Foundations of Operations Research

Master of Science in Computer Engineering

Roberto Cordone
roberto.cordone@unimi.it

Tuesday 13.15 - 15.15

Thursday 10.15 - 13.15

http://homes.di.unimi.it/~cordone/courses/2014-for/2014-for.html

Lesson 8: Maximum flow — Como, Fall 2013

A flow model

A company manages a hydraulic network to supply water to a big consumer.

The network is known: in particular, each link has a maximum flow capacity, the position of the water sources and of the consumer are given.

The aim is to estimate the maximum amount of water which is possible to supply to the consumer through the network.

The network can be modelled by an arc-weighted directed graph $G = (N, A, k)$

- the nodes stand for water sources and pipeline crossings, while a fictitious node s (source) represents the overall origin of water, and node t (sink) represents the consumer
- the arcs stand for pipelines (two opposite arcs for each pipeline), and fictitious arcs link s to each water source node
- the weight k_{ij} stands for the capacity of arc (i, j), i. e. the maximum feasible water flow in the pipeline (possibly different in the two directions)

A "human flow" model

An event will draw a large crowd in a square of the town centre.

The Civil Protection must define the maximum amount of people who can be admitted to the event, while guaranteeing that they will all be able to leave the town centre within a specified time.

The street network is known: in particular, the width of the streets and the relation between the width and the number of individuals per unit time who can move along each street are known.

We aim to estimate the maximum amount of people who can attend the event.

The network can be modelled by an arc-weighted directed graph $G = (N, A, k)$

- the nodes stand for crossings and squares:
  a source node s represents the square (origin of the crowd),
  a sink node t represents the external world (destination of the crowd)
- the arcs stand for streets (two opposite arcs for each street),
  plus fictitious arcs from each node on the border of the town centre to t
- the weight k_{ij} stands for the arc capacity (maximum number of individuals who can walk along the arc in a time unit)

The maximum flow problem

Given

- a directed graph $G = (N, A)$ with $n = |N|$ nodes and $m = |A|$ arcs
- a source node s with $\Delta_s^- = \emptyset$, and a sink node t with $\Delta_t^+ = \emptyset$ (the other nodes are denoted as transhipment nodes)
- a capacity function defined on the arcs: $k : A \to \mathbb{R}^+$

find a flow function defined on the arcs ($x : A \to \mathbb{R}^+$) such that

1. balance: in transhipment nodes the outgoing flow equals the ingoing flow

$$\sum_{(j,k)\in\Delta_j^+} x_{jk} = \sum_{(i,j)\in\Delta_j^-} x_{ij} \qquad j \in N \setminus \{s, t\}$$

2. capacity: the flow on each arc respects the capacity

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

3. the ingoing flow of the sink t is maximum:

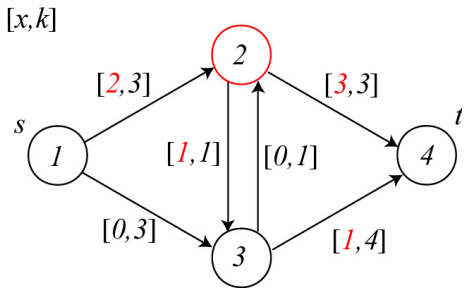
$$\max_{x:1 \text{ and } 2 \text{ hold}} f(x) = \sum_{(j,t)\in\Delta_t^-} x_{jt}$$

Examples

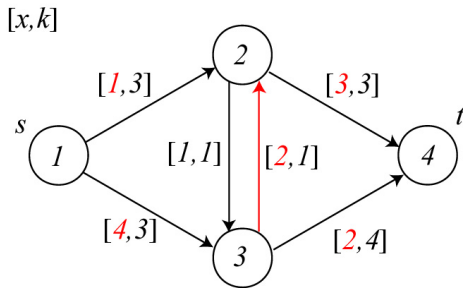
Let each arc (i, j) be labelled with its flow and capacity: $[x_{ij}, k_{ij}]$

x is not an optimal solution because it is. . .

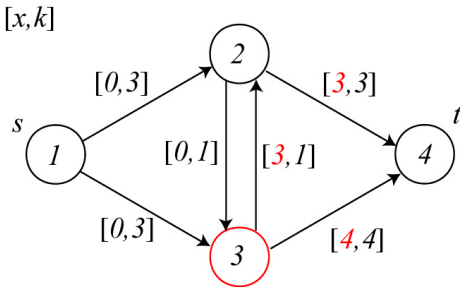
. . . nonbalanced



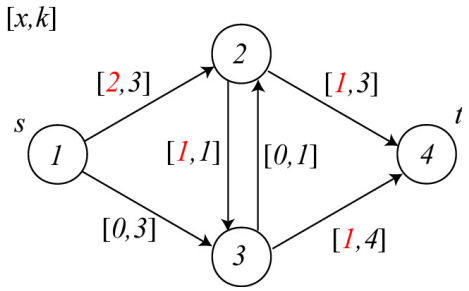
. . . violating a capacity



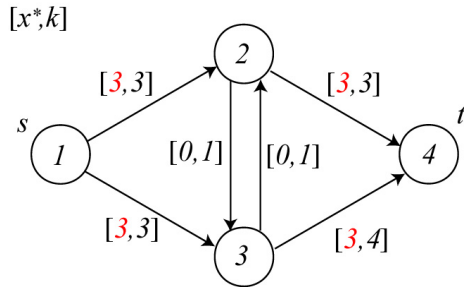
. . . nonbalanced



. . . nonminimal ($f(x) = 2$)



x^* is optimal ($f(x^*) = 6$)



A production planning model

During a planning horizon of various months $T = \{1, \ldots, n\}$, a factory produces an item, respecting a variable known production capacity.

The item is sold in batches at the end of each month, so as to satisfy (as much as possible) a known demand.

The unsold items are stored in a depot whose capacity is known.

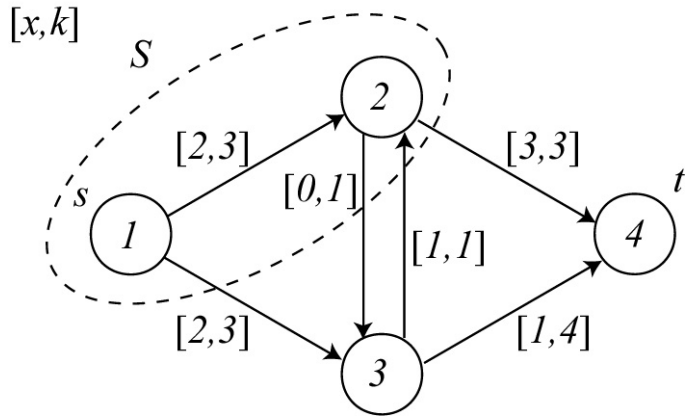
Determine the maximum number of items which remain in the depot at the end of the planning horizon.

The problem can be modelled by an arc-weighted directed graph $G = (N, A, k)$

- nodes: a factory node f_i, a storage node s_i and a demand node d_i for each month $i = 1, \ldots, n$, a final storage node s_{n+1}, a source s and a sink t
- arcs and arc capacities:
  - from s to all factory nodes: $k_{s,f_i} = +\infty$
  - from each factory node to the corresponding storage node: $k_{f_i,s_i} =$ production capacity in month i
  - from each storage node to the following one: $k_{s_i,s_{i+1}} =$ depot capacity
  - from each storage node to the corresponding demand node: $k_{s_i,d_i} =$ demand in month i
  - from each demand node and from the last storage node to t: $k_{d_i,t} = +\infty$ and $k_{s_n,t} =$ depot capacity

Capacity and flow associated to a cut

An (s,t)-cut is a cut induced by a subset of nodes including s and not t
A graph with n nodes admits 2^{n-2} (s,t)-cuts



$$S = \{1,2\} \Rightarrow \Delta_S^+ = \{(1,3),(2,3),(2,4)\} \text{ and } \Delta_S^- = \{(3,2)\}$$

Capacity of a cut is the sum of the capacities on the arcs of the outgoing cut

$$k_S = \sum_{(i,j)\in\Delta_S^+} k_{ij}$$

For example, $k_S = k_{13} + k_{23} + k_{24} = 7$

Flow across a cut is the difference of the outgoing and ingoing flow on the cut

$$x_S = \sum_{(i,j)\in\Delta_S^+} x_{ij} - \sum_{(i,j)\in\Delta_S^-} x_{ij}$$

For example, $x_S = x_{13} + x_{23} + x_{24} - x_{32} = 4$

Property 1: Uniformity of the flow

Given a feasible flow $x : A \to \mathbb{R}^+$, every (s,t)-cut is crossed by the same flow

Proof: *Given the outgoing flow of s, notice that no arc enters s*

$$x_{\{s\}} = \sum_{(s,k)\in\Delta_s^+} x_{sk} = \sum_{(s,k)\in\Delta_s^+} x_{sk} - \sum_{(i,s)\in\Delta_s^-} x_{is}$$

Given any subset S including s and not t ($s \in S \subseteq N \setminus \{t\}$), sum the balance constraints over all nodes in $S \setminus \{s\}$ and move both terms on the left side

$$\sum_{j\in S\setminus\{s\}} \sum_{(j,k)\in\Delta_j^+} x_{jk} - \sum_{j\in S\setminus\{s\}} \sum_{(i,j)\in\Delta_j^-} x_{ij} = 0$$

Sum the expression of $x_{\{s\}}$ obtained above

$$x_{\{s\}} = \sum_{j\in S} \sum_{(j,k)\in\Delta_j^+} x_{jk} - \sum_{j\in S} \sum_{(i,j)\in\Delta_j^-} x_{ij}$$

Split each sum, separating the arcs with both or only one extreme in S

$$x_{\{s\}} = \sum_{(j,k)\in A: j,k\in S} x_{jk} + \sum_{(j,k)\in\Delta_S^+} x_{jk} - \sum_{(i,j)\in A: i,j\in S} x_{ij} - \sum_{(i,j)\in\Delta_S^-} x_{ij} = x_S$$

The flow across any cut equals that going out of s; in particular, $f(x) = x_{\{s\}}$

Property 2: Relation between flow and capacity

Given any feasible flow $x : A \to \mathbb{R}^+$ and any (s, t)-cut, the capacity of the cut exceeds the flow across the cut

Proof: *Let S be any subset including s and not t: $s \in S \subseteq N \setminus \{t\}$.*

By definition

$$x_S = \sum_{(i,j)\in\Delta_S^+} x_{ij} - \sum_{(i,j)\in\Delta_S^-} x_{ij}$$

Since $x_{ij} \geq 0$ for all $(i, j) \in A$

$$x_S \leq \sum_{(i,j)\in\Delta_S^+} x_{ij}$$

Since $x_{ij} \leq k_{ij}$ for all $(i, j) \in A$

$$x_S \leq \sum_{(i,j)\in\Delta_S^+} k_{ij} = k_S$$

which is the thesis

Duality between maximum flow and minimum cut

The relation between flow and capacity is an example of duality, that is a general relation between a maximization and a minimization problem:

- weak duality when each solution of the minimization problem costs no less than each solution of the maximization problem; as a consequence, the minimum of the former is not smaller than the maximum of the latter
- strong duality when the two optima coincide

A weak duality relates two problems defined on graph $G = (N, A)$, on nodes s and t and on capacity function $k : A \to \mathbb{R}^+$:

- the minimum cut problem: find an (s, t)-cut of minimum capacity
- the maximum flow problem: find a flow of maximum size from s to t

The capacity of any (s, t)-cut is not smaller than any feasible flow from s to t: hence, the minimum cut capacity is not smaller than the maximum flow

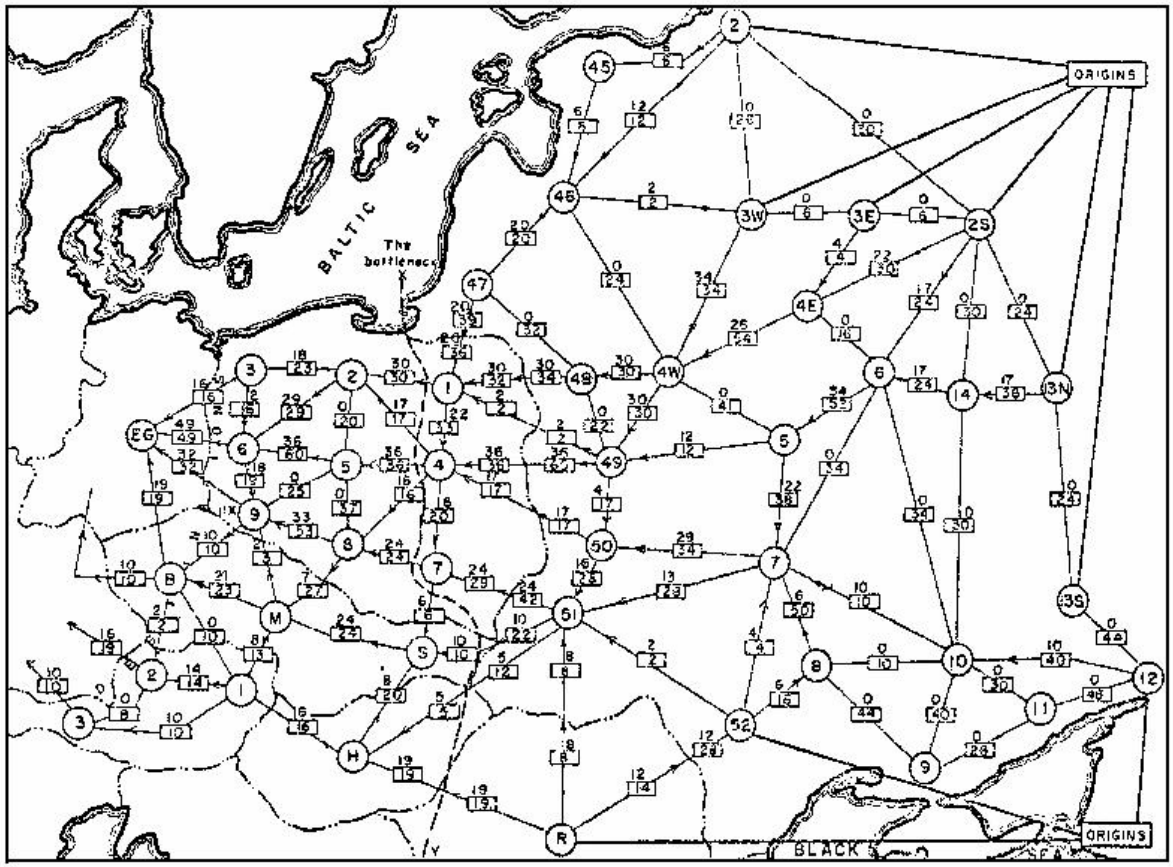
If the capacity of an (s, t)-cut equals a feasible flow x, both are optimal (the flow is maximum and the cut is minimum)

We will prove prove that also strong duality holds: the minimum capacity of an (s, t)-cut is always equal to the maximum flow from s to t

There is a subtle theoretical reason for that

A minimum cut model

What is the practical meaning of the minimum cut problem?



Find the bottleneck: links to destroy in order to reduce the flow

Conversely, select the links to improve in order to increase the flow

A load balancing model (1)

A set M of program modules must be distributed between two processors:

1. *the cost of executing module $i \in M$ on the first one is $\alpha_i = [\ 6\ 5\ 10\ 14\ ]$*
2. *the cost of executing module $i \in M$ on the second one is $\beta_i = [\ 14\ 10\ 3\ 8\ ]$*

A communication cost c_{ij} must be paid if modules i and j are assigned to different processors: $c_{12} = 5$, $c_{13} = 4$, $c_{14} = 4$, $c_{23} = 6$, $c_{24} = 3$, $c_{34} = 1$.

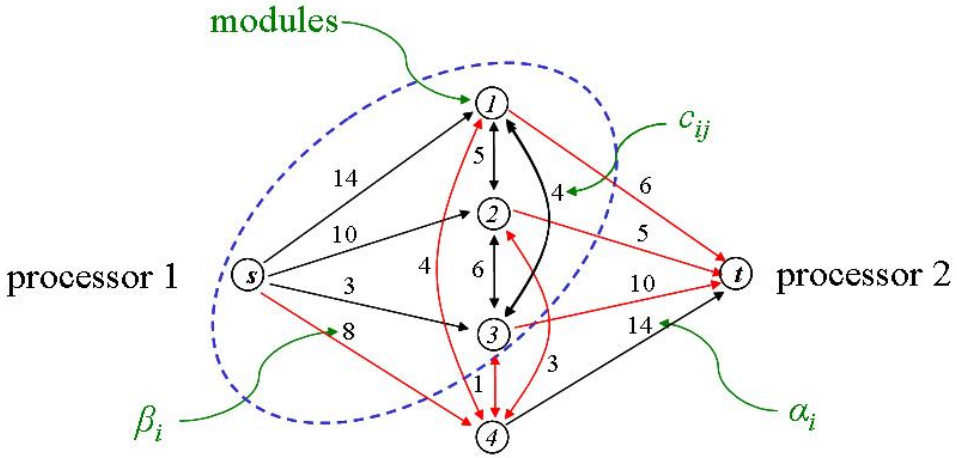
Distribute the modules minimizing the execution and communication cost.

Model the problem with an arc-weighted directed graph $G = (N, A, k)$:

- a node s for the first processor, a node t for the second processor, a node i for each module
- an arc from s to each module node, an arc from each module node to t, two opposite arcs between each pair of module nodes
- the weight function $k : A \to \mathbb{R}^+$ provides
  - the execution cost β_i for the arcs (s, i) with $i \in M$
  - the communication cost c_{ij} for the arcs $(i, j) \in M \times M$
  - the execution cost α_i for the arcs (i, t) with $i \in M$

Notice that the arcs express "separation", with the associated cost

A load balancing model (2)



Any (s, t)-cut describes a partition of the modules between the two processors:

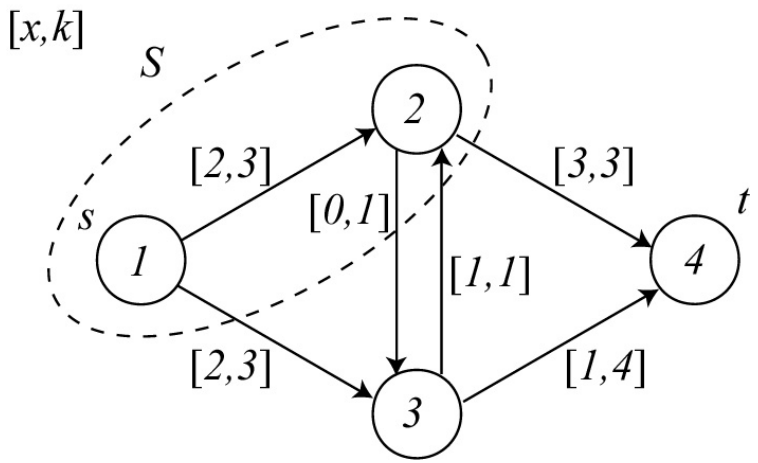
- S includes s and the modules assigned to the first processor
- $N \setminus S$ includes t and the modules assigned to the second processor
- the weights of the arcs in Δ_S^+ represent:
  - the execution costs β_j for the second processor
  - the communication costs c_{ij} between the two processors
  - the execution costs α_i for the first processor

Overall, the weight of any (s, t)-cut equals the cost of the assignment, and the minimum cut identifies the solution of minimum cost

Saturated and empty arcs

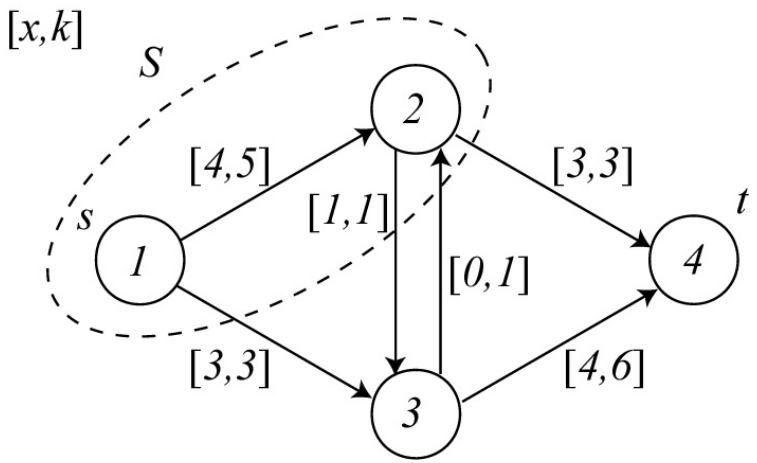
Given a feasible flow $x : A \to \mathbb{R}^+$, an arc (i, j) is

- saturated when its flow equals its capacity: $x_{ij} = k_{ij}$
  (*on a saturated arc the flow cannot increase*)
- empty when its flow is null: $x_{ij} = 0$
  (*on an empty arc the flow cannot decrease*)



Saturated arcs:
$(2, 4)$ and $(3, 2)$

Empty arcs: $(2, 3)$



Saturated arcs:
$(1, 3)$, $(2, 3)$ and $(3, 2)$

Empty arcs: $(3, 2)$

An optimality condition

The capacity of a cut and the flow across it are equal if and only if

- all outgoing arcs are saturated: $x_{ij} = k_{ij}$ for all $(i,j) \in \Delta_S^+$
- all ingoing arcs are empty: $x_{ij} = 0$ for all $(i,j) \in \Delta_S^-$

Proof: *The "if" part is trivial. As for the "only if" part:*

$$k_S = x_S \Rightarrow \sum_{(i,j)\in\Delta_S^+} k_{ij} = \sum_{(i,j)\in\Delta_S^+} x_{ij} - \sum_{(i,j)\in\Delta_S^-} x_{ij}$$

Since $x_{ij} \leq k_{ij}$ *for all* $(i,j) \in A$

$$\sum_{(i,j)\in\Delta_S^+} k_{ij} \leq \sum_{(i,j)\in\Delta_S^+} k_{ij} - \sum_{(i,j)\in\Delta_S^-} x_{ij} \Rightarrow \sum_{(i,j)\in\Delta_S^-} x_{ij} \leq 0$$

But since $x_{ij} \geq 0$ *for all* $(i,j) \in A$, *this implies* $x_{ij} = 0$ *for all* $(i,j) \in \Delta_S^-$

Consequently,

$$\sum_{(i,j)\in\Delta_S^+} k_{ij} = \sum_{(i,j)\in\Delta_S^+} x_{ij} \Rightarrow \sum_{(i,j)\in\Delta_S^+} (k_{ij} - x_{ij}) = 0$$

But since $k_{ij} - x_{ij} \geq 0$ *for all* $(i,j) \in A$, *this implies* $x_{ij} = k_{ij}$ *for all* $(i,j) \in \Delta_S^+$

An algorithmic idea

- Start with a trivial feasible flow: $x_{ij} = 0$ for all $(i, j) \in A$
- Increase the current flow while keeping it feasible
- If there is an (s, t)-cut whose capacity equals the flow, terminate; otherwise, go to step 2

How to increase the flow while keeping it feasible?

If the flow on arc (i, j) increases, one of the following effects must occur

- an increase in the flow on a subset of arcs going out of j
- a decrease in the flow on a subset of arcs going into j
- a combination of the previous two modifications

A similar effect must occur in i (just revert "outgoing" and "ingoing")

To keep it simple, modify the flow on a single arc for each extreme node:
the arcs with modified flow form a path between s and t,
because only in s and t it is feasible to change the flow on a single arc

Augmenting path

An augmenting path with respect to a feasible flow $x : A \to \mathbb{R}^+$ is an undirected path P from s to t, such that the flow value $f(x)$ can be increased modifying the flow on its arcs

In general the augmenting path is not directed; it consists of

- a set P^+ of forward arcs, directed from s to t, which must be nonsaturated
- a set P^- of backward arcs, directed from t to s, which must be nonempty

The flow value $f(x)$ increases by δ if

- the flow function x_{ij} increases by δ on all forward arcs $(i,j) \in P^+$
- the flow function x_{ij} decreases by δ on all backward arcs $(i,j) \in P^-$

The increase and decrease must respect the capacity along the whole path

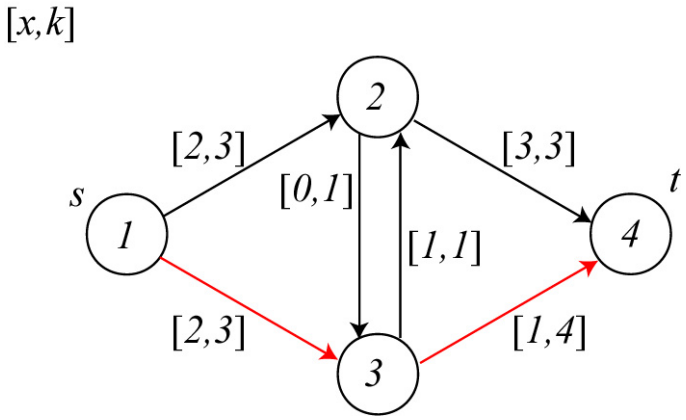
$$x_{ij} + \delta \leq k_{ij} \quad (i,j) \in P^+ \quad \text{and} \quad x_{ij} - \delta \geq 0 \quad (i,j) \in P^-$$

Therefore, the largest feasible flow modification is

$$\delta f(x,P) = \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right]$$

$\delta f(x,P) > 0$ if and only if all arcs are nonsaturated in P^+, nonempty in P^-

Example

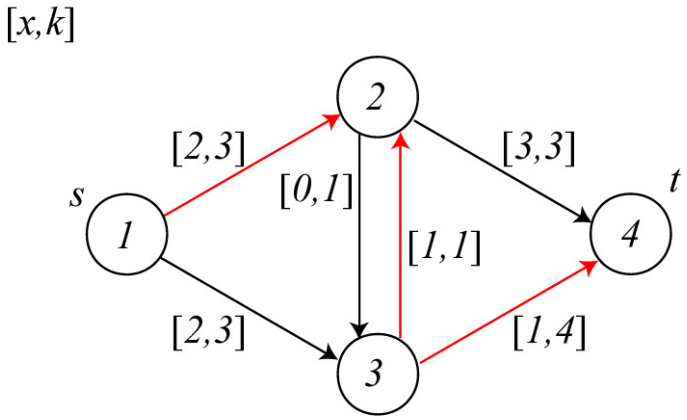


Augmenting path: $P = (1, 3, 4)$

Forward arcs: $P^+ = \{(1, 3), (3, 4)\}$

Backward arcs: $P^- = \emptyset$

$\delta f\,(x, P) = \min\,(k_{13} - x_{13}, k_{34} - x_{34}) = 1$



Augmenting path: $P = (1, 2, 3, 4)$

Forward arcs: $P^+ = \{(1, 2), (3, 4)\}$

Backward arcs: $P^- = \{(3, 2)\}$

$\delta f\,(x, P) = \min\,(k_{12} - x_{12}, x_{32}, k_{34} - x_{34}) = 1$

Ford-Fulkerson's algorithm (1956)

1. start from zero flow
2. if no augmenting path exists, terminate;
3. otherwise, add to the current flow the maximum feasible flow for the path found

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i,j) \in A$; { trivial feasible flow }

Stop := False;

Repeat

$P :=$ FindAugmentingPath(s, t, N, A, k, x);
If $\nexists P$
then Stop := True;
else

$$\delta := \min \left[\min_{(i,j)\in P^+} \left(k_{ij} - x_{ij}\right), \min_{(i,j)\in P^-} x_{ij} \right];$$

$f := f + \delta$;
$x_{ij} := x_{ij} + \delta$ for each $(i,j) \in P^+$;
$x_{ij} := x_{ij} - \delta$ for each $(i,j) \in P^-$;

until Stop = True;

Return (f, x);

How can one find an augmenting path? Why does this work?

Finding an augmenting path

Given graph $G = (N, A)$, the capacity function k and a feasible flow x, the auxiliary incremental network is an arc-weighted directed graph $\bar{G} = (N, \bar{A})$ with a residual capacity function $\bar{k} : \bar{A} \to \mathbb{R}^+$ defined as follows

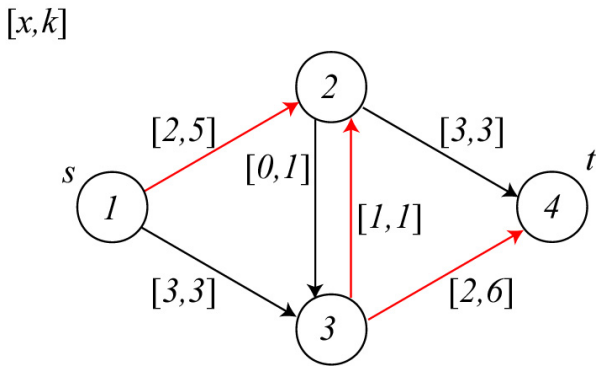
1. the nodes are the same as in the original graph
2. for each nonsaturated arc $(i, j) \in A$, add to $\bar{A}$ an arc (i, j) of residual capacity $\bar{k}_{ij} = k_{ij} - x_{ij}$
3. for each nonempty arc $(i, j) \in A$, add to $\bar{A}$ an arc (j, i) of residual capacity $\bar{k}_{ij} = x_{ij}$
4. resulting parallel arcs can be merged

The augmenting paths in the original network correspond one-to-one to the paths from s to t in the incremental network

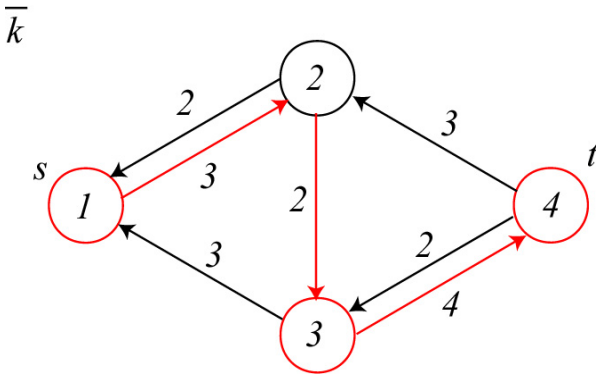
With a bit of intuition, one can work directly on the original graph: simply "see" where new flow can be added or existing flow can be reduced

Finding an augmenting path (example 1)

Given an original network $G(N, A, k)$ and a feasible flow x



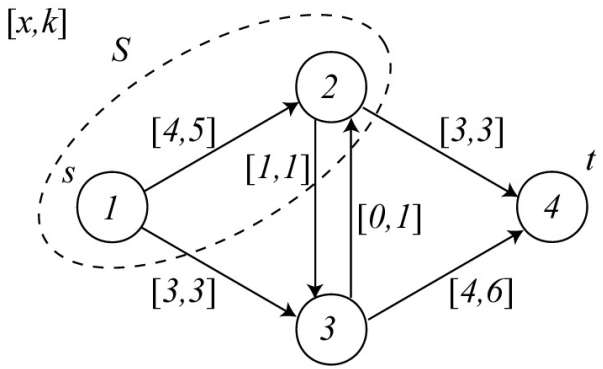
build the incremental network $G(N, \bar{A}, \bar{k})$: reverse the saturated arcs, keep the empty arcs, split in two the other arcs, merge a double arc $(2, 3)$



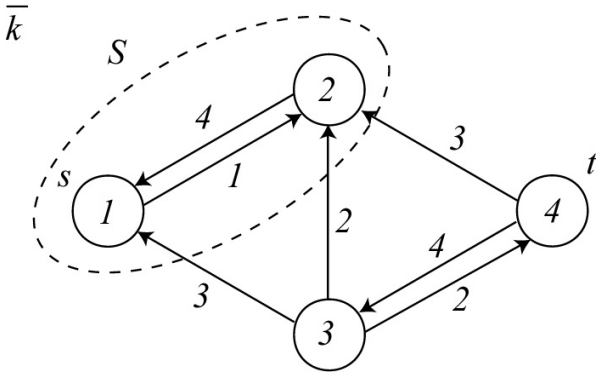
The augmenting path in the original network yields a path in the incremental network

Finding an augmenting path (example 2)

Given an original network $G(N, A, k)$ and a feasible flow x



build the incremental network $G(N, \bar{A}, \bar{k})$: reverse the saturated arcs, keep the empty arcs, split in two the other arcs, merge a double arc $(2, 3)$



No path in the incremental network $\Leftrightarrow$ No augmenting path the original one $\Leftrightarrow$ $\Leftrightarrow$ The cut induced by the subset S of nodes reachable from s has only saturated forward arcs and empty backward arcs $\Rightarrow$ the current flow is optimal!

Application of Ford-Fulkerson's algorithm (1)

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i, j) \in A$;

Stop := False;

Repeat

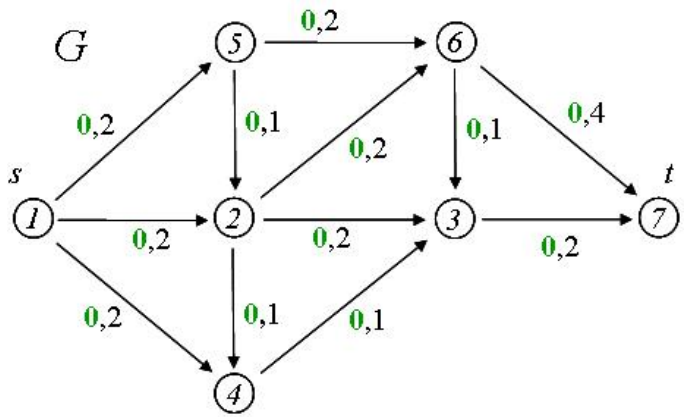
$P :=$ FindAugmentingPath(s, t, N, A, k, x);
If $\nexists P$
then Stop := True;
else

$$\delta := \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right];$$

$f := f + \delta$;
$x_{ij} := x_{ij} + \delta$ for each $(i, j) \in P^+$;
$x_{ij} := x_{ij} - \delta$ for each $(i, j) \in P^-$;

until Stop = True;

Return (f, x);



$x_{ij} := 0$ for all $(i, j) \in A$

Application of Ford-Fulkerson's algorithm (2)

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i, j) \in A$;

Stop := False;

Repeat

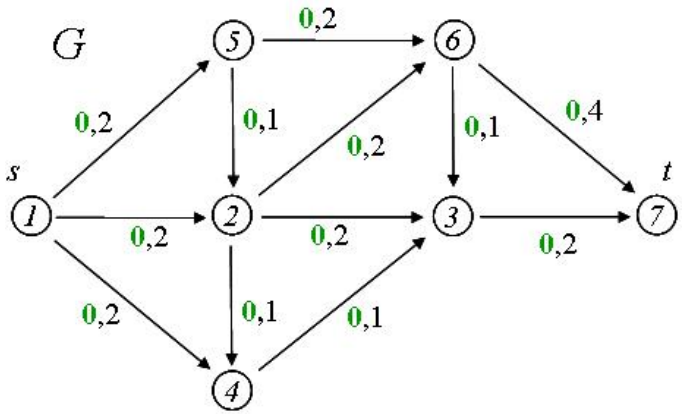
$P :=$ FindAugmentingPath(s, t, N, A, k, x);
If $\nexists P$
then Stop := True;
else

$$\delta := \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right];$$

$f := f + \delta$;
$x_{ij} := x_{ij} + \delta$ for each $(i, j) \in P^+$;
$x_{ij} := x_{ij} - \delta$ for each $(i, j) \in P^-$;

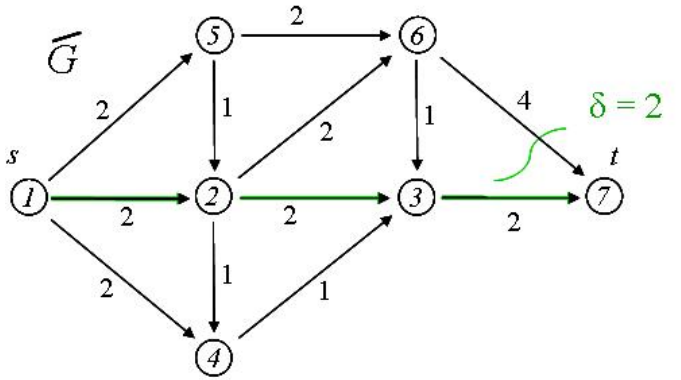
until Stop = True;

Return (f, x);



Augmenting path: $P = (1, 2, 3, 7)$

$\delta f\,(x, P) = 2$



Application of Ford-Fulkerson's algorithm (3)

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i, j) \in A$;

Stop := False;

Repeat

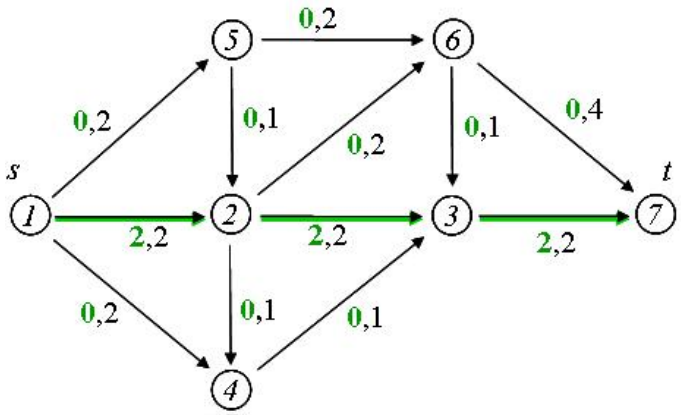
$P :=$ FindAugmentingPath(s, t, N, A, k, x);
If $\nexists P$
then Stop := True;
else

$$\delta := \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right];$$

$f := f + \delta$;
$x_{ij} := x_{ij} + \delta$ for each $(i, j) \in P^+$;
$x_{ij} := x_{ij} - \delta$ for each $(i, j) \in P^-$;

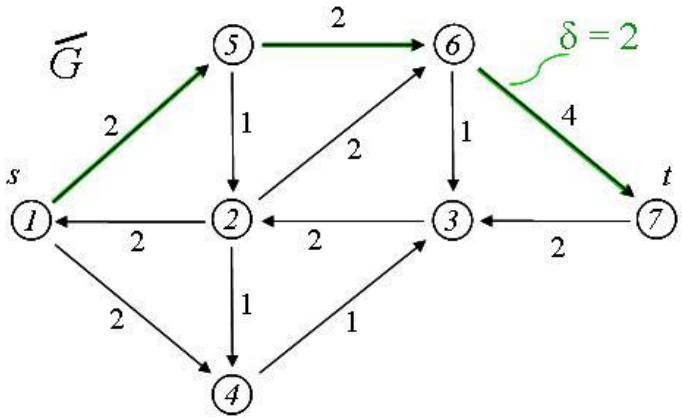
until Stop = True;

Return (f, x);



Augmenting path: $P = (1, 5, 6, 7)$

$\delta f\,(x, P) = 2$



Application of Ford-Fulkerson's algorithm (4)

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i, j) \in A$;

Stop := False;

Repeat

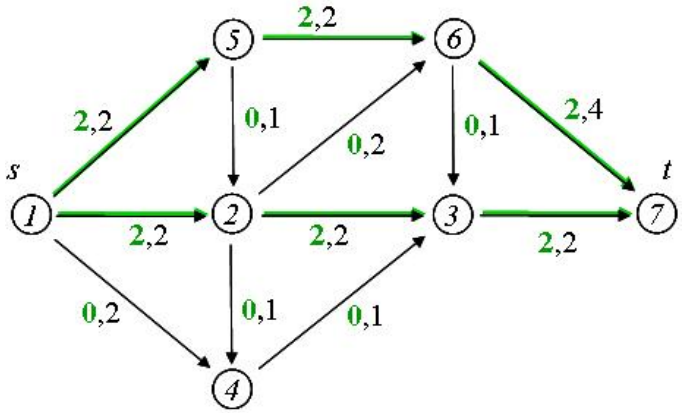
$P :=$ FindAugmentingPath(s, t, N, A, k, x);
If $\nexists P$
then Stop := True;
else

$$\delta := \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right];$$

$f := f + \delta$;
$x_{ij} := x_{ij} + \delta$ for each $(i, j) \in P^+$;
$x_{ij} := x_{ij} - \delta$ for each $(i, j) \in P^-$;

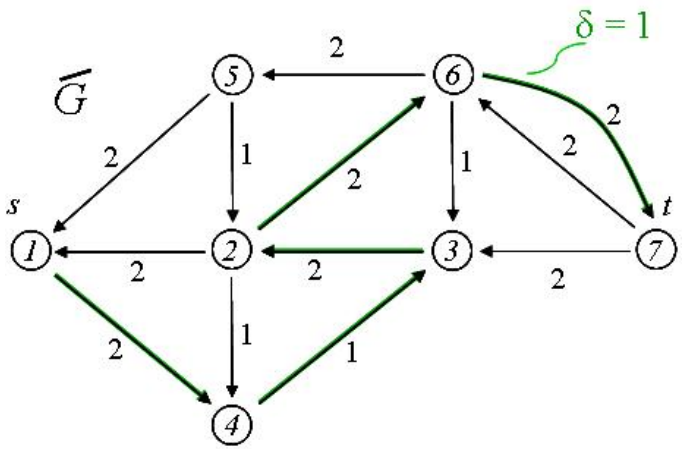
until Stop = True;

Return (f, x);



Augmenting path:
$P = (1, 4, 3, 2, 6, 7)$

$\delta f\,(x, P) = 1$



Application of Ford-Fulkerson's algorithm (5)

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i, j) \in A$;

Stop := False;

Repeat

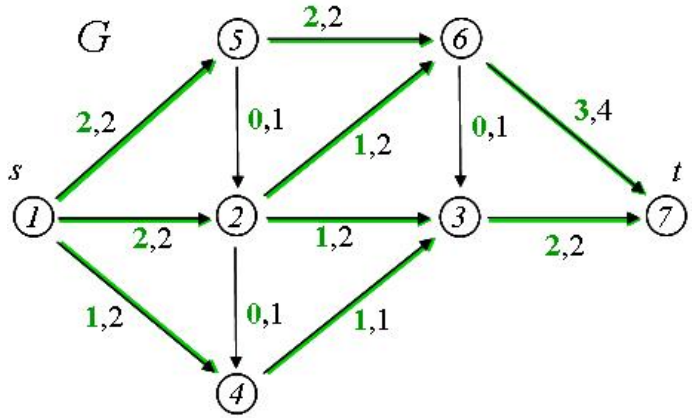
$P :=$ FindAugmentingPath(s, t, N, A, k, x);
If $\nexists P$
then Stop := True;
else

$$\delta := \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right];$$

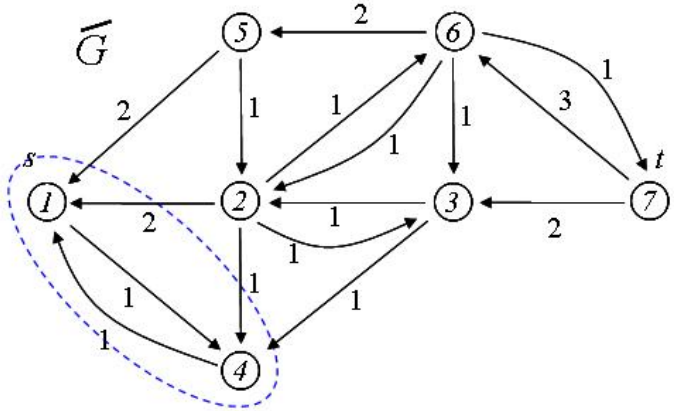
$f := f + \delta$;
$x_{ij} := x_{ij} + \delta$ for each $(i, j) \in P^+$;
$x_{ij} := x_{ij} - \delta$ for each $(i, j) \in P^-$;

until Stop = True;

Return (f, x);



No augmenting path:
The nodes reachable from s form $S = \{1, 4\}$ with $\bar{\Delta}_S^+ = \emptyset$



Optimal solution: $f(x) = x_S = k_S = 5$
all outgoing arcs are saturated
all ingoing arcs are empty

Computational complexity (1)

$$T = T_{\text{in}} + \sum_{i=1}^{i_{\max}} T_{\text{iter}}^{(i)}$$

FordFulkerson(N, A, k)

$f := 0$; $x_{ij} := 0$ for each $(i,j) \in A$; — $T_{\text{in}} \in O(m)$ (*set the flow on the arcs of* A)

Stop := False;

Repeat — How many iterations $i_{\max}$ of the loop?

  $P :=$ FindAugmentingPath(s, t, N, A, k, x); — $O(m)$ (*visit all arcs of* $\bar{A}$)

  If $\nexists P$

    then Stop := True;

    else

      $\delta := \min \left[\min_{(i,j)\in P^+} (k_{ij} - x_{ij}), \min_{(i,j)\in P^-} x_{ij} \right]$; — $O(n)$ (*scan the arcs of the augmenting path*)

      $f := f + \delta$; — $O(1)$

      $x_{ij} := x_{ij} + \delta$ for each $(i,j) \in P^+$; — $O(n)$ (*scan the arcs of the augmenting path*)

      $x_{ij} := x_{ij} - \delta$ for each $(i,j) \in P^-$; — $O(n)$ (*scan the arcs of the augmenting path*)

until Stop = True;

Return (f, x);

The overall complexity is $T \in O(i_{\max} m)$,

where $i_{\max}$ is the number of augmenting paths required to reach the optimum

Computational complexity (2)

How many iterations are necessary?

- at first, f and all x_{ij} are set to 0
- at each iteration, f increases by δ and all x_{ij} change by δ or keep unchanged
- if all capacities and flows are integer, then the flow update is integer, and the flows remain integer after the update

$$k_{ij}, x_{ij} \in \mathbb{N} \text{ for all } (i,j) \in A \Rightarrow \delta f(x,P) \in \mathbb{N} \Rightarrow x_{ij} \pm \delta f(x,P) \in \mathbb{N}$$

- since the path are augmenting, $\delta > 0 \Rightarrow \delta \geq 1$
- the maximum flow equals the minimum cut
- the minimum cut is lower than the total capacity of all the arcs, which is lower than $k_{\max}\ m$, where $k_{\max} = \max\limits_{(i,j)\in A} k_{ij}$

Consequently

$$i_{\max} \leq \frac{f^*}{\delta_{\min}} \leq \frac{m k_{\max}}{1} = m\ k_{\max}$$

and the overall complexity is $O\left(k_{\max} m^2\right)$

Pseudopolynomial algorithms

However, a maximum flow instance is given by a list of m arcs, and for each arc

- the index of the tail node, an integer coded into $\log_2 n$ bits
- the index of the head node, an integer coded into $\log_2 n$ bits
- the capacity, an integer coded into $\log_2 k_{\max}$ bits

Thus, the size of the instance is $m(2\log_2 n + \log_2 k_{\max})$ bits

Usually, $n < k_{\max}$ and the size is in $O(m \log k_{\max})$

The problem is that $k_{\max} m^2$ is

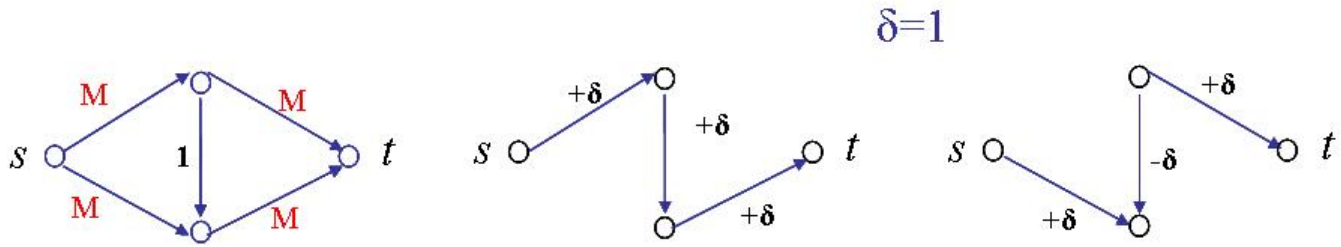
- polynomial with respect to m
- but but exponential with respect to $\log k_{\max}$

Consequently, Ford and Fulkerson's algorithm is not polynomial, but at least it is not exponential with respect to the number of nodes and arcs

A pseudopolynomial algorithm is an algorithm which is

- polynomial with respect to the number of elements of the problem
- nonpolynomial with respect to the values of the weight functions

An example of inefficient application



Ford and Fulkerson's algorithm might choose the following sequence of augmenting paths:

1. $P = (1, 2, 3, 4)$: $\delta f(x, P) = 1$ and $f(x) = 1$
2. $P = (1, 3, 2, 4)$: $\delta f(x, P) = 1$ and $f(x) = 2$
3. $P = (1, 2, 3, 4)$: $\delta f(x, P) = 1$ and $f(x) = 3$
4. ...

thus requiring $2M$ steps, where M can be set to any value

The maximum flow problem admits polynomial algorithms derived from Ford and Fulkerson's: the breakthrough is to choose a smart augmenting path (*for example, the path with the minimum number of arcs*)

A linear programming formulation

The maximum flow problem can be put into linear programming form with the following decision variables

- f: flow value (uniform across every (s,t)-cut)
- x_{ij}: flow on arc $(i,j) \in A$

$$\max \ f \tag{1}$$

$$\sum_{(i,j)\in\Delta_j^-} x_{ij} - \sum_{(j,k)\in\Delta_j^+} x_{jk} = \begin{cases} f & \text{if } j = s \\ 0 & \text{if } j \in N \setminus \{s,t\} \\ -f & \text{if } j = t \end{cases} \tag{2}$$

$$0 \leq x_{ij} \leq k_{ij} \qquad (i,j) \in A \tag{3}$$

It can be proved that this specific linear programming problem enjoys the integrality property: if the data are integer, the optimal solution is integer (if $k_{ij} \in \mathbb{N}$, then $x_{ij} \in \mathbb{N}$)

A project assignment model

A company with m workers has n ongoing projects. Each worker has specific skills, which can be appropriate or not to perform each project.

It is necessary to assign the projects to the workers so that:

- *each worker is assigned at most one project*
- *each project is assigned to at most one worker*

Assign the projects to the workers so that the maximum number of projects can be performed.

The problem can be modelled by an arc-weighted directed graph $G = (N, A)$:

- the nodes represent the workers and the projects, plus a fictitious source node s and a fictitious sink node t
- the arcs represent the compatibility relation between workers and projects, fictitious arcs link s to each worker node, and each project node to t
- the capacity function $k : A \to \mathbb{R}^+$ is defined as: $k_{si} = 1$ for all i, $k_{jt} = 1$ for all j, $k_{ij} = +\infty$ for all $i \neq s$, $j \neq t$

An arc with $x_{ij} = 1$ suggests to assign worker i to project j

Notice that the integrality property guarantees that $x_{ij} \in \{0, 1\}$ for all $(i, j) \in A$