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,...).

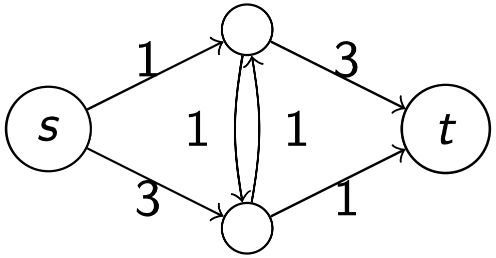
Many direct and indirect applications:

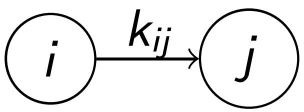
- telecommunication
- transportation (public, freight, railway, air, ...)
- logistics
- ...

2.4. Network flows

Definition 6

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 $(i, j) \in A$.





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

2.4. Network flows

Definition 7

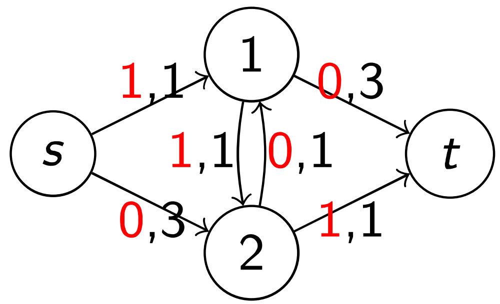
- A feasible flow x from s to t is a vector $x \in \mathbb{R}^m$ with a component x_{ij} for each arc $(i,j) \in A$ satisfying the capacity constraints

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

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

$$\sum_{(i,u)\in\delta^-(u)} x_{iu} = \sum_{(u,j)\in\delta^+(u)} x_{uj}, \quad \forall u \in N \backslash \{s,t\}$$

  (amount entering u = amount exiting u).
- The value of flow x is $\varphi = \sum_{(s,j)\in\delta^+(s)} x_{sj}$.
- Given a network and a feasible flow x, an arc $(i,j) \in A$ is saturated (empty) if $x_{ij} = k_{ij}$ ($x_{ij} = 0$).



Flow x of value $\varphi = 1$

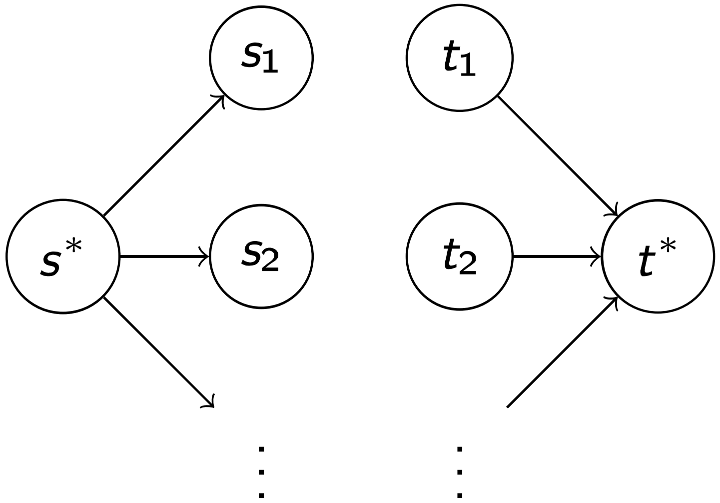
x_{1t} is empty.

x_{2t} is saturated.

2.4. Network flows

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 feasible flow from s to t of maximum value.

Note
If there are many sources/sinks with a unique type of product, dummy nodes s^* and t^* can be added:



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

$$k_{s^*i} = \text{availability limit, if any}$$

$$k_{jt^*} = +\infty$$

2.4. Network flows

Linear programming model

$$
\begin{array}{ll}
\max & \varphi \\
\text{s. t.} & \displaystyle\sum_{(u,j)\in\delta^+(u)} x_{uj} - \sum_{(i,u)\in\delta^-(u)} x_{iu} = \begin{cases} \varphi & u = s \\ -\varphi & u = t \\ 0 & \text{otherwise} \end{cases} \\
& 0 \le x_{ij} \le k_{ij}, \quad (i,j) \in A \\
& \varphi \in \mathbb{R}, x_{ij} \in \mathbb{R}, \quad (i,j) \in A
\end{array}
$$

where φ denotes the value of the feasible flow x (φ is also the amount exiting from s).

2.4. Network flows

Cuts, feasible flows and weak duality

Definition 8

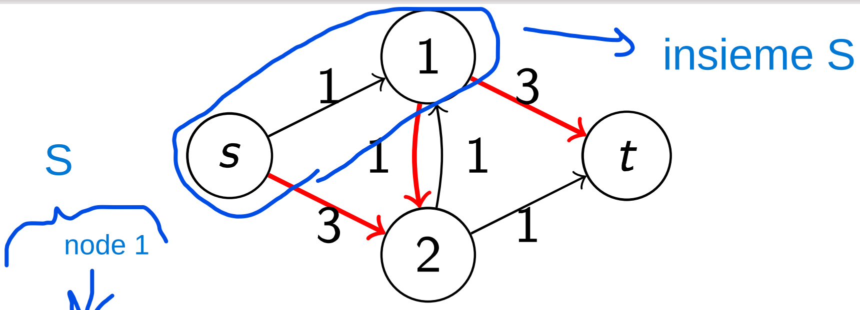
è un insieme di archi che vanno o vengono dall'insieme S, con s che fa parte di S e t che non fa parte di S.

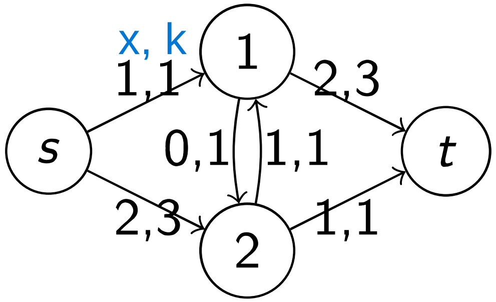
- A cut separating s from t is $\delta(S)$ of G with $s \in S \subset V$ and $t \in V \setminus S$.
  How many cuts separate s from t? 2^{n-2}, where $n = |V|$.
- Capacity of the cut $\delta(S)$ induced by S: $k(S) = \sum_{(i,j)\in\delta^+(S)} k_{ij}$.

  è la somma di tutti i flow constriant k presenti negli archi che escono dall'insieme S
- Given a feasible flow x from s to t and a cut $\delta(S)$ separating s from t, the value of the feasible flow x through the cut $\delta(S)$ is

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

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





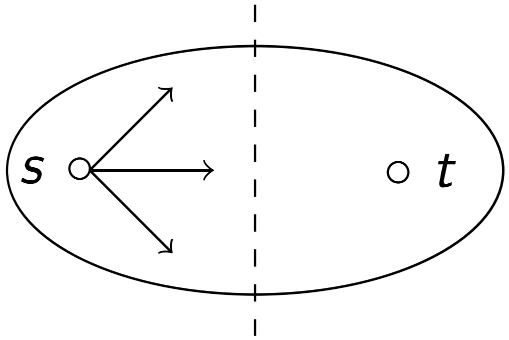
$\delta(\{s,1\}) = \{(s,2),(1,2),(1,t)\},\ k(S) = 7$

$\varphi(\{s,1\})) = 2 + 0 + 2 - 1 = 3$

Property 9

Given a feasible flow 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 $\forall v \in V \setminus \{s, t\}$.

2.4. Network flows

Property 10

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

$$\varphi(S) \leq k(S) \qquad \textit{(value of the flow} \ \leq \ \textit{capacity of the cut)}$$

Proof.

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

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

and because $0 \leq x_{ij} \leq k_{ij}$, for any $(i,j) \in A$,

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} - \sum_{(i,j)\in\delta^-(S)} x_{ij} \leq \sum_{(i,j)\in\delta^+(S)} k_{ij} = k(S).$$

Then, $\varphi(S) \leq k(S)$. □

2.4. Network flows

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

The property $\varphi(S) \leq k(S)$ for any feasible flow x and for any 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 arc capacities 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!

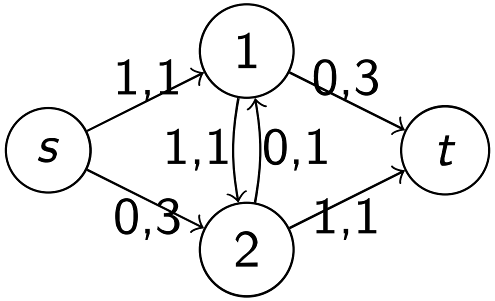
Ford-Fulkerson's algorithm

D. R. Fulkerson
(1924-1976)

Idea

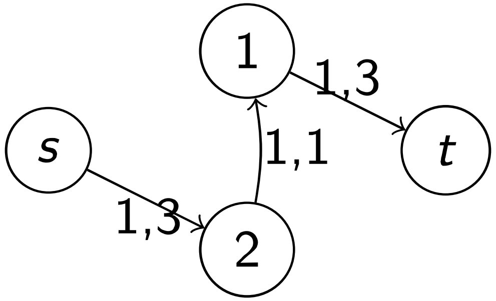
Start from a feasible flow x and try to iteratively increase its value φ by sending, at each iteration, an additional amount of product along a(n undirected) path from s to t with a strictly positive residual capacity.

Initial solution:

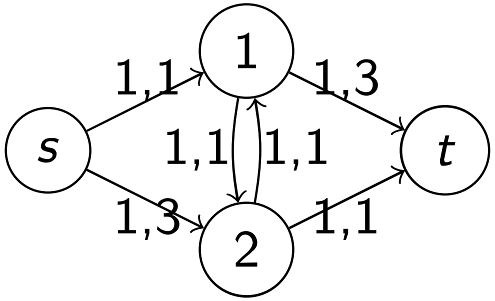


$$\varphi_0 = \varphi(\{s\}) = 1$$

Another path which is not "full"
Let's update the flow by one unit



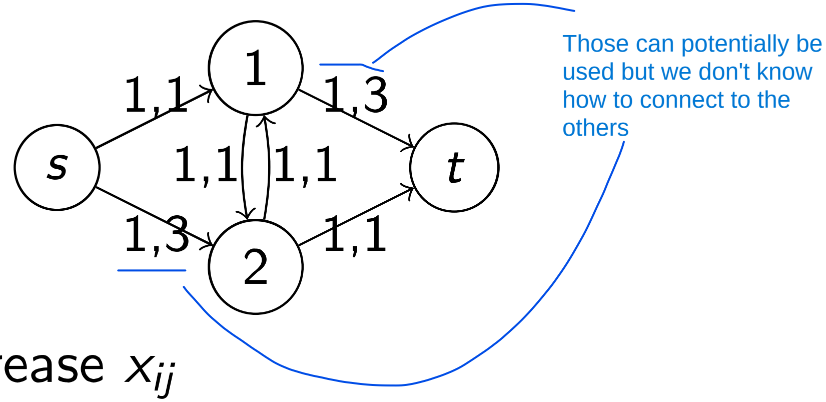
$$\Delta\varphi = \delta = 1$$



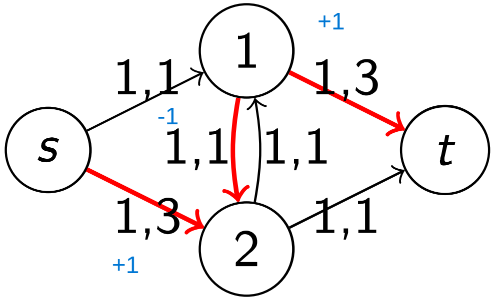
$$\varphi(\{s\}) = 2$$

2.4. Network flows

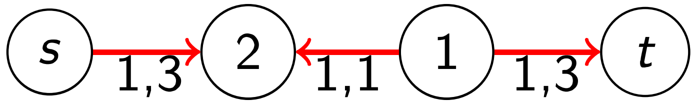
Can the value of the current feasible flow x be increased ($\varphi(\{s\}) = 2$)?



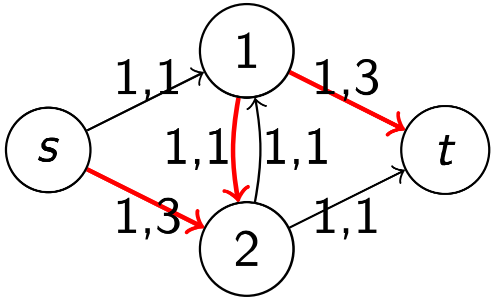
- If (i, j) is not saturated ($x_{ij} < k_{ij}$), we can increase x_{ij}
- (i, j) is not empty ($x_{ij} > 0$), we can decrease x_{ij} while respecting $0 \leq x_{ij} \leq k_{ij}$.



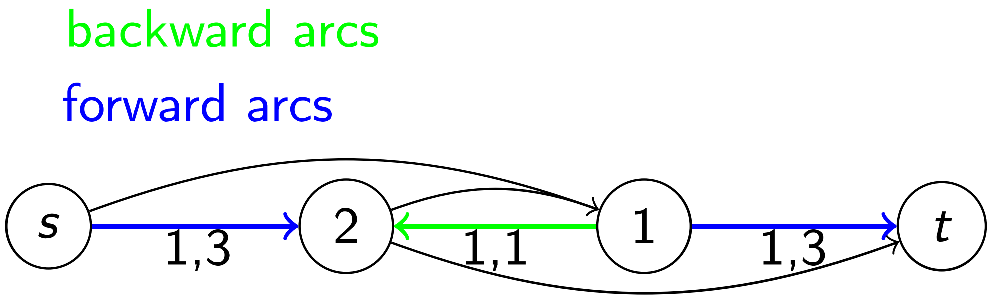
$\varphi(\{s\}) = 2$



2.4. Network flows



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

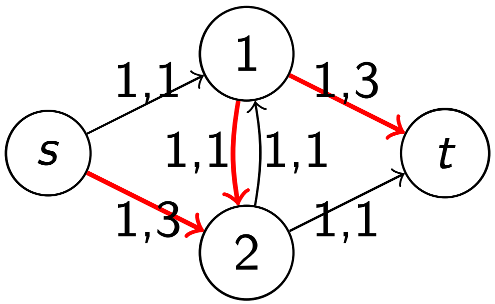


We can send $\delta = 1$ additonal units of product from s to t:

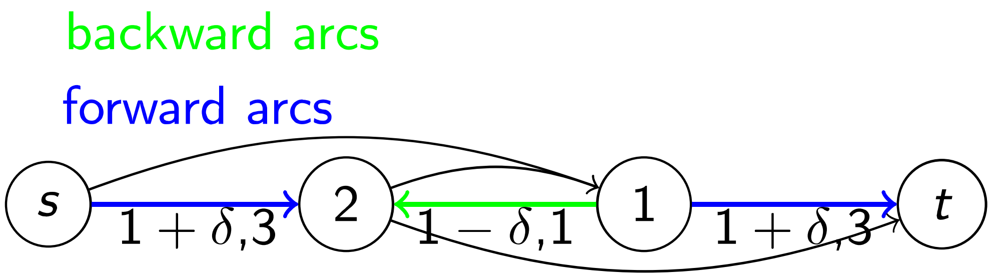
- $+\delta$ along forward arcs not saturated arcs
- $-\delta$ along backward arcs arcs that are not empty

Rationale

The unit of product that was going from 2 to 1 is redirected to t and the missing unit in 1 is supplied from s.



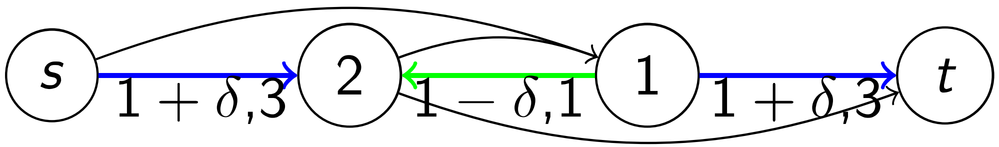
$\varphi(\{s\}) = 2 + \delta$



Definition 9

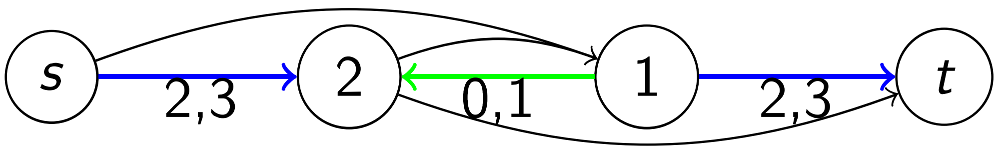
A path P from s to t is an augmenting path with respect to the current feasible flow x if $x_{ij} < k_{ij}$ for any forward arc and $x_{ij} > 0$ for any backward arc.
if there's a not saturated arc
not empty arc

Since the maximum additional amount of product that can be sent along the augmenting path $\langle (s,1),(1,2),(2,t)\rangle$



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

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



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

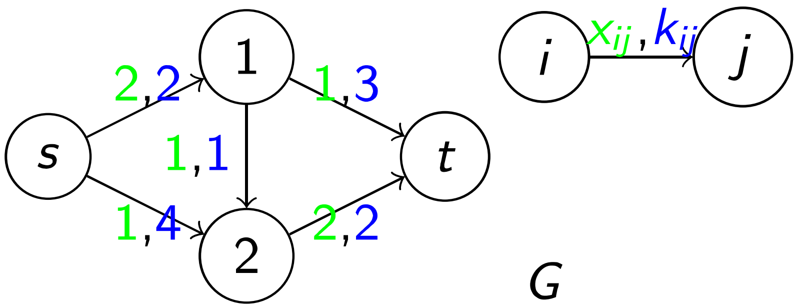
2.4. Network flows

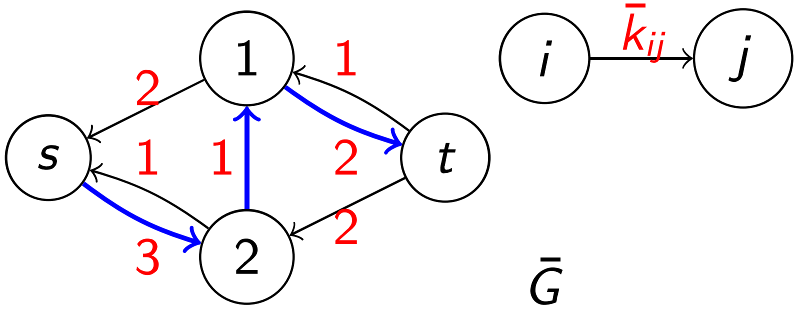
Given a feasible flow x for $G = (V, A)$, we construct the residual network $\bar{G} = (V, \bar{A})$ associated to x, which accounts for all possible flow variations w.r.t. x.

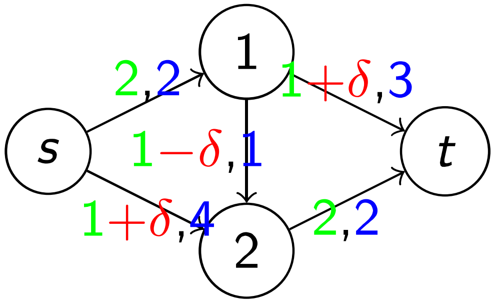
(arco opposto)

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

$\bar{k}_{ij}$ is called the residual capacity.







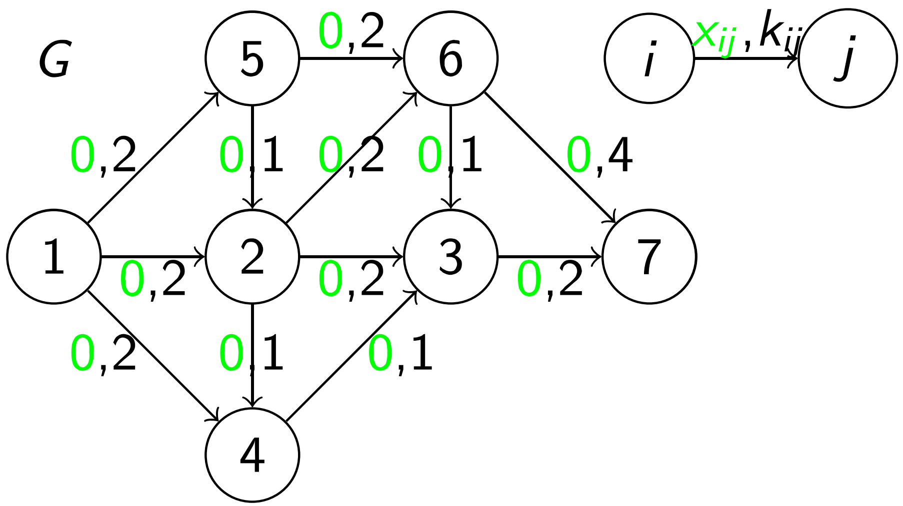
min{3,1,2}

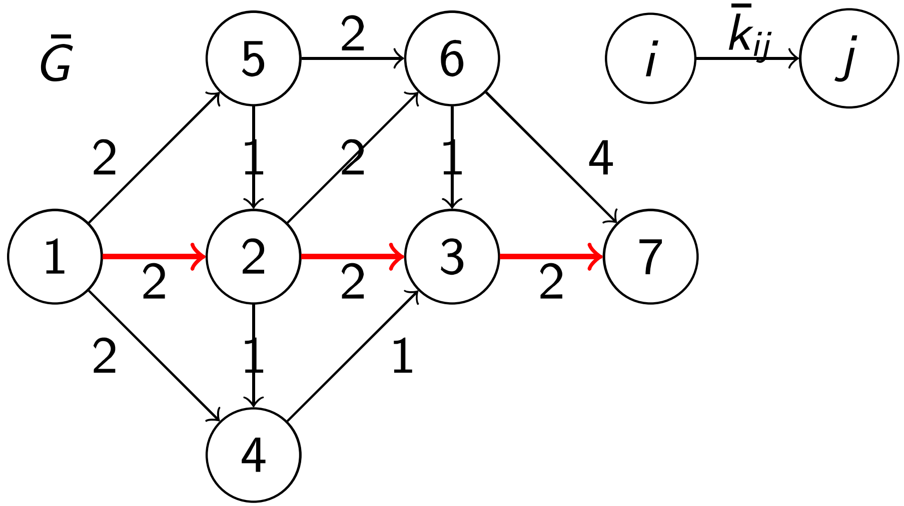
New feasible flow x of value $\varphi = 3 + \delta = 4$ ($\delta = 1$).

At each iteration:

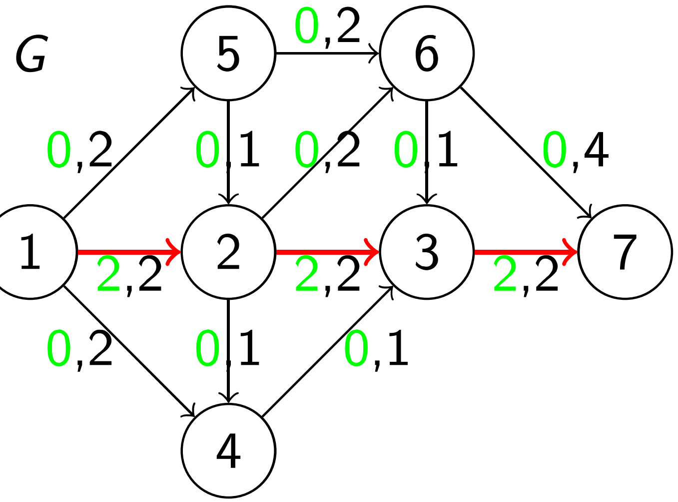
- To look for an augmenting path from s to t in G, we search for a path from s to t in $\bar{G}$.
- If there is an augmenting path from s to t, the current flow x is not of maximum value.

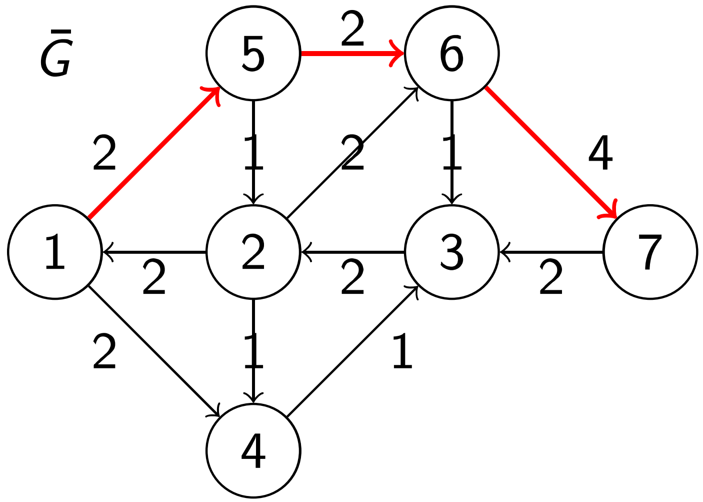
Example





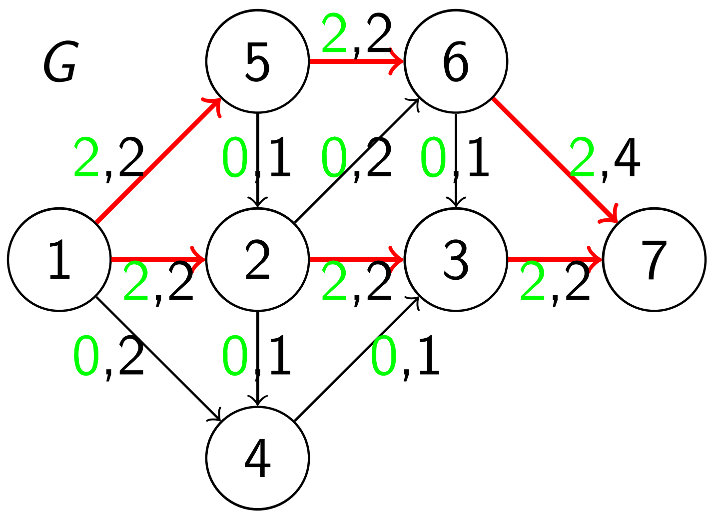
Augmenting path along which we can send $\delta = 2$ additional units of product

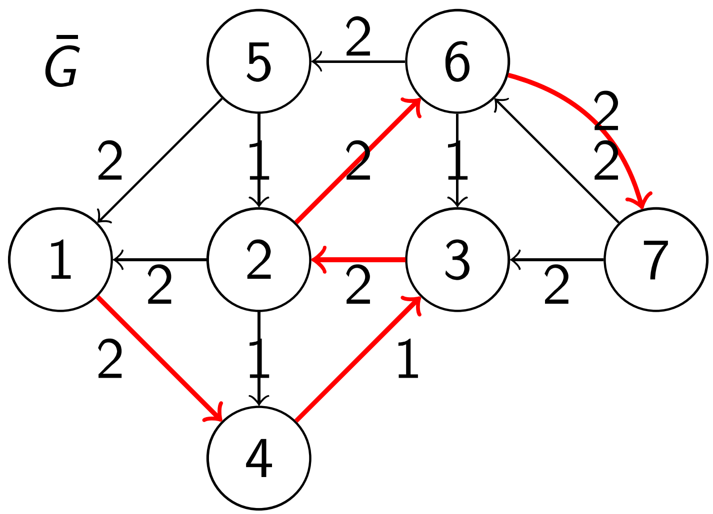




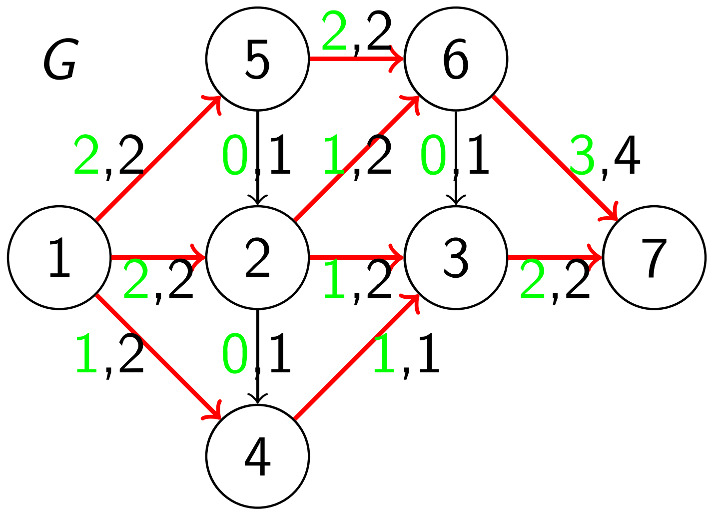
Augmenting path along which we can send $\delta = 2$ additional units of product

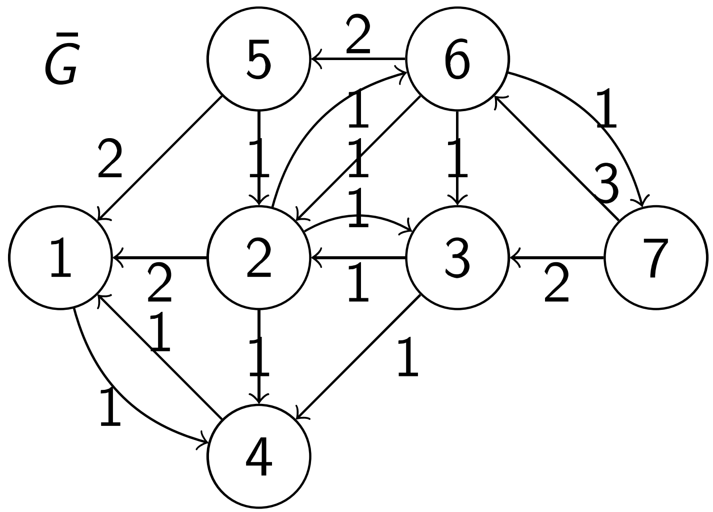
2.4. Network flows





Augmenting path along which we can send $\delta = 1$ additional units of product

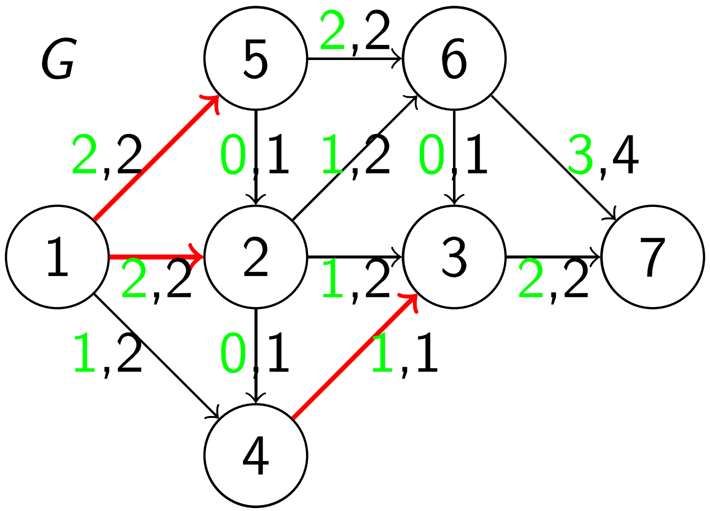


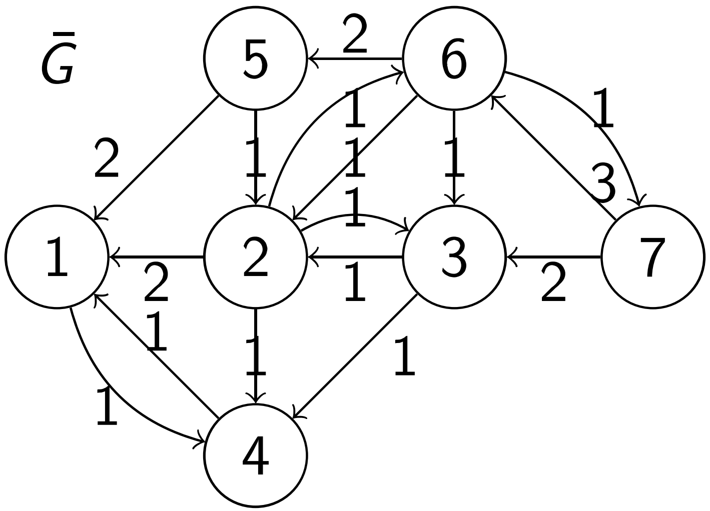


$S^* = \{1, 4\}$ subset of nodes reachable from s; $\delta^+_{\bar{G}}(S^*) = \emptyset$

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

2.4. Network flows





Cut $\delta_G(S^*)$ of capacity 5 and Feasible flow x of value $\varphi = 5$

Note

For $S^* = \{1, 4\}$, all outgoing arcs of $\delta_G(S^*)$ are saturated and all entering ones are empty.

2.4. Network flows

Proposition 5

Ford-Fulkerson's algorithm is exact.

Proof.

A feasible flow x has maximum value iff t is not reachable from s in the residual network associated to x.
$\Rightarrow$ If there is an aumenting path, then x is not optimal (of maximum value).
$\Leftarrow$ If t is not reachable from s, then there is a cut of $\bar{G}$ such that $\delta^+_{\bar{G}}(S^*) = \emptyset$.
By definition of $\bar{G}$, we have:

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

Therefore,

$$\varphi(S^*) = \sum_{(i,j)\in\delta^+_G(S^*)} \overbrace{x_{ij}}^{=k_{ij}} - \sum_{(i,j)\in\delta^-_G(S^*)} \overbrace{x_{ij}}^{=0} = \sum_{(i,j)\in\delta^+_G(S^*)} k_{ij} = k(S^*).$$

By weak duality, $\varphi(S) \leq k(S)$, $\forall x$ feasible, $\forall S \subset V$, with $s \in S$, $t \notin S$. Then, the flow x has maximum value and the cut induced by S^* has minimum capacity. $\square$

The algorithm implies:

Theorem 10 (Ford-Fulkerson/Strong duality)

The value of a feasible flow of maximum value = the capacity of a cut of minimum capacity.

Notes

- If all the capacities k_{ij} are integer ($\in \mathbb{Z}^+$), the flow 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).

2.4. Network flows

- Input: Graph $G = (N, A)$ with capacity $k_{ij} > 0$,for any $(i, j) \in A$, $s, t \in N$.
- Output: Feasible flow x from s to t of maximum value φ^*

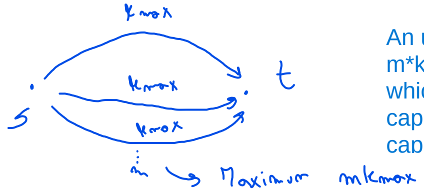
Algorithm 8: Ford-Fulkerson's algorithm for the maximum flow problem

$x \leftarrow 0$
$\varphi \leftarrow 0$ start from a initial flow of 0
$optimum \leftarrow false$ supposing we start from a flow of 0. This cycle gets repeated a number of times which is equal to the maximum value of the flow (phi*). Now, what is an upper bound for phi*?
repeat
m arcs
k_max, maximum capacity among all the capacities in the network
    Build residual network $\bar{G}$ associated to x
    $P \leftarrow$ path from s to t in $\bar{G}$ look for a path from in the residual network
    if *P is not defined* **then** optimum $\leftarrow$ true if the residual path do not exist then the feasible flow is the maximum
    else we can find an augmenting path
        $\delta \leftarrow \min\{\bar{k}_{ij} : (i, j) \in P\}$ we look for the minimum residual capacity on the path in the residual graph
        $\varphi \leftarrow \varphi + \delta$ update the flow
        for $(i, j) \in P$ **do**
            if (i, j) *is a forward arc* **then** $x_{ij} \leftarrow x_{ij} + \delta$
            else $x_{ji} \leftarrow x_{ji} - \delta$
until $optimum = true$

2.4. Network flows

Complexity

- Since $\delta > 0$, the value φ increases at each iteration (cycle).
- If all k_{ij} are integer, x and $\bar{k}_{ij}$ integer and $\delta \geq 1$, then there are at most φ^* increases. each time we increase at least by one
- Since



An upper bound for phi* is m*k_max, since is the case in which every arc is has the capacity of the maximum capacity among the arcs.

$$\varphi^* \leq k(\{s\}) \leq m\, k_{\max},$$

where $m = |A|$ and $k_{\max} = \max\{k_{ij} : (i,j) \in A\}$ and each loop is $O(m)$, the overall complexity is $O(m^2 k_{\max})$.

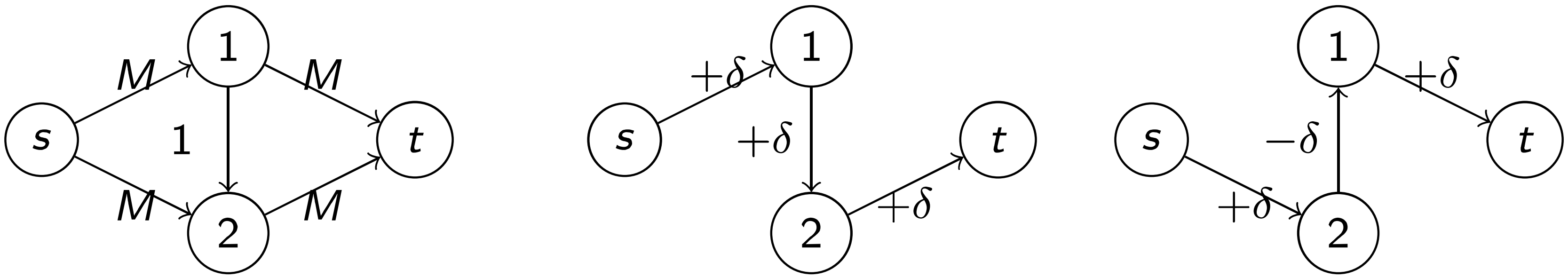
Definition 11

The size of an instance I, $|I|$, is the number of bits needed to describe the instance.

- Since $\lceil \log_2 i \rceil + 1$ bits are needed to store integer i, $|I| = O(m \log_2 k_{\max})$.
- $O(m^2 k_{\max})$ grows exponentially with $|I|$ because $k_{\max} = 2^{\log_2 k_{\max}}$.

2.4. Network flows

In some cases the algorithm may be very inefficient:



Assume M is very large

In the worst case ($\delta = 1$): $2M$ iterations!

Observation

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

$$\text{Edmonds and Karp } O(nm^2),\ \text{Dinic } O(n^2 m),\ \ldots$$

Also valid for the case where capacities are not integer.

2.4. Network flows

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 12 (Minimum cost flow problem)

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

Idea
Start from a feasible flow 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. Network flows

Indirect applications

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 such 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.

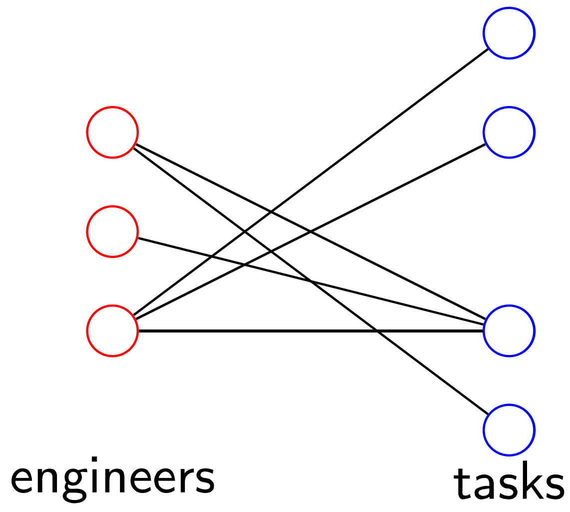
If the competences of the engineers are represented via a bipartite graph, what are we looking for in such a graph?

engineers are not connected between them
and tasks are not connected between them

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

2.4. Network flows

Graphical model: Bipartite graph of competences



Definition 13

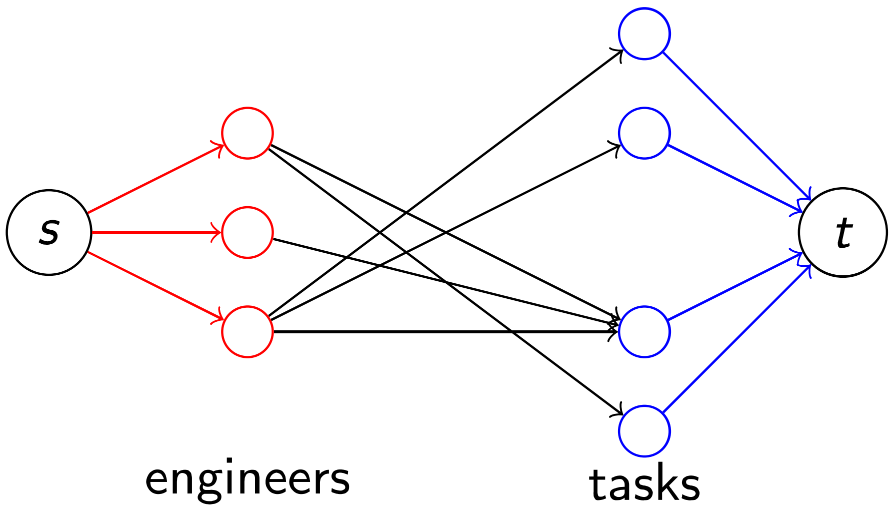
Given an undirected bipartite graph $G = (V, E)$, a matching $M \subseteq E$ is a subset of non-adjacent edges.

Problem 14

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

2.4. Network flows

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



$k_{sj} = k_{it} = 1$

There is a correspondence between the feasible flows (from s to t) of value φ and the matchings containing φ edges.

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

2.4. Network flows

Distributed computing

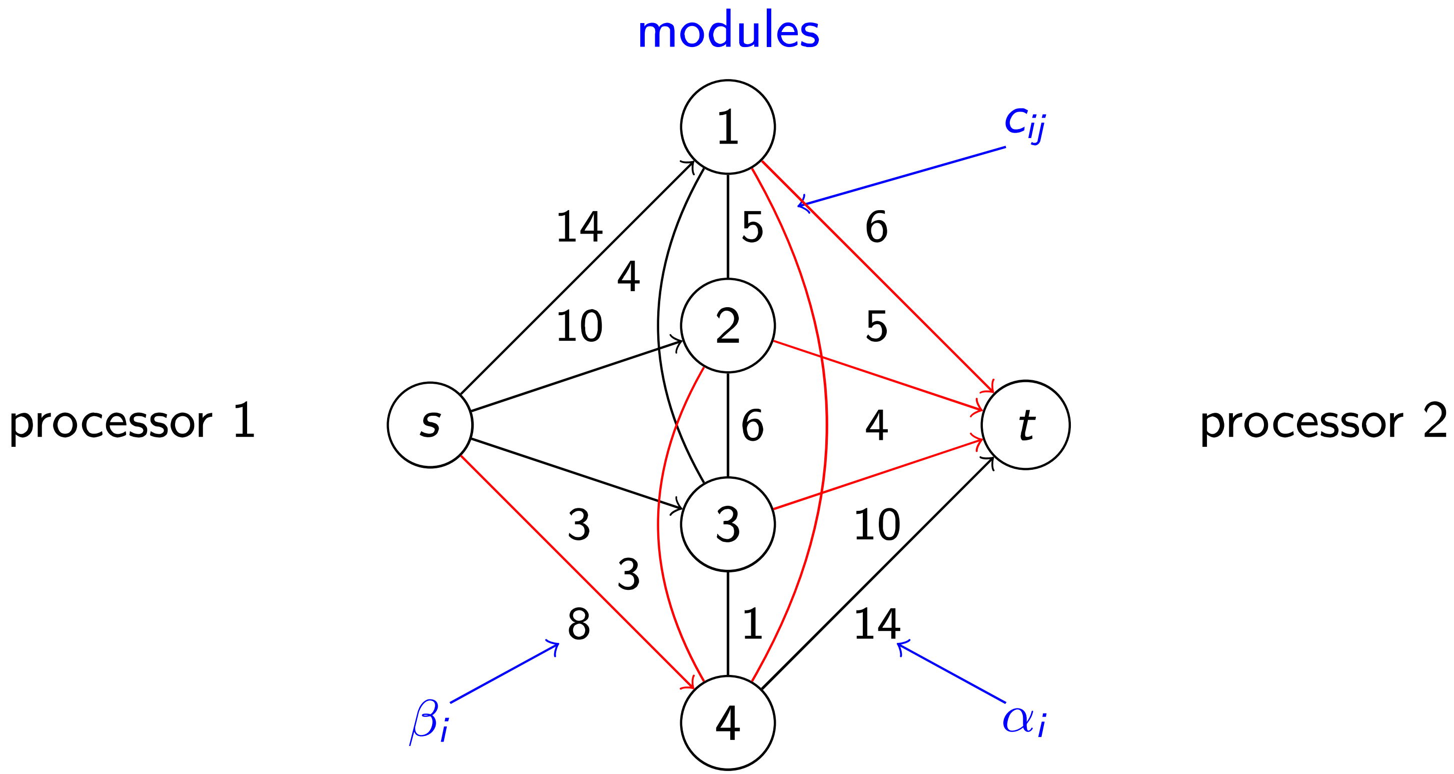
Assign n modules of a program to 2 processors so as to minimize the total cost (execution cost + communication cost).

Suppose we know:

- α_i = execution cost of module i on 1st processor, $1 \leq i \leq n$
- β_i = execution cost of module i on 2nd processor, $1 \leq i \leq n$
- 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.

2.4. Network flows



A cut separating s from t is an assignment of the n modules to the 2 processors

There is a correspondence between the $s - t$ cuts of minimum capacity and the minimum total cost assignments of the modules to the processors.