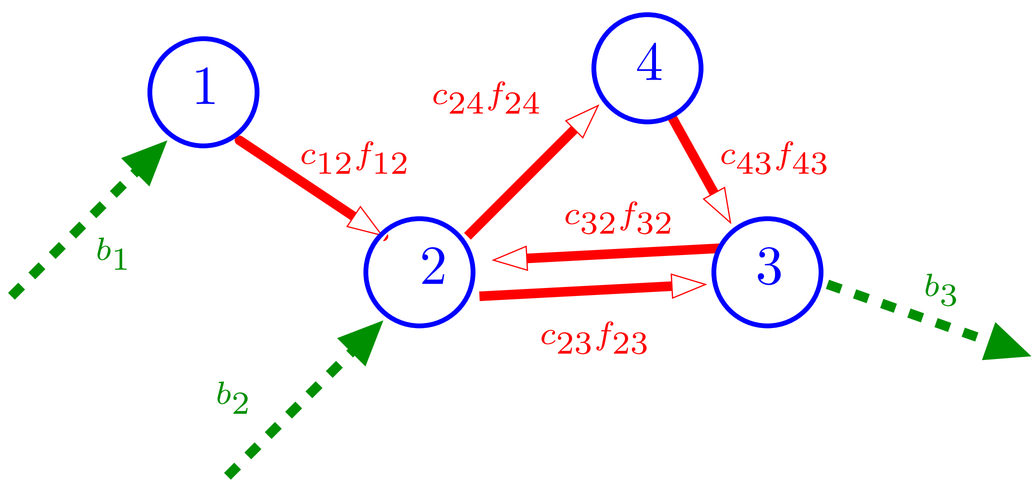
- "*what's taken from the environment goes back to the environment*"

Flow equation *objectives*

- Most network flow problems deal with the minimization of

$$\sum_{(i,j)\in\mathcal{A}} c_{ij} f_{ij}$$

- which is, again, linear in f.



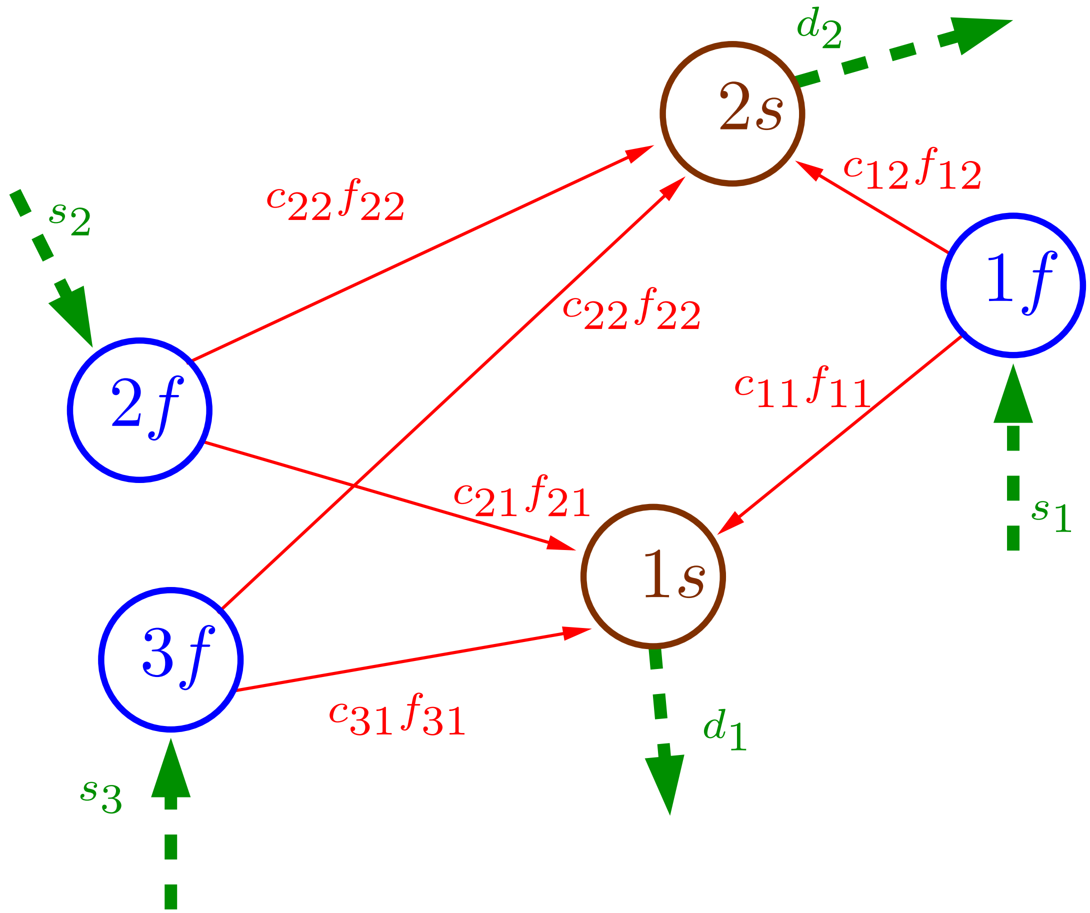
Major Examples of Network Flow Problems

The Transportation Problem

- Holes and piles of Dirt analogy.
- Old problem, formulated by Monge in 1781 and then Kantorovich in the late 30's.
- Suppose there are m factories and n shops that produce/sell computer units.
- Each factory i produces annually $s_i \geq 0$ computers and a shop j wants $d_j \geq 0$ of them.
- Each factory i has an arc directed towards each shop j.
- We suppose the total supply is equal to the demand, $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$.
- The transport problem is then

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} f_{ij} \\ \text{subject to} & f_{ij} \geq 0 \\ & \forall i = 1, \ldots, m, \quad s_{\boldsymbol{i}} = \sum_{\boldsymbol{j}=1}^{n} f_{\boldsymbol{ij}}, \\ & \forall j = 1, \ldots, n, \quad d_{\boldsymbol{j}} = \sum_{\boldsymbol{i}=1}^{n} f_{\boldsymbol{ij}}. \end{array}$$

The transportation Problem



- $1s, 2s$ stand for the shops and $1f, 2f, 3f$ is for factories.
- usually c_{ij} are proportional to distances.

The Assignment Problem

- Special case of the TP:
  - $m = n$, same number of suppliers and consumers.
  - supplies are all equal to 1, demands are all equal to 1.
  - problem is to assign one factory to one shop exactly, with minimal cost.

Next Time

- More examples.
- Provide a more concise description,
- Start describing particular types of solutions.