2.4 Network flows

Problems involving the distribution of a given "product" (e.g., water, gas, data,…) from a set of "sources" to a set of "users" so as to optimize a given objective function (e.g., amount of product, total cost,...).

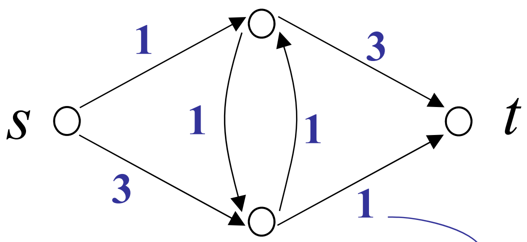
Many direct and indirect applications

- telecommunication
- transportation (public, freight, railway, air,…)
- logistics
- …

2.4.1 Maximum flow problem

Definitions:

- A *network* is a directed and connected graph $G = (V, A)$ with a source $s \in V$ and a sink $t \in V$ with $s \neq t$, and a capacity $k_{ij} \geq 0$ for each arc $(i, j) \in A$.

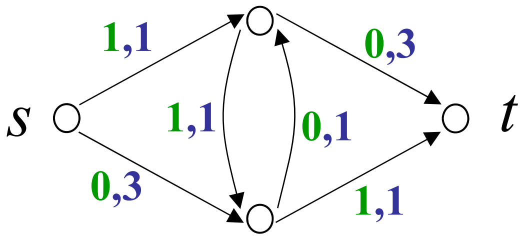


$\delta^-(s)=\delta^+(t)=\varnothing$

capacity k_{ij} of arc (i, j)

- A *feasible flow* $\underline{x}$ from s to t is a vector $\underline{x} \in \mathbb{R}^m$ with a component x_{ij} for each arc $(i,j) \in A$ satisfying the capacity constraints

$$0 \leq x_{ij} \leq k_{ij} \qquad \forall (i,j) \in A$$



Flow $\underline{x}$ of value φ=1

and the flow balance constraints at each intermediate node $h \in V \quad (h \neq s, t)$

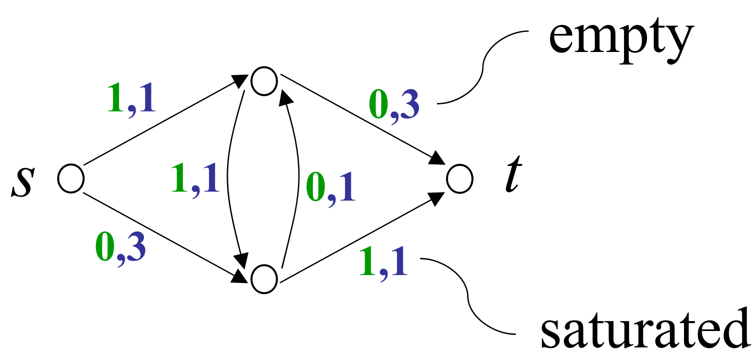
$$\sum_{(i,h)\in\delta^-(h)} x_{ih} = \sum_{(h,j)\in\delta^+ h)} x_{hj} \qquad \forall h \in V \setminus \{s,t\}$$

(amount entering in h = amount exiting from h)

- The *value of flow* $\underline{x}$: $\varphi = \sum_{(s,j)\in\delta^+(s)} x_{sj}$ with $\delta^+(s)$={ (s, j) : $(s, j)\in A$ }

- Given a network and a feasible flow $\underline{x}$,

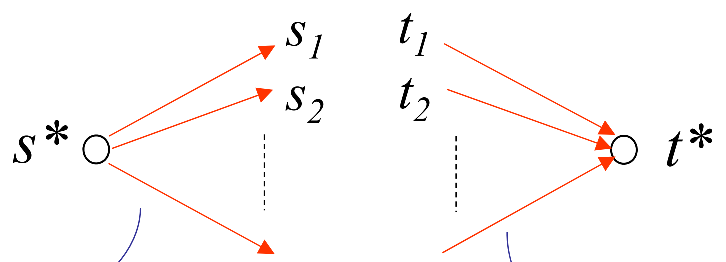
$$\text{an arc } (i, j) \in A \text{ is } \begin{cases} \underline{\textit{saturated}} \\ \\ \underline{\textit{empty}} \end{cases} \quad \text{if} \quad \begin{cases} x_{ij} = k_{ij} \\ \\ x_{ij} = 0 \end{cases}$$



Problem

Given a network $G = (V, A)$ with an integer capacity k_{ij} for each arc $(i,j) \in A$, and nodes s, $t \in V$, determine a <u>feasible flow</u> from s to t of <u>maximum value</u>.

Observation: If there are many sources/sinks with a unique type of product :



$\delta^-(s^*)=\emptyset$

$\delta^+(t^*)=\emptyset$

capacity = availability limit, if any

capacity = $+\infty$

Linear programming model

max φ

s.t.

$$\sum_{(h,j)\in\delta^+(h)} x_{hj} - \sum_{(i,h)\in\delta^-(h)} x_{ih} = \begin{cases} \varphi & \text{if } h = s \\ -\varphi & \text{if } h = t \\ 0 & \text{otherwise} \end{cases}$$

amount exiting from s

$$0 \le x_{ij} \le k_{ij} \qquad \forall\ (i,j) \in A$$

$$x_{ij} \in \mathbb{R},\ \varphi \in \mathbb{R}$$

where φ denotes the value of the feasible flow $\underline{x}$

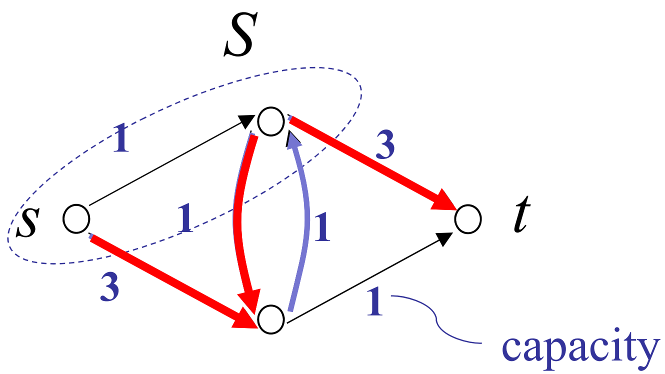
2.4.2 Cuts, feasible flows and weak duality

Definitions:

- A *cut* separating s from t is $\delta(S)$ of G with $s \in S \subset V$ and $t \in V \setminus S$.

  Number of cuts separating s from t ? 2^{n-2} with $n=|V|$

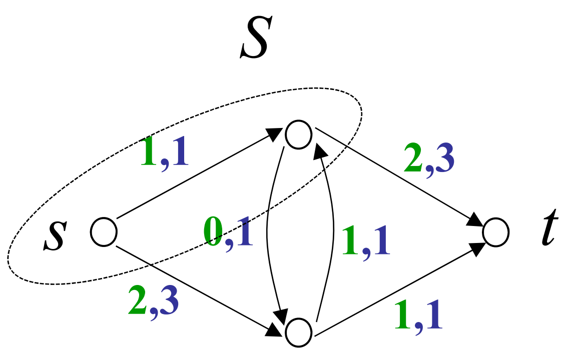
- *Capacity* of the *cut* $\delta(S)$ induced by S: $k(S) = \sum_{(i,j)\in\delta^{+}(S)} k_{ij}$



$k(S) = 7$

- Given a feasible flow $\underline{x}$ from s to t and a cut $\delta(S)$ with $s \in S$ and $t \notin S$, the *value of* the feasible *flow* $\underline{x}$ *through* the *cut* $\delta(S)$ is

$$\varphi(S) = \sum_{(i,j)\in\delta^+(S)} x_{ij} - \sum_{(i,j)\in\delta^-(S)} x_{ij}$$

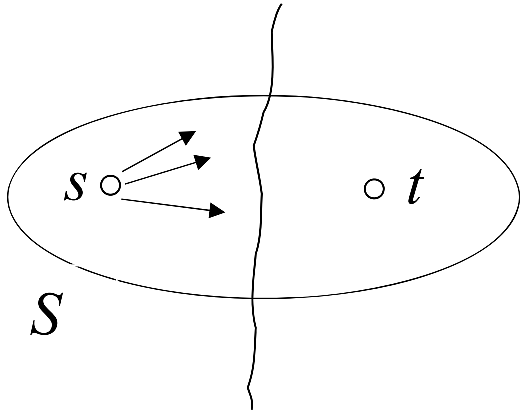


$\varphi(S) = 2 + 0 + 2 - 1 = 3$

With this notation the value of the flow $\underline{x}$ is $\varphi(\{s\})$.

Property Given a feasible flow $\underline{x}$ from s to t, for each cut $\delta(S)$ separating s from t, we have

$$\varphi(S) = \varphi(\{s\}).$$



Implied by the flow balance equations $\forall v \in V \setminus \{s, t\}$.

Property

For each feasible flow $\underline{x}$ from s to t and each cut $\delta(S)$, with $S \subseteq V$, separating s from t, we have

$$\varphi(S) \leq k(S)$$

value of the flow

capacity of the cut

Proof

Since

$$\varphi(S) = \sum_{(i,j)\in\delta^+(S)} \overset{\overset{k_{ij}}{\rotatebox{90}{$\geq$}}}{x_{ij}} - \sum_{(i,j)\in\delta^-(S)} \overset{\overset{0}{\rotatebox{90}{$\leq$}}}{x_{ij}} \leq \sum_{(i,j)\in\delta^+(S)} k_{ij} = k(S)$$

definition of value of the flow through the cut $\delta(S)$

Consequence: If $\varphi(S) = k(S)$ for a subset $S \subseteq V$ with $s \in S$ and $t \notin S$, then $\underline{x}$ is a flow of maximum value and the cut $\delta(S)$ is of minimum capacity.

The property $\varphi(S) \leq k(S)$ $\forall$ feasible flow $\underline{x}$ and $\forall$ cut $\delta(S)$ separating s from t, expresses a weak duality relationship between the two problems:

Primal problem: Given $G = (V, A)$ with integer capacities on the arcs and $s, t \in V$, determine a feasible flow of maximum value.

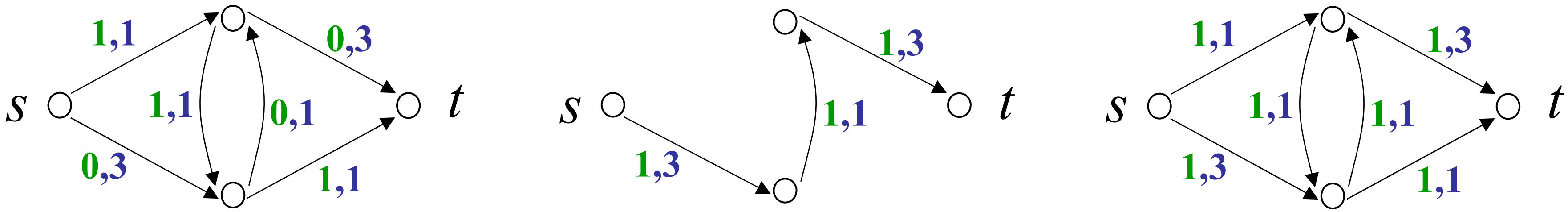
Dual problem: Given $G = (V, A)$ with integer capacities on the arcs and $s, t \in V$, determine a cut (separating s from t) of minimum capacity.

We shall see that such a relationship holds for any LP!

2.4.3 Ford-Fulkerson's algorithm

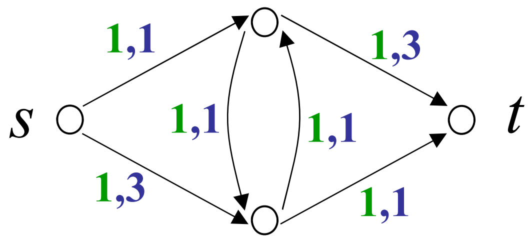
D. R. Fulkerson (1924-1976)

Idea: Start from a feasible flow $\underline{x}$ and try to iteratively increase its value φ by sending, at each iteration, an additional amount of product along a(n undirected) path from s to t with a strictly positive residual capacity.



$\varphi_0 = \varphi(\{s\}) = 1$ $\quad$ $\Delta\varphi = \delta = 1$ $\quad$ $\varphi(\{s\}) = 2$

Can the value of the current feasible flow $\underline{x}$ be increased?

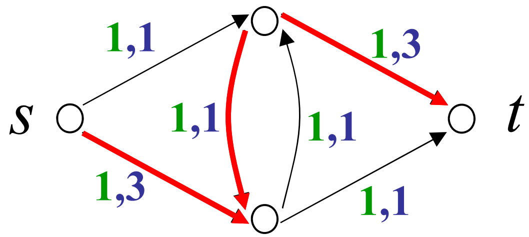


$\varphi(\{s\}) = 2$

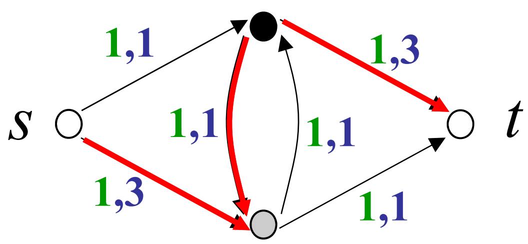
If (i, j) <u>is not saturated</u> $(x_{ij} < k_{ij})$, we can increase x_{ij}

If (i, j) <u>is not empty</u> $(x_{ij} > 0)$, we can decrease x_{ij} while respecting

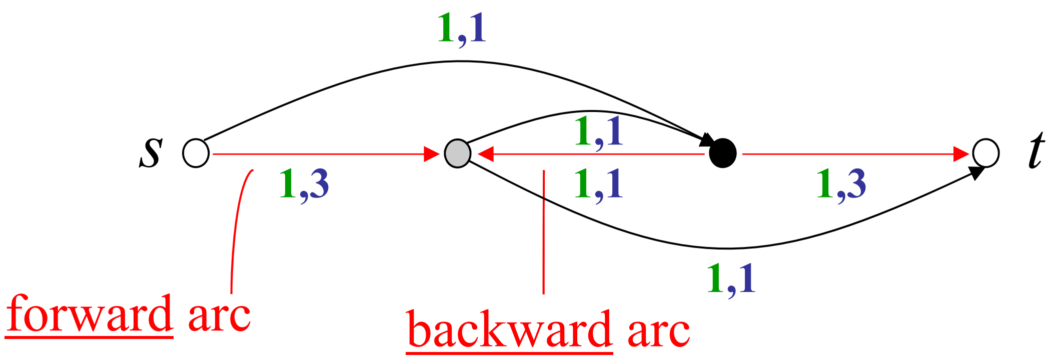
$$0 \le x_{ij} \le k_{ij}$$



$\varphi(\{s\}) = 2$



$\varphi(\{s\}) = 2$

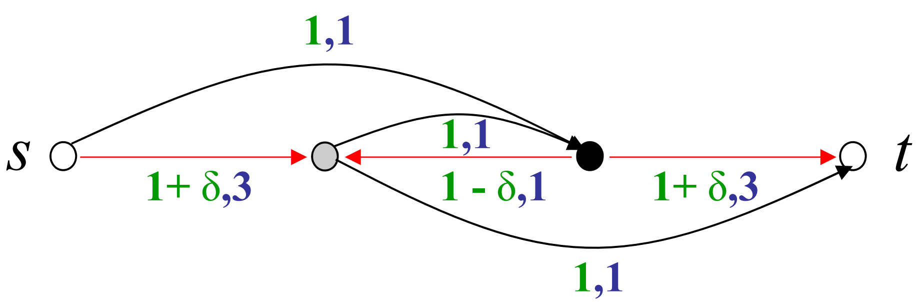


We can send $\delta = 1$ additonal units of product from s to t:

$+\delta$ along forward arcs

$-\delta$ along backward arcs

Rationale: The unit of product that was going from ● to ○ is redirected to t and the missing unit in ○ is supplied from s.

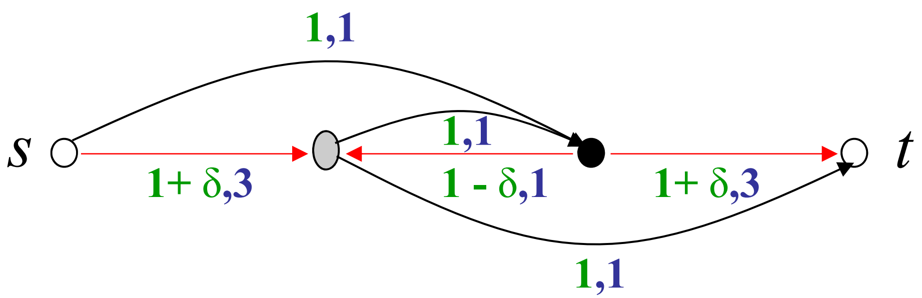


$\varphi(\{s\}) = 2 + \delta$

Definition: A path P from s to t is an *augmenting path* with respect to the current feasible flow $\underline{x}$ if

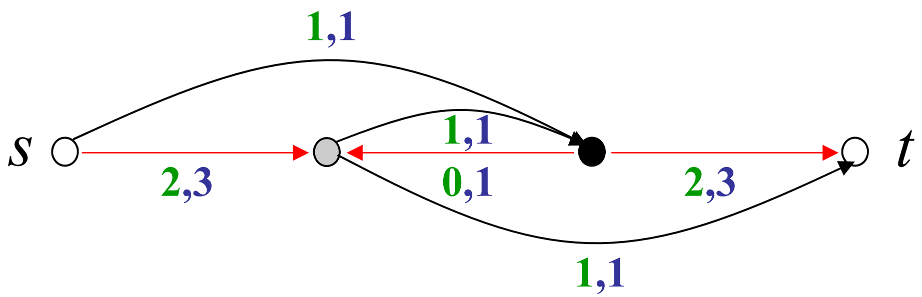
$$x_{ij} < k_{ij} \text{ for all forward arc and } x_{ij} > 0 \text{ for all backward arc.}$$

Since the maximum additional amount of product that can be sent along the augmenting path s-○-●-t



$\varphi(\{s\}) = 2+ \delta$

is equal to $\delta=1$, we obtain the new feasible flow $\underline{x}$



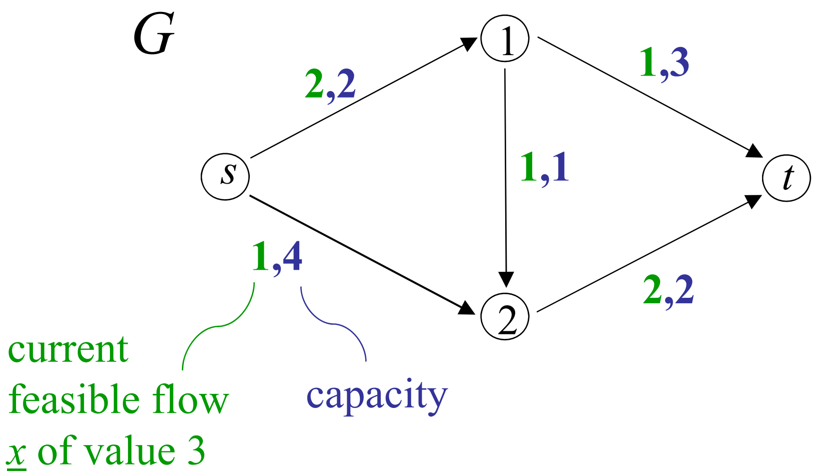
$\varphi(\{s\}) = 3$

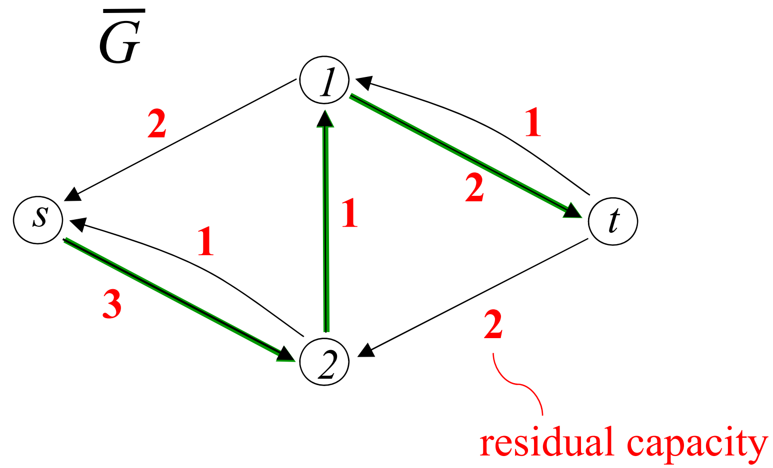
Given a feasible flow $\underline{x}$ for $G = (V, A)$, we construct the *residual network* $\overline{G} = (V, \overline{A})$ associated to $\underline{x}$, which accounts for all possible flow variations with respect to $\underline{x}$.

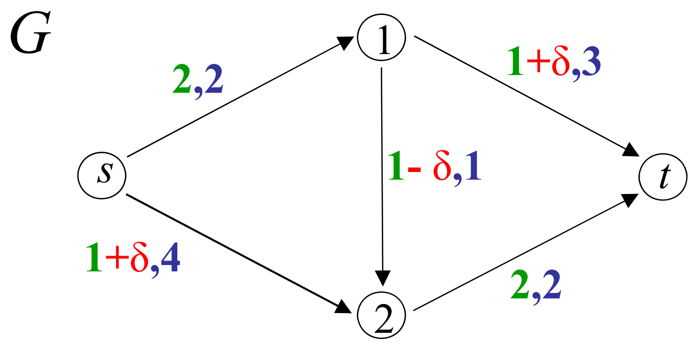
If $(i, j) \in A$ is not saturated, $(i, j) \in \overline{A}$ with $\overline{k}_{ij} = k_{ij} - x_{ij} > 0$.

residual capacity

If $(i, j) \in A$ is not empty, $(j, i) \in \overline{A}$ with $\overline{k}_{ji} = x_{ij} > 0$.





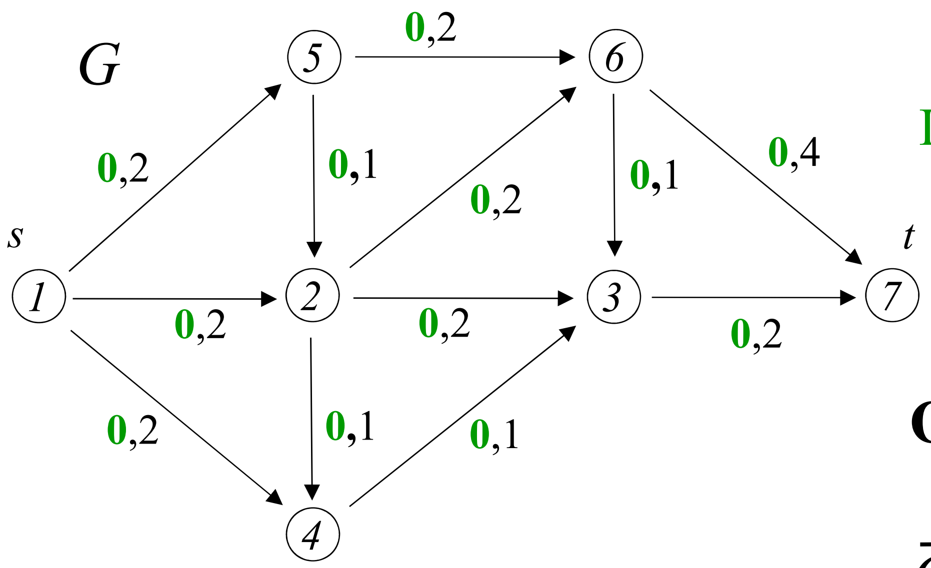


New feasible flow $\underline{x}$ of value
$\varphi = 3 + \delta \quad (\delta = 1)$

At each iteration: To look for an augmenting path from s to t in G, we search for a path from s to t in $\overline{G}$.

If $\exists$ an augmenting path from s to t, the current flow $\underline{x}$ is not optimal (of maximum value).

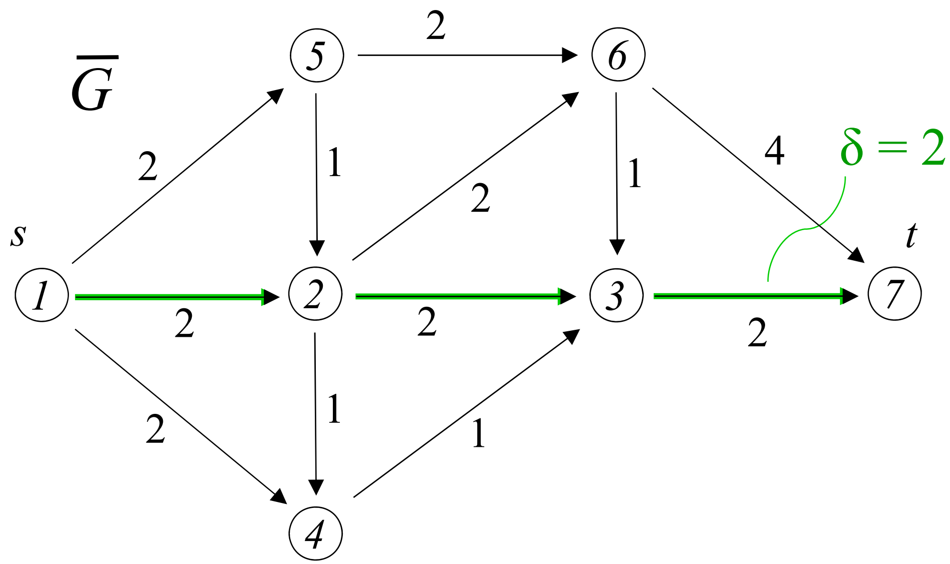
Example

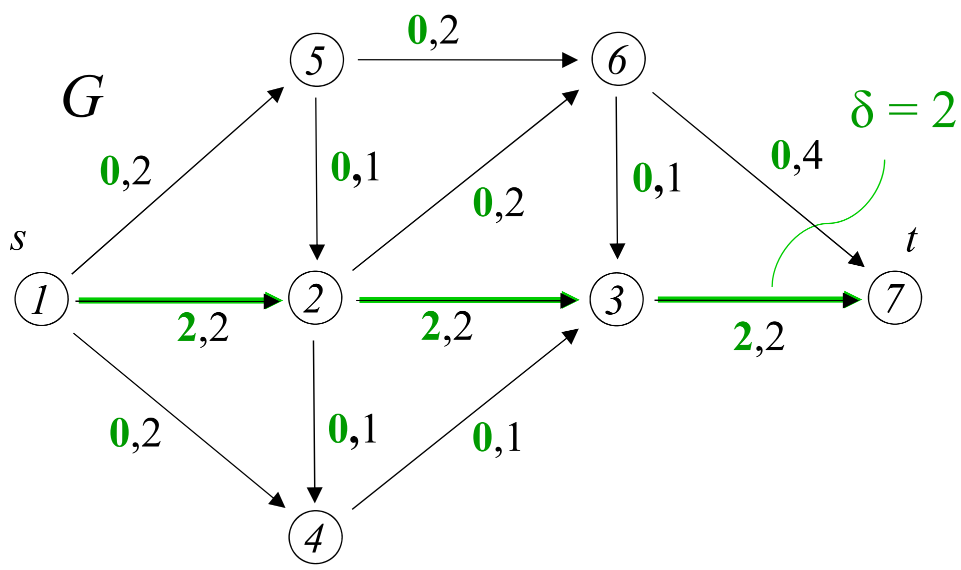


Initial feasible flow $\underline{x}_0 = \underline{0}$ of value $\varphi_0 = 0$

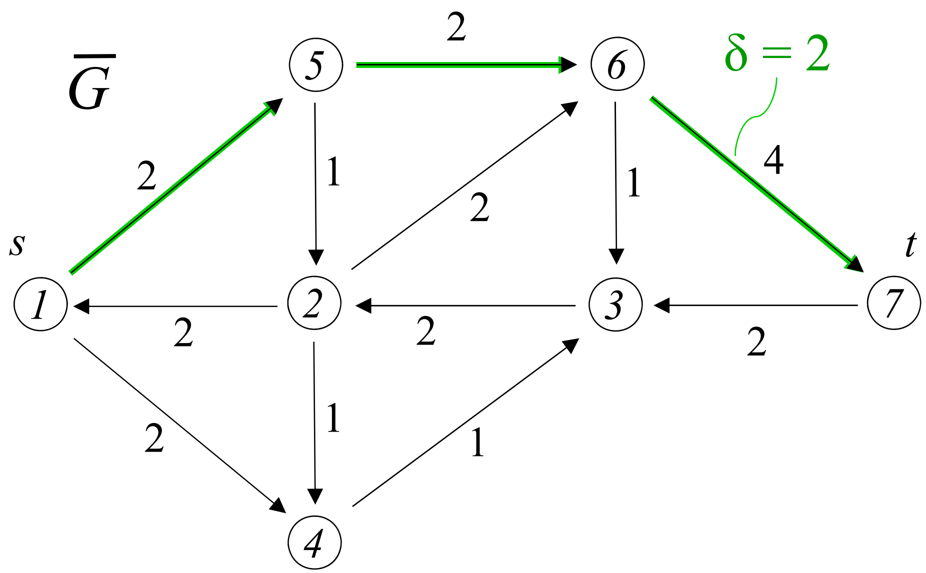
Observation: For $\underline{x} = \underline{0}$, $\overline{G} = G$

Augmenting path along which we can send $\delta = 2$ additional units of product

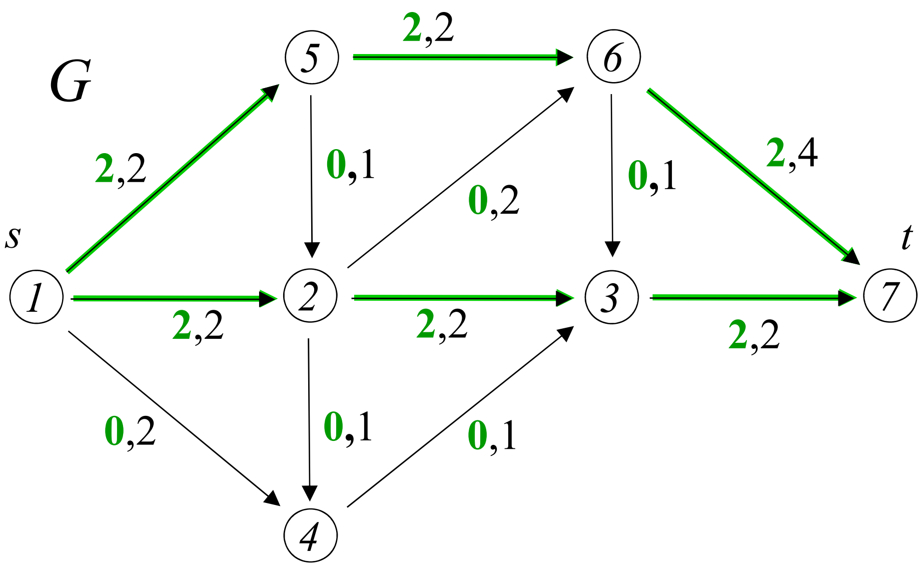




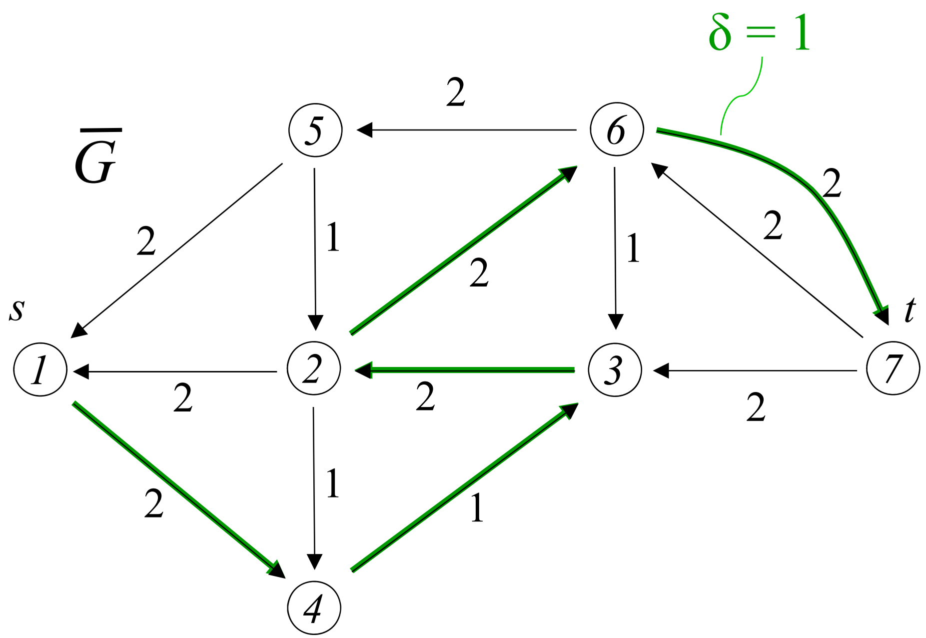
feasible flow $\underline{x}_1$ of value $\varphi_1 = 2$



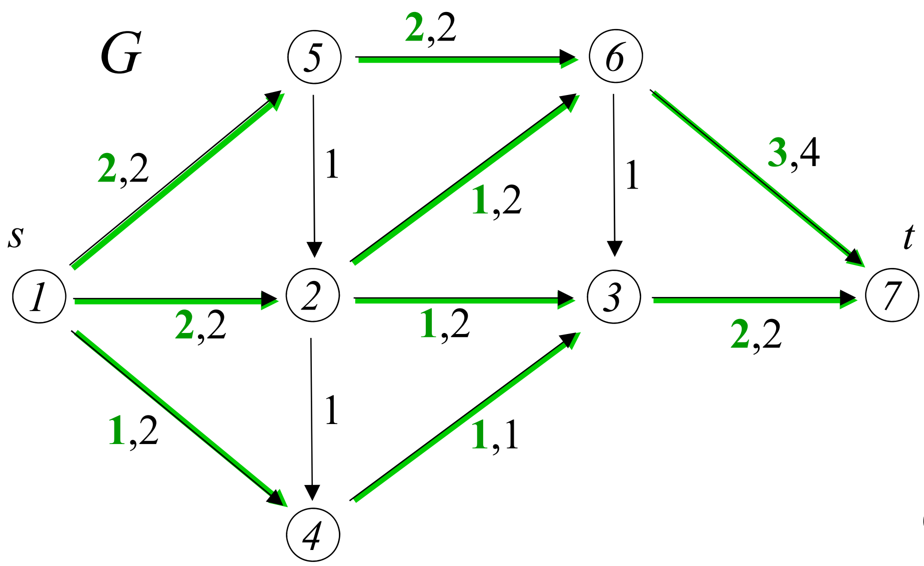
augmenting path with $\delta = 2$ with respect to the current feasible flow $\underline{x}_1$



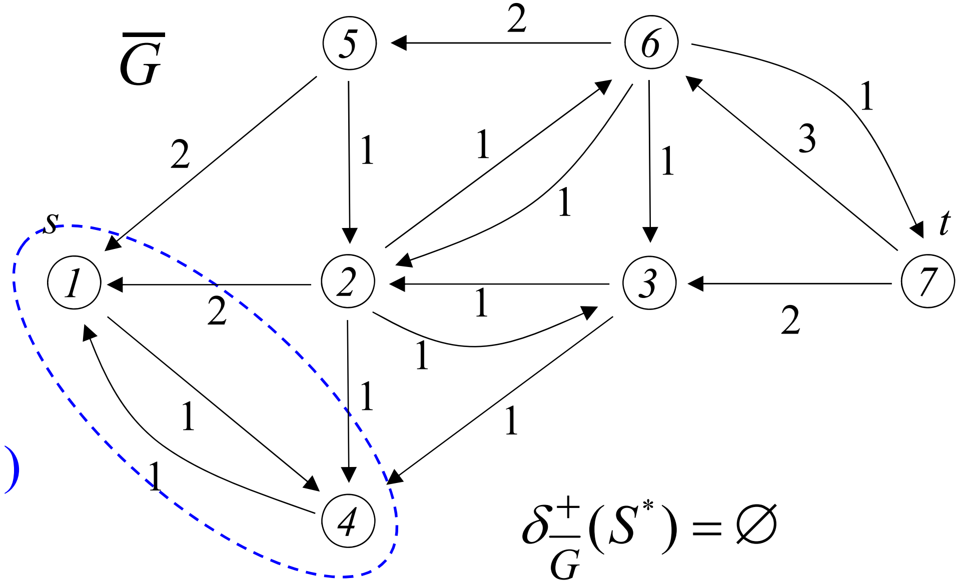
feasible flow $\underline{x}_2$ of value $\varphi_2 = 4$



augmenting path with $\delta = 1$ with respect to the current feasible flow $\underline{x}_2$



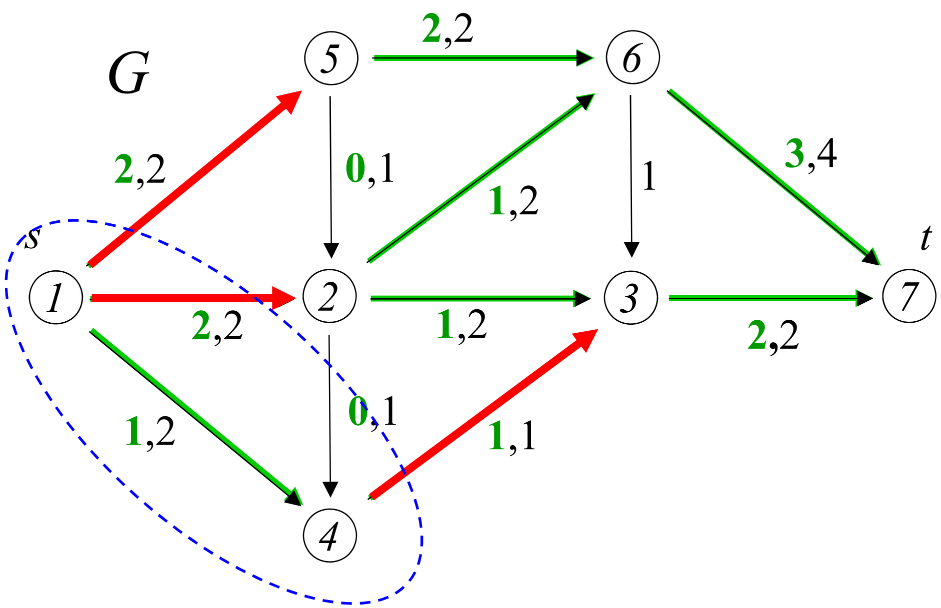
feasible flow $\underline{x}_3$ of value $\varphi_3 = 5$



$S^* = \{1, 4\}$ subset of all the nodes reachable from the source s (node 1)

$$\delta^+_{\overline{G}}(S^*) = \emptyset$$

7 is not reachable from 1 (only 4 is reachable) $\Rightarrow$ STOP



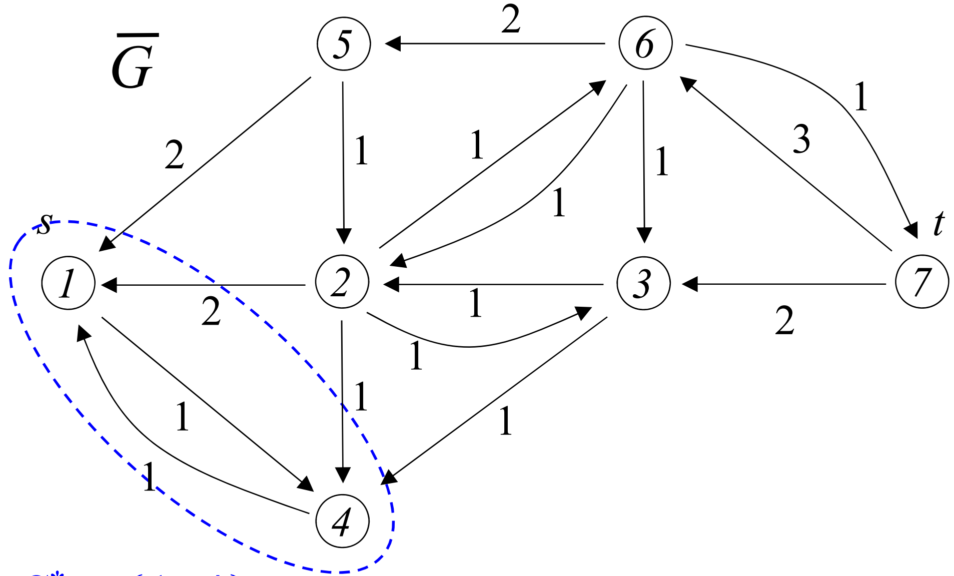
feasible flow $\underline{x}_3$ of value $\varphi_3 = 5$

Cut $\delta_G(S^*)$ of capacity = **5**

and

feasible flow $\underline{x}_3$ of value $\varphi_3 =$ **5** !

Observation: All <u>outgoing</u> arcs of $\delta_G(S^*)$ are <u>saturated</u> and all <u>entering</u> ones are <u>empty</u>



$S^* = \{1, 4\}$

Proposition: Ford-Fulkerson's algorithm is exact.

Proof

A feasible flow $\underline{x}$ is of maximum value $\Leftrightarrow$ t is not reachable from s in the residual network $\overline{G}$ associated to $\underline{x}$.

($\Rightarrow$) If $\exists$ an aumenting path, $\underline{x}$ is not optimal (of maximum value).

($\Leftarrow$) If t is not reachable from s, $\exists$ cut of $\overline{G}$ such that $\delta^+_{\overline{G}}(S^*) = \emptyset$

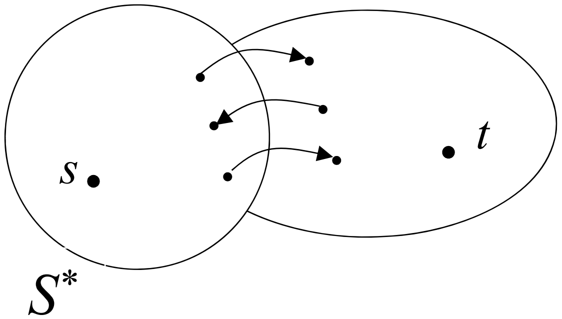
$s \in S^* \subseteq V$

By definition of $\overline{G}$, we have

- every $(i, j) \in \delta^+_G(S^*)$ is saturated
- every $(i, j) \in \delta^-_G(S^*)$ is empty.

Therefore

$$\varphi(S^*) = \sum_{(i,j)\in\delta_G^+(S^*)} \underbrace{\overset{\overset{k_{ij}}{\|}}{x_{ij}}}_{\text{all saturated}} - \sum_{(i,j)\in\delta_G^-(S^*)} \underbrace{\overset{\overset{0}{\|}}{x_{ij}}}_{\text{all empty}} = \sum_{(i,j)\in\delta_G^+(S^*)} k_{ij} = k(S^*)$$



Weak duality: $\varphi(S) \leq k(S)$ $\forall \underline{x}$ feasible

$\forall S \subseteq V, s \in S, t \notin S$

$\Rightarrow$ the feasible flow $\underline{x}$ is of <u>maximum value</u> and the <u>cut</u> induced by S^*, namely $\delta_G(S^*)$, is of <u>minimum capacity</u>.

The algorithm implies:

Theorem (Ford-Fulkerson)
The value of a feasible fow of maximum value = the capacity of a cut of minimum capacity.

strong duality

Observations:

- If all the capacities k_{ij} are integer ($\in \mathbb{Z}^+$), the flow $\underline{x}$ of maximum value has all x_{ij} integer and an integer value φ^*.
- Ford-Fulkerson's algorithm is not greedy (x_{ij} are also decreased).

Ford-Fulkerson's algorithm

Input $G = (V, A)$, capacity $k_{ij} > 0 \;\; \forall (i,j) \in A$, source $s \in V$, sink $t \in V$

Output Feasible flow $\underline{x}$ from s to t of maximum value φ^*

```
BEGIN
  x:=0; φ:=0; optimum:=false;    /* initialization */
  REPEAT
    Build residual network Ḡ associated to x;                    O(m)
    Determine, if ∃, a path P from s to t in Ḡ;                  O(n+m)
    IF  P does not exist  THEN  optimum := true;
    ELSE
      δ:= min {k̄_ij : (i,j) ∈ P};  φ:= φ + δ;                   O(n)
      FOR EACH (i,j) ∈ P DO
        IF (i,j) is forward THEN  x_ij := x_ij + δ;              O(n)
        ELSE  x_ji := x_ji - δ; END-IF
    END-IF
  UNTIL optimum = true;
END
```

Maximum number of cycles?

Complexity

Since $\delta > 0$, the value φ increases at each iteration (cycle).

If all k_{ij} are integer, $\underline{x}$ and $\overline{k}_{ij}$ integer and $\delta \geq 1 \Rightarrow$ at most φ^* increases.

Since $\varphi^* \leq \underbrace{k(\{s\})}_{\text{capacity of the cut } \delta(\{s\})} \leq m\, k_{max}$

where $m = |A|$ and $k_{max} = \max\{k_{ij} : (i, j) \in A\}$

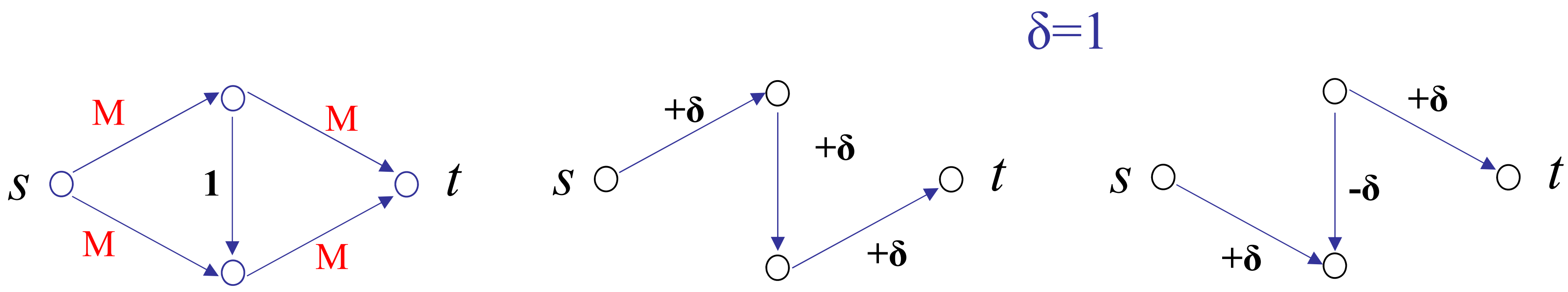
and each cycle is $O(m)$, the overall complexity is $O(m^2 k_{max})$.

Definition: The *size of an instance* I, denoted by $| I |$, is the number of bits needed to describe the instance.

Since $\lceil \log_2(i) \rceil + 1$ bits needed to store integer i, $| I | = O(m \log_2(k_{max}))$

$O(m^2 k_{max})$ grows exponentially with $| I |$ because $k_{max} = 2^{\log_2(k_{max})}$.

In some cases the algorithm is very inefficient:



M very large

In the worst case: 2M iterations!

Observation: The algorithm can be made polynomial by looking for augmenting paths with a minimum number of arcs.

Edmonds and Karp $O(nm^2)$, Dinic $O(n^2m)$,…

Also valid for the case where capacities are not integer.

Polynomial time algorithms for flow problems

More efficient algorithms exist, based on augmenting paths, pre-flows (relaxing the node flow balance constraints) and capacity scaling.

Problem

Minimum cost flow problem

Given a network with a unit cost c_{ij} associated to each arc (i,j) and a value φ>0, determine a feasible flow from s to t of value φ and of minimum total cost.

Idea: Start from a feasible flow $\underline{x}$ of value φ and send, at each iteration, an additional amount of product in the residual network (respecting the residual capacities and the value φ) along cycles of negative cost.

2.4.4 Indirect applications

1) Assigment (matching) problem

Given m engineers, n tasks and for each engineer the list of tasks he/she can perform. Assign the tasks to the engineers so that:

- each engineer is assigned at most one task,
- each task is assigned to at most one engineer,

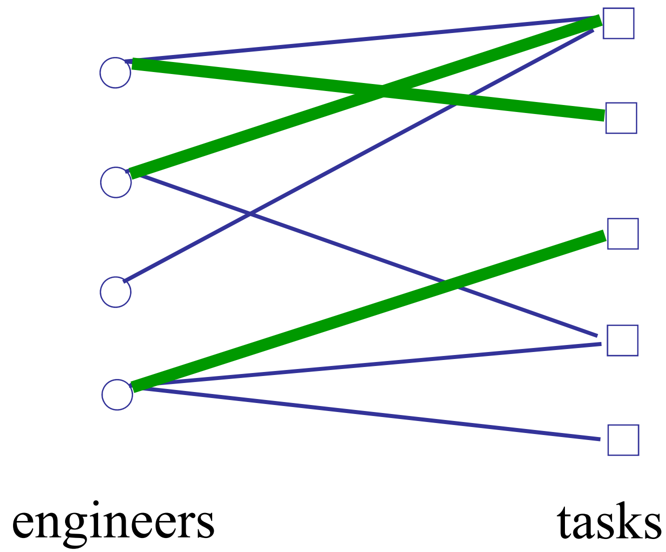
and the number of tasks that are executed (engineers involved) is maximized.

If the competences of the engineers are represented via a bipartite graph, what are we looking for in such a graph?

How can we reduce this problem to the problem of finding a feasible flow of maximum value in an ad hoc network?

Graphical model:

Bipartite graph
of competences

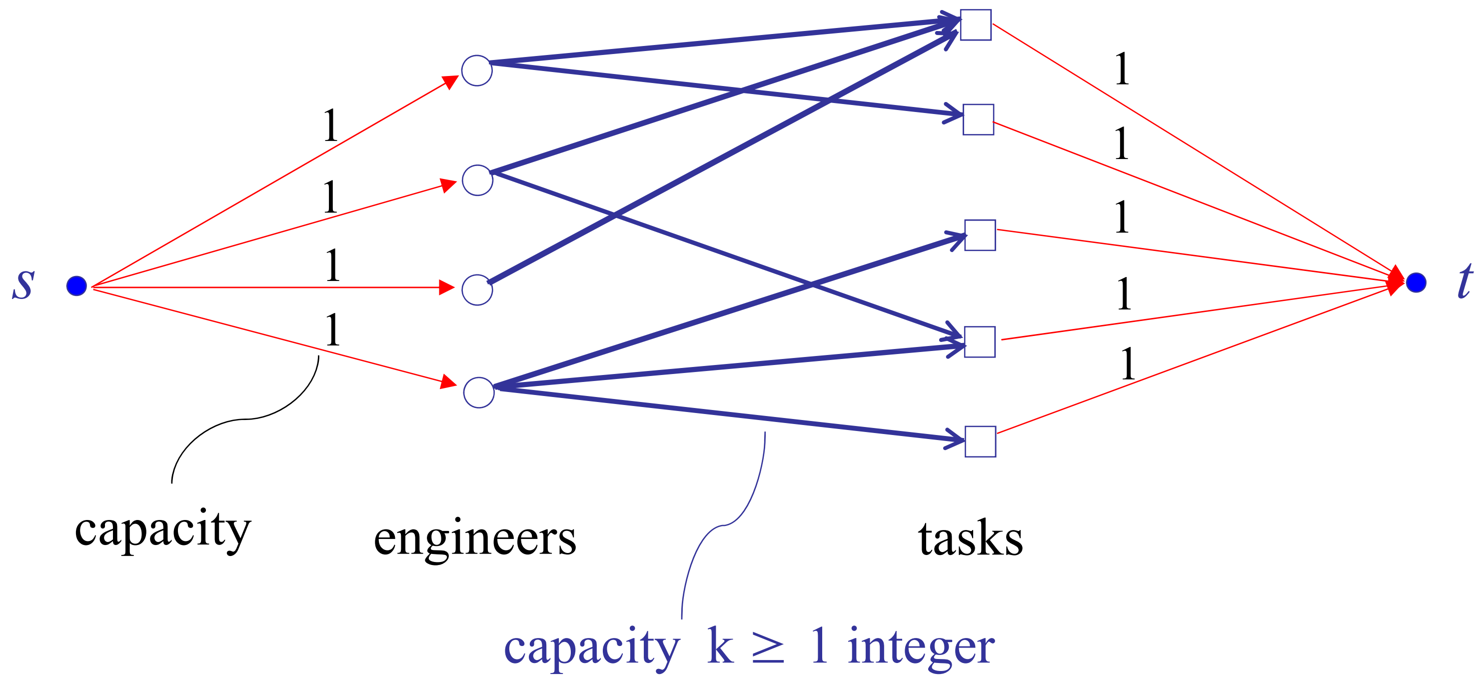


Definition: Given an undirected bipartite graph $G = (V, E)$, a *matching* $M \subseteq E$ is a subset of non adjacent edges.

Problem

Given a bipartite graph $G=(V, E)$, determine a matching with a maximum number of edges.

This problem can be reduced to the problem of finding a feasible flow of maximum value from s to t in the following network:



Correspondence between the <u>feasible flows</u> (from s to t) of <u>value φ</u> and the <u>matchings</u> containing <u>φ edges</u>.

Indeed: integer capacities $\Rightarrow$ optimal flow has integer x_{ij} and integer maximum value φ^*.

2) Distributed computing

Assign n modules of a program to 2 processors so as to minimize the total cost (execution cost + communication cost).

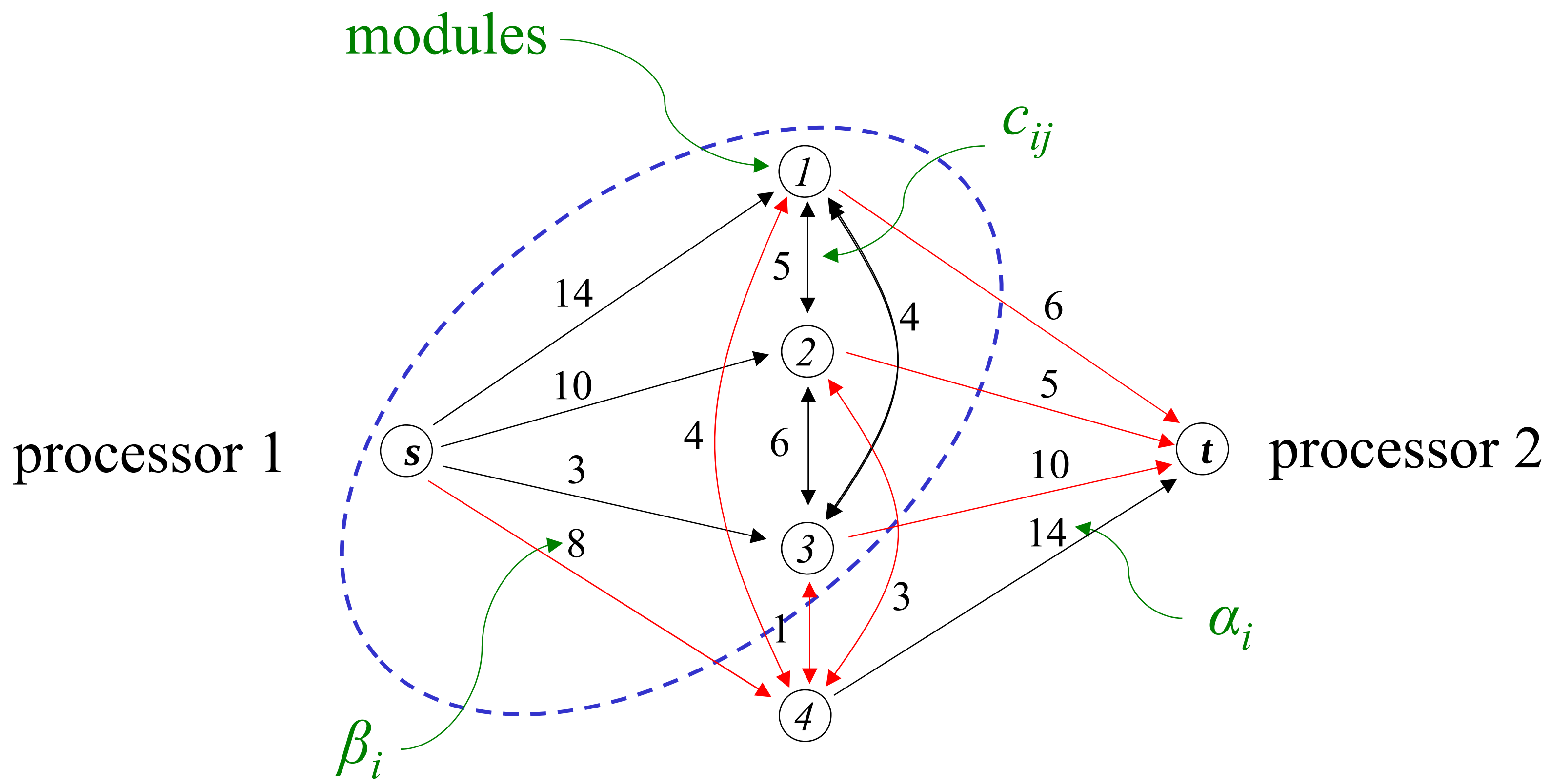
Suppose we know:

α_i = execution cost of module i on 1st processor $\quad 1 \leq i \leq n$

β_i = execution cost of module i on 2nd processor $\quad 1 \leq i \leq n$

c_{ij} = communication cost if modules i and j are assigned to different processors $1 \leq i, j \leq n$.

Reduce this problem to that of finding a cut of minimum total capacity in an ad hoc directed network.



cut separating *s* from *t* ⟷ assignment of the n modules to the 2 processors

Correspondence between the *s*-*t* cuts of minimum capacity and the minimum total cost assignments of the modules to the processors.